სოხუმის სახელმწიფო უნივერსიტეტი
ი. ვეკუას სახელობის სოხუმის ფიზიკა-ტექნიკის ინსტიტუტი

რამაზ შამუგია

INTERNET და WEB ტექნოლოგიების საფუძვლები

(HTML, CSS, My SQL, PHP, Dreamweaver MX)

რედაქტორისაგან

წინამდებარე სახელმძღვანელოში, **PHP-MySQL** პლატფორმაზე ინტერაქტიული ვებ-საიტების შემუშავებისათვის, საბაზისო საშუალების სახით, შერჩეულია პოპულარული ვებ-რედაქტორი-**Macromedia Dreamweaver MX**.

სახელმძღვანელოში, მაგალითის სახით განხილულია პროგრამების არქივისა და ელექტრონული სტატიების ბიბლიოთეკისათვის საიტის შექმნა. მასალა წარმოდგენილია პრინციპით: მარტივიდან რთულისაკენ. მარტივი სტატიკური ვებ-გვერდები შექმნილია **DreamweaverMX** რედაქტორის საშუალებით, გზადაგზა მოყვანილია **HTML** ენის აღწერა. MySQL მონაცემთა ბაზებიდან, მონაცემების ამომკრები მარტივი სერვერული გვერდები, აგრეთვე შექმნილია **DreamweaverMX** რედაქტორში. ამასთან დაწვრილებით არის განხილული **DreamweaverMX**-ის მიერ შექმნილი **PHP**-ს ყველა სცენარი, და აღწერილია მათი მუშაობა. პარალელურად არის მოცემული შესავალი მონაცემთა ბაზებში და მოცემულია **PHP** ენის მოკლე აღწერა. უფრო რთული ვებ-გვერდები შექმნილია **PHP-MySQL**-ის საშუალებებით, **DreamweaverMX**-ის გამოყენებლად. მოცემულია განვითარებული ვებ-პორტალის ელემენტების შემუშავების მაგალითები: ასარჩევი ფერების გამა, ვებ-ინტერფეისის საშუალებით ფაილების მართვა, საკუთარი გზავნილების სია და სხვა.

სახელმძღვანელო განკუთვნილია ინფორმატიკის სპეციალობის სტუდენტებისა და მაგისტრანტების, აგრეთვე ინტერნეტ და ვებ ტექნოლოგიებით დაინტერესებული სხვა სპეციალისტებისათვის, ვისაც გააჩნია საბაზისო ცოდნა საინფორმაციო-ტექნოლოგიების დარგში.

რედაქტორი:
აკადემიური დოქტორი, სტუ-ს სრული პროფესორი ზ.მიქაძე

რეცენზენტები:

ტ.მ. დოქტორი, სტუ-ს სრული პროფესორი, ზ. გასიტაშვილი,

აკადემიური დოქტორი, სტუ-ს ასოც. პროფესორი ლ. ადამია

სარჩევი

რაჟდნ (რაჟიკო) აქვსენტის-ძე
შამუგიას ნათელ ხსოვნას.

წინასიტყვაობა

წინამდებარე სახელმძღვანელო "**Internet დაWeb ტექნოლოგიების საფუძვლები (HTML,CSS,My SQL,PHP,,,Dreamweaver MX")** წარმოადგენს სტატიკური და ინტერაქტიული დინამიკური საიტების შექმნის სახელმძღვანელოს. იგი წარმოდგენილია ორ ნაწილად. წიგნში ძირითადად განხილულია ისეთი საკითხები, როგორიცაა ვებ-დაპროგრამების საწყისები, შესავალი მონაცემთა ბაზებში, ვებ-გვერდების შემუშავება **PHP** ენის საშუალებებით, სერვერული გვერდების შექმნა ვიზუალური **HTML** რედაქტორის-**DreamweaverMX**-ის გამოყენებით. თეორიული მასალის საილუსტრაციოდ წიგნში მოცემულია ვებ-საიტების აგების კონკრეტული მაგალითები. პირველ ნაწილში, რომელიც ოთხი თავისაგან შედგება, ინტერნეტის და ვებ-ტექნოლოგიების ზოგად საკითხებთან ერთად, განხილულია სტატიკური ვებ-გვერდების აგებისათვის, ჰიპერტექსტური მონიშვნის ენის-**HTML**, კასკადური სტილების ცხრილების-**CSS** და ვიზუალური **HTML** რედაქტორის **DreamweaverMX**-ის ძირითადი მეთოდების, ხერხებისა და საშუალებების გამოყენების პრინციპები. მეორე ნაწილში, რომელიც ხუთი თავისაგან არის შემდგარი, განხილულია იმავე ენებისა და რედაქტორის გამოყენების საკითხები, აქტიური სერვერული ვებ-გვერდების ასაგებად. წიგნს ასევე დართული აქვს ოთხი დანართი. დანართებში წარმოდგენილია ვებ-სერვერ **Appache**-თან, **PHP**-დამმუშავებელთან (ინტერპრეტატორთან) მონაცემთა ბაზების **MySQL** სერვერთან და **phpMyAdmin**-თან პრაქტიკული მუშაობის ასათვისებლად განკუთვნილი მასალა, რომელიც აუცილებელია დინამიური ვებ-საიტების შექმნისა და ინტერნეტ-ქსელში მათი მუშაობის უზრუნველსაყოფად.

PHP-MySQL პლატფორმაზე ინტერაქტიული ვებ-საიტების შემუშავების საბაზო საშუალების სახით შერჩეულია ერთ-ერთი ყველაზე მძლავრი და ყველაზე პოპულარული ვიზუალური ვებ-რედაქტორი **Macromedia Dreamweaver MX.**

წიგნში, მაგალითის სახით, განხილულია საიტის შექმნა პროგრამების არქივისა და ელექტრონული სტატიებისთვის. მასალა წარმოდგენილია პრინციპით: მარტივიდან რთულისაკენ. მარტივი

სტატიკური ვებ-გვერდები შექმნილია **DreamweaverMX** რედაქტორის საშუალებით, გზადაგზა მოყვანილია **HTML** ენის აღწერა. **DreamweaverMX** რედაქტორშია აგრეთვე შექმნილი **MySQL** მონაცემთა ბაზებიდან მონაცემების ამომკრები მარტივი სერვერული ვებ-გვერდები. ამასთან დაწვრილებით არის განხილული **DreamweaverMX**-ის მიერ შექმნილი **PHP**-ს ყველა სცენარი და აღწერილია მათი მუშაობა. პარალელურად მოცემულია შესავალი მონაცემთა ბაზებში და მოყვანილია **PHP** ენის მოკლე აღწერა. უფრო რთული ვებ-გვერდები შექმნილია **PHP-MySQL**-ის საშუალებებით, **DreamweaverMX**-ის გამოუყენებლად. მოყვანილია განვითარებული ვებ-პორტალის ელემენტების შემუშავების მაგალითები: ფერების გამის არჩევა, ვებ-ინტერფეისის საშუალებით ფაილების მართვა, საკუთარი გზავნილების სია და სხვა.

წიგნი განკუთვნილია მომხმარებლების და პროგრამისტების, სტუდენტების და მასწავლებლების, აგრეთვე ვებ-დიზაინით დაინტერესებულ სხვა პირთა ფართო წრისათვის, ვისაც გააჩნია საბაზო ცოდნა ინტერნეტ-ტექნოლოგიების შესახებ. წიგნის წაკითხვის შემდეგ, მკითხველს შეეძლება ცხრილების, გრაფიკების, ანიმაციების, ბგერების და მუსიკის შემცველი, პროფესიონალური ვებ-საიტებისა და ვებ-გვერდების შემუშავება, აგრეთვე ინტერაქტიული ელემენტების შემცველი ვებ-საიტების ინტერნეტში განთავსება.

წიგნში წარმოდგენილი მასალა გასაგები იქნება დამწყებთათვის და ამავე დროს უფრო გამოცდილი ვებ-პროგრამისტებიც იპოვიან მასში მათთვის საინტერესო და სასარგებლო მასალას.

ვფიქრობ, რომ წარმოდგენილი წიგნი მნიშვნელოვნად დაეხმარება ინფორმატიკის, მართვის სისტემებისა და საინფორმაციო ტექნოლოგიების სპეციალობის სტუდენტებს და ვებ-დიზაინით დაინტერესებულ სხვა პირებს, მათ შორის ვებ და ინტერნეტ პროგრამისტებს, საინფორმაციო ტექნოლოგიების შესახებ აუცილებელი ცოდნის მიღების, განმტკიცების და გაღრმავების საქმეში.

წარმოდგენილი სახელმძღვანელო არის ინტერნეტ და ვებ დაპროგრამების ძირითადი საკითხების, სპეციფიკური მიდგომით, ქართულ ენაზე გადმოცემის ერთ-ერთი პირველი მცდელობა, ამიტომ ის არ შეიძლება დაზღვეული იყოს ხარვეზებისაგან. აღნიშნულიდან გამომდინარე, ავტორი მადლიერებით მიიღებს წიგნში გადმოცემული მასალის დახვეწისა და სრულყოფის მიზნით

გამოთქმულ შენიშვნებსა და მოსაზრებებს. მათი გამოგზავნა მსურველებს შეუძლიათ მისამართზე: rmz.shamugia@gmail.com.

შესავალი

სერვერული **PHP** **Web**-გვერდებისა და **MySQL**–მონაცემთა ბაზების გამოყენებით, **Web**-საიტების შექმნის შესახებ, უკანასკნელ პერიოდში, ძალზე ბევრი სახელმძღვანელოა დაწერილი.
”PHP”, **”MySQL”**, ” **PHP** და **MySQL** ”,” **MySQL** და **PHP** ”-ასეთი დასახელების წიგნებს მრავლად შევხვდებით მაღაზიების დახლებზე. ღირს კი, კიდევ ერთი წიგნის დაწერა ამავე თემაზე?
საქმე იმაში კი არ გახლავთ, ღირს, თუ არა დაწერა. არც იმაში, თუ რის შესახებ უნდა დაიწეროს. არამედ იმაში, თუ როგორ უნდა დაიწეროს! დიახ სწორედ იმაში, როგორ უნდა დაიწეროს მსგავსი წიგნები.
მაგრამ, მოდით ყველაფერზე ვისაუბროთ თანმიმდევრულად. პოტენციურ მკითხველს, ალბათ უამრავი კითხვა დაუგროვდა, და ავტორს მოუწევს მათზე პასუხების გაცემა. ხოლო პასუხის გაცემა ჯობია აუჩქარებლად, მშვიდად, ზედმეტი ხმაურის გარეშე. მაშ ასე...

რაზეა საერთოდ საუბარი?

მართლაც, რა ხილია **PHP**, **MySQL** და **Dreamweaver MX** და რასთან მიირთმევენ მათ. დავიწყოთ სულ თავიდან.

MySQL-ეს არის მონაცემთა სერვერი, განსაკუთრებული პროგრამა, რომელიც მოწესრიგებული მონაცემების, მონაცემთა ბაზებად წოდებულ, განსაკუთრებულ სტრუქტურებში შენახვის საშუალებას იძლევა. მაგალითად ასეთ ბაზაში შესაძლოა ჩაიწეროს საინვენტარო წიგნი, მოსამსახურეების სია, რამდენიმე წლის საგამოცდო უწყისები, ბიბლიოთეკაში არსებული წიგნების ნუსხა და მრავალი სხვა. და, არა მხოლოდ ჩაიწეროს და შენახულ იქნას, არამედ მარტივად და სწრაფად მოძებნილ იქნან სასურველი მონაცემები პირველივე მოთხოვნისამებრ.

PHP-ეს არის პლატფორმა სერვერული Web-გვერდების შესაქმნელად. უხეშად რომ ვთქვათ, მისი საშუალებით შესაძლებელია დაიწეროს პროგრამა, რომელიც მოახდენს **Web**-გვერდების ფორმირებას, საიტის დამთვალიერებელის მოთხოვნის შესაბამისად და მათ გაგზავნას ისევ მომხმარებლისათვის. ასეთი **Web**-გვერდები შესაძლოა შეიცავდნენ მონაცემებს, რომლებიც მონაცემთა ბაზებში, მათ შორის **MySQL**-ში ინახებიან.

MySQL და **PHP** ცალცალკეც ძალზე შთამბეჭდავები არიან. მაგრამ ორივეს ერთად-უბრალოდ სასწაულების მოხდენა შეუძლიათ! მაგალითად, ინტერნეტ-მაღაზიისთვის **PHP**-ზე დაწერილ სერვერულ პროგრამას, შეუძლია **MySQL** მონაცემთა ბაზებიდან საქონლის ნუსხის ამოღება და მისგან ლამაზი **Web**-გვერდის ფორმირება. ხოლო იმავე ინტერნეტ-მაღაზიის შემადგენელი მეორე სერვერული პროგრამა, დამთვალიერებელს ეკითხება თუ რომელი საქონლის შეკვეთა სურს მას და რა მისამართზე უნდა იქნას გაგზავნილი არჩეული საქონელი. შემდგომ, თვითონვე წერს იმავე ბაზაში მონაცემებს ახალი შეკვეთის შესახებ.

გასაკვირი არ არის, რომ **PHP** და **MySQL** ასეთი პოპულარულები არიან. უფრო მეტიც, ისინი საკმაოდ მძლავრი პროგრამებია, რომელთა საშუალებითაც შესაძლებელია პრაქტიკულად **Web**-გვერდების შექმნასთან დაკავშირებული ნებისმიერი ჩანაფიქრის რეალიზება. აღსანიშნავია, აგრეთვე ის, რომ ისინი შესანიშნავად მუშაობენ ერთად, და-ლეგალური პროგრამების გამოყენების მომხრეების საყურადღებოდ!- აბსოლიტურად უფასონი არიან.

დიდი ინფორმაციული შემადგენლობის მქონე **Web**-საიტების შემქმნელებისათვის აღნიშნული პროგრამები ნამდვილ მიგნებას წარმოადგენენ. ამავე დროს მათი საშუალებით შესაძლებელია სრულყოფილი, კომერციული დანიშნულების ინტერნეტ-მაღაზიების შექმნაც.

DreamweaverMX-ი ჩვეულებრივი (**სტატიკური**) და სერვერული (**დინამიკური**),მათ შორის **PHP**-ზე დაწერილი **Web**-გვერდების შესამუშავებელი უმძლავრესი პროგრამული პაკეტია. მისი საშუალებით შესაძლებელია ურთულესი **PHP Web**-გვერდების შექმნა ისე, რომ ადამიანს არ გააჩნდეს რაიმე ცოდნა თვით **PHP**-ენისა და მისი საშუალებით პროგრამირების პრინციპების შესახებ. რაც შეეხება შესაძლებლობებს, **DreamweaverMX**-ს იმდენი შესაძლებლობა გააჩნია, რომ მათზე რამოდენიმე სქელტანიანი წიგნის დაწერა შეიძლება და მაინც აუცილებლად რაიმე მნიშვნელოვანი გამოგვრჩება.

DreamweaverMX-ი შექმნილია ფირმა **MacromediaMX**-ის მიერ, მაგრამ სამწუხაროდ იგი ფასიანია. მიუხედავად ამისა ხელმისაწვდომია სადემონსტრაციო ვერსიები, რომლებიც 30-დღის განმავლობაში მუშაობენ. ასე, რომ ყველა მსურველს შეუძლია მათი მოსინჯვა პრაქტიკაში, ასე ვთქვათ მისთვის "ხელის შევლება".

მაგრამ ადამინი, რომელსაც ჭიპი აქვს მოჭრილი **PHP**-ზე და **MySQL**-ზე, უკმაყოფილოდ შეიშმუშნება. "**PHP** და **MySQL**-გასაგებიაო-იტყვის ის. მაგრამ რა შუაშიაო აქ DreamweaverMX-ი".

DreamweaverMX: "აკეთე, როგორც მე "

აი აქ კი, ჩვენ მჭიდროდ მივუახლოვდით კითხვას პასუხზე-თუ როგორ უნდა დაიწეროს წიგნი. ზოგადად, ავტორის აზრით, ეს მთავარი კითხვაა, რომელზეც არა მხოლოდ კომპიუტერული თემატიკის, არამედ ნებისმიერი წიგნის ავტორმა უნდა იცოდეს პასუხი.

PHP-ზე და **MySQL**-ზე წიგნების წერა შესაძლებელია ბევრნაირად. შესაძლებელია დავიწყოთ ენის აღწერით, გავაგრძელოთ მარტივი გვერდების მაგალითებით, და დავამთავროთ **MySQL** მონაცემთა ბაზებთან მუშაობის აღწერით. შესაძლებელია აგრეთვე, დავიწყოთ პირდაპირ მონაცემთა ბაზებთან მუშაობის აღწერით, გავაგრძელოთ მათდამი მოთხოვნების შედგენის წვრილმანებით და დავასრულოთ ისეთი ჯურღმულებით, რაც მკითხველებში ციებ-ცხელებას

გამოიწვევს. კიდევ ასეც შეიძლება: მთლიანად ყურადღება გავამახვილოთ **MySQL**-ზე, ხოლ **PHP** აღვწეროთ ზედაპირულად, მხოლოდ იმისთვის რომ "რაიმე მაინც იყოს".
შეიძლება სულ სხვანაირადაც-როგორც ეს წიგნია დაწერილი. რა შუაშია აქ **DreamweaverMX**-ი? აი თუ რა შუაშია....
ადრე ნახსენები იყო, რომ **DreamweaverMX**-ს შეუძლია თვითონ შექმნას **PHP**-ს მარტივი სერვერული **Web**-გვერდები, რომლებიც უზრუნველყოფენ **MySQL**-ბაზებიდან მონაცემების ამოკრეფას. ამიტომ, დავიწყებთ იმით, რომ ჩვენი პირველი გვერდების შესაქმნელად ვისარგებლებთ **DreamweaverMX**-ის მომსახურებით და ყურადღებით დავაკვირდებით **PHP**- კოდს, რომელსაც იგი შექმნის. ამაში ჩვენ დაგვეხმარება **PHP**-ის ინგლისურენოვანი სახელმძღვანელო, რომელიც შეგვიძლია მოვიპოვოთ აღნიშნული პლატფორმის ოფიციალურ საიტზე-**http://www.php.net.**
როდესაც ამ გზით ჩვენ შევიძენთ გარკვეულ გამოცდილებას, ჩვენ თვითონ დავიწყებთ სერვერული გვერდების კოდების წერას.
ბოლოსდაბოლოს **DreamweaverMX**-ის შესაძლებლობები **PHP**-პროგრამირებაში არც თუ ისე დიდია. ჩვენთვის საჭირო ბევრ რამეს ის უბრალოდ ვერ შეძლებს. ჩვენ კი შევძლებთ.
ფაქტიურად **PHP**-ს სერვერული გვერდების წერას ჩვენ ვისწავლით მზა მაგალითებზე. ეს პროგრამირების სწავლის საუკეთესო გზაა. ავტორმა თვითონ სწორედ ასევე დაიწყო თავის დროზე, **DreamweaverMX**-ის მიერ შექმნილი **PHP**-კოდების შესწავლით.

"აკეთე როგორც მე! ისწავლე ჩემგან!"-ბრძანებს **DreamweaverMX**-ი. ღირს ამ მოწოდების გათვალისწინება, თუნდაც საწყის ეტაპზე.
კომპიუტერების სპეციალისტი მხრებს აიჩეჩავს: "პროგრამირება-ეს უპირველეს ყოვლისა ხომ შემოქმედებაა, ხოლო კომპიუტერები ხელოვნებას არ არიან ნაზიარებნი"-იტყვის ის. სადავო არაფერია. მაგრამ ნუ დაგვავიწყდება, რომ პროგრამებს ქმნიან ადამიანები. ხელოვნებაც ადამიანების შემოქმედებაა...
მაგალითის სახით ჩვენ შევქმნით **Web**-საიტს-სტატიების და ფაილების (პროგრამების, მულტიმედიისა და ა.შ.) არქივს. სტატიების და ფაილების ნუსხას ჩვენ განვათავსებთ **MySQL** ბაზებში. ბაზებიდან მათ ამოკრეფას განახორციელებენ **PHP** სერვერული გვერდები, რომელთა შექმნაშიც **DreamweaverMX**-ი დაგვეხმარება.

ნაწილი 1

ჩვენი პირველი საიტი

თავი1. თანამედროვე ინტერნეტ-ტექნოლოგიები
თავი2. **HTML-Web**-გვერდების საწერი ენა
თავი3. მუშაობა **Macromedia DreamweaverMX**-თან
თავი4. **Web**-საიტთან მუშაობა **DreamweaverMX**-ში

პირველ ნაწილში ჩვენ დავკავდებით იმით, რომ შევქმნით ჩვენს უპირველეს **Web**-საიტს. დიახ! ჯერ რათქმა უნდა იქნება ინტერნეტ-ტექნოლოგიების შესავალი კურსი, შემდეგ ჩვენ გავარკვევთ, თუ როგორ იქმნებიან **Web**-გვერდები, მერე შევისწავლით საკუთრივ პროგრამულ პაკეტს-**Macromedia DreamweaverMX** -იგი ძალიან დაგვეხმარება მუშაობაში.
ასეთი ფუნდამენტალური ცოდნით აღჭურვის შემდეგ, ჩვენ შევუდგებით საიტზე მუშაობას.
ჩვენი პირველი საიტი ძალიან მარტივი იქნება. ჩვენთვის ხომ ამ ეტაპზე მთავარია სწავლა. ასე არ არის?

თავი 1

თანამედროვე ინტერნეტ-ტექნოლოგიები

ყოველ პრაქტიკას, წინ თეორია უნდა უსწრებდეს. თეორიის გარეშე შორს ვერ წავალთ. სანამ რაიმეს კეთებას დავიწყებდეთ, უნდა გავარკვიოთ რისთვის, როგორ და რატომ კეთდება ეს "რაიმე" მაინცდამაინც ასე. სხვანაირად არაფელი არ გამოგვივა.
ასეა ინტერნეტ-ტექნოლოგიებთან დაკავშირებითაც. ცოდნის გარკვეული მარაგის გარეშე ჩვენ ვერაფრის შექმნას ვერ შევძლებთ (ყოველ შემთხვევაში არაფელი ღირებული არ გამოგვივა). მაშ ასე, გამოვრთოთ კომპიუტერი-დაისვენოს!-და დავიწყოთ კითხვა.

ინტერნეტის მუშაობის ძირითადი პრინციპები.

ჯერ ჩვენ ვისაუბრებთ იმის შესახებ, თუ რა არის ინტერნეტი და როგორ მუშაობს იგი-განვიხილავთ გარკვეულ ზოგად საკითხებს.

რა არის ინტერნეტი. ინტერნეტის სერვისები

პირველივე კითხვა, რომელზედაც ჩვენ პასუხი უნდა მივიღოთ არის: რას წარმოადგენს საკუთრივ ინტერნეტი, რისი მოცემა შეუძლია მას ჩვენთვის. სწორედ მისი მუშაობის ძირითადი პრინციპების განხილვა იქნება ჩვენი უახლოესი ამოცანა.

მაშ ასე, **ინტერნეტი**-ეს მსოფლიო კომპიუტერული ქსელია. სხვათაშორის მას ხშირად ასეც უწოდებენ: მსოფლიო ქსელი, ანუ უფრო მარტივადაც-ქსელების ქსელი. მთელ დედამიწას მოდებული სპილენძის მავთულების, ოპტიკურ-ბოჭკოვანი ხაზების და რადიო არხების ობობა, რომელიც ერთმანეთთან აკავშირებს მრავალრიცხოვან კომპიუტერებს-აი, თუ რა არის ინტერნეტი. ბუნებრივია, რომ ყოველივე ეს ექვემდებარება ზოგად სტანდარტებს (მათ შესახებ ვისაუბრებთ ოდნავ მოგვიანებით). წინააღმდეგ შემთხვევაში აღნიშნული სუპერქსელი უბრალოდ ვერ იმუშავებდა.

თუ უფრო ზუსტები ვიქნებით, ინტერნეტი-ეს განუყოფელი ქსელი კი არ არის, არამედ იგი უფრო მცირე ქსელების ერთიანობას წარმოადგენს, რომლებიც ერთმანეთთან საერთო არხებითა და სტანდარტებით არიან დაკავშირებულნი. ასეთი ქსელები უამრავია: უზარმაზარი ტერიტორიული ქსელები, რომლებიც მთელ ოლქებს, შტატებს და სახელმწიფოებს მოიცავენ, და უწყებრივი ქსელები, რომლებიც აერთიანებენ მონათესავე ორგანიზაციებს, აგრეთვე ცალკეული ორგანიზაციების ლოკალური კომპიუტერული ქსელები და ეგრეთ წოდებული კამპუსური ქსელები-ქსელები, რომლებიც აერთიანებენ ქალაქების ერთი ან რამოდენიმე ერთმანეთთან ახლოს მდებარე რაიონების ქსელებს.

მათ შორის არსებული, კავშირგაბმულობის მაღალსიჩქარიანი არხების საშუალებით, ისინი ქმნიან ერთიან და მთლიან წარმონაქმნს, რომელსაც **ინტერნეტი** ეწოდება.

ინტერნეტ-ქსელს ერთი შესანიშნავი თავისებურება გააჩნია-იგი ძალზე მდგრადია შეფერხებების მიმართ. ანუ, თუ სადმე გაწყდება ქსელის ორი მონაკვეთის (როგორც ქსელის პროფესიონალები ეძახიან-**სეგმენტის**) დამაკავშირებელი ხაზი, ჩვენ ამას ვერც კი შევნიშნავთ. ვერ შევნიშნავთ იმის გამო, რომ ჩვენს მიერ მოთხოვნილი მონაცემები, ამ შემთხვევაში გამოემართებიან სხვა არხების გავლით. სპეციალისტები ამბობენ, რომ ინტერნეტი დეცენტრალიზებულია-მას მონაცემების გადაცემის და მართვის ერთიანი ცენტრი არ გააჩნია, ამიტომ ავარიის შემთხვევაში ხდება

ქსელის კონფიგურაციის ავტომატური შეცვლა და იგი აგრძელებს ნორმალურ ფუნქციონირებას.

ინტერნეტის კიდევ ერთი საინტერესო თავისებურება მისი გლობალურობა და მსოფლიოს მასშტაბით განვრცობადობაა. კომპიუტერიდან აუდგომლად, ჩვენ შეგვიძლია მოგზაურობა მთელი დედამიწის ზურგზე, აშშ-ში, ავსტრალიაში, გერმანიაში, ზიმბაბვეში და ანტარქტიკაშიც კი. ამისათვის საჭიროა მხოლოდ სწორი მისამართის აკრეფა.

მაშ ასე, ჩვენ უკვე გავარკვიეთ თუ რა არის ინტერნეტი. ახლა გადავინაცვლოთ წარსულში და ვნახოთ თუ როგორ იწყებოდა ყოველივე.

ინტერნეტს საკმაოდ ხანგრძლივი და საინტერესო ისტორია გააჩნია. იგი მე-20 საუკუნის 70-იანი წლების პირველ ნახევარში გაჩნდა, როდესაც ამერიკის თავდაცვის სამინისტრომ დააფინანსა მტყუნებებისადმი მდგრადი კომპიუტერული ქსელის შექმნასთან დაკავშირებული პროექტი. ბუნებრივია, რომ ეს ქსელი იქმნებოდა თავდაცვის საჭიროებებისათვის და სახელწოდებაც შესაბამისი ჰქონდა-**ARPANET**. უფრო მოგვიანებით, 80-იანი წლების დასაწყისში ეს ქსელი გადავიდა მეცნიერების ხელში, ხოლო სამხედროები შეუდგნენ სხვა ქსელის შექმნას, რომელსაც დღესაც იყენებენ. დროის იმავე პერიოდში **ARPANET**-ს გადაერქვა სახელი და მას **INTERNETI** ეწოდა.

დასაწყისში, ჯერ-კიდევ **ARPANET**-ის პერიოდში, ეს ქსელი გამოიყენებოდა ელექტრონული ფოსტის გადასაგზავნად და ფაილების ურთიერთ გაცვლისათვის. **Web**-გვერდები, რომელთა გამოც ჩვენ, ქსელში მოგზაურობა გვიწევს, გაჩნდნენ მხოლოდ 80-იანი წლების ბოლოს. სწორედ მაშინ გახდა ინტერნეტი ფართო მასებისათვის ხელმისაწვდომი, და შეწყდა მისი მხოლოდ მეცნიერების მიერ გამოყენება. იგი იქცა ყველასათვის ხელმისაწვდომ, ნამდვილად სახალხო ქსელად.

საქართველოში, ისევე, როგორც ყოფილ საბჭოთა კავშირში ინტერნეტი გაჩნდა 1990 წლების დასაწყისში, მაგრამ კომპიუტერის სპეციალისტების ფართო მასებში პოპულარობა მხოლოდ 90-იანი წლების შუახანებში მოიპოვა.

დღეისათვის მსოფლიოში ასობით მილიონოი ადამიანი სარგებლობს ქსელის მომსახურებებით ერთმანეთთან ურთიერთობისა და ინფორმაციის მოძიებისათვის.

რადგანაც ჩვენ ვახსენეთ ინტერნეტის მიერ შემოთავაზებული მომსახურებები, ანუ როგორც პროფესიონალები ამბობენ ***ინტერნეტ-სერვისები***, მოდით შევეცადოთ გავიგოთ მათ შესახებ უფრო მეტი. ბოლოს და ბოლოს ჩვენ უნდა ვისარგებლოთ ამ სერვისებით.

მაშ ასე, ყველაზე ძველი და ყველაზე პოპულარული ინტერნეტ-სერვისი-ეს არის ***ელექტრონული ფოსტა*** (**e-mail**). ყოველდღიურად მსოფლიოში მილიონობით ელექტრონული წერილის გაგზავნა და მიღება ხდება, და მომავალში ეს რიცხვი მხოლო გაიზრდება. მართლაც, ელექტრონული ფოსტა ხელმისაწვდომია, მოხერხებულია, სწრაფია და უფასოა "ქაღალდის" ფოსტისაგან განსხვავებით, რომელსაც ინტერნეტის მომხმარებლებმა მისდამი სიძულვილის ნიშნად "ნიჟარისებრი" უწოდეს (ინგლისურად-**snail mail**). რა თქმა უნდა ამ ხელმისაწვდომობას, მოხერხებულობას, სისწრაფეს და უფასოობას გარკვეული ისეთი უხერხულობებიც ახლავს, როგორიც არის "სპამი"-არასანქცირებული სარეკლამო გზავნილები, მაგრამ მათთან ბრძოლა შესაძლებელია.

ინტერნეტის კიდევ ერთი სერვისი, თითქმის ისეთივე ძველი, როგორიც ფოსტაა-ეს არის **ფაილების გადაგზავნა.** ინტერნეტის მომხმარებლები მას **FTP**-ს **(File Transfer Protocol**, ფაილების გადაცემის პროტოკოლი; თუ რატომ ეძახიან ასე-ჩვენ მოგვიანებით გავიგებთ) ეძახიან.

დღეისათვის **FTP** არ არის ისეთივე პოპულარული, როგორც ინტერნეტის გარიჟრაჟზე იყო, მაგრამ მაინც ხშირად გამოიყენება ფაილების გადასაგზავნად.

WWW. ინტერნეტის მესამე სერვისი, რომელსაც ჩვენ შემდგომში დაწვრილებით განვიხილავთ-არის მსოფლიო ობობა, ანუ **WWW** (**World Wide Web**-ყველგან განფენილი ობობა), ან მარტივად-**Web**. ეს არის სწორედ ის **Web**-გვერდები და **Web**-საიტები, რომლებსაც ჩვენ ვათვალიერებთ Web-დამთვალიერებლების საშუალებით. ეს მართლაც ყველაზე შთამბეჭდავი სერვისია, რომელმაც თავის დროზე, მხოლოდ მეცნიერებისათვის ხელმისაწვდომი ქსელის, ყველასათვის ხელმისაწვდომ ქსელად გარდაქმნა განაპირობა და შეძინა მას ისეთი პოპულარობა, როგორიც დღეისათვის გააჩნია. ძირითადად, სწორედ **WWW**-ს შესახებ გვექნება საუბარი ამ წიგნში.

კიდევ არსებობს რამოდენიმე, ნაკლებად პოპულარული ინტერნეტ-სერვისები. მათ შორის :

ახალი ამბები. ეს სერვისი რაღაცით ელექტრონულ ფოსტას წააგავს: მომხმარებელი წერს წერილს ეგრეთ წოდებული ახალი ამბების ჯგუფში-განცხადებების ელექტრონული დაფის მაგვარ სერვისში. სხვა მომხმარებლები კითხულობენ ამ წერილს და წერენ პასუხებს, რომლებსაც ამავე "დაფაზე" ათავსებენ. საწყისი წერილისა და მასზე პასუხების ერთობლივობა ქმნიან განხილვას (ინგლისურად-**thread**), რომელშიც მონაწილეობის მიღება შეუძლია ნებისმიერ მომხმარებელს, რომელიც იქნება ახალი ამბების მოცემული ჯგუფის ხელისმომწერი. ოდესღაც ახალი ამბების სერვისი საკმაოდ პოპულარული იყო, მაგრამ ახლა თანდათანობით თმობს თავის პოზიციებს.

ნაკადობრივი მაუწყებლობა. ეს ინტერნეტის საშუალებით ტელე და რადიო მაუწყებლობის სახეობაა, რომელიც სულ ცოტა ხნის წინათ გაჩნდა, მაგრამ ძალიან სწრაფად ხდება პოპულარული. ერთად-ერთი სერიოზული ნაკლოვანება, რომელიც მას გააჩნდა-სწრაფი არხების არსებობის აუცილებლობა-უკანასკნელ პერიოდში აღარ არის აქტუალური.

ინტერნეტ-პეიჯერები. ეს სერვისი აგრეთვე წააგავს ელექტრონულ ფოსტას: მომხმარებლები ერთმანეთს უგზავნიან მოკლე "შეტყობინებებს" ჩვეულებრივი პეიჯერების ანალოგიურად. ინტერნეტ-პეიჯერები, როგორც წესი უფრო სწრაფად მუშაობენ ვიდრე ჩვეულებრივი ელექტრონული ფოსტა და დროდადრო ქმნიან უშუალო ურთიერთობის ილუზიას. მაგალითის სახით შეიძლება გავიხსენოთ უპოპულარესი **ICQ** და მისი ნაკლებად ცნობილი "კოლეგები" : **Miranda, Odigo** და სხვა.

ჩათები (ინგლისურიდან **chat**-„ჭორაობა"). ეს თავისებური "საუბარია" ინტერნეტის საშუალებით. იგი კიდევ უფრო მეტად წააგავს უშუალო ურთიერთობას. მომხმარებელი კლავიატურაზე კრიბავს ტექსტს, რომელიც თვალის დახამხამებაში გადაეგზავნება მის თანამოსაუბრეს ან თანამოსაუბრეებს. პოპულარობის მიხედვით ჩათები აღემატებიან ინტერნეტ-პეიჯერებს და უტოლდებიან **WWW**-ს.

მაშ ასე, ამით დავასრულოთ. ინტერნეტის ნაკლებად ცნობილ და ვიწრო სპეციალიზაციის მქონე სერვისებს აღარ განვიხილავთ. ბოლოს და ბოლოს, მათ შესახებ ინფორმაცია (ისევე, როგორც ბევრ

სხვა რამეზე) შესაძლებელია მოძიებულ იქნას იმავე ინტერნეტში. ამ წიგნის თემა კი სულ სხვა რამ არის.

კლიენტები და სერვერები

განვაგრძოთ ჩვენი ჩაშვება ინტერნეტის ელექტრონულ ჯურღმულებში. ამჯერად საუბარი გვექნება პროგრამების ორ ნაირსახეობაზე, რომელთა საშუალებითაც ხორციელდება ინტერნეტ-მომსახურებები.

მართლაც, როგორ ვსარგებლობთ ჩვენ ყველა იმ სიმდიდრით, რასაც იძლევა მსოფლიო ქსელი? განსაკუთრებული პროგრამების საშუალებით! ესენია **Web**-დამთვალიერებელი, ელექტრონული ფოსტის კლიენტები, ინტერნეტ-ტელევიზიის დათვალიერების პროგრამა და ინტერნეტ-რადიო, **ICQ** და სხადასხვა ''წამკითხველი'' პროგრამები. ყველა მათგანი ჩვენთვის კარგად არის ნაცნობი. მაგრამ ჩვენთვის ინტერნეტ-მომსახურებების მოსაწოდებლად გამოსაყენებელი პროგრამების რაოდენობა გაცილებით მეტია. ძალიან ბევრი მათგანი თუ შეიძლება ასე ითქვას ''უხილავია''. ანუ ჩვენ არ ვურთიერთობთ მათთან პირდაპირ. ზოგადად არსებობს ინტერნეტ-პროგრამების ორი ერთმანეთისაგან პრინციპულად განსხვავებული კატეგორია. ჩვენ ახლა გადავალთ მათ განხილვაზე.

პროგრამები, რომლებიც პირველ კატეგორიას განეკუთვნებიან, ურთიერთქმედებენ უშუალოდ ინტერნეტის მომხმარებლებთან და ეხმარებიან მათ სხვადასხვა ინფორმაციის მიღებაში: ელექტრონული ფოსტა, **Web**-გვერდები, ინტერნეტ-პეიჯერების შეტყობინებები, ჩათები და სხვა. ესენია **Web**-დამთვალიერებლები, ელექტრონული ფოსტის, ჩათების, ინტერნეტ-პეიჯერების **კლიენტები**, ანუ ყველა ის პროგრამები, რომლებთანაც ჩვენ ვურთიერთობთ ჩვენს კომპიუტერებზე მუშაობისას. ასეთ პროგრამებს უწოდებენ **პროგრამა-კლიენტებს** (ხოლო კომპიუტერებს, რომლებზეც ისინია განთავსებული-მათ შორის კერძოდ ჩვენს კომპიუტერს!- **კლიენტურ კომპიუტერებს**).

ინფორმაცია, რომელთანაც ჩვენ ვმუშაობთ პროგრამა-კლიენტების საშუალებით-**Web**-საიტები, წერილები, ხმოვანი და ვიდეო-ფაილები-ინახება სხვა, ეგრეთ წოდებულ **სერვერულ** კომპიუტერებზე. აღნიშნული ინფორმაციის კლიენტური პროგრამებისათვის, ანუ იგივე ჩვენთვის, გადაცემაზე პასუხისმგებლები არიან პროგრამები, რომლებიც მეორე კატეგორიას განეკუთვნებიან-ეგრეთ წოდებული **სერვერული პროგრამები**

(**სერვერები**). ყოველი ინტერნეტ-სერვისისათვის არსებობს სერვერული პროგრამების (***სერვერების***), (და როგორც უკვე ვიცით, **კლიენტების**) საკუთარი კლასი: ელექტრონული ფოსტის, ჩათის, ინტერნეტ-პეიჯერის, ნაკადური მაუწყებლობის და ა.შ. **Web**-სერვერები.

შენიშვნა: *ძალიან ხშირად ცნება "სერვერი" ვრცელდება სერვერულ კომპიუტერზეც და თვით პროგრამა-სერვერზეც. ზოგადად ეს არასწორია, ვინაიდან ერთ სერვერულ კომპიუტერზე შესაძლებელია დაყენებული იყოს რამოდენიმე პროგრამა-სერვერი, მაგრამ პრაქტიკაში ასეა შესული.*

მოდით, ახლა უფრო დაწვრილებით ვისაუბროთ იმაზე, თუ როგორ ურთიერთქმედებენ კლიენტები სერვერებთან. თუმცა მონაცემების მიღებისა და გაგზავნის პროცესებს ჩვენ ცალკე განვიხილავთ.
ზოგადად კლიენტის მიერ სერვერისაგან ინფორმაციის მიღების პროცესი ხუთი ნაბიჯისაგან შედგება:

1.მომხმარებელი, პროგრამა-კლიენტის საშუალებით მოითხოვს რაიმე ინფორმაციას სერვერისაგან;

2.კლიენტი ამყარებს კავშირს სერვერთან და უგზავნის მას განსაკუთრებულ ინფორმაციულ ბლოკს, რომელსაც **კლიენტის მოთხოვნა** ეწოდება. ამ მოთხოვნის სტრუქტურა მკაცრად არის სტანდარტიზებული, რათა სერვერისათვის ის გასაგები იყოს;

3.სერვერი იღებს მოთხოვნას და ახდენს მის გაშიფვრას;

4.სერვერი პოულობს კლიენტისათვის საჭირო ფაილს ან ფაილში ჩაწერილ მონაცემთა ფრაგმენტს და უგზავნის მას კლიენტს უკვე სხვა ინფორმაციული ბლოკის-**სერვერის პასუხის** სახით.

5.კლიენტი იღებს სერვერისაგან პასუხს, ახდენს მის დეშიფრაციას და მიღებულ ინფორმაციას აწვდის მომხმარებელს. სერვერისაგან პასუხის მიღების შემდეგ კლიენტი წყვეტს მასთან კავშირს.

კლიენტის მიერ სერვერისათვის მონაცემების გაგზავნის პროცესი დაახლოებით იგივე ხუთი ნაბიჯისაგან შედგება:

1.მომხმარებელი რაიმე ხერხით აწვდის პროგრამა-კლიენტს გასაგზავნ ინფორმაციას;

2.კლიენტი ამყარებს კავშირს სერვერთან და კლიენტური მოთხოვნის შემადგენლობაში აწვდის მას გასაგზავნ ინფორმაციას. ამასთან, გასაგზავნი ინფორმაცია, როგორც წესი, იშიფრება განსაკუთრებული წესით;

3.სერვერი იღებს მოთხოვნას, ახდენს მის გაშიფვრას და ამოალაგებს გამოგზავნილ ინფორმაციას;

4.სერვერი კლიენტის მიერ გამოგზავნილ ინფორმაციას წერს ფაილში. ამის შემდეგ, წარმატებული ჩაწერის შემთხვევაში, იგი უგზავნის კლიენტს პასუხის შემადგენლობაში, ეგრეთ წოდებულ დასტურს-განსაკუთრებულ ინფორმაციულ ბლოკს, რომელიც ატყობინებს კლიენტს იმის შესახებ, რომ ყველაფერმა ნორმალურად ჩაიარა. წარუმატებელი ჩაწერის შემთხვევაში კლიენტს ეგზავნება შეტყობინება შეცდომის შესახე.

5.კლიენტი იღებს პასუხს სერვერისაგან, ახდენს მის დეშიფრაციას და აცნობებს მომხმარებელს მონაცემების წარმატებული ან წარუმატებელე გაგზავნის შესახებ, ან სიტუაციის გამოსწორების მიზნით, ეცდება რაიმე ქმედების განხორციელებას დამოუკიდებლად. სერვერისაგან პასუხის მიღების შემდეგ კლიენტი წყვეტს კავშირს მასთან.

კლიენტისა და სერვერის ”ურთიერთობის” მთლიან პროცესს, კლიენტის მიერ მოთხოვნის გაგზავნიდან დაწყებული, სერვერისაგან პასუხის მიღებით დამთავრებული, **სეანსი** ეწოდება.
ადრე ნათქვამი იყო, რომ კლიენტის მიერ სერვერისათვის მოთხოვნის გაგზავნამდე, ე.ი სეანსის დაწყებამდე, კლიენტი უნდა დაუკავშირდეს სერვერს. სხვა სიტყვებით რომ ვთქვათ, დაამყაროს კავშირი სერვერთან. ეს კავშირი გრძელდება ზუსტად იმდენ ხანს, რამდენიც საჭიროა სეანსის დასრულებისათვის, ამიტომ მას **სეანსური** ან **დროებითი** კავშირი ეწოდება.
ყოველი კავშირი მოითხოვს კომპიუტერულ რესურსებს. სერვერმა უნდა შეინახოს თავის მეხსიერებაში ცნობები იმ კლიენტის შესახებ,

რომელიც მას დაუკავშირდა, ხოლო თუ ასეთი კავშირები ბევრი იყო დამყარებული (ხშირად ასეც ხდება), მაშინ ძალზე ბევრი მეხსიერების მოხმარება ხდება. ამიტომ სეანსურ კავშირებს დიდი უპირატესობა გააჩნიათ-ისინი მცირე დროის განმავლობაში გრძელდებიან, ზუსტად იმდენი ხანი, რამდენიც საჭიროა სერვერული პასუხის გასაგზავნად, რის შემდეგაც წყდებიან, და ხდება მათთვის განკუთვნილი კომპიუტერული მეხსიერების ავტომატურად გამოთავისუფლება.
მაგრამ სეანსურ კავშირებს უდიდესი ნაკლოვანებაც გააჩნიათ-მათი საშუალებით მონაცემების გაცვლის სეანსის დაწყება შეუძლიათ მხოლოდ კლიენტებს. ხოლო, ჩათის და ინტერნეტ-პეიჯერის შემთხვევაში, მაგალითად, სერვერს თვითონ უწევს სეანსის დაწყება, რათა გაუგზავნოს ახალი შეტყობინებები "დასვენების რეჟიმში" მყოფ კლიენტებს. ასეთ შემთხვევებში გამოიყენებიან **მუდმივი** ან **მრავალსეანსური** ჩართვები, მუშაობის განსხვავებული სცენარებით.

1.კლიენტი ამყარებს მუდმივ კავშირს სერვერთან;
2.კლიენტი და სერვერი ახორციელებენ მონაცემების ურთიერთგაცვლას. ამასთან ურთიერთგაცვლის სეანსი შესაძლოა დაწყებულ იქნას როგორც კლიენტის მიერ, ასევე სერვერის მიერ;
3.კლიენტი წყვეტს კავშირს სერვერთან. მონაცემების გაცვლა დასრულებულია.

აქ ყურადღება უნდა მიექცეს იმას, რომ კავშირის დამყარება ხდება მხოლოდ კლიენტის მიერ. სერვერი კი მუშაობს უკვე დამყარებული კავშირის მეშვეობით.

როგორც წესი, სერვერული კომპიუტერები-ნამდვილ მონსტრებს წარმოადგენენ, რომლებიც შეიცავენ რამოდენიმე პროცესორს, შთამბეჭდავი მოცულობის უზარმაზარ დისკურ მასივებს, ინტერნეტთან მიერთების მძლავრ საშუალებებს და სპეციალურ პროგრამულ უზრუნველყოფას, რომელსაც საკმარისი "ძალა გააჩნია", რათა მართოს მთელი ეს სიძლიერე. მათში ყველაფერი მიმართულია იქითკენ, რომ მოხდეს რაც შეიძლება მეტი კლიენტის მომსახურება, დამუშავდეს რაც შეიძლება მეტი მოთხოვნა, რომ მომხმარებლებმა მოთხოვნილი ინფორმაცია მიიღონ განკუთვნილ დროში.
მაგრამ ხშირ შემთხვევებში, როდესაც კლიენტებისა და მოთხოვნების რაოდენობა ძალიან ბევრი აღმოჩნდება და სერვერული

კომპიუტერის რესურსები არასაკმარისი იქნება მათ დასაკმაყოფილებლად, იწყება პრობლემები.
პრობლემებმა შეიძლება თავი იჩინონ იმაში, რომ სერვერი უბრალოდ უარს განაცხადებს "ზედმეტი" კლიენტების მომსახურებაზე, და შესთავაზებს მათ გარკვეული დროის განმავლობაში დალოდებას, იმ დრომდე სანამ დატვირთვა ცოტათი მაინც არ შემცირდრბა; ხშირ შემთხვევაში იმაშიც, რომ მძლავრი სერვერი უბრალოდ "ჩამოეკიდება". ასეთი რამეც ხდება და არც თუ ისე იშვიათად.
მოდით ნუ ვისაუბრებთ სამწუხარო ფაქტებზე. არ ღირს ინტერნეტ-ტექნოლოგიების მიმზიდველი სამყაროს გაცნობის დაწყება ისეთი არასასიამოვნო მოვლენებით, როგორებიც არიან სისტემური მტყუნებები. რაც უფრო მცირეა მათი რაოდენობა და რაც უფრო ნაკლებად ხდება მათი დაფიქსირება, მით უკეთესია ყველა ჩვენთაგანისათვის.
მაშ ასე, ჩვენ წეღან გავეცანით განსაკუთრებულ **არქიტექტურას** (კომპიუტერული ქსელების აგების პრინციპს), რომელსაც **ორკვანძოვანი** ან "**კლიენტ-სერვერული**" არქიტექტურა ეწოდება. ეს არქიტექტურა გამოიყენება პრაქტიკულად ყველა თანამედროვე ინტერნეტ-სერვისების რეალიზაციისას და ჯერ-ჯერობით ამართლებს თავის დანიშნულებას.

შენიშვნა: *ზოგიერთი ინტერნეტ-სერვისები, კერძოდ კი ეგრეთწოდებული ფაილების ურთიერთგაცვლის ქსელები (Napster, Gnutella, Kazaa და სხვა), იყენებენ პრინციპიალურად სხვა, ერთკვანძიან არქიტექტურას. აქ ინტერნეტთან მიერთებული და მისი სერვისების განმახორციელებელი ყველა კომპიუტერი ფაქტიურად ტოლფასოვანია; ნებისმიერი მათგანი შეიძლება გამოვიდეს როგორც კლიენტური, (მოითხოვოს ინფორმაცია სხვა კომპიუტერებისაგან), ასევე როგორც სერვერული (გასცეს მასში შენახული ინფორმაცია სხვა კომპიუტერებს) კომპიუტერის როლში. თავისთავად ცხადია, რომ აქ გამოიყენება განსაკუთრებული პროგრამული უზრუნველყოფა, რომელსაც როგორც კლიენტის, ასევე სერვერის ფუნქციები გააჩნია.*

პროტოკოლები

ადამიანები, რათა გაუგონ ერთმანეთს, ერთსა და იმავე ენაზე უნდა საუბრობდნენ. ზუსტად ასევეა ქსელში ჩართული კომპიუტერების

შემთხვევაშიც, არა აქვს მნიშვნელობა რომელ ქსელზეა საუბარი- მსოფლიო თუ ლოკალურზე.
ასეთ ქსელებში მონაცემების ურთიერთ გაცვლა უნდა ხდებოდეს ერთიანი სტანდარტების მიხედვით, წინააღმდეგ შემთხვევაში ყველაფერი აირევა ერთმანეთში.
სტანდარტს, რომლის მიხედვითაც ხდება ქსელში გასაგზავნი მონაცემების კოდირება **პროტოკოლი ეწოდება**.
ინტერნეტში მონაცემთა გაცვლისათვის გამოიყენება რამოდენიმე პროტოკოლი, რომელთაც ჩვენ აქ მოკლედ მიმოვიხილავთ.
ინტერნეტის, ყველაზე მთავარი პროტოკოლი, თუ შეიძლება ასე ეწოდოს მას-არის **TCP/IP** (**Transfer Control Protokol/Internet Protocol**, გადაცემის მართვის პროტოკოლი/ინტერნეტი- პროტოკოლი). ეს არის ეგრეთ წოდებული დაბალი დონის პროტოკოლი, რომელიც განსაზღვრავს გადასაცემი მონაცემების, მხოლოდ ყველაზე ძირითად პარამეტრებს: მონაცემების ცალკეული პორციების (პაკეტების) სიგრძეს, კოდირების ხერხს, გამგზავნისა და მიმღების მისამართებს, აგრეთვე შეცდომებისაგან დაცვის მეთოდებს. შეიძლება ითქვას, რომ **TCP/IP**, კონკრეტულად ინტერნეტის არხებში მონაცემების გადაცემით არის დაკავებული, და მას არ აინტერესებს გადასაცემი ინფორმაციის შინაარსობრივი მხარე.
TCP/IP პროტოკოლს ეფუძნებიან სხვა უფრო მაღალი დონის პროტოკოლები. აღნიშნული პროტოკოლები აღწერენ კლიენტური მოთხოვნების და სერვერული პასუხების ფორმატებს: მონაცემების მოთხოვნის ან გადაცემის შემთხვევაში, კლიენტის მიერ სერვერისათვის გაგზავნილ განსაკუთრებულ ბრძანებებს და გადასაცემი ინფორმაციის წარმოდგენის ხერხებს.
გადასაცემი ინფორმაციის, შეშფოთებებისადმი განსაკუთრებული მდგრადობის უზრუნველმყოფი კოდირებით, მისი პაკეტებად დაყოფით და საკუთრივ გადაცემით, დაკავებულია "შავი მუშა" ჩვენთვის უკვე ნაცნობი **TCP/IP**.

შენიშვნა: *თუ უფრო მკაცრად განვიხილავთ, არსებობენ კიდევ ფიზიკური დონის პროტოკოლები, რომლებიც მდებარეობენ TCP/IP-ზე "დაბლა." ისინი განსაზღვრავენ სიგნალების, კაბელების, განმხოლოებლების და სხვათა ელექტრულ პარამეტრებს.*

ინტერნეტის ყოველი სერვისი იყენებს თავის საკუთარ მაღალდონიან პროტოკოლს (ხშირად რამდენიმესაც, რომლებიც სხვადასხვა

ამოცანებისთვის არიან განკუთვნილი და ერთმანეთის კონკურენტი ორგანიზაციების მიერ იწარმოებიან). მოდით განვიხილოთ პროტოკოლები, რომლებსაც ჩვენ შესაძლოა მომავალში წავაწყდეთ.
ჩვენ რა თქმა უნდა დავიწყებთ **WWW**-თი. მონაცემების გადაცემისათვის მსოფლიო ობობა იყენებს **HTTP** (**HyperText Transfer Protokol**, ჰიპერტექსტის გადაცემის პროტოკოლი) პროტოკოლს. ის განსაზღვრავს კლიენტის (**Web**-დამთვალიერებლის) მიერ **Web**-სერვერისათვის გადასაგზავნი მონაცემების მოთხოვნისა და მართვის ბრძანებების ანაკრეფს და ორივე მიმართულებით გადასაგზავნი მონაცემების წარმოდგენის ხერხებს. უნდა აღინიშნოს, რომ ეს ინტერნეტის ყველაზე ცნობილი და ფართოდ გავრცელებული პროტოკოლია; ყველა მეტ-ნაკლებად წიგნიერი ინტერნეტის მომხმარებლისათვის ნაცნობია ეს აბრევიატურა.
ფაილების გადაგზავნის სერვისი **FTP**-იყენებს პროტოკოლს, რომელსაც ასევე **FTP-პროტოკოლი** ეწოდება. იგი ასევე განსაზღვრავს სერვერზე ფაილების მართვის (ჩატვირთვა, სერვერზე განთავსება, კოპირება, გადაადგილება, წაშლა და ა.შ.) ბრძანებათა ანაკრებს და კავშირგაბმულობის არხების საშუალებით გადაცემისათვის, ფაილების კოდირების ხერხებს. ამ თვალსაზრისით **HTTP** და **FTP** პროტოკოლები ძალიან წააგავენ ერთმანეთს.
ელექტრონული ფოსტა კი, ერთდროულად ორ პროტოკოლს იყენებს. პირველი პროტოკოლი **SMTP** (**Simple Mail Transfer Protocol**, ფოსტის გადაგზავნის მარტივი პროტოკოლი)-გამოიყენება კლიენტის მიერ, სერვერისათვის ფოსტის გადასაგზავნად. სერვერისგან ფოსტის მისაღებად კი კლიენტი მასთან ურთიერთქმედებს **POP3** (**Post-Office Protocol**, ფოსტის პროტოკოლის) პროტოკოლის საშუალებით.

შენიშვნა: *საჭიროა აგრეთვე გავიხსენოთ IMAP (Internet Message Access Protocol, ინტერნეტ ფოსტაში შეღწევის პროტოკოლი), რომელიც გამოიყენება ასევე კლიენტის მიერ სერვერისაგან ფოსტის მისაღებად. უფრო ძველ POP3-თან შედარებით იგი მეტ შესაძლებლობებს იძლევა, მაგრამ ნაკლებად არის გავრცელებული.*

ახალი ამბების სერვისი მუშაობისათვის იყენებს **NNTP** (**Network News Transfer Protokol**, ქსელური სიახლეების გადაცემის პროტოკოლი). დანარჩენი სერვისებიც თავიანთ პროტოკოლებს იყენებენ. მაგრამ ჩვენ მათზე არ შევჩერდებით.

პროტოკოლის ცნებასთან ძალიან მჭიდროდ არის დაკავშირებული **TCP/IP პორტის** ცნება. **პორტი**-ეს თავისებური წარმოსახვითი არხია, რომლის საშუალებითაც ხდება მონაცემების გადაცემა, მაღალი დონის პროტოკოლებიდან ერთ-ერთის გამოყენებით. შეიძლება ითქვას, რომ ინტერნეტის მონაცემთა გადაცემის ნებისმიერი არხი დაყოფილია 65 535 დანომრილ პატარ-პატარა არხებად-სწორედ ამდენივე პორტის არსებობას ითვალისწინებს **TCP/IP**-პროტოკოლი. მაღალი დონის ყოველი არსებული პროტოკოლი მონაცემთა გადაცემისათვის იყენებს თავის საკუთარ პორტს (ეგრეთ წოდებული წინასწარ დასახელებული პორტი).
ცხრილში 1.1. ჩამოთვლილია ზოგიერთი პროტოკოლები და მათ მიერ "დაკავებული" პორტები.

ცხრილი 1.1. ზოგიერთი მაღალი დონის პროტოკოლების მიერ მონაცემების გადაცემისათვის გაჩუმებით გამოყენებული **TCP/IP** პორტები

პროტოკოლი	**გამოყენებული პორტი**
HTTP	80
FTP	21
SMTP	25
POP3	110

მაგრამ რატომ ასეთი უცნაური სახელწოდება-**გაჩუმებითი** (წინასწრ განსაზღვრული, **default**) პორტი. მოდით გავერკვეთ უფრო დეტალურად.
საქმე იმაშია, რომ ყველა მეტ-ნაკლებად სერიოზული სერვერი იძლევა პროტოკოლის მიერ გამოყენებული პორტის სხვა პორტით შეცვლის შესაძლებლობას. მაგალითად **Web**-სერვერი შეიძლება დაყენებული იქნას ისე, რომ კლიენტებთან "ურთიერთობისათვის" გამოიყენოს არა მე-80 პორტი, არამედ ვთქვათ მე-8000 პორტი. (დრო და დრო შეიძლება შეგვხდეს ამგვარად დაყენებული სერვერები). ეს სრულებით შესაძლებელია, მაგრამ რეკომენდებული არ არის, რათა შეცდომაში არ იქნან შეყვანილი მომხმარებლები. ამიტომ ასეთი მიდგომა ძალზე იშვიათად გამოიყენება, მხოლოდ უკიდურეს შემთხვევებში.
თუ სერვერს სპეციალურად არ დავაყენებთ სხვა პორტის გამოყენებაზე, მაშინ ის გამოიყენებს სწორედ წინასწარ

განსაზღვრულ (გაჩუმებით) პორტს. იმთავითვე ასეა გათვალისწინებული მის პარამეტრებში.

ინტერნეტ-მისამართები

ახლა მოდით ვისაუბროთ იმაზე, თუ რანაირად ხდება ინტერნეტში ჩართული კომპიუტერების იდენტიფიცირება. კერძოდ კი- ინტერნეტ-მისამართებზე.

ინტერნეტ-მისამართი-არის უნიკალური რიცხვითი ან სიმბოლური მნიშვნელობა, რომელიც ქსელში ჩართული კომპიუტერის ზუსტად იდენტიფიცირების საშუალებას იძლევა. სწორედ ინტერნეტ-მისამართის მიხედვით პოულობს კლიენტი მისთვის სასურველ სერვერს. სწორედ ინტერნეტ-მისამართის მიხედვით ხდება მონაცემების გაგზავნა. ინტერნეტ-მისამართი-სერვერის თავისებური სახელია.

თავდაპირველად, ინტერნეტის ეპოქის გარიჟრაჟზე, ინტერნეტ-მისამართის ნაცვლად გამოიყენებოდა **IP**-მისამართი-**TCP/IP** პროტოკოლისათვის კომპიუტერის მაიდენტიფიცირებელი რიცხვითი მნიშვნელობა. თუ გავიხსენებთ, **TCP/IP,** გადასაცემ ინფორმაციას ყოფს პაკეტებად. ასე რომ, ყოველი ასეთი პაკეტი შეიცავს გამგზავნი და მიმღები კომპიუტერების **IP**-მისამართებს. **IP**-მისამართი შესანიშნავი რამ არის კომპიუტერებისათვის, მაგრამ ძალიან ცუდია ადამიანებისათვის. მას ასეთი სახე აქვს: 192.168.1.10. არც თუ ისეთი თვალსაჩინოა, ხომ მართალია? სწორედ ამიტომ, ინტერნეტის გაფართოებასთან ერთად მწყობრში ჩადგა ინტერნეტ-მისამართების ახალი სისტემა, რომლითაც დღემდე ვსარგებლობთ. ეს ეგრეთ წოდებული **დომენური მისამართებია,** რომელთა შესახებაც დაწვრილებით საუბარს აზრი აქვს.

მაგრამ, მანამ სანამ ჩვენ დავიწყებდეთ საუბარს დომენურ მისამართებზე, მოდით გავარკვიოთ, თუ რა არის დომენი. **დომენი**, ან **დომენური ზონა**-არის ინტერნეტის ცალკეული უბანი, რომელიც იქმნება მისი მართვის გასამარტივებლად. ასეთი უბანი შეიძლება იყოს დიდი ან მცირე, ან სულაც ერთი კომპიუტერისაგან შედგებოდეს. ყოველი დომენი აღინიშნება ინგლისური ასოებისაგან შემდგარი ტექსტური სტრიქონით. დომენების სტრუქტურა რუსულ "მატრიოშკა"-ს წააგავს: მცირე დომენები "ჩალაგებულია" დიდ დომენებში, ხოლო თავის მხრივ დიდი დომენები, კიდევ უფრო დიდ "გიგანტურ" დომენებში. გიგანტურ დომენებს უწოდებენ **მაღალი ან**

ზედა დონის დომენებს, ხოლო მათში ჩალაგებულ უფრო მცირე დომენებს-**ქვედა ან დაბალი დონის დომენებს.**
ზედა დონის დომენები შეიძლება იყოს ინტერნაციონალური ან ნაციონალური. ინტერნაციონალური დომენები კომპიუტერებს აერთიანებენ რაიმე ნიშნის მიხედვით: მათ მიეკუთვნებიან დომენები **com**(კომერციული სერვერები), **edu** (საგანმანათლებლო), **mil**(სამხედრო), **org**(ორგანიზაციები, რომლებიც არ არიან დაკავშირებული კომპიუტერებთან და ინტერნეტთან), **net** (ორგანიზაციები, რომლებიც დაკავშირებული არიან კომპიუტერთან და ინტერნეტთან) და ზოგიერთი სხვა. ნაციონალური დომენები აერთიანებენ კომპიუტერებს ტერიტორიული ნიშნით და გაიცემიან ქვეყნებზე; ესეთი დომენებია **us**(აშშ), **uk**(დიდი ბრიტანეთი), **fr**(საფრანგეთი), **de**(გერმანია), **ru**(რუსეთი) და სხვა.
რაც შეეხება ქვედა დონის დომენებს, ისინი როგორც წესი გაიცემიან ცალკეულ ორგანიზაციებზე ან იგივე ტერიტორიული პრინციპით. მათი ტექსტური აღნიშვნა ხშირად ემთხვევა იმ ორგანიზაციის ან რაიონის დასახელებას, რომლისთვისაც გაიცემიან.
ახლა ჩვენ თუ ჩავწერთ ჩვენთვის საინტერესო კომპიუტერის ადგილმდებარეობის განმსაზღვრელი ყველა დომენის აღნიშვნებს, მცირედან მსხვილი დომენის მიმართულებით, მათი ერთმანეთისაგან წერტილებით გამოყოფით, ჩვენ მივიღებთ ამ კომპიუტერის დომენურ სახელს. ასე მაგალითად, თვით კომპიუტერის სახელია **Comp45**, განყოფილება რომელშიც ის დგას **buh**(ბუღალტერია), ორგანიზაცია რომელშიც ეს განყოფილება შედის-**department**, ხოლო ქვეყანა-**ge** (საქართველო), ჩვენ მივიღებთ ასეთ დომენურ სახელს:

Comp45.buh.department.ge

მერწმუნეთ, რომ ამის დამახსოვრება გაცილებით იოლია, ვიდრე უცნაური **IP**-მისამართის.
კი, მაგრამ, პრობლემა იმაშია, რომ **TCP/IP** პროტოკოლი ვერ გებულობს დომენურ სახელებს. რა უნდა ვქნათ? როგორ გარდავმნათ დომენური სახელი მისთვის გასაგებ **IP**-მისამართად?
ამისათვის იყენებენ განსაკუთრებულ პროგრამებს, რომელთაც **DNS (Domain Name System,** ქსელური სახელების სისტემა) სერვერები ეწოდებათ. ისინი კომპიუტერებისაგან, რომელთაც **TCP/IP** პროტოკოლების საშუალებით მონაცემების სადმე გაგზავნა

ესაჭიროებათ, იღებენ დომენურ სახელებს და კომპიუტერს უბრუნებენ ამ დომენური სახელების შესაბამის **IP**-მისამართებს. ასეთი **DNS** სერვერები ყველა დომენს გააჩნია. გარდა ამისა, რამოდენიმე, მსოფლიოში ყველაზე მძლავრი **DNS** სერვერი, ყველა სხვა დომენებზე "მაღლა" იმყოფება. და ყველა მათგანს სამუშაო ყოფნის.
ახლა დავუბრუნდეთ ისევ დომენურ სახელებს და განვიხილოთ ზოგიერთი დეტალი, რომელიც დაგვეხმარება მომავალში.

დომენური სახელი ახდენს თვით სერვერული კომპიუტერის იდენტიფიცირებას და არა მასზე მომუშავე პროგრამა-სერვერისას. მსგავსი პროგრამა-სერვერი ერთსა და იმავე კომპიუტერზე რამოდენიმე შეიძლება იყოს: **Web**, **FTP**, **ფოსტა**, **ჩათი** და **სხვა**.
საჭირო სერვერისათვის მისამართად, სხვების გვერდის ავლით, დომენური სახელების წინ მიუთითებენ იმ პროტოკოლის აღნიშვნას, რომლითაც ეს სერვერი "ურთიერთქმედებს" კლიენტებთან.
აი ასე გამოიყურება იგი (პროტოკოლის აღნიშვნა გამოყოფილია მუქი შრიფტით):

http:// comp45.buh.department.ge

ftp:// comp45.buh.department.ge

პირველ შემთხვევაში ჩვენ მივმართავთ **Web**-სერვერს, ხოლო მეორე შემთხვევაში **FTP** სერვერს, რომლებიც ერთსა და იმავე კომპიუტერზე - **comp45.buh.department.ru** მდებარეობენ.

შენიშვნა: *მართალია არსებობს ერთი და იმავე კომპიუტერისათვის ერთდროულად რამოდენიმე დომენური სახელის და IP-მისამართის მინიჭების შესაძლებლობა, მაგრამ იგი იშვიათად გამოიყენება, თანაც განსაკუთრებულ შემთხვევებში. მომავალში სიმარტივისათვს ჩვენ ჩავთვლით, რომ ერთი დომენური სახელი (და ერთი IP-მისამართი)-ეს ერთი კომპიუტერია.*

თუკი რომელიმე სერვერი იყენებს წინასწარ განსაზღვრულისაგან (გაჩუმებითისაგან) განსხვავებულ პორტს, მაშინ საჭირო პორტი იწერება სერვერული კომპიუტერის დომენური სახელის შემდეგ და

გამოიყოფა მისგან ორი წერტილით. აი ასე (პორტის ნომერი გამოყოფილია მუქი შრიფტით):
http://comp45.buh.department.ge**:8000**

მაშ ასე, ინტერნეტის მუშაობის ძირითად პრინციპებს ჩვენ უკვე გავეცანით. ახლა მოდით ყურადღება **WWW**-ზე გავამახვილოთ- ჩვენ ძირითადად, სწორედ ამ სერვისით ვისარგებლებთ მთელი ამ წიგნის მანძილზე.

WWW-ს ძირითადი ცნებები

აქ ჩვენ გავარკვევთ ყველაფერს **Web**-გვერდებისა და **Web**-საიტების შესახებ, გავიგებთ თუ რითი განსხვავდება ვებ-საიტი ვებ-გვერდისაგან, ვისაუბრებთ **Web**-კლიენტებზე და **Web**-სერვერებზე, და გავეცნობით რამოდენიმე ახალ ცნებას.

Web-გვერდები და Web-საიტები

რა არის **Web**-გვერდი? ამ კითხვაზე პასუხის გაცემა ბევრს შეუძლია. ეს არის ინტერნეტ დოკუმენტი, რომელიც განკუთვნილია ინტერნეტში გასავრცელებლად **WWW**-სერვისის საშუალებით. თუ კი ყველასათვის გასაგებ ენაზე ვიტყვით, ეს არის ის, რასაც უჩვენებს თავის ფანჯარაში **Web**-გვერდების დათვალიერების პროგრამა-**Web** დამთვალიერებელი (ბრაუზერი).
ტექნიკური თვალსაზრისით, **Web-გვერდი**-არის ტექსტური ფაილი, რომელიც შეიცავს საკუთრივ ტექსტს, მისი ფორმატირების რამოდენიმე ბრძანებას და ინახება სერვერული კომპიუტერის მყარ დისკოებზე. **Web** დამთვალიერებლისაგან **HTTP** პროტოკოლით მოთხოვნის მიღებისას, **Web**-სერვერი (**WWW**-სერვისის უზრუნველმყოფი სერვერული პროგრამა) ამოიღებს აღნიშნულ ფაილს და უგზავნის მას **Web**-დამთვალიერებელს.
მაგრამ, როგორ აგებინებს **Web**-დამთვალიერებელი **Web**-სერვერს, თუ რომელი **Web**-გვერდი სჭირდება მას? ძალზე მარტივად- მოთხოვნის შემადგენლობაში ვებ-დამთვალიერებელი უგზავნის სერვერს აღნიშნული გვერდის შემცველი ფაილის დასახელებას და მის მიმანიშნებელ სრულ გზას (ბილიკს). მაგალითად ასე:

http://comp45.buh.department.ge/somepage.html

ეს მოთხოვნა აიძულებს **Web**-სერვერს ამოიღოს და გაუგზავნოს **Web**-დამთვალიერებელს ფაილი **somepage.html**.

რა არის **Web-საიტი**? **ეს საერთო თემატიკით გაერთიანებული ვებ-გვერგების ერთობლივობაა.** ვებ-საიტი ასევე სერვერული კომპიუტერის მყარ დისკოზე განთავსებულ საქაღალდეებში ჩალაგებული ფაილების ანაკრების სახით ინახება. (რა თქმა უნდა საქაღალდეების გამოყენება სავალდებულო არ არის, მაგრამ ასე უფრო მოხერხებულია, განსაკუთრებით, თუ ფაილების რაოდენობა დიდია და ყველა მათგანი სხვადასხვა ტიპისაა). როგორც ვხედავთ, წმინდა ტექნიკური განსხვავება ვებ-გვერდებსა და ვებ-საიტებს შორის არც თუ ისე დიდია.

ახლა კი ვისაუბროთ სერვერული კომპიუტერის დისკოზე საიტების დამახსოვრების ზოგიერთ ტექნიკურ დეტალებზე.

უპირველეს ყოვლისა, საიტის შემადგენელი ყველა ფაილის შენახვისათვის (ან ვებ გვერდის, თუ მას განვიხილავთ, როგორც ვებ-საიტს "გადაგვარებულ" ნაირსახეობას), სერვერული კომპიუტერის დისკოზე იქმნება სპეციალური საქაღალდე, რომელსაც **ძირეული საქაღალდე** (root folder) ეწოდება. ყველაფერი, რაც ამ საქაღალდეში არ იმყოფება, ვებ-სერვერის მიერ ამორიცხული იქნება საიტის შემადგენლობიდან.

მაშ ასე, ვებ-საიტის შემადგენელი ყველა ფაილი ინახება სერვერული კომპიუტერის დისკოზე სპეციალურად შექმნილ საქაღალდეში. საიტისათვის ამ საქაღალდეს ქმნის ადამიანი, რომელიც პასუხისმგებელია პროგრამა ვებ-სერვერის (ან მთლიანად სერვერული კომპიუტერის) გაწყობაზე (გამართვაზე) და მის მომსახურებაზე-**ადმინისტრატორი.**

მოცემულ საქაღალდემდე სრული გზა (მასთან მიმავალი ბილიკი) შეიტანება ვებ-სერვერის პარამეტრებში, რათა ამ უკანასკნელმა საჭიროების შემთხვევაში შეძლოს მისი მოძებნა. საიტის მთელი შიგთავსი მოთავსებული უნდა იყოს **ძირეულ საქაღალდეში.**

შენიშვნა: *ზოგადად, ვებ-სერვერების ყველა სერიოზული პროგრამა იძლევა ეგრეთ წოდებული ვირტუალური საქაღალდეების შექმნის შესაძლებლობას.* ***ვირტუალური საქაღალდე****-ეს არის საქაღალდე, რომელიც განთავსებულია კომპიუტერის ფაილური სისტემის ნებისმიერ ადგილას, მაგრამ ვებ-სერვერის მიერ აღიქმება როგორც საიტის შემადგენელი ნაწილი. შემდგომში ჩვენც ვირტუალურ*

საქაღალდეებს, საიტების შემადგენელ ნაწილებად განვიხილავთ, თუ სხვა რამე არ იქნება სპეციალურად მითითებული.

როდესაც ვებ-დამთვალიერებელი ვებ-სერვერს უგზავნის ასეთი სახის მოთხოვნას:

http://www.somesite.ge/somepage.html,

ვებ-სერვერი პოულობს somepage.html ფაილს საიტის ძირეულ საქაღალდეში და უგზავნის მას ვებ-დამთვალიერებელს. თუ ვებ-დამთვალიერებელს დასჭირდება ფაილი, რომელიც მოთავსებულია არა თვით ძირეულ, არამედ ერთ-ერთ-მასში ჩალაგებულ საქაღალდეში, მან უნდა გამოაგზავნოს ასეთი მოთხოვნა:

http://www.somesite.ge/somefolder1/somefolder2/somerage.html.

ამ შემთხვევაში ვებ-სერვერი უგზავნის ვებ-დამთვალიერებელს ფაილს **somepage.html**, რომელიც მოთავსებულია საქაღალდეში **somefolder1/somefolder2**. ეს უკანასკნელი კი თავის მხრივ, ასევე საიტის ძირეულ საქაღალდეშია მოთავსებული.

შენიშვნა: *ვირტუალურ საქაღალდეში განთავსებული ფაილისადმი მიმართვისათვის, გამოიყენება ანალოგიური მოთხოვნა:*

http://www.somesite.ge/somevirtualfolder1/somepage.html

მაშ ასე, ყველაფერი კარგად არის, ყველაფერი არაჩვეულებრივია და ყველაფერი გასაგებია. მაგრამ ჩვენ ხომ ძალიან იშვიათად ვკრიფავთ ვებ-დამთვალიერებლის ინტერნეტ-მისამართის შეტანის ველში ისეთ მოთხოვნებს, რომლებიც უშუალოდ მიუთითებენ ჩვენთვის საჭირო ვებ-გვერდზე. უფრო ხშირად ჩვენი მოთხოვნები გამოიყურებიან უფრო "მორიდებულად", მაგალითად ასე:

http://www.somesite.ge

ანუ ისინი არ მიუთითებენ უშუალოდ ფაილზე. როგორ იქცევა სერვერი ამ შემთხვევაში?

საქმე იმაშია, რომ საიტის ერთ-ერთი გვერდი წინასწარ განისაზღვრება, როგორც საიტის მთავარი, ეგრეთ წოდებული **წინასწარ განსაზღვრული (გაჩუმებითი) გვერდი.** სწორედ ის ეგზავნება ვებ-დამთვალიერებელს, თუ მას თავის მოთხოვნაში კონკრეტული გვერდი (ან ფაილი) არა აქვს მითითებული. ამ გვერდის ფაილის სახელს განსაზღვრავს ვებ-სერვერის ადმინისტრატორი პარამეტრების შერჩევისას-როგორც წესი ეს არის **default.htm(l)** ან **index.htm(l).**
თუ ჩვენი ვებ-დამთვალიერებლის ინტერნეტ-მისამართის შესატან ველში ჩვენ ავკრიფავთ ამდაგვარ მისამართს:

http://www.somesite.ge,

Web-დამთვალიერებელი გამოგვიგზავნის ჩვენი საიტის ძირეულ საქაღალდეში არსებულ გვერდს-**default.html**.

ადრე ჩვენ განვიხილეთ ეგრეთ წოდებული აბსოლუტური ინტერნეტ-მისამართები, რომლებიც მოიცავენ როგორც თვით ვებ-სერვერის მისამართებს, ასევე საჭირო ვებ-გვერდების ფაილების სახელებს. მაგრამ ფაილის ინტერნეტ მისამართის მითითება შეიძლება ვებ-დამთვალიერებელში უკვე გახსნილი (მიმდინარე) გვერდის მიმართაც: **page2.htm1**
ამ მოთხოვნის მიღებისთანავე ვებ-სერვერი გამოგვიგზავნის ჩვენ **page2.htm1** გვერდს, რომელიც იმავე საქაღალდეშია, რაც მიმდინარე გვერდი. აღსანიშნავია, რომ სერვერის სახელს ეს მისამართი არ შეიცავს, ვინაიდან იგულისხმება, რომ ფაილი **page2.htm1** მდებარეობს იმავე სერვერზე, რაც მიმდინარე გვერდის ფაილი: **folder/page2.html**.
ეს მოთხოვნა აიძულებს ვებ-სერვერს საქაღალდე **folder**-ში მოძებნოს **page2.htm1** გვერდი, რომელიც მიმდინარე ვებ-გვერდის შემცველ საქაღალდეშია მოთავსებული.
ასეთი სახის: **…/folder2/page3.html** მოთხოვნა კი დაგვიბრუნებს **page3.html** გვერდს **folder2** საქაღალდედან, რომელიც იმავე საქაღალდეშია მოთავსებული, რაც ის საქაღალდე, რომელშიც მიმდინარე ვებ-გვერდი ინახება.
დაბოლოს ავღნიშნოთ, რომ ინტერნეტ მისამართს, რომელიც რაიმე ფაილის სახელს, მიუთითებს მიმდინარე გვერდის შემცველი

ფაილის მიმართ და არ შეიცავს თვით სერვერის სახელს- **ფარდობითი მისამართი ეწოდება**.
ახლა კი შეგვიძლია უფრო დაწვრილებით ვისაუბროთ პროგრამა ვებ-დამთვალიერებელსა და პროგრამა ვებ-სერვერზე.

Web-დამთვალიერებლები

ჩვენ უკვე ვიცით, რომ ვებ-დამთვალიერებლები, ვებ-გვერდებისა და ვებ-საიტების დასათვალიერებელი პროგრამებია. მათი ძირითადი ამოცანაა-გაუგზავნონ ვებ-სერვერებს კორექტულად, ყველა სტანტარტების მოთხოვნების შესაბამისად ფორმულირებული კლიენტური მოთხოვნები, მიიღონ სერვერული პასუხები და გამოიტანონ მიღებული გვერდები ეკრანზე. ამისათვის ვებ-დამთვალიერებლის ფანჯარა შეიცავს ინტერნეტ-მისამართის შესატან ველს და არეს რომელშიც გამოიტანება ვებ-გვერდი (ცხადია იგი შეიცავს სათაურს, მენიუსა და ინსტრუმენტების პანელს, ისევე, როგორც **Windows**-ის ნებისმიერი სხვა გამოყენებითი პროგრამების მრავალრიცხოვანი ფანჯრები).
სერვერიდან ვებ-გვერდის ფაილის მიღების შემდეგ (და მასთან დაკავშირებული ყველა სხვა ფაილისა-ვინაიდან გვერდი შესაძლებელია შედგებოდეს მრავალი ფაილისაგან) ვებ-დამთვალიერებელი ინახავს მათ კომპიუტერ-კლიენტის მყარი დისკოს განსაკუთრებულ უბანში, რომელსაც **ქეში** ეწოდება. ამ ქეშს შესაძლოა გააჩნდეს როგორც ჩვეულებრივი საქაღალდის სახე (**Microsoft Internet Explorer**-ის და **Opera**-ს ქეშები), ასევე დიდი ფაილის სახე (**Netscape**-ისა და **Mozilla**-ს ქეშები).
რისთვის არის ეს საჭირო? თუნდაც იმისთვის, რომ შემდგომში ჩვენ გვქონდეს მოცემული გვერდის დათვალიერების შესაძლებლობა ინტერნეტში ჩართვის გარეშე. ყველა თანამედროვე ვებ-დამთვალიერებლები უზრუნველყოფენ ეგრეთ-წოდებულ **ავტონომიურ რეჟიმს** (**offline mode**), როდესაც მათ ეკრანზე გამოაქვთ მხოლოდ ის გვერდები, რომლებიც ქეშებშია მოთავსებული. (უნდა აღინიშნოს, რომ ძალზე მოხერხებული რამეა!) თუ ჩვენ შევეცდებით დავათვალიეროთ გვერდი, რომელიც არ არის ქეშში დამახსოვრებული, ვებ-დამთვალიერებელი შემოგვთავაზებს ინტერნეტში ჩართვას და შესაბამისი გვერდის ჩატვირთვას.
ახლა გავეცნოთ დღეისათვის ყველაზე პოპულარულ ვებ-გვერდების დათვალიერების პროგრამებს. ყველა მათგანი ძირითადად ერთსა და იმავე სტანდარტებს აკმაყოფილებენ და ერთმანეთისაგან

განსხვავდებიან მხოლოდ აღნიშნულ სტანდარტებში გაუთვალისწინებელი დეტალებით, ასევე მომხმარებლებისათვის მათი მოხერხებულობის დონით.

ვირტუალური სივრცეების ნამდვილი მეფე-ეს, რათქმა უნდა **Microsoft Internet Explorer**-ია. ის **Windows**-ზე მომუშავე ყველა კომპიუტერშია დაყენებული (რამაც, როგორც ავი ენები ამბობენ, განაპირობა მისი პოპულარობა). ეს ძალიან მძლავრი, სწრაფი, რესურსებისადმი ნაკლებად მომთხოვნი და საოცრად მოხერხებული პროგრამაა. ინტერნეტ-მომხმარებლების უმრავლესობა ვებ-გვერდების დასათვალიერებლად სწორედ **Internet Explorer**-ს იყენებს. დღეისათვის ხელმისაწვდომია 7.0 ვერსია და გავრცელებული ხმების მიხედვით, მოსალოდნელია ახალი ვერსია, რომელიც შევა ახალი **Windows**-ის შემადგენლობაში, თუმცა ფირმა **Microsoft**-ი ჩვეულებისამებრ სდუმს ამ თემასთან დაკავშირებით.

პოპულარულობის მიხედვით მეორე ადგილი უკავია ნორვეგიულ პროგრამას **Opera**-ს.

ეს საკმაოდ მძლავრი და ძალიან სწრაფი პროგრამაა, რომელიც უზრუნველყოფს ყველა ოფიციალური ვებ-სტანდარტების მოთხოვნებს. მიუხედავად ამისა ძალიან მომთხოვნია სისტემური რესურსებისადმი, განსაკუთრებით რთული ვებ-გვერდების ასახვის შემთხვევაში. გარდა ამისა იგი ფასიანია, უფასო ვერსია კი უჩვენებს რეკლამას. უკანასკნელად გამოსული ვერსიის ნომერია 7.60 და როგორც ჩანს ძალიან მალე დაძველდება, ვინაიდან **Opera**-ს ვერსიები ძალიან სწრაფად იცვლებიან.

ერთ დროს **WWW**-სივრცის ბელადი **Netscape Navigator** ახლა უკვე აღარ არის პოპულარული-მას ინტერნეტ-მომხმარებლების მხოლოდ 1% იყენებს. თუმცა **Navigator**-ის უკანასკნელი ვერსია 7.2 არც თუ ისე ცუდად გამოიყურება, უზრუნველყოფს ინტერნეტის თანამედროვე ყველა სტანდარტის მოთხოვნებს, კორექტულად ასახავს ვებ-გვერდების უმეტესობას და სისტემური რესურსებისადმიც არც თუ ისე დიდად მომთხოვნია. მაგრამ მაინც **Navigator** მრავალი პარამეტრით ჩამორჩება როგორც **Internet Explorer**-ს, ასევე **Opera**-ს.

ყველაზე "უმცროსი" ვებ-დამთვალიერებელი ატარებს სახელს-**Mozilla**. ეს პროგრამა ვრცელდება უფასოდ. უფრო მეტიც მისი კოდები ღიაა შესწავლისა და მოდიფიცირებისათვის. ის იმავე პროგრამულ ბირთვზეა აგებული, რაც **Navigator-7**, უფრო ზუსტად პირიქით-**Navigator-7** არის აგებული **Mozilla**-ს ბაზაზე. (თავდაპირველად **Mozilla** იქმნებოდა **Navigator**-ის ახალი

პროგრამული ბირთვის საცდელად, მაგრამ შემდგომში დამოუკიდებელ პროდუქტად იქცა). ეს ახალი პროდუქტი საკმაოდ კარგია, უზრუნველყოფს ყველა ვებ-სტანდარტის მოთხოვნას, არ არის მომთხოვნი სისტემური რესურსებისადმი, საკმაოდ სწრაფია და გააჩნია მრავალი საინტერესო და ძალზე სასარგებლო შესაძლებლობები, როგორებითაც ჯერ-ჯერობით ვერც ერთი მისი კონკურენტი ვერ დაიტრაბახებს. მართალია, ჯერ ის არც თუ ისე პოპულარულია, მაგრამ მომავალში, შესაძლოა გამოიჩინოს თავი.

უნდა ითქვას ისიც, რომ ყველა ზემოთ ჩამოთვლილი პროგრამები-სინამდვილეში პროგრამულ პაკეტებს წარმოადგენენ. სახელდობრ ვებ-დამთვალიერებლების გარდა ისინი უამრავ სხვა პროგრამებსაც შეიცავენ: საფოსტო კლიენტებს, ახალი ამბების კლიენტებს, მულტიმედია ფაილების გამთამაშებლებს, ჩათების კლიენტებს, ინტერნეტ-პეიჯერებს და ვებ-საიტების შემქმნელებისათვის სპეციალურ უტილიტებსაც.

ერთ პაკეტში ინტერნეტთან სამუშაო მრავალი სხვადასხვა პროგრამების გაერთიანების პრაქტიკა, უკვე დიდი ხანია არსებობს, მაგრამ ახლახან მას წერტილი დაუსვეს **Mozilla**-ს შემქნელებმა. მათ ამ პაკეტიდან გამოყვეს თვით ვებ-დამთვალიერებელი (კიდევ რამდენიმე მცირე უტილიტა) და დაარქვეს მას **Firefox**. ამ დროისათვის ის, თანდათან იკრებს პოპულარობას, და ჩრდილში აქცევს, როგორც თავის "წინამორბედ" **Mozilla**-ს (მითუმეტეს, რომ შემქმნელები მომავალში საერთოდ აპირებენ ამ უკანასკნელის დახურვას), ასევე **Internet Explorer**-საც.

სულ ახლახანს ფირმა **Apple**-მა, რომელიც ვიწრო წრეებში ფართოდ ცნობილ კომპიუტერებს-**Macintosh**-აწარმოებს, განაცხადა თავისი საკუთარი ვებ-დამთვალიერებლის **Safari-**ს გამოშვების შესახებ. ამტკიცებენ, რომ ეს ყველაზე, მათ შორის **Opera**-ზეც, სწრაფი ვებ-დამთვალიერებელია. მისი კოდები ასევე ღიაა სწავლისა და მოდიფიცირებისათვის.

ამ დროისათვის **WWW**-სივრცეებს "სერავენ" პრაქტიკულად მხოლოდ ზემოთ აღნიშნული ხუთი პროგრამა (ექვსი თუ ჩავთვლით **Firefox**-ს ცალკე პროექტად). არსებობს კიდევ რამდენიმე ნაკლებად ცნობილი ვებ-დამთვალიერებელი და ასევე პროგრამების საკმაოდ მრავალრიცხოვანი კოჰორტა, რომლებიც აგებულნი არიან **Internet Explorer**-ის პროგრამული ბირთვის საფუძველზე და აფართოებენ მის შესაძლებლობებს. ჩვენ მათ არ განვიხილავთ.

მხოლოდ ისღა დაგვრჩა სათქმელი, რომ ვებ-დამთვალიერებლის არჩევა-ეს ყველა ადამიანის პირადი საქმეა. ყველა მათგანი აკმაყოფილებს ერთი და იგივე სტანდარტების მოთხოვნებს (მართალია, ხშირად თავისებურად) და უზრუნველყოფს მომხმარებლებს დაახლოებით ერთი და იმავე შესაძლებლობებით (თუმცა ყველა ვერ ჩათვლის მათ დამაკმაყოფილებლად და მოხერხებულად). ასე, რომ როგორც სიმღერაშია ნათქვამი "იფიქრეთ თვითონ, გადაწყვიტეთ თვითონ".

Web-სერვერები

ვინაიდან ჩვენ უკვე არა მხოლოდ ვებ-საიტების მომხმარებლები, არამედ თითქმის შემქმნელებიც ვართ, უნდა გვაინტერესებდეს არა მხოლოდ ვებ-დამთვალიერებლები, არამედ ვებ-სერვერებიც. მოდით ვისაუბროთ მათზე.

ვებ-სერვერების "ზოოპარკი" არაფრით არ ჩამოუვარდება ვებ-დამთვალიერებლების (ან თუ გავითვალისწინებთ, რომ ვებ-დამთვალიერებლები ერთმანეთთან ხისტ კონკურენციაში არიან-"სერპანტარიუმს"), ასე რომ გვაქვს შესაძლებლობა, პროგრამა შევარჩიოთ ჩვენი გემოვნების მიხედვით. ვებ-დამთვალიერებლებისაგან განსხვავებით, ვებ-სერვერებს შორის არ არსებობს აშკარად გამოკვეთილი ლიდერი-მათ შორის ყველაზე გავრცელებულებიც კი ვერ იკავებენ ბაზრის ნახევარზე მეტს.
ჩვენი მოკლე მიმოხილვა დავიწყოთ ფირმა **Microsoft**-ის ორი პაკეტით: **Personal Web Server** და **Internet Information Server**. ორივე ეს პაკეტი მომხმარებელს მიეწოდება **Microsoft Windows**-ის შემადგენლობაში. მათ შორის პირველი **Windows 98**-ის და **Me**-ს შემადგენლობაში, ხოლო მეორე **Windows NT**, **2000**, **XP** და **2003**-ის შემადგენლობაში. თავიანთ მოვალეობებს ისინი ძალიან კარგად ართმევენ თავს, არ ფლანგავენ სისტემურ რესურსებს, მარტივად ხდება მათი დაყენება, უზრუნველყოფენ მრავალი მოწინავე ინტერნეტ-ტექნოლოგიების მოთხოვნებს და შესაბამისად სწორად გამართვის შემთხვევაში ადვილად უსწორდებიან კონკურენტებს. სახელდობრ ვებ-სერვერის გარდა, ისინი შეიცავენ აგრეთვე **FTP** და ფოსტის სერვერებს, აგრეთვე დამატებითი პროგრამების გარკვეულ რაოდენობას.
ვებ-სერვერი **Apache**-შეიძლება ითქვას, რომ ყველაზე გავრცელებულია. მის ღირსებებს შორის შეიძლება დასახელდეს:

უფასო გავრცელება (უფრო მეტიც მისი კოდები ღიაა), გამართვის სიმარტივე, საკმაოდ მაღალი წარმადობა, კარგი მხარდაჭერა. ყოველ შემთხვევაში შედარებით დაბალი დატვირთვის მქონე ვებ-საიტებისათვის ის იდეალური არჩევანია. თავის დროზე ფირმა **Netscape**-მა -ცნობილი ვებ-დამთვალიერებლის **Navigator**-ის შემქმნელმა-შექმნა ვებ-სერვერიც, რომელსაც ერქვა-**Netscape Web-Server**. წარმადობისა და თანამედროვე ინტერნეტ-ტექნოლოგიების მხარდაჭერის თვალსაზრისით იგი არაფერს არ უთმობს თავის კონკურენტ **Microsoft**-ს, მაგრამ სამწუხაროდ ვერ მოიპოვა მსგავსი პოპულარობა. კიდევ არის ერთი ფრიად საყურადღებო **Web**-სრვერი-**Sambar**. მას ისეთი რაოდენობის ინტერნეტ-ტექნოლოგიების მხარდაჭერა შეუძლია (ბევრი მათგანი ექსკლუზიურია), რომ უბრალოდ საკვირველია-რანაირად და რაში უნდა გამოიყენოს ადამიანმა მთელი ეს სიმდიდრე. **Sambar**-ს მხოლოდ ორი ნაკლოვანება გააჩნია: დაბალი ცნობადობა და გამართვის შედარებითი სირთულე. ნაკლებად ცნობილი და სპეციალიზირებული სერვერების განხილვას ჩვენ არ შევუდგებით. მათი რაოდენობა ძალიან დიდია. უმჯობესია ვილაპარაკოთ იმაზე, თუ როგორ განვათავსოთ ჩვენი საიტი ინტერნეტში.

Web-საიტის პუბლიკაცია ინტერნეტში.

ჰოსტინგ-პროვაიდერები

მაშ ასე, დავუშვათ, რომ ჩვენ შევქმენით ჩვენი საიტი (ჩვენ აუცილებლად შევქმნით მას, სანამ დავამთავრებთ ინტერნეტ ტექნოლოგიების შესწავლას ამ წიგნის მიხედვით). აწი ჩვენ უნდა გავაკეთოთ ისე, რომ საკუთარი თვალებით მისი ხილვის ყველა მსურველს, ჰქონდეს ამის შესაძლებლობა-განვათავსოთ ან როგორც გამოცდილი ინტერნეტის სპეციალისტები ამბობენ **გამოვაქვეყნოთ** ის ინტერნეტში. ამისათვის ჩვენ გვჭირდება ინტერნეტში ჩართვა და ვებ-სერვერი.

თუ ჩვენი კომპიუტერი ჩართულია ინტერნეტში ჩქაროსნული არხის საშუალებით ან ინტერნეტის სტანდარტებით მომუშავე ლოკალურ ქსელში (ეგრეთ წოდებულ **ინტრანეტში),** მაშინ ჩვენ უბრალოდ შეგვიძლია მასზე ვებ-სერვერის დაყენება და ჩვენი ვებ-საიტის იქვე მოთავსება. ეს ყველაზე მარტივი შემთხვევაა, თუმცა, რათქმა უნდა ჩვენ მოგვიწევს გზადაგზა ადმინისტრატორის პროფესიის ათვისება.

იმ "ბედნიერი გამონაკლისებისათვის", ვინც ინტერნეტში სატელეფონო არხების საშუალებით გადიან, სამი ხერხი არსებობს, რათა მათ შეძლონ თავიანთი ვებ-ქმნილება წარუდგინონ მასზე მოწყურებულ მასებს. მოდით ჩამოვთვალოთ ისინი უფრო მარტივიდან უფრო რთულის მიმართულებით.

ინტერნეტ-პროვაიდერების სოლიდური უმრავლესობა, ინტერნეტში დაშვების გარდა, თავაზობს თავის კლიენტებს სხვა მომსახურებებსაც: ელექტრონულ ფოსტას, მათ საიტზე დაშვებას, სადაც ახალი ამბები, სხვადასხვა დოკუმენტაცია და ფაილური არქივებია განთავსებული და სხვა. მაშ ასე ამ "სხვათა" შორის არის ისეთი მომსახურებაც, როგორიც არის კლიენტების ვებ-საიტების განთავსებისათვის სერვერული კომპიუტერის მყარ დისკოებზე ადგილების გამოყოფა. პირველი ხერხი სწორედ იმაში მდგომარეობს, რომ გავარკვიოთ ამა თუ იმ ინტერნეტ-პროვაიდერის სერვერზე საიტების პუბლიკაციის პირობები და ამ პირობების დაცვით გამოვაქვეყნოთ საიტი.

თუ ინტერნეტ-პროვაიდერი რაიმე მიზეზის გამო ხელმომჭირნეობს, შეიძლება მივმართოთ მეორე ხერხს. ინტერნეტში საკმაოდ ბევრი საიტი არსებობს, რომლებიც კლიენტებს საიტების გამოსაქვეყნებლად უფასოდ სთავაზობენ ადგილს თავიანთ სერვერებზე. პროცესი ამ შემთხვევაშიც მარტივია: შევდივართ ასეთი სერვერის საიტზე, გავდივართ რეგისტრაციას, ვარკვევთ საიტის პუბლიკაციის პირობებს და ვაქვეყნებთ საიტს.

ყველაფრით კარგია უფასო სერვერები: ფულსაც არ იღებენ და საიტების პუბლიკაციის საშუალებასაც იძლევიან. მაგრამ უფასო ყველი ბევრი არ შეიძლება იყოს. როგორც წესი, ასეთ სერვერებზე საიტისათვის გამოსაყოფი დისკის მოცულობა ძალზე შესღუდულია, ე.ი. დიდ საიტს ასეთნაირად ვერ გამოაქვეყნებ. ასევე ადმინისტრატორს შეუძლია შესღუდოს იმ მომხმარებლების რაოდენობა, რომლებსაც შეუძლიათ თქვენს საიტზე ერთდროულად შემოსვლა. ამასთან ვებ-საიტების შექმნისათვის გამოსაყენებელი ზოგიერთი ვებ-ტექნოლოგიების მხარდაჭერის საკითხიც შეიძლება არ იყოს მოწოდების დონეზე.

თუ ჩვენ უფრო მეტი გვინდა, ვიდრე უფასო სერვერების შემოთავაზებებია, მაშინ მოგვიწევს საკუთარი საფულეს ამოღება და მესამე გზით წასვლა-გამოვაქვეყნოთ საიტი ფასიან სერვერზე. ღმერთის წყალობით ასეთები საკმაოდ ბევრია და მათი მომსახურებაც შედარებით იაფია. ორგანიზაციებს, რომლებიც

კლიენტებს სთავაზობენ ადგილებს თავიანთ სერვერებზე ვებ-საიტების პუბლიკაციისათვის, **ხოსტინგ-პროვაიდერებს** უწოდებენ.

რა იქნება ამის მერე?

ესეც ასე. ინტერნეტ-ტექნოლოგიებს მოვრჩით. ზოგადი საკითხები ჩვენ უკვე განვიხილეთ, დროა შევუდგეთ ვებ-გვერდებისა და ვებ-საიტების შექმნას.

შემდეგ თავში, ჩვენ გავარკვევთ თუ რას წარმოადგენენ ვებ-გვერდები და როგორ ქმნიან მათ. ჩვენ შევისწავლით ვებ-გვერდების საწერ ენას **HTML**-ს, გავიგებთ, რისგან უნდა შედგებოდეს ყოველი, საკუთარი თავის პატივისმცემელი ვებ-გვერდი, როგორ ვიმუშაოთ ტექსტთან და გრაფიკასთან და რა საშუალებებით არის შესაძლებელი ჩვენი ქმნილების რამდენადმე "შელამაზება".

მაგრამ ჯერ, ჩვენ უნდა გავარკვიოთ, თუ რა მაგალითზე უნდა ვისწავლოთ ვებ-გვერდების შექმნა. სახელდობრ, უნდა განვსაზღვროთ ჩვენი პირველი ვებ-საიტის სტრუქტურა.

თავი 2

HTML-Web-საიტების საწერი ენა

ჩვენ უკვე მივუახლოვდით იმ მომენტს, როდესაც უნდა შევქმნათ ჩვენი პირველი ვებ-საიტი. ჯერ-ჯერობით ეს მხოლოდ პატარა მაგალითი იქნება, რომელიც იმისთვის გვჭირდება, რომ შევისწავლოთ ისეთი ვებ-დოკუმენტების შექმნის პრინციპები, რომლებიც ნორმალურად აისახებიან ყველა ვებ-დამთვალიერებლებში (**ბრაუზერებში**). თვით საიტის შექმნით კი ჩვენ უფრო მოგვიანებით დავკავდებით.

სინამდვილეში ვებ-გვერდების შექმნის პროცესში არ არის არაფელი განსაკუთრებით რთული. თუ გავითავისებთ რამოდენიმე ზოგად ტერმინს და ხელთ გვექნება შესაბამისი ცნობარები და ერთი-ორი პროგრამა, შესაძლებელია ვებ-გვერდების, როგორც ორცხობილების დაშტამპვა. გასაკვირი არ არის, რომ ახლა ვებ-დიზაინერის პროფესია ასე ფართოდ არის გავრცელებული.

სხვა საქმეა-კარგი ვებ-საიტი. მისი შექმნა იწყება საკმაოდ ხანგრძლივი დაგეგმარებით. საჭიროა მისი სტრუქტურის მოფიქრება, ყველა საჭირო მასალების (რომელთა ინტერნეტში

გამოქვეყნებაც გვაქვს გადაწყვეტილი-ხომ სწორად ამისთვის იქმნება საიტი) მოგროვება და მათი იმ სახემდე მიყვანა, რომელიც ვებ-დამთვალიერებლისთვის მისაღები იქნება. საიტში შემავალი ყველა ვებ-გვერდის შექმნის შემდეგ საჭიროა მათი ერთმანეთთან დაკავშირება და ამ კავშირების აუცილებელი შემოწმება შეცდომების არსებობაზე. შესაძლებელია მოხდეს ისე, რომ ზოგიერთ ვებ-გვერდზე დამთვალიერებელმა საერთოდ ვერ მოახერხოს შესვლა.
ეხლა მოდით ჩავრთოთ კომპიუტერი-მან უკვე საკმარისად დაისვენა-და შევუდგეთ საქმეს. დავიწყოთ ვებ-გვერდების საწერი ენით-**HTML**-ით.

შესავალი HTML ენაში

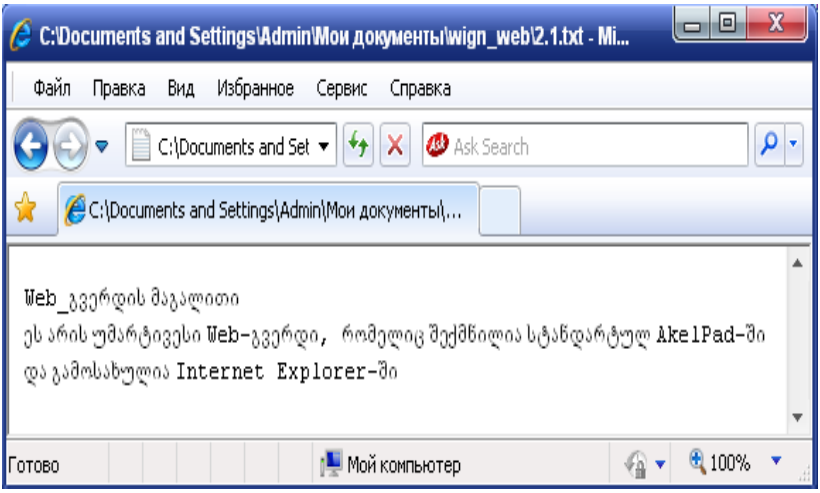
HTML (**Hyper Text Markup Language**, ჰიპერტექსტური მონიშვნის ენა)-ეს განსაკუთრებული ენაა რომელიც გამოიყენება ვებ-საიტების დასაწერად. ახლა ჩვენ გავეცნობით მას და ამავდროულად მოვსინჯავთ ჩვენს ძალებს, პირველი, ჯერ-ჯეროზით ძალიან მარტივი ვებ-გვერდების დაწერაში.

HTML-ის ტეგები. ტექსტის ფორმატირება.

ჩავრთოთ პატარა ტექსტური რედაქტორი **NotePad**-ი (შესაძლებელია **AkelPad**), რომელიც შედის **Windows**-ის შემადგენლობაში და ავკრიფოთ მასში ასეთი ტექსტი:

Web_გვერდის მაგალითი
ეს არის უმარტივესი Web_გვერდი,რომელიც შექმნილია
სტანდარტულ AkelPad-ში და გამოსახულია Internet Explorer-ში.

შევინახოთ ეს ტექსტი ფაილში **2.1.txt**. შემდეგ გავხსნათ იგი ვებ-დამთვალიერებელში-**Microsoft Internet Explorer**, რომელიც ასევე შედის **Windows**-ის სტანდარტულ კომპლექტაციაში. ჩვენ დავინახავთ იმას, რაც გამოსახულია ნახ.2.1-ზე.



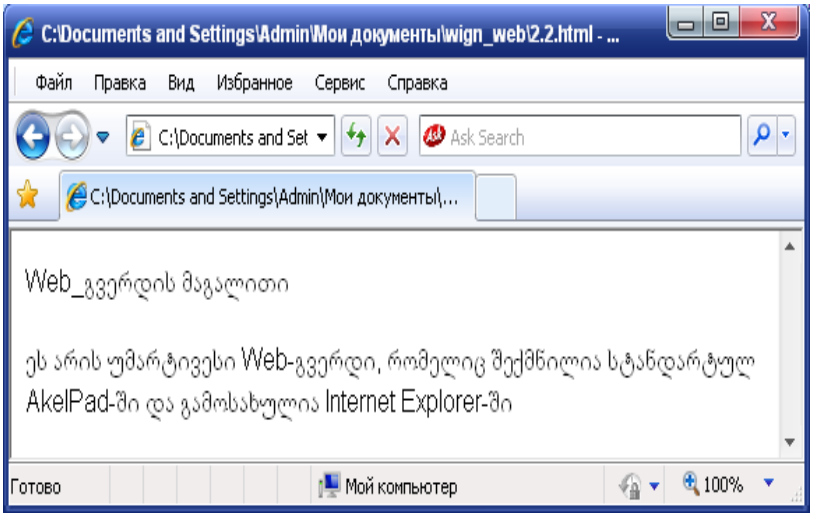
ნახ.2.1 Internet Explorer-ში გამოსახული მარტივი ტექსტი

არა ეს ჯერ-კიდევ ვებ-გვერდი არ არის. ეს ვებ-დამთვალიერებელში გახსნილი მარტივი ტექსტური ფაილია (ვებ დამთვალიერებლებს აგრეთვე შეუძლიათ მარტივი ტექსტური ფაილების გახსნაც). მოდით გადავაქციოთ იგი ნამდვილ ვებ-გვერდად.
დასაწყისისათვის მოვახდინოთ ფაილ **2.1. txt**-ს რედაქტირება, რათა მისი შიგთავსი გამოიყურებოდეს ასე (დამატებული ფრაგმენტები გამოყოფილია მუქი შრიფტით):

<P>Web_გვერდის მაგალითი**</P>**
<P>ეს არის უმარტივესი Web_გვერდი,რომელიც შექმნილია სტანდარტულ AkelPad-ში და გამოსახულია Internet Explorer-ში.**</P>**

ამის შემდეგ დავიმახსოვროთ რედაქტირებული ტექსტი მეორე ფაილში-**2.2. html**.
(დამახსოვრების სტანდარტულ ფანჯარაში ფაილის სახელის შეტანისას, იგი ბრჭყალებში უნდა ჩავსვათ. წინააღმდეგ შემთხვევაში ტექსტური რედაქტორი თავისი სულიერი სიკეთის გამო დაუმატებს მას გაფართოებას **txt** და ჩვენი ფაილი მიღებს სახელს **1.1 . htm. txt**.). თუ გავხსნით მას **Internet Explorer**-ში, დავინახავთ იმას, რაც გამოსახულია ნახ.2.2-ზე. შეამჩნიეთ სხვაობა? ტექსტური ფაილი **2.1. txt**, **Internet Explorer**-მა გამოიტანა "როგორც არის", ყოველგვარი

შელამაზების გარეშე. კერძოდ, მან არ დაყო ძალზე გრძელი აბზაცი ორ სტრიქონად (თუმცა ასე აჯობებდა). ყოველივე ეს იმის გამო, რომ ჩვეულებრივი ტექსტი არ შეიცავს ბრძანებებს მისი ფორმატირებისათვის.



ნახ.2.2. Internet Explorer-ში გახსნილი ჩვენი პირველი ვებ-გვერდი.

სულ სხვაა ფაილ **2.2 html**-ში დამახსოვრებული **HTML**-კოდი. **Internet Explorer**-მა მოახდინა ორი აბზაცის ფორმირება-ეს კარგად ჩანს ნახ.2.2-ზე-ძალიან გრძელი ტექსტი დაყო 2 სტრიქონად ისე, რომ ტექსტის სიგანე არ აღემატება ფანჯრის სიგანეს. ეს მოხდა იმიტომ, რომ მას ტექსტში შეხვდა ორი მისთვის ნაცნობი ბრძანება, რომლებიც აღწერენ ტექსტის ფორმატირებას, ეგრეთ წოდებული **HTML** ტეგები. ესენია ტექსტური აბზაცის ტეგები **<P>** და **</P>**.
ზოგად შემთხვევაში **HTML** ტეგები, წარმოადგენენ ”<” და ”>” ნიშნებს შორის მოთავსებულ ინგლისურ სიტყვებს. ამ ტეგებიდან პირველი (ჩვენ შემთხვევაში- **<P>**) განსაზღვრავს ტეგის მოქმედების არეში მოხვედრილი ტექსტის ფრაგმენტის დასაწყისს და მას **გამხსნელი ტეგი** ეწოდება. მეორე ტეგი **</P>** განსხვავდება პირველისაგან იმით, რომ მასში ”<” ნიშანსა და თვით ტეგის დასახელებას შორის დგას ნიშანი ”/”-სლეში; ის განსაზღვრავს ტექსტის ფრაგმენტის დასასრულს და მას **დამხურავი ტეგი** ეწოდება.

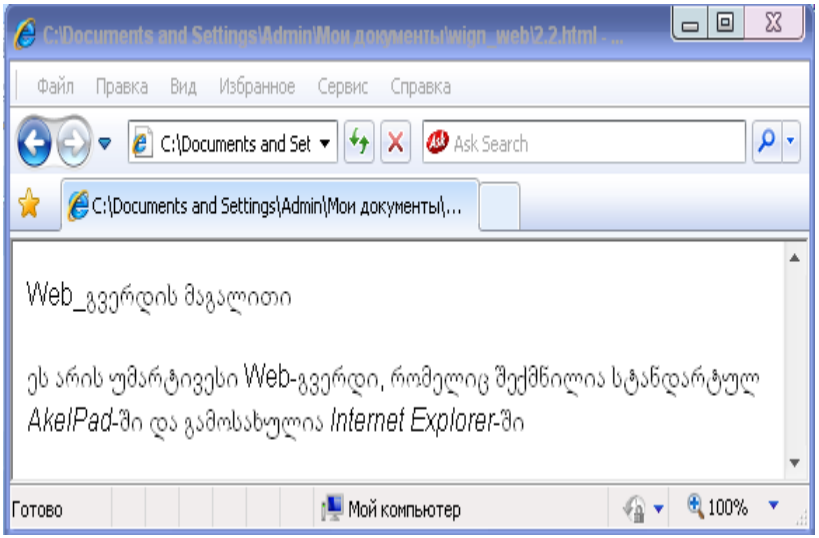
ხოლო თვით ტექსტის ფრაგმენტს რომელზეც ტეგი ახდენს გავლენას, **ტეგის შიგთავსი** ეწოდება.

HTML ტეგების აბსოლუტური უმრავლესობა წყვილ-წყვილად გამოიყენება, ანუ მათ გააჩნიათ გამხსნელი ტეგი, დამხურავი ტეგი და შიგთავსი. არსებობენ აგრეთვე, ცალცალი ტეგები, მაგალითად **<IMG>** სახის, რომლებიც მხოლოდ ერთი ტეგისაგან შედგებიან და შიგთავსიც არ გააჩნიათ, მაგრამ მათი რაოდენობ არც თუ ისე დიდია. ჩვენ მათ მოგვიანებით გავეცნობით.

ზემოთ ნათქვამიდან გამომდინარე, ჩვენ შეგვიძლია ვებ-გვერდის განმარტების ფორმულირება. მაშ ასე, **ვებ-გვერდი**-ეს არის ნებისმიერ მარტივ ტექსტურ რედოქტორში შექმნილი ჩვეულებრივი ტექსტური ფაილი (**Windows**-ის სტანდარტულ კომპლექტაციაში შემავალი უმარტივესი ტექსტური რედაქტორით-**Not-Pad**-ით ან **AkelPad**-ით შექმნილი ტექსტიც), რომელიც დამახსოვრებულია **htm** ან **html** გაფართოებით.

HTML ტეგების მანიპულირებით, ჩვენ შეგვიძლია ჩვენი პირველი ვებ-გვერდის ტექსტის ფორმატირება ჩვენივე სურვილის შესაბამისად. მაგალითად, გამოვყოთ მათში დასახელებული ორი პროგრამის დასახელებები კურსივით. ამისათვის ჩვენ დაგვჭირდება **<EM>...</EM>** წყვილი ტეგის გამოყენება. აი ასე (დამატებული ტეგები გამოყოფილია მუქი შრიფტით):

```
<P>Web_გვერდის მაგალითი</P>
<P>ეს არის უმარტივესი Web_გვერდი,რომელიც შექმნილია
სტანდარტულ <EM>AkelPad-ში</EM> და გამოსახულია <EM>
Internet Explorer-ში.</EM></P>
```



ნახ.2.3. ჩვენი პირველი ვებ-გვერდი.პროგრამების დასახელებები გამოყოფილია კურსივით

ჩვენი ქმედებების შედეგი ასახულია ნახ.2.3-ზე.

თუ ჩვენ მოგვინდება დამთვალიერებლის დასახელების **"Internet Explorer"** მუქი ფერით გამოყოფა, ჩვენ უნდა გამოვიყენოთ წყვილი ტეგი **<STRONG>...</STRONG>**. აი ასე:

```
<P>Web_გვერდის მაგალითი</P>
<P>ეს არის უმარტივესი Web_გვერდი,რომელიც შექმნილია
სტანდარტულ <EM>AkelPad-ში</EM> და გამოსახულია
<EM><STRONG>Internet Explorer-ში.</STRONG></EM></P>
```

აქ ჩვენ ტეგი **<STRONG>** ჩავალაგეთ **<EM>** ტეგში. აღნიშნულის შედეგად სიტყვები **"Internet Explorer"** მიიღებენ მუქ ფერს. ზოგადად **HTML** ტეგების ერთმანეთში ჩალაგება-ჩვეულებრივი მოვლენაა და ჩვენ რაც შეიძლება სწრაფად უნდა შევეგუოთ მას.
დამწყები ვებ დიზაინერები ხშირად ასეთ შეცდომას უშვებენ: დამხურავ ტეგებს ათავსებენ გამხსნელი ტეგების არა მკაცრად საპირისპირო მიმართულებით. ასე მაგალითად, ქვემოთ მოცემული

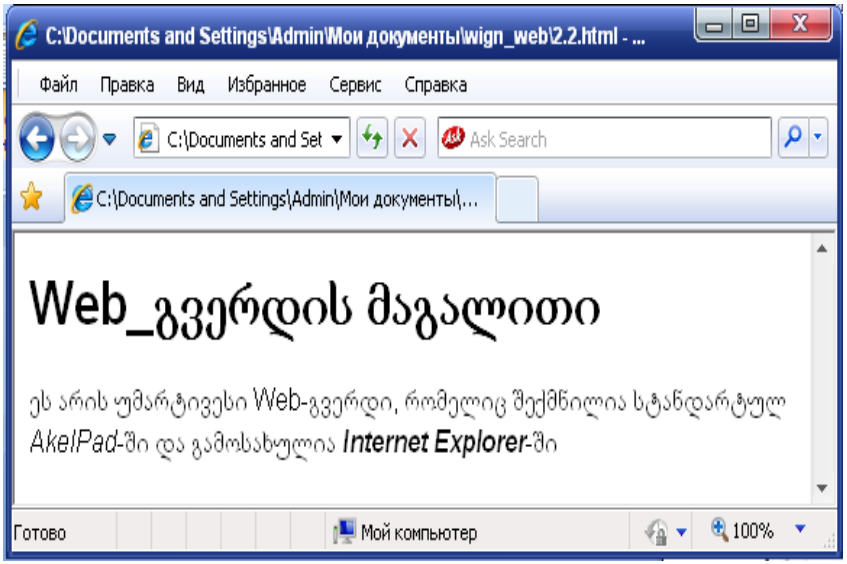
კოდი შეიცავს მსგავს შეცდომას: ტეგებმა **</EM>** და **</STRONG>** შეიცვალეს ადგილები:

<P>Web_გვერდის მაგალითი</P>
<P>ეს არის უმარტივესი Web_გვერდი,რომელიც შექმნილია
სტანდარტულ <EM>AkelPad-ში</EM> და გამოსახულია
<EM><STRONG>Internet Explorer-ში.**</EM></STRONG>**.</P>

ასეთ კოდთან შეხვედრისას ვებ-დამთვალიერებლის ქცევა არაპროგნოზირებადი ხდება (თუმცა **Internet Explorer**-ი განთქმულია ვებ-დიზაინერის მსგავსი შეცდომების შესწორების შესაძლებლობებით). ასე, რომ სამომავლოდ ღირს შემდეგი მარტივი წესის დამახსოვრება: დამხურავი ტეგები უნდა მეორდებოდნენ მათი შესაბამისი გამხსნელი ტეგების საპირისპირო მიმართულებით. დაბოლოს, მოდით ჩვენს ვებ-გვერდს მივანიჭოთ დიდი "მყვირალა" სათაური. ამისათვის ტექსტის პირველ აბზაცში შევცვალოთ ტეგი **<P>...</P>** ტეგით **<H1>…</H1>:**

<H1>Web_გვერდის მაგალითი**</H1>**
<P>ეს არის უმარტივესი Web_გვერდი,რომელიც შექმნილია
სტანდარტულ <EM>AkelPad-ში</EM> და გამოსახულია
<EM><STRONG>Internet Explorer-ში. **</STRONG></EM>**</P>

ჩვენი პირველი ვებ-გვერდის საბოლოო ვარიანტი მოცემულია ნახ.2.4-ზე. ჩანს, რომ ვებ-დამთვალიერებელმა თვითონ გამოყო **<H1>** ტეგის შიგთავსი ძალიან დიდი შრიფტით, რათა ის უფრო თვალში საცემი ყოფილიყო.



ნახ.2.4. ჩვენი პირველი ვებ-გვერდი სათაურით.

შევნიშნოთ, რომ **<H1>** ტეგს, ასეც უწოდებენ-პირველი დონის სათაურის ტეგი. რატომ პირველი? იმიტომ, რომ **HTML** სტანდარტი ითვალისწინებს სათაურების ექვს დონეს, დაწყებული ყველაზე მაღალიდან და დამთავრებული ყველაზე დაბალით. ეს სათაურები განისაზღვრებიან შემდეგი ტეგებით: **<H2>**(მეორე დონე), **<H3>**(მესამე) და ასე შემდეგ **<H6>**-მდე (მეექვსე, ყველაზე დაბალი დონეა). პირველი დონის სათაურებით, როგორც წესი განისაზღვრებიან მთლიანად ვებ-გვერდების დასახელებები, მეორე დონის სათაურებით-ვებ-გვერდების ცალკეული ნაწილების დასახელებები, და ა.შ..

HTML კოდს კიდევ ერთი უპირატესობა გააჩნია ჩვეულებრივ ტექსტთან შედარებით. ჩვეულებრივი ტექსტი, როგორც ჩვენ უკვე ვნახეთ ნახ.2.1.-ზე, გამოისახება ისევე, როგორც დაწერილია-ყველა დაშორებებითა და სტრიქონების წყვეტებით. ვებ-დამთვალიერებელმა არცკი შეიწუხა თავი, რათა გადაეტანა საკმაოდ გრძელი სტრიქონი, რომ გავეთავესუფლებინეთ საკუთარი ფანჯრის შიგთავსის დათვალიერებისას ჰორიზონტალურად გადახვევის აუცილებლობისაგან.

HTML-კოდის შემთხვევაში ვებ-დამთვალიერებელი ახდენს **<P>** ტეგის შიგთავსის ფორმირებას ისე, რომ ის დაეტიოს მის ფანჯარაში, რათა მომხმარებელს არ მოუხდეს ფანჯრის შიგთავსის გადახვევა ჰორიზონტალურად. ვებ-დამთვალიერებელი თვითონ ყოფს აბზაცს სტრიქონებად, მიუხედავად იმისა, თუ როგორ დავყავით ჩვენ იგი გვერდის **HTML**-კოდში. ამასთან სტრიქონებს შორის წყვეტებს ის აღიქვამს ჰარებად, რაც ძალზე მოსახერხებელია ჩვენთვის.
ვთქვათ, რომ ჩვენ გვინდა ჩვენი **2.2.html** გვერდის **HTML**-კოდის რედაქტირება ასეთნაირად:

<H1>Web_გვერდის მაგალითი**</H1>**
<P>ეს არის უმარტივესი Web_გვერდი,რომელიც შექმნილია
სტანდარტულ <EM>AkelPad-ში</EM> და გამოსახულია
<EM><STRONG>Internet Explorer-ში.**</STRONG></EM>**.</P>

ჩვენ დავყავით ძალიან გრძელი მეორე აბზაცი რამოდენიმე უფრო მოკლე სტრიქონებად-ასე უფრო მოხერხებულია მათი დანახვა (შემდგომშიც ასე მოვიქცეთ ხოლმე). თუ ჩვენ ამის შემდეგ გავხსნით დარედაქტირებულ გვერდს **2.2.html** ვებ-დამთვალიერებელში, დავინახავთ, რომ ჩვენი კოდის მისებურ წარმოდგენაში არაფელი არ შეცვლილა-იხილეთ ნახ.2.4.
ახლა კი მოდით ცოტა დავისვენოთ და ამასობაში რაიმე ახალიც გავიგოთ.
გასაგებია, იმისათვის რომ ერთსა და იმავე ვებ-გვერდს ვებ-დამთვალიერებლები ერთნაირად წარმოაჩენდნენ, **HTML**-ენა უნდა იყოს სტანდარტიზებული. მისი სტანდარტიზაციით (აგრეთვე ინტერნეტის მრავალი სხვა სტანდარტებით) დაკავებულია განსაკუთრებული ორგანიზაცია, რომელსაც **World Wide Web Consortium** ან შემოკლებით **WWWC** ან კიდევ უფრო ხშირად **W3C** ეწოდება. ეს დასახელება ასე იკითხება: "Web კომიტეტი".

W3C გამოსცემს საკმაოდ ვრცელ დოკუმენტებს, რომლებშიც აღწერილია **HTML** სტანდარტის სხვადასხვა ვერსიები. ამ ენის უკანასკნელი ვერსია-4.01-გამოვიდა გასული საუკუნის 90-იანი წლების ბოლოს. ყველა თანამედროვე **Web**-დამთვალიერებლები სწორად ამ ვერსიას ეფუძნებიან.

შენიშვნა: *როგორც ჩანს HTML-ის 4.01 ვერსია მართლაც უკანასკნელი იქნება. შემდგომში HTML-ენა თანდათანობით შეიცვლება თავისი შთამომავლით-XHTML (eXtensible HyperText Markup Language-ჰიპერტექსტური მონიშვნის გაფართოებული ენა). ფაქტიურად XHTML-ეს უფრო მკაცრი, სრულყოფილი და მოძველებული ტეგებისაგან გაწმენდილი ენაა. ყოველ შემთხვევაში, ტეგების ანაკრეფების მიხედვით ეს ორივე ენა ძალიან გავს ერთმანეთს. დღეისათვის XHTML-ენა ნაკლებად არის გავრცელებული და როგორც ჩანს მეტნაკლებად მნიშვნელოვან პოპულარობას არცთუ ისე მალე მიიღებს.*

გრაფიკა Web-გვერდებზე. ჩანერგილი ელემენტები

განვაგრძოთ ჩვენი ექსპერიმენტები **HTML**-ენასთან და ვებ-გვერდებთან. ამჯერად ვისაუბრებთ იმაზე, თუ როგორ მოვათავსოთ ჩვენს ვებ-გვერდზე გრაფიკული გამოსახულებები.

მაგრამ, ჯერ მოდით შემოვიღოთ კიდევ ერთი ტერმინი, რომელიც დაგვეხმარება შემდგომ მუშაობაში. ვთქვათ, რომ ტექსტის ნებისმიერ აბზაცს, მის ნებისმიერ ფრაგმენტს, რომელიც წყვილი ტეგების შიგთავსს წარმოადგენს, აგრეთვე ყველაფერს, რაც ვებ-გვერდზე თავსდება ერთეულოვანი ტეგების საშუალებით-**ვებ-გვერდის ელემენტები** ეწოდება.

ადრე ჩვენ საქმე გვქონდა მხოლოდ ვებ-გვერდების ტექსტურ ელემენტებთან. ასეთი ელემენტების მაგალითებია ტექსტების აბზაცები, სათაურები, კურსივი ფრაგმენტები და მუქი ტექსტები. მათი შექმნა ხდება ჩვენთვის უკვე ცნობილი **HTML**-კოდების საშუალებით, რომლებიც ტექსტურ რედაქტორში იწერება და ინახება ტექსტურ ფაილებში **htm(l)** გაფართოებებით. შემდგომში საუბარი გვექნება გვერდების ისეთ ელემენტებზე რომელთა კოდირება არ არის შესაძლებელი **HTML**-კოდების საშუალებით-ანუ **ჩანერგილ** ელემენტებზე. ყველა ჩანერგილი ელემენტი ინახება ცალკე ფაილში, ხოლო თვით **HTML**-კოდში მიეთითება მინიშნება საჭირო ფაილზე. **ჩანერგილი ელემენტები**, კერძოდ გრაფიკული გამოსახულებებიც შეიძლება იყონ. სინამდვილეში გარაფიკული ინფორმაცია არ შეიძლება შენახული იქნას ტექსტური სახით; ნახატების, ფოტოსურათების, სქემების შესანახად მოგონილია სპეციალური ფორმატები-**GIF**, **JPEG**, **BMP**, **TIFF** და სხვა. მათთან სამუშაოდ ტექსტური რედაქტორები არ გამოდგება-საჭიროა სპეციალური პროგრამები-**გრაფიკული რედაქტორები.**

ვთქვათ გვაქვს რაიმე სურათი და გვინდა მისი ჩასმა ჩვენს ვებ-გვერდზე. რა უნდა გავაკეთოთ ამისათვის?

უპირველეს ყოვლისა, ჩვენ უნდა განვსაზღვროთ ვებ-გვერდის რა ადგილას უნდა მოთავსდეს სურათი და მოვძებნოთ **HTML**-კოდის შესაბამისი ადგილი. ამის შემდეგ საჭიროა კოდის იმ ადგილას ჩაისვას განსაკუთრებული ერთეულოვანი კოდი **<IMG>**:

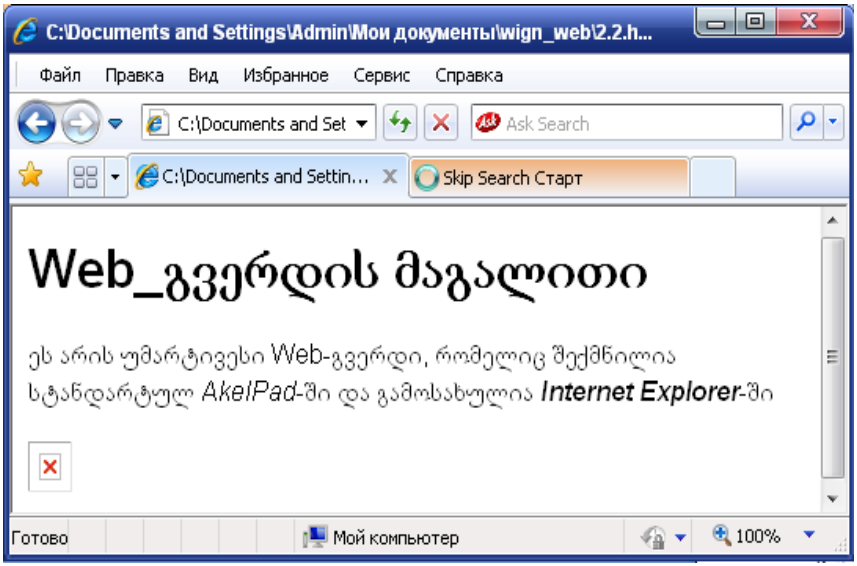
```
<H1>Web_გვერდის მაგალითი</H1>
<P>ეს არის უმარტივესი Web_გვერდი,რომელიც შექმნილია
სტანდარტულ <EM>AkelPad-ში</EM> და გამოსახულია
<EM><STRONG>Internet Explorer-ში.</STRONG></EM>.</P>
<IMG SRC="exclam.gif">
```

ამ ტეგის დანახვისას, ვებ-დამთვალიერებელი გაუგზავნის **Web**-სერვერს კიდევ ერთ მოთხოვნას-მოძებნოს ფაილი **exclam.gif**. ამ ფაილის მიღებისთანავე, **Web**-დამთვალიერებელი მის შიგთავსს გადააგზავნის ვებ-გვერდის იმ ადგილას სადაც დგას ტეგი **<IMG>** (იხილეთ ნახ.2.5)

აქ ჩვენ წავაწყდით ეგრეთ წოდებულ **ტეგის ატრიბუტებს**-ტეგების განსაკუთრებულ დამატებით პარამეტრებს, რომლებიც განსაზღვრავენ ზოგიერთ დამატებით პირობებს ან მნიშვნელობებს. ჩვენს შემთხვევაში ტეგი **<IMG>** შეიცავს **SRC** ატრიბუტს, რომელსაც მინიჭებული აქვს მნიშვნელობა **exclam.gif**, იმ ფაილის სახელი, რომელიც ჩვენ გვჭირდება. (ყველა მნიშვნელობა იწერება ორმაგ ბრჭყალებში "...").

ატრიბუტები არსებობენ **სავალდებულო** და **არასავალდებულო**. ტეგი სავალდებულო ატრიბუტს აუცილებლად უნდა შეიცავდეს. ტეგში არასავალდებულო ატრიბუტის არსებობა არ არის აუცილებელი; თუ კი ის არ არის მითითებული, ვებ-დამთვალიერებელი თვლის, რომ მას რაიმე მნიშვნელობა წინასწარ აქვს მინიჭებული. **<IMG>** ტეგის **SRC** ატრიბუტი სავალდებულოა. იმისათვის, რომ ვებ-გვერდზე დავსვათ გამოსახულება, მას უნდა მიეთითოს თუ სად მდებარეობს გამოსახულება. არასავალდებულო ატრიბუტის ტიპიური მაგალითია იმავე **<IMG>** ტეგის **ALT** ატრიბუტი, რომელიც განსაზღვრავს ვებ-გვერდზე იმ შემთხვევაში გამოსატან ტექსტს, თუ რაიმე მიზეზის გამო ვერ ხერხდება გამოსახულების შემცველი ფაილის ჩატვირთვა (**ტექსტი-შემცვლელი**).

<IMG SRC=”exclam.gif” ALT “აქ უნდა იყოს logo>



ნახ.2.5. ჩვენი პირველი გამოსახულებიანი Web-გვერდი

Web-გვერდებზე ხდება ისეთი გრაფიკული გამოსახულებების გამოყენება, რომლებიც ქვემოთ ჩამოთვლილი ფორმატებიდან რომელიმე ერთ-ერთის საშუალებით არიან წარმოდგენილი მეხსიერებაში: **GIF**, **JPEG**, **BMP**. მოდით მოკლედ განვიხილოთ ისინი. ფორმატი **GIF (Graphic Interchange Format)**, გრაფიკის ურთიერთ გაცვლის ფორმატი). მშვენიერი ფორმატია შტრიხული გამოსახულებების შესანახად, რომლებიც სხვადასხვა სიგრძის ხაზების, ფორმების და ფერებისაგან შედგებიან. ამიტომ მას იყენებენ ვებ-გვერდების გაფორმებასთან დაკავშირებული ელემენტების (ხაზები, სიების მარკერები და ა.შ.), ყოველგვარი სქემების, ნახაზების, ფანქრით ნახატების და ა.შ. შესანახად. გარდა ამისა, ფორმატ **GIF**-ს გააჩნია ერთი ძალიან საინტერესო თვისება-მისი საშუალებით შესაძლებელია ერთ ფაილში მთელი ფილმის შენახვა (ანიმირებული **GIF**-ი), რაც შეუცვლელი თვისებაა რეკლამებისათვის გამოსაყენებლად.

ფორმატი **JPEG (Joint Pictures Encoding Group**-უძრავი გამოსახულებების კოდირების ჯგუფი**),** წარმატებით გამოიყენება ნახევარტონალური გამოსახულებების შესანახად, რომლებიც შედგებიან სხვადასხვა ფერის მრავალი წერტილებისაგან. ამიტომაც სურათებსა და სკანირებულ ფოტოებს სწორად ამ ფორმატში ინახავენ.

ფორმატი **PNG (Portable Network Graphics**, გადაადგილებადი ქსელური გრაფიკა), როგორც მისი შემქმნელები ამტკიცებენ, აერთიანებს **GIF** და **JPEG** ფორმატების შესაძლებლობებს და თავისუფალია მათთვის დამახასიათებელი ნაკლოვანებებისაგან. უნდა აღინიშნოს, რომ ჩანაფიქრი განხორციელდა წარმატებით: **PNG**-ში შესაძლებელია შტრიხული და ნახევარტონალური გამოსახულებების დამახსოვრება ხარისხის დაკარგვის გარეშე. მიუხედავად ამისა მან, ჯერ-ჯეროზით ვერ მოიპოვა დამსახურებული პოპულარობა.

შენიშვნა. *ინტერნეტში ზემოთ ჩამოთვლილი სამის გარდა გრაფიკის სხვა ფორმატებიც გამოიყენება. კერძოდ, ძალიან პოპულარულია ფორმატი Shockwave/Flash, რომელიც ფირმა Macromedia-ს მიერ არის შექმნილი და სტატიკური გამოსახულებების გარდა, ასევე ფილმებისა და მთელი პროგრამების შექმნის საშუალებას იძლევა. მაგრამ GIF, JPEG, BMP-საგან განსხვავებით ვებ-დამთვალიერებლის მიერ არ ხდება ამ ფორმატების პირდაპირი მხარდაჭერა და ისინი მოითხოვენ დამატებითი პროგრამების გამოყენებას, ხოლო მათი ვებ-გვერდებზე განთავსებისათვის სხვა ტეგები გამოიყენება.*

ჰიპერმინიშნებები

ადრე ნათქვამი იყო, რომ ვებ-საიტი-ეს არის ერთმანეთთან დაკავშირებული ვებ-გვერდების ერთობლივობა. მაგრამ არ იყო არაფერი ნათქვამი იმის შესახებ, თუ როგორ უკავშირდებიან ისინი ერთმანეთს. დროა გავარკვიოთ ეს საკითხიც.

აღნიშნული მიზნით იყენებენ **ჰიპერმინიშნებებს**- განსაკუთრებულ კავშირებს, რომელთაც გადავყავართ ერთი ვებ-გვერდიდან მეორეზე. მათ განსაკუთრებული წესით გამოყოფილი (როგორც წესი ფერითა და ხაზის გასმით) ტექსტის ფრაგმენტებს სახე აქვთ. თუ ასეთ ჰიპერმინიშნებაზე დავაწკაპუნებთ ”მაუსით”, ვებ-დამთვალიერებელი ჩატვირთავს სხვა ვებ-გვერდს, რომლის მისამართიც ჰიპერმინიშნების პარამეტრებშია მითითებული.

ჰიპერმინიშნებების შექმნა ხდება სპეციალური წყვილი ტეგის **<A>**-საშუალებით და ასეთი სახე გააჩნიათ:

<A HREF=”page125.html”>გვერდიN125</A>

ჩანს, რომ გვერდის ინტერნეტ-მისამართი, რომელზედაც უნდა გადაგვიყვანოს ჰიპერმინიშნებამ, განისაზღვრება **HREF** ატრიბუტის საშუალებით. მოცემულ შემთხვევაში ჰიპერმინიშნებას მივყავართ **page125.html** გვერდზე, რომელიც იგივე საქაღალდეშია მოთავსებული, რაც მიმდინარე გვერდი (მოცემულ მომენტში ვებ-დამთვალიერებელში გახსნილი გვერდი).
თუ საჭიროა დაშვების მიღება ვებ-გვერდზე, რომელიც მოთავსებულია სხვ საქაღალდეში, ჰიპერმინიშნების კოდს ექნება ასეთი სახე:

<A HREF=”/folder1/page126.html”>გვერდიN126</A>

თუ საჭიროა ვებ-გვერდი, რომელიც საერთოდ სხვა საიტზე მდებარეობს, მაშინ ინტერნეტ-მისამართი უნდა შეიცავდეს მის დომენურ სახელს (ან **IP**-მისამართს):
<A
HREF=”www.othersite.ru/folder1/page3.html”>გვერდიN3
</A>
ახლა მოდით ჩავატაროთ პრაქტიკული ექსპერიმენტები ჰიპერმინიშნებებზე. ფაილ **2.2.html**-ში შენახული გვერდის **HTML**-კოდში ჩავამატოთ მუქი შრიფტით გამოყოფილი სტრიქონი:

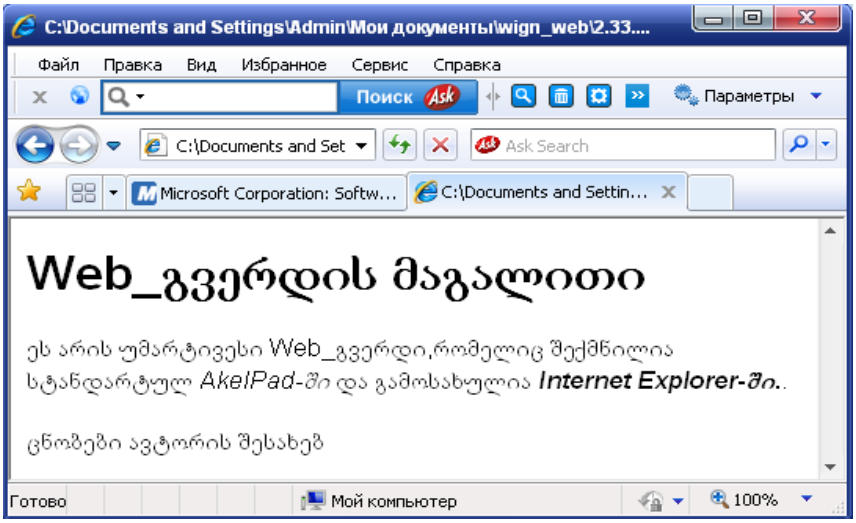
<H1>Web_გვერდის მაგალითი</H1>
<P>ეს არის უმარტივესი Web_გვერდი,რომელიც შექმნილია
სტანდარტულ <EM>AkelPad-ში</EM> და გამოსახულია
<EM><STRONG>Internet Explorer-ში.</STRONG></EM>.</P>
<P><A HREF=”2.4.html”></A>ცნობები ავტორის შესახებ</P>

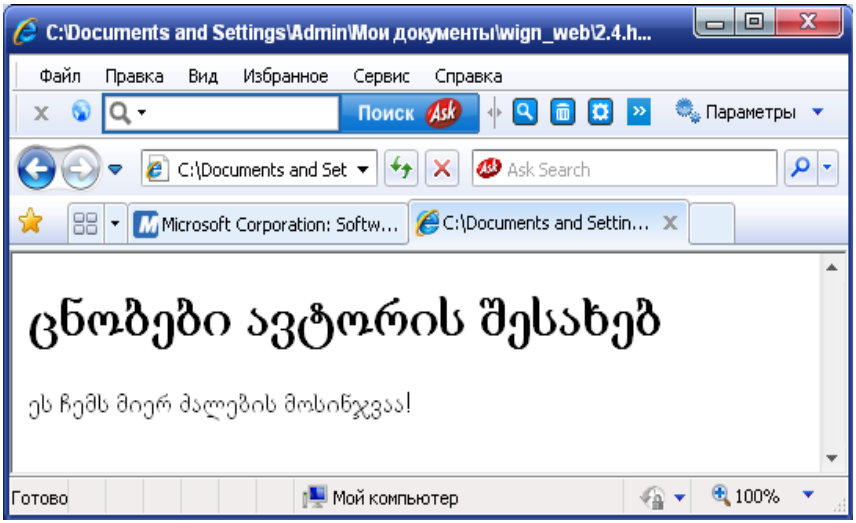
შევინახოთ ახალი გვერდი სხვა ფაილში სახელწოდებით **2.3.html**.
ახლა შევქმნათ გვერდი **2.4.html**. მისი **HTML**-კოდი ძალიან მარტივია:

<H1>ცნობები ავტორის შესახებ</H1>
<P>ეს ჩემს მიერ ძალების მოსინჯვაა!</P>

ამის შემდეგ გავხსნათ ვებ-დამთვალიერებელში ფაილი **2.3.html** და დავაწკაპუნოთ ჰიპერმინიშნებაზე **ცნობები ავტორის შესახებ**. ვებ-დამთვალიერებლის ფანჯარაში გაჩნდება გვერდი ცნობებით ავტორის შესახებ, რომელიც **2.4.html** ფაილშია შენახული.
ახლა კი ჩავატაროთ პატარა ფოკუსი. შევცვალოთ ჰიპერმინიშნების კოდი, **2.3.html** გვერდის **HTML**-კოდში ასეთნაირად (ცვლილებები გამოყოფილია მუქი შრიფტით):

<P><A HREF="2.4.html" TARGET="_blank">ცნობები ავტორის შესახებ</A></P>





ნახ.2.6. ვებ-დამთვალიერებლის ორი ფანჯარა, რომლებშიც ჩატვირთულია სხვადასხვა ვებ-გვერდები.

ჩვენ **<A>** ტეგში მოვათავსეთ არასავალდებულო ატრიბუტი **TARGET**, რომელიც განსაზღვრავს ჰიპერმინიშნების **მიზანს**. მიზანი კი განსაზღვრავს, თუ სად უნდა იქნას გამოტანილი ვებ-გვერდი რომელზეც მიუთითებს ჰიპერმინიშნება. თუ ამ ატრიბუტს მივანიჭებთ მნიშვნელობას**_blank**, ვებ-გვერდი გამოტანილი იქნება ვებ-დამთვალიერებლის ცალკე ფანჯარაში. იმისათვის რომ განვსაზღვროთ ჰიპერმინიშნების ჩვეულებრივი ქცევა (ახალი გვერდი გამოიტანება იგივე ფანჯარაში, რაც მიმდინარე გვერდი) საკმარისია ატრიბუტ **TARGET**-ს მივანიჭოთ წინასწარ განსაზღვრული მნიშვნელობა**_self** ან სულაც მოვაშოროთ ეს ატრიბუტი ჰიპერმინიშნებას.

თუ ახლა შევინახავთ შესწორებულ ვებ-გვერდს **2.3.html**, შემდეგ ხელახლა გავხსნით მას ვებ-დამთვალიერებელში და დავაწკაპუნებთ ჰიპერმინიშნებაზე ***ცნობები ავტორის შესახებ,*** ვებ-დამთვალიერებელი გახსნის ახალ ფანჯარას და ჩატვირთავს მასში **2.4.html** გვერდს. ორივე ეს ფანჯარა ნაჩვენებია ნახ.2.6.-ზე. მომავალში ჩვენ კიდევ დავუბრუნდებით ჰიპერმინიშნებებს, როდესაც შევუდგებით საკუთარი საიტის შექმნას. ჯერ კი მოდით

გავერკვეთ იმ წესებში, რომელთა შესაბამისადაც უნდა ხდებოდეს ვებ-გვერდების გაფორმება.

სწორად გაფორმებული ვებ-გვერდები

მოდით, კიდევ ერთხელ გადავავლოთ თვალი ჩვენი პირველი ვებ-გვერდის **HTML**-კოდს.

```
<H1>Web_გვერდის მაგალითი</H1>
<P>ეს არის უმარტივესი Web_გვერდი,რომელიც შექმნილია
სტანდარტულ <EM>AkelPad-ში</EM> და გამოსახულია
<EM><STRONG>Internet Explorer-ში.</STRONG></EM>.</P>
```

ის ძალზე კომპაქტურია, კორექტულად აისახება როგორც **Internet Explorer**-ში, ასევე სხვა ვებ-დამთვალიერებლებში (იგი შემოწმებულია **Firefox**-შიც და ნორმალურად მუშაობს). მიუხედავად ამისა ჩვენს პირველ ვებ-ქმნილებას შეიძლება ეწოდოს მხოლოდ **კორექტული ვებ-გვერდი.** კორექტული გვერდები სწორად აისახებიან ვებ-დამთვალიერებლებში, მაგრამ არ აკმაყოფილებენ ყველა იმ მოთხოვნებს, რომლებსაც **W3C**-სტანდარტები უყენებენ ვებ-გვერდებს.

რა უნდა გაკეთდეს იმისათვის, რომ ჩვენი გვერდი აკმაყოფილებდეს ყველა სტანდარტებს, ანუ სხვანაირად რომ ვთქვათ, იყოს **სწორად გაფორმებული.** ამისათვის საჭიროა, რომ მის კოდს დაემატოს რამოდენიმე ახალი ტეგი, რომლებსაც უხილავი ტეგები ეწოდებათ. უხილავი ტეგების შიგთავსები, ხილული ტეგების შიგთავსებისაგან განსხვავებით, არანაირად არ აისახებიან ვებ-დამთვალიერებლების ფანჯრებში, მაგრამ გავლენას ახდენენ მთელი გვერდის ასახვაზე.
ქვემოთ მოყვანილია სწორად გაფორმებული ვებ-გვერდის **HTML**-კოდი (დამატებული ფრაგმენტები ტრადიციულად მუქი შრიფტით არიან გამოყოფილი):

```
<H1>Web_გვერდის მაგალითი</H1>
<P>ეს არის უმარტივესი Web_გვერდი,რომელიც შექმნილია
სტანდარტულ <EM>AkelPad-ში</EM> და გამოსახულია
<EM><STRONG>Internet Explorer-ში.</STRONG></EM>.</P>
</BODY>
</HTML>
```

მოდით, შემდგომი გამოყენებისათვის შევინახოთ ეს გვერდი ფაილში სახელწოდებით **2.5.html** და ყურადღებით დავაკვირდეთ მის **HTML**-კოდს.

შევნიშნავთ, რომ ჩვენი საწყისი კოდი მოთავსებულია რაღაცნაირ უცნაურ წყვილ ტეგში **<BODY>**. ეს ტეგი განსაზღვრავს ვებ-გვერდის ეგრეთ წოდებული **სხეულის სექციას**-სახელდობრ მის შიგთავსს, რომელიც ეკრანზე გამოდის. ყველაფერი, რისი დანახვაც ჩვენ გვსურს ვებ-დამთვალიერებლის ფანჯარაში, მოთავსებული უნდა იყოს ამ სექციაში-ანუ ტეგში **<BODY>**.

სხეულის სექციის გარდა, ნებისმიერი თავისითავის პატივისმცემელი ვებ-გვერდი ასევე უნდა შეიცავდეს წყვილი ტეგით **<HEAD>**-ით განსაზღვრულ **სათაურის სექციას,** სადაც გარკვეული სახის დამხმარე ინფორმაცია უნდა იყოს მოთავსებული. (იგი არ უნდა შეგვეშალოს ჩვეულებრივ ტექსტურ სათაურში, რომელიც **<H1>...<H6>!** ტეგებიდან ერთ-ერთით განისაზღვრება). ეს ინფორმაცია ვებ-დამთვალიერებლის მიერ ეკრანზე კი არ გამოიტანება, არამედ გამოიყენება მის მიერ შიდა მოხმარებისათვის, კერძოდ განსაზღვრავს ვებ-გვერდის ზოგიერთ პარამეტრს.

ჩვენს შემთხვევაში ზემოთ აღნიშნული დამხმარე ინფორმაციიდან გვაქვს მხოლოდ ვებ-გვერდის დასახელება, რომელიც წყვილი ტეგით **<TITLE>** არის განსაზღვრული. ეს დასახელება ვებ-დამთვალიერებელს გამოაქვს თავისი ფანჯრის სათაურში; გარდა ამისა იგი შეიტანება ვებ-დამთვალიერებლის ისტორიაში-მის მიერ ადრე დათვალიერებული ვებ-გვერდების სიაში. ვებ-გვერდის დასახელების გარდა, სათაურის სექცია ხშირ შემთხვევაში, შეიცავს ვებ-დამთვალიერებლისათვის საჭირო სხვა მრავალ მონაცემს. ჩვენ მათ მოგვიანებით განვიხილავთ.

დაგვრჩა სათქმელი მხოლოდ ის, რომ ვებ-გვერდის მთელი **HTML**-კოდი-როგორც სათაურის სექცია, ისე სხეულის სექცია-მოთავსებული უნდა იყოს წყვილი ტეგის **<HTML>**-ის შიგნით.

HTML-ტეგების იერარქია.

HTML-ენის შესახებ სულ ცოტა რამ დაგვრჩა სათქმელი. მაგრამ სანამ ვიტყოდეთ, შემოვიღოთ კიდევ რამოდენიმე ტერმინი, რომლებიც შემდგომში აუცილებლად დაგვჭირდება.

ამ თავის დასაწყისში ჩვენ ვსაუბრობდით ტეგების ერთმანეთში ჩალაგებულობის შესახებ. ადრე განხილული **<HTML>** კოდი-ასეთი

ჩალაგებულობის კარგი მაგალითია. ერთმანეთში ჩალაგებული ტეგები საკმაოდ რთულ სტრუქტურას, ან როგორც პროფესიონალი პროგრამისტები ამბობენ **იერარქიას** ქმნიან. ასე მაგალითად, ტეგი **<STRONG>** ჩალაგებულია ტეგ **<EM>**-ში, ტეგი **<EM>**-ტეგ **<P>**-ში, ტეგი **<P>**-ტეგ **<BODY>**-ში, ტეგი **<BODY>**-ტეგ **<HTML>**-ში. ეს მხოლოდ უმარტივესი ვებ-გვერდია-უფრო რთულ ვებ-გვერდებზე იერარქია გაცილებით უფრო რთულია.
ამიტომ, მათში უკეთ გარკვევის მიზნით, **<HTML>**-კოდში, ჩალაგებული ტეგები, როგორც ზემოთ იყო ნაჩვენები, ხშირად ერთმანეთისაგან დაშორებულად იწერებიან. ხანდახან ასეთნაირადაც წერენ:

<HTML>
    <HEAD>
        <TITLE>
    <BODY>
        <H1>
        <P>
        <EM>
        <STRONG>

ანუ, აცლიან ხელშემშლელ დამხურავ ტეგებს და მათ შიგთავსებს.
ახლა კი-შეპირებული ახალი ტერმინები. ზემოთ ნაჩვენებ სქემაზე, ტეგებს შორის დაშორების სიდიდე განსაზღვრავს ამა თუ იმ ტეგის **ჩალაგებულობის დონეს.** ასე მაგალითად, **<HTML>**-ტეგს გააჩნია ჩალაგებულობის ნულოვანი დონე, ვინაიდან საერთოდ არსად არ არის ჩალაგებული, ტეგ **<BODY>**-ის ჩალაგებულობის პირველი დონე, ხოლო ტეგ **<H1>**-ს მეორე და ა.შ..
ჩალაგებულობის ზედა (წინა) დონეების ტეგებს უწოდებენ **მშობლიურ ან მშობელ ტეგებს,** ხოლო მომდევნო დონეების ტეგებს-**შვილობილ ან შთამომავალ** ტეგებს. მაგალითად, **<HEAD>** ტეგისათვი მშობელი ტეგი იქნება **<HTML>** ტეგი, ხოლო შვილობილი ტეგი-**<TITLE>. <BODY>** ტეგისათვის მშობელი ტეგი იქნება **<HTML>**, ხოლო **შვილობილები-<H1>,<P>,<EM>** და **<STRONG>.**

მაშ ასე, დავასრულეთ. **<HTML>**-ენის საწყისებს ჩვენ უკვე გავეცანით. გადავიდეთ უფრო რთულ საკითხებზე, კერძოდ ვებ-გვერდების

გაფორმების საშუალებებზე. ამისათვის ჩვენ გამოვიყენებთ სხვა ენას და სხვა ტექნოლოგიას, რომელსაც **CSS** ტექნოლოგია ეწოდება.

Web -გვერდების გაფორმების საშუალებები

კარგია! ჩვენ უკვე შევქმენით ჩვენი პირველი ვებ-გვერდი და საკმარისად დავტკბით მისი ხილვით. ახლა კი გვსურს ცოტათი მაინც მისი გალამაზება-ფერების დამატება, შრიფტების შეცვლა და ა.შ.. რა საშუალებების გამოყენება შეგვიძლია ამისათვის?

სტილების კასკადური ცხრილები (CSS).

ვებ-გვერდის, საკუთარი სურვილის მიხედვით გაფორმების, ყველაზე თანამედროვე საშუალება დღეისათვის-ეს არის ეგრეთწოდებული **სტილების კასკადური ცხრილების** (ან უბრალოდ სტილების ცხრილები) გამოყენება. ინგლისურად მათ **CSS (Cascading Style Sheet)** ეწოდებათ.

სტილების კასკადური ცხრილების შესაქმნელად გამოიყენება არა **HTML**, არამედ სხვა ენა, რომელსაც ასევე ეწოდება-**CSS**. ჩვენ ახლა სწორედ მას გავეცნობით.

სტილების შექმნა

ვთქვათ, რომ ჩვენ გვინდა მუქი შრიფტით გამოყოფილი მთელი ტექსტის, ანუ ყველა **<STRONG>** ტეგების შიგთავსის მწვანე ფრად გამოყოფა. ამისათვის საკმარისი იქნება ჩვენი **2.5.html** ვებ-გვერდის სათაურის სექციაში კოდის ასეთი ფრაგმენტის ჩაწერა (გამოყოფილია მუქი შრიფტით):

```
<HEAD>
   <STYLE>
      STRONG { color: green; }
   </STYLE>
</HEAD>
```

ახლა, ჩვენ თუ გავხსნით ამ გვერდს ვებ-დამთვალიერებელში, დავინახავთ, რომ სიტყვა **“Microsofti”** გამწვანდა. მაშ, რა გავაკეთეთ ჩვენ? და რა არის ის უცნაური ტექსტი, რომელიც მოთავსებულია წყვილ ტეგში **<STYLE>**.

სწორედ ეს არის სტილების ცხრილი. მოცემულ შემთხვევაში იგი მდებარეობს იმავე ფაილში, რაც თვით ვებ-გვერდი და ამიტომ მას **შიდა სტილების ცხრილი** ეწოდება.
სტილების ჩვენი პირველი ცხრილი შეიცავს მხოლოდ ერთ სტილს-გვერდის რომელიმე ელემენტის აღწერის ფორმატს.
ჩვენი ეს სტილი ცვლის **<STRONG>** ტეგის შიგთავსის გარეგნულ სახეს და მას ასეც ეწოდება **ტეგის ხელახლა განსაზღვრის** (შეცვლის) ტეგი.
ყოველი სტილი შედგება **დასახელებისა** და **აღწერისაგან**. დასახელება ცალსახად განსაზღვრავს სტილს და ჩვენს შემთხვევაში ემთხვევა თვით ტეგის სახელწოდებას. **სტილის აღწერა** იწერება სახელის შემდგომ ფიგურულ ფრჩხილებში და შეიცავს სტილის **ატრიბუტების** ანაკრებს და მათ მნიშვნელობებს. ატრიბუტსა და მის მნიშვნელობას შორის უნდა იდგეს ორწერტილის ნიშანი, ხოლო წყვილი "ატრიბუტი-მნიშვნელობა" ერთმანეთისაგან წერტილ-მძიმის ნიშნით უნდა გამოიყოს.
ჩვენი ერთად-ერთი სტილის აღწერა შეიცავს ერთად-ერთ ატრიბუტს **color**(ტექსტის ფერი), რომელსაც მინიჭებული აქვს მნიშვნელობა **green** (მწვანე). ჩვენ შეგვიძლია დავუმატოთ სტილის აღწერაში კიდევ ერთი ატრიბუტი, ვთქვათ **text-decoration** (ტექსტის დამატებითი "შელამაზებისათვის"), რომელსაც ექნება მნიშვნელობა **underline** (ხაზგასმული), აი ასე:

```
<HEAD>
   <STYLE>
      STRONG { color: green;
text-decoration: underline; }
   </STYLE>
</HEAD>
```

ახლა ვნახოთ, თუ რას გვიჩვენებს ვებ-დამთვალიერებელი. მართლაც, იმუშავა!

ყურადღება! *ვებ-გვერდების შემუშავებისას თავი უნდა ავარიდოთ ტექსტების ხაზგასმას. პრაქტიკულად ყველა ვებ-დამთვალიერებლები ხაზგასმულად უჩვენებენ ჰიპერმინიშნებებს და თუ ვებ-გვერდი შეიცავს ხაზგასმული ტექსტების სხვა*

ფრაგმენტებსაც, რომლებიც არ წარმოადგენენ ჰიპერ-მინიშნებებს, საიტის დამთვალიერებლები დაიბნევიან.

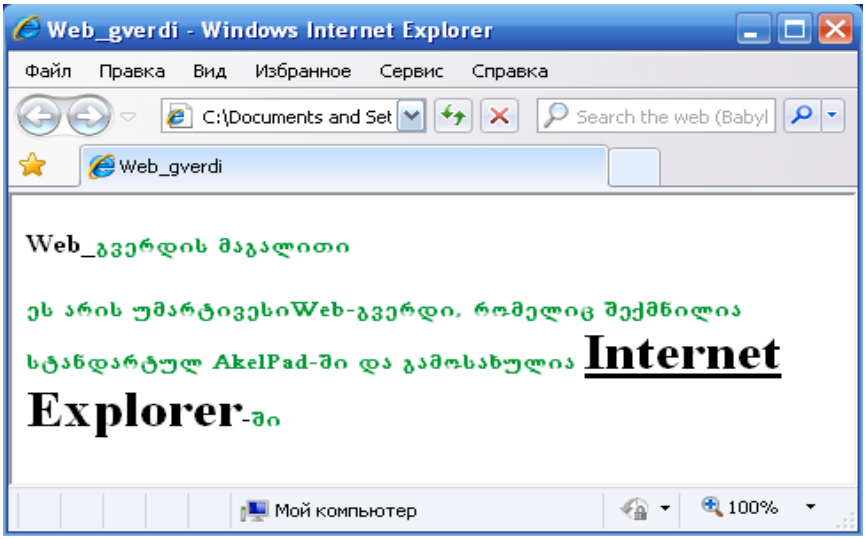
კი, მაგრამ, თუ ჩვენ გვინდა რაიმე ტეგში მოთავსებული ტექსტური ფრაგმენტის გაფორმება, მაშინ როგორ უნდა მოვიქცეთ? ამისათვის უნდა შეიქმნას სხვა ტიპის სტილი, რომელსაც **სტილურ კლასს უწოდებენ.** სტილური კლასი შეიძლება გამოყენებული იქნას ნებისმიერი **HTML** ტეგის მიმართ.
დავუშვათ, რომ ჩვენ გვინდა შრიფტის შეცვლა, რომლითაც აკრეფილია ტექსტი **"Internet Explorer"**-ი (**<EM>** ტეგის შიგთავსი). ამისათვის **2.5html** ვებ-გვერდის სტილების ცხრილში, ჩვენ უნდა შევქმნათ სტილური კლასი **bigfont**:

.bigfont { font-size: 14pt; }

სტილური კლასის სახელი ყოველთვის უნდა იწყებოდეს წერტილით-ეს მნიშვნელოვანია. რაც შეეხება ატრიბუტს **font-size**, იგი განსაზღვრავს შრიფტის ზომას. ხოლო მნიშვნელობა 14pt აღნიშნავს 14 პუნქტის ტოლი შრიფტის ზომას.
ახლა საჭიროა როგორმე მივაბათ ჩვენს მიერ შექმნილი სტილური კლასი **<EM>**-ტეგს, რომელიც შეიცავს ტექსტს **"Internet Explorer"**. ამისათვის გამოვიყენოთ ყველა ტეგის მიერ გამოყენებადი ატრიბუტი **CLACC**, რომელსაც პარამეტრის სახით მიეთითება საჭირო კლასის სტილის დასახელება წერტილის გარეშე. დავამატოთ იგი ტეგში **<EM>**, როგორც ქვემოთ არის ნაჩვენები:

<EM CLASS="bigfont"><STRONG>Internet Explorer</STRONG></EM>

დავიმახსოვროთ შესწორებული ვებ-გვერდი **2.5.html** და გავხსნათ იგი ვებ-დამთვალიერებელში. ჩვენ დავინახავთ იმას, რაც ნაჩვენებია ნახ. 2.7-ზე.



ნახ. 2.7. ჩვენი ვებ-გვერდი სტილების გამოყენების შემდეგ.

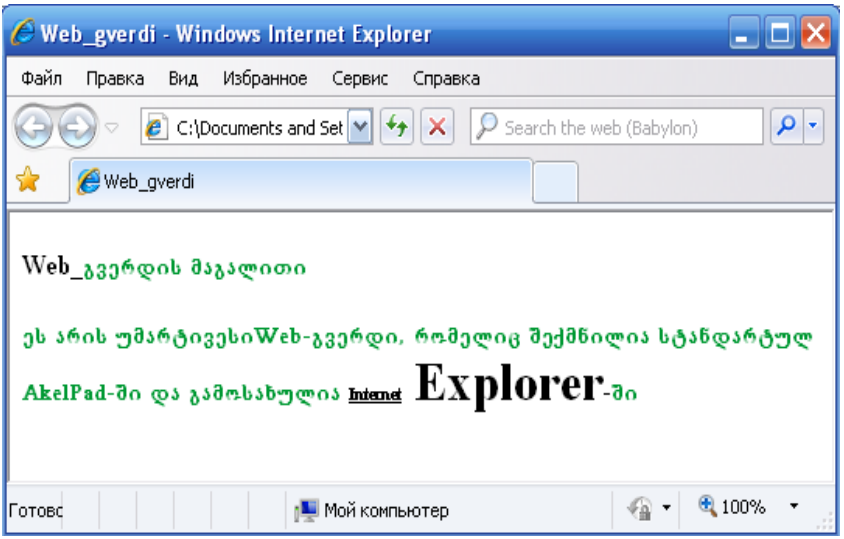
ჩანს, რომ ტექსტი **"Internet Explorer"** გაიზარდა ზომებში, რაც იმას ნიშნავს, რომ სტილური კლასი **bigfont** მოქმედებს. ამასთან ერთად, აგრეთვე აგრძელებს მოქმედებას ჩვენს მიერ ადრე შექმნილი **<STRONG>** ტეგის გარდაქმნის (ხელახლა განსაზღვრის) სტილი; სიტყვები **"Internet Explorer"**, გარდა იმისა, რომ გაიზარდნენ ზომებში, მათ შეინარჩუნეს მწვანე ფერი და დარჩნენ ხაზგასმული. გამოდის, რომ მოხდა სტილების თითქოს ერთმანეთზე დადება. (წინსწრებით უნდა აღინიშნოს, რომ ეს კასკადურობის წესის მოქმედების შედეგია, რომლის შესახებაც ჩვენ მოგვიანებით გვექნება საუბარი).

განვაგრძოთ ჩვენი ექსპერიმენტები სტილებზე. მოდით ცოტათი შევცვალოთ **<STRONG>** ტეგის გარდაქმნის სტილი, ასეთნაირად:

```
STRONG { color: green;
text-decoration: underline;
font-size: 9pt; }
```

აქ ჩვენ მუქი ტექსტისათვის განვსაზღვრეთ 9 პუნქტის ტოლი შრიფტის ზომა. ახლა, თუ გავხსნით ჩვენს მრავალტანჯულ **2.5.html**

გვერდს ვებ-დამთვალიერებელში, დავინახავთ იმას, რაც ნაჩვენებია ნახ. 2.8.-ზე.



ნახ. 2.8. სტილების კასკადურობის დემონსტრაცია

ახლა სიტყვები **"Internet Explorer"** გარდა მწვანე ფერისა და ხაზგასმულობისა, გამოირჩევიან კიდევ შრიფტის შემცირებული ზომით. გამოდის, რომ **<STRONG>**-ტეგის გარდაქმნის სტილში ჩვენს მიერ გამოყენებულმა ატრიბუტმა **font-size**, გააუქმა იმ ატრიბუტის მოქმედება **bigfont** სტილურ კლასში, რომელიც მიბმული იყო მშობელ ტეგზე **<EM>.**
მაგრამ, ამასთან ერთად, სტილური კლასის-**bigfont**-ის დანარჩენი მითითებები, გადავიდნენ **<STRONG>**-ტეგის შეცვლის სტილში. (სწორედ ეს არის სტილების ცხრილების კასკადურობა, რომლის შესახებაც შემდგომში ვისაუბრებთ).
CSS სტანდარტი უშვებს ტეგების შეცვლის ასეთი სტილების შექმნასაც:

EM STRONG { color: green: }

მსგავსი სტილი გამოყენებული იქნება **<EM>** ტეგში ჩალაგებული **<STRONG>** ტეგის მიმართ. სხვა ტეგების მიმართ ის არ იქნება

გამოყენებული. ასე რომ, ქვემოთ მოყვანილ მაგალითში სიტყვა "მწვანე" გახდება მწვანე, ხოლო დანარჩენი ტექსტი დარჩება შავად.

<EM>კურსივი<STRONG>მწვანე</STRONG></EM>

შესაძლებელია ტეგის შეცვლის სტილისა და სტილური კლასის კომბინირება გიბრიდული სტილის შექმნით. აი ასე:

EM.Bigfont { font-size: 14pt }

მსგავსი სტილი შესაძლებელია გამოყენებულ იქნას მხოლოდ <EM> ტეგის მიმართ, რომელსაც გააჩნია ატრიბუტი CLASS, რომლის მნიშვნელობაც ტოლია bigfont-ის. სხვა ტეგების მიმართ ეს სტილი გამოყენებული არ იქნება. ასე რომ, ქვემოთ მოყვანილ მაგალითში სიტყვები "დიდი ტექსტი" აკრეფილი იქნება გადიდებული შრიფტით (14 პუნქტი), ხოლო სიტყვები "ჩვეულებრივი ტექსტი"- ჩვეულებრივი, წინასწარ განსაზღვრული შრიფტით.

<EM CLASS="bigfont">დიდი ტექსტი</EM>
<STRONG CLASS="bigfont">ჩვეულებრივი ტექსტი</STRONG>

სტილების განსაზღვრის სამი ხერხი

სტანდარტი CSS უშვებს სტილების განსაზღვრას სამი სხვადასხვა ხერხით. ჩვენ ახლა მათ განვიხილავთ.

პირველი ხერხი-სტილების განთავსება სტილების შიდა ცხრილში. ჩვენ უკვე ვიცით, რომ სტილების შიდა ცხრილი მდებარეობს ვებ-გვერდის დასახელების სექციაში, <STYLE> წყვილი ტეგის შიგნით. სტილების შიდა ცხრილები გამოიყენება, იმ შემთხვევაში, თუ საჭიროა სტილების რაიმე ანაკრეფის გამოყენება მხოლოდ ამ გვერდის ფარგლებში.
მეორე ხერხი-საჭირო სტილების განთავსება სტილების გარე ცხრილში. **სტილების გარე ცხრილი** ინახება ცალკე ფაილში **.css** გაფართოებით და უერთდება ვებ-გვერდს განსაკუთრებული ერთეულოვანი ტეგის <LINK> საშუალებით. აი ასე:

<LINK REL="stylesheet" HREF="styles.css" TYPE="text/css">

ამ ტეგის სამი ატრიბუტიდან ჩვენ გვაინტერესებს მხოლოდ ატრიბუტი **HREF**. ის განსაზღვრავს ფაილს, რომელშიც ინახება სტილების გარე ცხრილი. დანარჩენი ატრიბუტები დამხმარე ინფორმაციის მატარებლები არიან და განსაკუთრებულ შემთხვევებში გამოიყენებიან.
სტილების ცხრილის ფაილის შიგთავსი-იგივეა, რაც **<STYLE>** ტეგის შემთხვევაში. ერთადერთი სხვაობა: თვითონ ტეგის მითითება ფაილში, ამ შემთხვევაში საჭირო არ არის.

სტილების გარე ცხრილების გამოყენება მიზანშეწონილია, თუ საჭიროა ერთი და იგივე სტილების ანაკრეფის გამოყენება ერთდროულად რამოდენიმე ვებ-გვერდის მიმართ.

და ბოლო **მესამე ხერხი**-ყველაზე საინტერესო. ეს **ჩაშენებული სტილებია,** რომელთა აღწერილობა თავსდება პირდაპირ საჭირო ტეგში; უფრო ზუსტად თუ ვიტყვით, თავსდება განსაკუთრებულ ატრიბუტში **STYLE**, რომელიც პრაქტიკულად ყველა ხილული **HTML** ტეგების მიერ გამოიყენება.

<P STYLE="font-size: 9pt; color: green;">პატარა და მწვანე</P>

ეს მაგალითი უჩვენებს, თუ როგორ გამოიყენება ჩაშენებული სტილი. ფაქტიურად ის შეიცავს მხოლოდ სტილის აღწერას. სახელი აქ არ მიეთითება, ვინაიდან არ არის სავალდებულო ამ სტილის ცალსახად იდენტიფიცირება-ისედაც ნათელია თუ რომელი ტეგის მიმართ უნდა იყოს ის გამოყენებული.
თუ საჭიროა რაიმე სტილის გამოყენება ტექსტის ნებისმიერი ისეთ ფრაგმინტისადმი, რომელიც არცერთი ტეგის შიგთავსს არ წარმოადგენს, მაშინ ეს უნდა გაკეთდეს სპეციალური წყვილი ტეგის **<SPAN>**-ის საშუალებით. ეს ტეგი დაკავებულია მხოლოდ იმით, რომ მონიშნავს ტექსტის ფრაგმენტს, მისთვის სტილის მიერთებისათვის.

<P>ეს არის<SPAN STYLE="text-decoration: underline;">ხაზგასმული </SPAN>ტექსტი.</P>

ტექსტის გასაფორმებელი ფრაგმენტი უბრალოდ თავსდება **<SPAN>** ტეგის შიგნით, რის შემდეგაც ჩვენ შეგვიძლია მასზე ნებისმიერი ქმედების განხორციელება.

ერთსა და იმავე ვებ-გვერდზე შესაძლებელია სტილების განსაზღვრის სამივე ხერხის ერთდროულად გამოყენება. ანუ, შეიძლება ვებ-გვერდს მიუერთდეს **სტილების გარე ცხრილი**, სათაურის სექციაში-განთავსდეს **სტილების შიდა ცხრილი**, ხოლო ზოგიერთი ტეგებს მიმართ-გამოყენებული იქნას **ჩაშენებული სტილები**. ყოველივე ეს ვებ-გვერდების გაფორმების პროცესის მოქნილად მართვის საშუალებას იძლევა.

რატომ "კასკადური"?

როგორ ერკვევა ვებ-საიტი სტილების მთელ ამ "ზოოპარკში"? მოდით გავარკვიოთ ეს საკითხი. ამავდროულად, გავიგოთ ისიც, თუ რატომ ეწოდებათ სტილების ცხრილებს კასკადური.
დავუშვათ, რომ, ჩვენ გვაქვს სტილების ასეთი გარე ცხრილი:

```
P               {   font-size:  10 pt;  }
H1              {   font-size:  20 pt;
                    text-align: center:  }
.copyright      { font-style: italic;   }
```

ატრიბუტი **font-style** განსაზღვრავს შრიფტის მოხაზულობას, ხოლო მისი მნიშვნელობა **italic** შრიფტს გადააქცევს კურსივად **<EM>** ტეგის გამოყენების გარეშე. ატრიბუტი **text-align** კი განსაზღვრავს ტექსტის აბზაცის განლაგებას, ჩვენს შემთხვევაში-ცენტრში (მნიშვნელობა **center**).
სტილების ჩვენს მიერ შექმნილი ცხრილი დავიმახსოვროთ ფაილში სახელწოდებით **2.1.css.** ახლა შევქმნათ პატარა **Web**-გვერდი ასეთი **HTML** კოდით:

```
<HTML>
    <HEAD>
        <TITLE>ექსპერიმენტები სტილებზე</TITLE>
        <LINK REL="stylesheet" HREF="2.1.css" TYPE="text/css">
        <STYLE>
            H1 { font-size: 14pt; color: red; }
        </STYLE>
    </HEAD>
    <BODY>
```

```
        <H1>სტილები<H1>
         <P>ვატარებთ ექსპერიმენტებს სტილებზე</P>
         <P CLASS=”copyright>
          საავტორო<SPAN STYLE=”font-style: normal”> უფლებები
</SPAN> ჩვენ გვეკუთვნის.
 </P>
     </BODY>
 </HTML>
```

ყველაფერი ეს ჩვენთვის უკვე ცნობილია. ერთად-ერთი **font-style** სტილის ატრიბუტის მნიშვნელობა **normal** აუქმებს შრიფტის კურსივულ მოხაზულობას.

მაშ ასე, სულ პირველად ვებ-დამთვალიერებელი ტვირთავს **2.1.css** სტილის გარე ცხრილს, რომელიც ვებ-გვერდთან მიერთებულია **<LINK>** ტეგის საშუალებით. ამგვარად, **<P>** და **<H1>** ტეგების შეცვლის სტილი და სტილების კლასი copyright გამოყენებული იქნებიან ვებ-გვერდის მიმართ იმთავითვე.

შემდგომ ვებ-დამთვალიერებელი კითხულობს სტილების შიდა ცხრილს და არკვევს, რომ მასშიც არის ასევე მოთავსებული **<H1>** ტეგის გარდაქმნის სტილი. ანუ წარმოიშობა სტილების კონფლიქტი და ძალაში შედის კასკადურობის წესი, რომელიც მოაწესრიგებს ამ კონფლიქტს. თან, მოაწესრიგებს ისე კოხტად, რომ შეუძლებელია არ აღფრთოვანდეთ. (ნეტა ასე კოხტად გვარდებოდეს ყველა კონფლიქტი დედამიწაზე).

გარე ცხრილში გაკეთებულ სტილის განსაზღვრებას, ვებ-დამთვალიერებელი უმატებს განსაზღვრებას, რომელიც შიდა ცხრილშია გაკეთებული. ხოლო, თუ განსაზღვებრა ეხება ერთსა და იმავე ატრიბუტს (ჩვენს შემთხვევაში **font-size**, რომლითაც შრიფტის ზომაა განსაზღვრული), აიღება ის განსაზღვრება, რომელიც შიდა ცხრილშია გაკეთებული. წესი ”საკუთარი პერანგი სხეულთან უფრო ახლოა” მუშაობს სტილების ცხრილების შემთხვევაშიც.

ვებ-დამთვალიერებლის მიერ შიდა მოხმარებისათვის ფორმირებული **<H1>** ტეგის მარეზულტირებელ სტილს, ასეთი სახე ექნება:

```
H1    {  font-size: 14 pt;
         text-align: center;
          Color: red }
```

თუ ჩვენ დავაკვირდებით ვებ-გვერდისა და სტილების ცხრილის კოდს, შევნიშნავთ სტილების კიდევ ერთ კონფლიკტს. მისი მოწესრიგება ხდება ისევე, როგორც წინა შემთხვევაში.

ჯერ ვებ-დამთვალიერებელი იღებს **P** სტილის განსაზღვრებას და უმატებს მას სტილური კლასის **copyright** განსაზღვრებას.

ამ შემთხვევაში, სტილურ კლასში განსაზღვრულ ატრიბუტებს, გააჩნიათ პრიორიტეტი, ტეგის შეცვლის სტილში განსაზღვრულ ატრიბუტებთან შედარებით. აქ კიდევ ერთი წესი მოქმედებს, რომლის დამახსოვრებაც კარგი იქნებოდა: "(კერძო განსაზღვრებებს გააჩნიათ პრიორიტეტი, ზოგად განსაზღვრებებთან შედარებით).

სტილური კლასები კი-უფრო კერძოა, ვინაიდან შესაძლებელია მათი მიბმა ნებისმიერ ტეგთან.

შემდგომ, მიღებული სტილი მიებმება **copyright** მნიშვნელობის მქონე, ატრიბუტ **CLASS**-ის შემცველ **<P>** ტეგს და ამ ტეგის შიგთავსი აკრეფილი იქნება 10 პუნქტის ზომის კურსივი შრიფტით. მაგრამ ამ ტეგის შიგთავსში ჩვენ ვხედავთ ჩალაგებულ ტეგს **<SPAN>**, რომლისთვისაც განსაზღვრული იყო კურსივული მოხაზულობის უარმყოფელი ჩაშენებული სტილი. ვინაიდან ჩაშენებული სტილი-კიდევ უფრო კერძოა, ვიდრე სტილური კლასი, მის განსაზღვრებებს გააჩნიათ უმაღლესი პრიორიტეტი. ამტომ სიტყვა "უფლებები" (**<SPAN>** ტეგის შიგთავსი) უკვე აკრეფილი იქნება ჩვეულებრივი შრიფტით და არა კურსივით.

ყოველივე ეს შეიძლება რთულად მოგვეჩვენოს. სინამდვილეში, დამწყები ვებ-დიზაინერებისათვის (ხშირად გამოცდილებისთვისაც) კასკადურობა-ღმერთის რისხვაა. მაგრამ თუ გავერკვევით სტილების კონფლიკტების მოწესრიგების ხერხებში, შესაძლებელი გახდება სტილების ცხრილების რაოდენობის მინიმუმამდე შემცირება. ასე, რომ ერთი რომელიმე ვებ-გვერდის რაიმე ელემენტის შესაცვლელად, საკმარისი იქნება შიდა ცხრილში და ჩაშენებულ სტილში, მხოლოდ ზოგიერთი, საჭირო ატრიბუტების შეცვალა. ცხადია, რომ ეს გაცილებით მარტივი და სწრაფი გზაა, ვიდრე სტილის ცალკე ცხრილის დაწერა სპეციალურად ამ გვერდისათვის.

ჩვენ კიდევ დავუბრუნდებით სტილების ცხრილებს მომავალში, როდესაც დავიწყებთ უშუალოდ ჩვენი საიტის შექმნას. ახლა კი მოდით დავუბრუნდეთ ჩვენთვის უკვე ნაცნობ **HTML**-ს და განვახორციელოთ მოგზაურობა ამ ენის ისტორიაში.

HTML-ის ფიზიკური ფორმატირების ტეგები

სტილების ცხრილები-ეს საკმაოდ დაგვიანებული სიახლეა ვებ-დიზაინში. თუ გავითვალისწინებთ იმას, რომ **HTML** გაჩნდა 1989 წელს, სტილების ცხრილები-გაჩნდნენ მხოლოდ 1997 წელს. მაგრამ მეტ-ნაკლებად ფართო პოპულარობა მათ მიიღეს მხოლოდ უკანასკნელი რამოდენიმე წლის განმავლობაში. რა იყო მანამდე? ”სტილებისწინა” ეპოქაში, ვებ-გვერდებზე ტექსტების გაფორმებისათვის იყენებდნენ **HTML**-ის ეგრეთწოდებულ **ფიზიკური ფორმატირების ტეგებს.** ამ ტეგების საშუალებით ვებ-დიზაინერს შეეძლო განესაზღვრა ტექსტის შრიფტი, ფერი, სიმუქე და დახრილობა. პრინციპში, ეს ტეგები დღემდეც არსებობენ-ისინი არავის არ გაუუქმებია, უბრალოდ ისინი გამოცხადებულია გამოყენებისათვის არარეკომენდებულად. მათ ნაცვლად რეკომენდებულია უფრო მძლავრი და სამუშაოდ მოსახერხებელი სტილების ცხრილების გამოყენება.

კი მაგრამ, მინც საინტერესოა, თუ რას წარმოადგენენ ფიზიკური ფორმატირების ტეგები? როგორ ხდება მათი გამოყენება? (ვინ იცის იქნებ მოგვიწიოს სხვისი **HTML** კოდის ჩასწორება).

გავიხსენოთ ორი, ჩვენთვის უკვე ნაცნობი ტეგი, რომლებიც ტექსტის ფორმატირების საშუალებას იძლევიან. ესენია ტეგი **<STRONG>**, რომელიც ამუქებს ტექსტის შრიფტს და ტეგი **<EM>**, რომელიც განსაზღვრავს მის დახრილობას. პრინციპში მათი საშუალებითაც არის შესაძლებელი ტექსტის ფორმატირება, რასაც ჩვენ აქამდე წარმატებით ვაკეთებდით.

მაგრამ საქმე იმაშია, რომ ტეგები **<STRONG>** და **<EM>** არა მხოლოდ აფორმატებენ ტექსტს. ისინი ასევე გამოყოფენ მას განსაკუთრებულად: ტეგი **<STRONG>**-უფრო მეტად, ხოლო ტეგი **<EM>** უფრო ნაკლებად. პრინციპში ისინი არა ფორმატირების, არამედ ტექსტის გამოყოფის ტეგებია. ვებ-დამთვალიერებელი უბრალოდ ცდილობს დაგვანახოს ეს ”გამოყოფილობა” შრიფტის შეცვლის საკუთარი საშუალებებით. ვებ-დამთვალიერებლების სხვადასხვა პროგრამებს შეუძლიათ დაგვანახონ ეს ”გამოყოფილობა” სხვადასხვა ხერხებით: ფერით, ვებ-გვერდზე ადგილმდებარეობით, ხმის ინტონაციით (უკვე არსებობენ ეკრანიდან ტექსტის წაკითხვის ტექნოლოგიები) და სხვა. ამიტომაც ამბობენ, რომ **<STRONG>** და **<EM>**-წარმოადგენენ ტექსტის **ლოგიკური ფორმატირების ტეგებს.**

მათ საპირისპიროდ, **ფიზიკური ფორმატირების ტეგები** უბრალოდ ამუქებენ, ხრიან და ფერს უცვლიან ტექსტს. როგორც კი წააწყდება

ასეთ ტეგს, ვებ-დამთვალიერებელი უბრალოდ მოახდენს ტექსტის დაფორმატებას ისე, როგორც ეს ტეგით არის განსაზღვრული, მაგრამ არ გამოყოფს მას სხვა ხერხებით. (ასევე ხდება სტილების ცხრილების დამუშავებაც-ვებ-დამთვალიერებელი, უბრალოდ, ახდენს ვებ-გვერდის ელემენტის ფორმატირებას ისე, როგორც ეს სტილით არის გათვალისწინებულია).
ლოგიკური ფორმატირების ტეგებს **<STRONG>** და **<EM>** შეესაბამებიან ფიზიკური ფორმატირების წყვილი ტეგები **<B>** და **<I>**; პირველი ამუქებს ტექსტს, ხოლო მეორე-ხდის მას დახრილს. არსებობს აგრეთვე წყვილი ტეგი **<U>**, რომელიც ტექსტს ხდის ხაზგასმულს. წყვილი ტეგი **<FONT>**, ტექსტის შრიფტის დასახელების, ზომისა და ფერის განსაზღვრის საშუალებას იძლევა.
კი მაგრამ, რატომ არ უნდა გამოვიყენოთ ფიზიკური ფორმატირების ტეგები სტილების ცხრილების მაგივრად? პრინციპში გამოყენება შესაძლებელია, მაგრამ სასურველი არ არის. აი, თუ რატომ?
-ფიზიკური ფორმატირების ტეგები **W3C** კომიტეტის მიერ გამოცხადებულია, როგორც გამოსაყენებლად არა რეკომენდებული. ეს იმას ნიშნავს, რომ შეიძლება დადგეს ისეთ მომენტი, როდესაც უახლესი ვებ-დამთვალიერებელი (ან სხვა **HTML** კოდის გადამამუშავებელი პროგრამები) უბრალოდ არ განახორციელებენ მათ მხარდაჭერას. რათქმა უნდა, თუ ასეთი რამ, მოხდება, ეს არც თუ ისე მალე იქნება, მაგრამ, როგორც ამბობენ: "ციგებს ზაფხულში ამზადებენ".

შენიშვნა! *XHTML ენის სტანდარტი უკვე აღარ შეიცავს ფიზიკური ფორმატირების ტეგებს, მათი მოძველებულობის გამო.*

სტილების ცხრილებთან მუშაობა გაცილებით მოსახერხებელია. რთული **HTML** კოდის წაკითხვა ისედაც ძნელია, და თუ კი ის, ამავე დროს ფიზიკური ფორმატირების ტეგებით იქნება გაჯერებული, საერთოდ შეუძლებელი იქნება. გარდა ამისა, თუ ჩვენ მოულოდნელად დაგვჭირდება მუქი ფერით აკრეფილი მთელი ტექსტის გაფერადება მუქ-ლურჯად? ამ შემთხვევაში, ხომ გაცილებით მოსახერხებელია **CSS** კოდის ერთი სტრიქონის - **<STRONG>** ტეგის გარდასახვის სტილის დაწერა, ვიდრე მთელს კოდში **<B>** ტეგების მოძიება და მათი შიგთავსების კიდევ **<FRONT>** ტეგებში ჩასმა.

სტილების ცხრილები მეტ შესაძლებლობებს გვთავაზობენ. მაგალითად, **CSS**-კოდის რამოდენიმე სტრიქონის დაწერით, ჩვენ შეგვიძლია ტექსტური აბზაცის, სიმპათიურ ჩარჩოში ჩასმა. **HTML**-ის საშუალებებით პირდაპირ ამის გაკეთება შეუძლებელია.
სტილების ცხრილები ვითარდებიან, ხოლო ფიზიკური ფორმატირების ტეგები-არა. ახლა უკვე მიღებულია სტანდარტი კასკადური სტილების მეორე ვერსიაზე-**CSS2**. მასში იმდენი ახალი შესაძლებლობებია, რომ ჯერ-ჯერობით, ვერც ერთი ვებ-დამთვალიერებელი ვერ ახდენს სრულად მათ მხარდაჭერას.

ამით, საუბარი ვებ-გვერდების გაფორმების თაობაზე, შეგვიძლია ჩავთვალოთ დასრულებულად. მაგრამ, მოდით, კიდევ ერთ მნიშვნელოვან საკითხზე-რუსული ტექსტების კოდირებაზე შევაჩეროთ ჩვენი ყურადღება.

ტექსტის კოდირება. რუსული კოდირების პრობლემები.

ჯერ გავარკვიოთ, თუ როგორ ხდება ტექსტის წარმოდგენა კომპიუტერის მეხსიერებაში. საქმე იმაშია, რომ კლავიატურიდან შესატან, ეკრანზე გამოსატან და ფაილში შესანახ ყოველ სიმბოლოს შეესაბამება უნიკალური ნომერი, რომელსაც **სიმბოლოს კოდს უწოდებენ.** ასე, რომ კომპიუტერის მეხსიერებაში ტექსტი ფაქტიურად წარმოადგენს კოდების ანაკრებს. ეკრანზე უხილავ დამხმარე სიმბოლოებს, მაგალითად ტექსტის ყოველი სტრიქონის დასასრულის აღმნიშვნელ, ეტლის დაბრუნებისა და სტრიქონის გადატანის სიმბოლოებს, აგრეთვე გააჩნიათ თავიანთი კოდები.
სიმბოლოების კოდების ერთობლივობა, აღწერილობასთან ერთად, თუ რომელ სიმბოლოს რომელი კოდი შეესაბამება, ქმნის **კოდირების სისტემას,** ანუ **კოდების ცხრილს. ყოველ კოდირების სისტემას გააჩნია თავისი სახელი, მაგალიტად 1251 ან КОИ-8.**

ვინაიდან ნებისმიერი ენა, იყენებს სიმბოლოების საკუთარ ანაკრებს, ყოველი ენისათვის კოდირების სისტემები, როგორც წესი განსხვავდებიან ერთმანეთისაგან. (გამონაკლისს წარმოადგენენ ზოგიერთი დასავლეთ ევროპული ენები, რომლებიც იყენებენ ერთსა და იმავე კოდირებას). რუსული ენა ერთდროულად რამოდენიმე კოდირებას იყენებს. სწორედ ამ კოდირებებთან არის დაკავშირებული **რუნეტის-** ინტერნეტის რუსული ნაწილის, ერთ-ერთი უმთავრესი პრობლემა.

ყველაფერი იმის გამოა, რომ სხვადასხვა ოპერაციული სისტემების რუსული ვერსიები იყენებენ სხვადასხვა კოდირებას. ასე მაგალითად, ვერსია **Windows** იყენებს კოდირებას **1251**, ხოლო **MS-DOS**-ის რუსული ვერსია 866-ს (იგივე **ISO-8859-5**). თუ ამას დავუმატებთ კიდევ კოდირებას, რომელსაც იყენებს **UNIX** ოპერაციული სისტეის რუსული ვერსია-**КОИ-8** და **Macintosh**-ის კომპიუტერების რუსული ვერსიის ოპერაციული სისტემა **MacCyrillic**, კოდირებების რაოდენობა ოთხს მიაღწევს. ეს მხოლოდ მთავარი კოდირებებია. ადრე კიდევ რამოდენიმე ნაკლებად გავრცელებული კირილიკური კოდირება არსებობდა (”ძირითადი”, ”ბულგარული”, ”ამერიკული”, ”იუგოსლავიური” და სხვა).

შენიშვნა! *10-12 წლის წინ, შემუშავებული იქნა უნივერსალური კოდირება Unicode, რომელიც დედამიწაზე არსებული ყველა ენის მხარდაჭერას ახორციელებს. მიუხედავად იმისა, რომ დღეისათვის ეს კოდირება საკმაოდ პოპულარული გახდა, სხვადასხვა კოდირებების პრობლემა ჯერ-ჯერობით მაინც რჩება.*

რა საშიშროებას შეიცავს ყოველივე ეს? საქმე იმაშია, რომ სხვადასხვა კოდირებაში ერთ და იმივე სიმბოლოებს (ეს ეხება მხოლოდ რუსული ენის სიმბოლოებს-ლათინურებთან დაკავშირებით ყველაფერი რიგზეა) სხვადასხვა კოდები შეესაბამებათ. აღნიშნულის შედეგად ერთი კოდირების სისტემაში აკრეფილი ტექსტი, სხვა კოდირების სისტემაში დათვალიერებისას აბსოლუტურად არაწაკითხვადი ხდება.

ყველა ჩვენთაგანს უცდია ალბათ, **NotePad**-ში აკრეფილი ტექსტური დოკუმენტის **Norton Commander**-ში გახსნა და უნახავს, თუ რა ხდება ამ დროს-ტექსტი იქცევა გაუგებარი სიმბოლოების ერთობლივობად. ყოველივე ეს იმის გამო ხდება, რომ რუსული კოდირებები **866 (MS-DOS)**, რომელსაც იყენებს **Norton Commander**-ი და **1251 (Windows),** რომელსაც იყენებს **NotPad**, არ ემთხვევა ერთმანეთს. მათში ერთი და იგივე კოდი სხვადასხვა სიმბოლოებს შეესაბამება.

რაში მდგომარეობს გამოსავალი?

გამოსავალი არ არსებობს. შეიძლება მხოლოდ იმის იმედი გვქონდეს, რომ რომელიმე კოდირება გახდება სტანდარტი და თანდათანობით განდევნის კონკურენტებს. ჯერ-ჯერობით ასეთი სტანდარტის როლზე პრეტენზია გააჩნია 1251 კოდირებას, თუმცა ძველი თაობის

ინტერნეტ-სპეციალისტები, რომლებიც **UNIX**-თავსებად სისტემებს იყენებენ, ამ როლზე **КОИ-8**-ს უწევენ რეკომენდაციას. ყოველ შემთხვევაში, დღეისათვის ქსელის რუსულ სეგმენტში არსებული ვებ-გვერდების უმრავლესობა დაწერილია 1251 კოდირების სისტემაში. აქვე უნდა აღინიშნოს, რომ პრაქტიკულად ყველა თანამედროვე ვებ-დამთვალიერებლები ახორციელებენ ყველა არსებული კოდირებების სისტემების მხარდაჭერას.
სპეციალურად კოდირების მისათითებლად, რომელშიც ვებ-გვერდია აკრეფილი, **W3C** კომიტეტმა **HTML** ენაში გაითვალისწინა განსაკუთრებული ტეგი **<META>**. იმისათვის, რომ ვებ-დამთვალიერებელს გავაგებინოთ, რომ ვებ-გვერდი აკრეფილია 1251 კოდირებაში, საჭიროა სათაურის სექციაში ასეთი კოდის ჩასმა:

<META HTTP-EQUIV="Content-Type" CONTENT="text/html; Charset=windows-1251">

თუკი ჩვენ, ვებ-გვერდს შევქმნით **КОИ-8** კოდირებაში, კოდირების სისტემის მისათითებლად უნდა გამოვიყენოთ ასეთი კოდი:

<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=koi8-r">

თუ ვებ-გვერდი აკრეფილია რომელიმე ევროპულ ენაზე, რომელიც იყენებს კოდირებას 1250, ტეგს **<META>** ექნება ასეთი სახე:

<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=windows-1250">

ინგლისურენოვანი ვებ-გვერდებისათვის პრინციპში კოდირების მითითება აუცილებელი არ არის. მაგრამ თუ გაგვიჩნდება ასეთ სურვილი, მაშინ ვებ-გვერდის სათაურის სექციაში შეიძლება ჩაისვას ასეთი სტრიქონი:

<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">

რუსულენოვანი ვებ-გვერდებისათვის კოდირების მითითება ყოველთვის სასურველია. ეს დაიცავს შესაძლო პრობლემებისაგან ვებ-დიზაინერებს და საიტების მომხმარებლებსაც.

საიტების აგების საწყისები.

როგორც უკვე ვიცით, **ვებ-საიტი-ეს არის საერთო თემატიკით გაერთიანებული და ჰიპერმინიშნებებით დაკავშირებული მრავალი ვებ გვერდის ერთობლივობა**.
ნამდვილი საიტის ასაგებად საჭიროა **HTML** და **CSS** ენების ცოდნა, და რათქმა უნდა საიტის თემატიკის მოფიქრება და მისი სტრუქტურის დაგეგმვა.

Web-საიტის დაგეგმვა

დაგეგმვის ეტაპი-უპირველესი და უმთავრესია ნებისმიერი საიტის შექმნისას. ეს მტკიცება სამართლიანია, როგორც მარტივი ერთგვერდიანი, ასევე ტრანსნაციონალური სუპერკომპანიების, როგორებიც არიან **Intel** და **Coca-Cola**, გიგანტური საიტების შემთხვევაშიც. მართლაც, სანამ რაიმეს კეთებას დავიწყებდეთ , საჭიროა კარგად გავერკვეთ იმაში, თუ რისი მიღება გვსურს შედეგად. და ყოველივე ამის გარკვევა აუცილებელია საწყის ეტაპზე, მანამ სანამ შევუდგებოდეთ უშუალოდ საქმეს. როდესაც სამუშაო უკვე შესრულებულია, მასში რაიმეს შეცვლა გაცილებით უფრო რთულია, ვიდრე თავიდანვე ყველაფრის ისე კეთება, როგორც ეს საჭიროა.
ამიტომ სანამ შევუდგებოდეთ ჩვენი მომავალი საიტის დაგეგმვას, მოდით საერთოდ გამოვრთოთ კომპიუტერი, რათა არ წაგვძლიოს ვებ-გვერდების ეგრევე კომპიუტერზე აკინძვის სურვილმა. ამ ეტაპზე ჩვენ მხოლოდ ფანქარი და ქაღალდი დაგვჭირდება. ან თუ ჩვენ მაინცა და მაინც უპირატესობას კომპიუტერს მივანიჭებთ, მაშინ მხოლოდ ტექსტური რედაქტორის პროგრამა და საიტების პროექტირების **Microsoft Visio**-ს მსგავსი, სპეციალური პაკეტები შეიძლება დაგვჭირდეს.
მაშ ასე, რა უნდა გავარკვიოთ, ჩვენს საიტზე მუშაობის დაწყების წინ?

1.უპირველეს ყოვლისა, მკაცრად განვსაზღვროთ საიტის ამოცანები. რა უნდა აკეთოს მან: მოგვითხროს ვინმეს ან რამეს შესახე,

მოიზიდოს კლიენტები, დაგვეხმარის რაიმე პრობლემების გადაწყვეტაში, გაგვანათლოს თუ გაგვართოს. ამოცანებიდან გამომდინარე საიტის სტრუქტურა შეიძლება ძალიან განსხვავდებოდეს ერთმანეთისაგან.

2.განვსაზღვროთ, კონკრეტულად რა ინფორმაცია უნდა განთავსდეს საიტზე და რა არა. აქ მთავარი პრინციპია -არაფელი ზედმეტი, მხოლოდ ის, რაც სჭირდება პოტენციურ მომხმარებლებს.

3. შევაგროვოთ ყველა საჭირო ინფორმაცია. ეს დასაწყისშივე უნდა გავაკეთოთ, რათა არ მოგვიწიოს მისი მოძიება საიტზე მუშაობის დროს-ჩვენ მაშინ ამისთვის არ გვეცლება. ყველა ტექსტი, გამოსახულება, ფაილი, რომელთა განთავსებასაც ვაპირებთ საიტზე, ჩვენივე კომპიუტერში უნდა იყოს მოთავსებული.

4.გადავწყვიტოთ, თუ როგორ სტილში უნდა იყოს საიტის დიზაინი შესრულებული. იქნება ის კონსერვატიული, მკაცრი თუ სხვა. შესაბამისად, საშინაო გვერდი უნდა ასახავდეს ავტორის ესთეტიკურ მიდრეკილებებს, გასართობი საიტი უფრო მხიარულ ტონებში უნდა იყოს შესრულებული, ხოლო ახალი ამბებისათვის განკუთვნილი საინფორმაციო საიტი -უფრო მოკრძალებული, რათა დიზაინის სიმარტივემ არ დაჩრდილოს მთავარი-ინფორმაცია. ამ ეტაბზე სასარგებლო იქნება ვებ-გვერდების ესკიზების გაკეთება ქაღალდზე.

5.მოვიფიქროთ საიტის ლოგიკური სტრუქტურა, და სასურველია დავხატოთ იგი.

აქ აჯობებს, რომ არ დავიწყოთ ველოსიპედის აღმოჩენა, არამედ შევიდეთ მსგავსი თემატიკის რომელიმე უკვე არსებულ და პოპულარულ ვებ-საიტზე და ვნახოთ თუ როგორ არის ის ორგანიზებული.

მაგალითად, საოჯახო საიტისათვის იდეალური იქნება ასეთი სტრუქტურა: თავფურცელი (საშინაო გვერი) მოკლე ცნობებით საიტის მეპატრონის შესახებ. ამავე გვერდზე შესაძლოა განთავსდეს მიპატიჟება სხვა გვერდების დასათვალიერებლად, შესაბამისი მინიშნებებით, ხოლო დანარჩენ გვერდებზე განთავსდეს ინფორმაცია ავტორის გატაცებების, პროექტების, ნამუშევრების

შესახებ (იმის და მიხედვით, თუ ვინ არის ავტორი-პროგრამისტი, მუსიკოსი, მხატვარი, მითითებული უნდა იყოს მინიშნებების სია პროგრამებზე, აუდიო კლიპებზე ან სურათების ფაილებზე). სასურველია აგრეთვე, რომ საიტის შესაბამის გვერდებზე განთავსებული იყოს ფოტოგალერეა და მინიშნებები "მეგობარ" საიტებზე, ასევე ავტორის შესახებ უფრო დაწვრილებითი ინფორმაცია, მისი საფოსტო მისამართი და ფოტოსურათი.

6.შევამოწმოთ, ხომ არაფელი გამოგვრჩა. ეს საიტის დაგეგმვის ბოლო ეტაპია, მაგრამ არანაკლებ მნიშვნელოვანი, ვიდრე სხვები.

ახლა კი ზუსტად ის დროა, რომ დავიწყოთ საუბარი საიტის ლოგიკურ სტრუქტურაზე, ანუ იმზე, თუ რა განყოფილებებისაგან უნდა შედგებოდეს საიტი და როგორ უნდა იყვნენ ეს განყოფილებები ერთმანეთთან დაკავშირებული. ეს თემა მარტივი არ არის და შესაბამისად საუბარიც ხანგრძლივი გვექნება.

Web-საიტის ლოგიკური სტრუქტურა

მაშ ასე, საიტის ლოგიკური სტრუქტურა აღწერს საიტის სხვადასხვა ვებ-გვერდების დანიშნულებას და მათ შორის არსებულ კავშირებს. ის განისაზღვრება უპირველესყოვლისა იმით, თუ როგორ არის ორგანიზებული საიტზე ინფორმაცია. სხვანაირად, რომ ვთქვათ ყველა შესამუშავებელი საიტისათვის უნდა იქნას მოფიქრებული მისი საკუთარი სტრუქტურა.
რათქმა უნდა, არსებობენ საიტის სტრუქტურიზაციის ზოგიერთი ზოგადი პრინციპები, რომელთა დაცვა ყოველთვის აუცილებელია. ჩვენ ახლა მათ განვიხილავთ. ქვემოთ მოცემულია კარგად გააზრებული საიტის მაგალითი:

მთავარი გვერდი
  საიტის სიახლეები
    სიახლეების არქივი
 განყოფილება1
    გვერდი1
    გვერდი2
 .....
განყოფილება2
   გვერდი1

გვერდი2
.....
.....
ინფორმაცია დამპროექტებლების შესახებ
საკონტაქტო მონაცემები
საიტის რუქა.

ახლა ვისაუბროთ საიტის ყოველი გვერდის შესახებ ცალცალკე.

მთავარი ან საწყისი გვერდი, როგორც წესი, წარმოადგენს წინასწარ განსაზღვრულ (default) გვერდს. დამთვალიერებელი (ბრაუზერი), საიტის ინტერნეტ-მისამართის აკრეფის შემდეგ სწორედ ამ გვერდზე ხვდება. ის საიტის მოკლე აღწერას, საიტის შესახებ მოკლე შესავალ ინფორმაციას, სიახლეებს (სავალდებულო არ არის) და საიტის სხვა გვერდებზე გადამყვან ჰიპერმინიშნებების სიას (ე.წ. სანავიგაციო ზოლს) შეიცავს. ხანდახან მთავარ გვერდზე ათავსებენ ცნობებს საიტის შემქმნელებისა და მათი უფლებების შესახებ, აგრეთვე ცნობებს საიტის მესაკუთრეებთან და საიტზე მოხსენიებულ სხვა პირებთან და ორგანიზაციებთან კონტაქტების შესახებ.
შეიძლება ითქვას, რომ მთავარი გვერდი-ეს არის მთელი საიტის "სახე".
ამიტომ ცდილობენ, რომ ის რაც შეიძლება პატარა იყოს, რათა დამთვალიერებელი საიტიდან დროზე ადრე არ წავიდეს, ისე რომ, არ დაელოდოს მის სრულად ჩატვირთვას. მაგრამ მეორე უკიდურესობაშიც არ უნდა ჩავვარდეთ-რომ, მთავარი გვერდი იმდენად "სპარტანული" გამოვიდეს შინაარსის მიხედვით, რომ დამთვალიერებელი ვერ მიხვდეს სად მოხვდა. მთავარი გვერდი დამთვალიერებელს უნდა აწვდიდეს საკმარის ინფორმაციას საიტის შესახებ, მაგრამ ამასთან ერთად არ უნდა გადაიტვირთოს იგი ზედმეტი ცნობებით და თავი არ უნდა მოაბეზროს დამთვალიერებელს თავისი ხანგეძლივი ჩატვირთვით-ეს ძალიან მნიშვნელოვანია.

საიტის სიახლეებს ხშირად მთავარ გვერდზეც ათავსებენ. ისინი წარმოადგენენ საიტზე გაკეთებული ყველა დამატებებისა და განახლებების ქრონოლოგიურ სიას. როგორც წესი აქ აისახება მხოლოდ გარკვეული პერიოდის შესაბამისი (თვე, კვარტალი, წელი, იმისდა მიხედვით, თუ რამდენად ხშირად ხდება საიტის განახლება)

სიახლეები. უფრო ძველ სიახლეებზე წვდომისათვის გათვალისწინებულია ეგრეთ წოდებული **სიახლეების არქივის** ვებ-გვერდი.

საიტის სასარგებლო შიგთავსი-ეს ის ინფორმაციაა, რომლისთვისაც საიტი იყო შექმნილი. მისი სტრუქტურირება ხდება ისევე, როგორც წიგნში: რაიმე თემასთან დაკავშირებული ცალკეული აბზაცები ერთიანდებიან თავებად, თავები თავიანთ მხრივ-განყოფილებებად. ამგვარად, საიტის დამთვალიერებელს მაშინათვე შეუძლია იპოვოს მისთვის საინტერესო ინფორმაცია, განყოფილებებიდან თავებისკენ, თავებიდან აბზაცებისკენ მოძრაობით, მანამ, სანამ ვერ მიაგნებს იმას, რისთვისაც შემოვიდა საიტზე.

ცნობები საიტის დამპროექტებლების შესახებ შეიძლება განთავსებულ იქნას როგორც ცალკე გვერდზე, ასევე მთავარ გვერდზეც. თუ დამპროექტებლების რაოდენობა დიდი არ არის (ან სულაც ერთია), უფრო მიზანშეწონილია მეორე ვარიანტი. ამ შემთხვევაში მათ შესახებ ცნობები თავსდება ვებ-გვერდის ბოლოს, საავტორო უფლებების შესახებ ცნობების გვერდით. ამასთან, აუცილებლად მიეთითება ელექტრონული ფოსტის მისამართი, რომელზეც საიტის მომხმარებელს შეეძლება იმ პრობლემების შესახებ შეტყობინებების გაგზავნა, რომელთაც იგი საიტის დათვალიერებისას წააწყდა (არაჩატვირთვადი საიტი, "ცარიელი" გამოსახულებები, "არსაითკენ" გადამყვანი ჰიპერმინიშნებები, შეცდომები ტექსტში და სხვა.).
საიტის მფლობელთან კონტაქტის შესახებ ცნობები საჭიროა იმ შემთხვევაში, თუ მოცემული საიტი სარეკლამო მიზნებს ემსახურება. მაგალითად, თუ ეს სავაჭრო ორგანიზაციის საიტია, სავალდებულოა საკონტაქტო მონაცემების მითითება. სხვანაირად პოტენციური მომხმარებლებიდან ვერცერთი ვერ დაუკავშირდება გამყიდველს. ამ შემთხვევაში საჭიროა, როგორც ჩვეულებრივი "ქაღალდის", ასევე ელექტრონული ფოსტის მისამართების მითითება, აგრეთვე ტელეფონი, ფაქსი და პეიჯერი-მოკლედ, ყველა მონაცემი, რომელებზეც პოტენციურ კლიენტებს შეეძლებათ მომართვა.

საიტის რუქა-ეს არის ვებ-გვერდი, რომელზეც განთავსებულია საიტის ლოგიკური სტრუქტურის შესაბამისად ორგანიზებული, საიტის ყველა გვერდზე გადამყვანი ჰიპერმინიშნებების

ერთობლივობა. საიტის რუქა გამოიყენება იმისათვის, რომ მომხმარებელს, რომელმაც ზუსტად იცის თუ რა უნა, შეეძლოს მისთვის საჭირო ინფორმაციის სწრაფად მიგნება. რუქა როგორც წესი გააჩნია ყველა მეტნაკლებად დიდ და ჩახლართულ საიტებს; მარტივ საიტებზე ისინი ზედმეტია.

ვაპროექტებთ ჩვენს პირველ ვებ-საიტს

ახლა, სანამ ფანქარი და ქაღალდი ან ტექსტური რედაქტორის პროგრამა ჯერ კიდევ ჩვენს ხელთაა, მოვხაზოთ ჩვენი პირველი ვებ-საიტის სტრუქტურა. მის შექმნას ჩვენ შევუდგებით მხოლოდ მესამე თავში, მაგრამ დაგეგმვა უმჯობესია განვახორციელოთ წინასწარ.

ვთქვათ, რომ ჩვენი საიტი წარმოდგენილია კომპიუტერული თემატიკის ფაილებისა და არქივების სახით. საიტის შემადგენლობაში გვექნება ვებ-გვერდები, ჰიპერმინიშნებებით ფაილებზე (პროგრამები, დოკუმენტები, არქივები და სხვა) და სტატიებზე, მაგრამ თვით ფაილები და სტატიები არა (ასე ჩვენ თავს დავაღწევთ საავტორო უფლებებთან დაკავშირებულ პრობლემებს).
მთავარ გვერდზე, სანავიგაციო ზოლისა და შესავალი ტექსტის გარდა, გამოტანილი იქნება აგრეთვე საიტის სიახლეები და ცნობები დამპროექტებლების შესახებ. საიტის რუქას ჩვენ არ გავაკეთებთ-ის ზედმეტი იქნება ჩვენი უმარტივესი საიტისათვის.
დავწეროთ ქაღალდზე ან ტექსტურ რედაქტორში ჩვენი საიტის ლოგიკური სტრუქტურა:

მთავარი გვერდი (საიტის სიახლეები, დამპროექტებლების შესახებ ცნობები)
ფაილები
ინტერნეტი
ოფისი
მულტიმედია
სისტემა
პროგრამირება
თამაშები
სხვადასხვა
სტატიები
ინტერნეტი

ოფისი
მულტიმედია
სისტემა
პროგრამირება
გართობა
სხვადასხვა

ამ ჩამონათვალის ყოველი სტრიქონი უნდა წარმოადგენდეს ცალკე ვებ-გვერდს.
გვერდები ***ფაილები*** და ***სტატიები*** უნდა შეიცავდნენ კატეგორიების სიებს (ინტერნეტი, ოფისი და ა.შ.); ამ სიების ყოველი პოზიცია უნდა წარმოადგენდეს ჰიპერმინიშნებას იმ გვერდზე, რომელიც შეიცავს მინიშნებებს თვით ამ კატეგორიას მიკუთვნებულ ფაილებზე და სტატიებზე. გავაკეთოთ ისე, რომ ყოველი სტატია იხსნებოდეს ვებ-დამთვალიერებლის ცალკე ფანჯარაში-მომხმარებლისათვის ასე უფრო მოხერხებული იქნება.

რა იქნება ამის მერე?

მაშ ასე, როგორც ჩანს საიტის სტრუქტურასთან დაკავშირებით ყველაფერი ნათელია. აწი შეიძლება შევუდგეთ საქმეს. მაგრამ ჩვენ რა, ხელით ვაპირებთ ჩვენი ვებ-საიტის ყველა ვებ-გვერდის შექმნას? რატომ? ხომ არსებობენ მშვენიერი პროგრამები, რომლებიც დაგვეხმარებიან ჩვენ ამ საქმეში. მათ **ვებ-რედაქტორები ეწოდებათ.** ჩვენ სწორად მათგან ერთ-ერთს-**Macromedia DreamweaverMX**-ს გამოვიყენებთ ჩვენი საქმისთვის. შევუდგეთ საქმეს!

თავი 3

მუშაობა Macromedia DreamweaverMX-თან.

კაცობრიობა მუდმივად მიისწრფვის თავისთვის ცხოვრების გამარტივებისაკენ, ამისათვის მძიმე, მონოტონური და არასაინტერესო სამუშაოების შესასრულებლად სულ უფრო და უფრო სრულყოფილ ინსტრუმენტებს იგონებს. ეს გასაგებიც არის. ხომ შეიძლება, რომ გამონთავისუფლებული დრო, უფრო მეტი ეფექტურობით იქნას გამოყენებული ადამიანის მიერ, ვთქვათ თვითგანათლებისთვის, სპორტისთვის, ხელოვნების ან სულაც

კარგად გამოძინებისთვის. ხოლო, მთელი რუტინა შეასრულონ მანქანებმა.

ასეა ვებ-დიზაინშიც. გულმოდგინებით **HTML** ტეგების წერას, მათი ერთმანეთში ჩალაგებულობის წესის დაცვაზე თვალთვალს, შეცდომების გასწორებას, მიღებული შედეგების ყოვეწუთიერად შემოწმებას ვებ-დამთვალიერებელში, კაცობრიობას ურჩევნია საჭირო ტექსტის მარტივად დაწერეა და მზა ვებ-გვერდის მიღება. სწორად ამისათვის წერენ პროგრამისტები განსაკუთრებულ პროგრამებს, რომელთა დანიშნულება მდგომარეობს ვებ-გვერდების შექმნაში-ეგრეთ წოდებულ **ვებ-რედაქტორებს.**
ერთ-ერთი ასეთი პროგრამა დაწერილია ფირმა **Macromedia**-ს პროგრამისტების მიერ და მას **Macromedia Dreamweaver MX**-ი ეწოდება.
პირველი მისი ვერსია გამოვიდა შორეულ 1998 წელს; **DreamweaverMX**-ი საოცრად მძლავრი და ამავე დროს გამოსაყენებლად მარტივი პროგრამაა; მასთან მუშაობა შეუძლია დამწყებ მომხმარებელსაც კი, რომელსაც წარმოდგენაც კი არა აქვს **HTML**-ენაზე, მაგრამ პროფესიონალებიც კმაყოფილები რჩებიან მისი გამოყენებით. მრავალი ვებ-პროგრამისტი მუშაობს **DreamweaverMX**-თან, პირველი ვერსიიდან დაწყებული და ჯერჯერობით არც უფიქრიათ მისი შეცვლა რაიმე სხვა პროგრამით.
DreamweaverMX-ი, **WYSIWYG** (**What You See Is What You Get**, "რასაც ხედავ იმას მიიღებ") პრინციპით მომუშავე **ვიზუალური ვებ-რედაქტორების** ტიპიური წარმომადგენელია. ამ შემთხვევაში, მომხმარებელს მის მიერ ფორმირებული ტექსტის დანახვა მაშინათვე შეუძლია რედაქტორის ფანჯარაში. ამავე პრინციპით მუშაობს ცნობილი ტექსტური რედაქტორიც **Microsoft Word**.

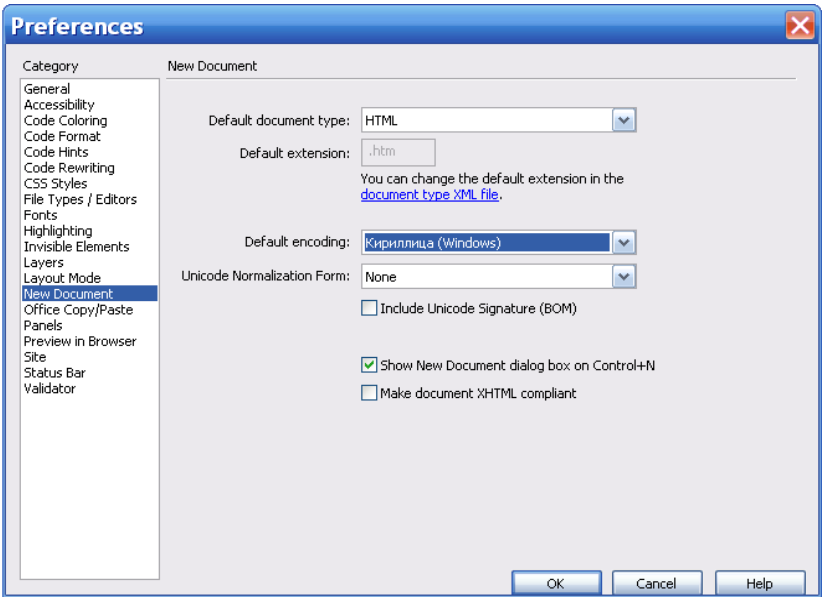
არავიზუალური ვებ-რედაქტორები (იგივე-**HTML-რედაქტორები**) აგებული არიან სხვა პრინციპზე. ისინი უშუალოდ **HTML**-კოდთან მუშაობენ და ამავე დროს მომხმარებელს სხვადასხვა დამატებით შესაძლებლობებს აძლევენ: ტეგების სწრაფად ჩასმა; ატრიბუტებისა და მათი მნიშვნელობების მოხერხებულად განსაზღვრა; წინასწარ განსაზღვრული შაბლონების ანაკრებებს, ვებ-გვერდების სტანდარტული ელემენტების შესაქმნელად; სტილების ცხრილებისათვის სპეციალურ რედაქტორებს; შედეგების სწრაფად

დათვალიერების შესაძლებლობას და სხვა. ამ შემთხვევაში, ისინი წააგავენ **NotePad**-ს, მაგრამ მნიშვნელოვნად გაფართოებულს.
შეიძლება ვინმემ დაასკვნას, რომ ვიზუალური რედაქტორები უფრო დამწყები ვებ-დიზაინერებისათვის არიან განკუთვნილნი, ხოლო არავიზუალური რედაქტორები კი მათი უფრო გამოცდილი კოლეგებისათვის. მაგრამ ეს მთლად სწორი დასკვნა არ იქნება.
პროფესიონალი, როგორც წესი იყენებს ერთსაც და მეორესაც; მას შეუძლია შექმნას ვებ-გვერდის ჩონჩხი ვიზუალურ რედაქტორში, ხოლო მისი დახვეწა განახორციელოს-არავიზუალურში. ასე გაცილებით მარტივია, განსაკუთრებით თუ ვებ-გვერდი საკმარისად რთულია.
უნდა აღინიშნოს, რომ პრაქტიკულად ყველა მეტ-ნაკლებად სერიოზულ ვებ-რედაქტორს გააჩნია თვით **HTML**-კოდის ჩასწორების რეჟიმი (ანუ ფაქტიურად ჰიბრიდულ ვებ-რედაქტორებს წარმოადგენენ). ამიტომ, უკვე პრაკტიკულად ყოველთვის, როცა "ვიზუალურ ვებ-რედაქტორებზე" საუბრობენ, სწორედ ჰიბრიდულ პროგრამებს გულისხმობენ. ცხადია, რომ მათ რიცხვს პირველ რიგში მიეკუთვნება **DreamweaverMX**-ი, რომელსაც დროა უკვე, რომ უფრო ახლოს გავეცნოთ.

Dreamweaver MX-ის წინასწარი გამართვა (გაწყობა)

DreamweaverMX-ბრწყინვალე პროგრამაა, რომელიც მომხმარებელს "პირველი სიტყვისთანავე" უგებს. მაგრამ მიუხედავად ამისა, ვებ-საიტზე მუშაობის დაწყებამდე ჩვენ მოგვიწევს მისი გაწყობა (გამართვა,პარამეტრების განსაზღვრა, დაყენება). საბედნიეროდ სავალდებულოდ გასაწყობი პარამეტრების რაოდენობა არც თუ ისე ბევრია, მაგრამ მათი განსაზღვრა აუცილებელია, სხვა შემთხვევაში ჩვენ ვერ მივიღებთ სასურველ შედეგს.
DreamweaverMX-ის ყველა პარამეტრის განსაზღვრა ხორციელდება მრავალფუნქციონალურ დიალოგურ ფანჯარაში **Preferences**, რომელიც მართვის სხვადასხვა ელემენტების შემცველი, მრავალი ჩანართისაგან შედგება. მისი გამოძახებისათვის მენიუ **Edit**-ში პუნქტი **Preferences** უნდა ავირჩიოთ ან დავაჭიროთ **<Ctrl>+<U>** კლავიშების კომბინაციას.
პირველ რიგში ჩვენ უნდა განვსაზღვროთ კოდირების წესი ახალი ვებ-გვერდებისათვის. ამისათვის სიაში **Category**, რომელიც ფანჯრის მარცხენა ნაწილში მდებარეობს, ავირჩიოთ პუნქტი **New Document**. ამის შემდეგ ფანჯრის მარჯვენა მხარეს გაჩნდებიან მართვის

ელემენტები, რომელთა საშუალებითაც განისაზღვრებიან ახლად შესაქმნელი ვებ-გვერდების პარამეტრები. (ნახ.3.1).
კოდირების წესი აირჩევა განშლად სიაში-**Default encoding**. აქ წესი ძალიან მარტივია: თუ არ არსებობს სპეციალური მოთხოვნები ვებ-სერვერზე ვებ-საიტის განთავსებისადმი, აღნიშნული სიიდან ვირჩევთ პუნქტს **Кириллица(Windows)**, რომელიც კოდირება 1251 შეესაბამება. ინგლისურენოვანი ვებ-გვერდებისათვის უნდა ავირჩიოთ პუნქტი **Западноевропейская** (დასავლეთევროპული, კოდირება **iso-8859-1**, რომელიც ინგლისური ენისთვის გამოიყენება) ან **Центральноевропейская** (ცენტრალურევროპული,კოდირება **1250**).
ხანდახან ხდება ისე, რომ ვებ-სერვერის ადმინისტრატორი ითხოვს, რომ მის ვებ-სერვერზე გამოქვეყნებული ვებ-გვერდების კოდირება იყოს **КОИ-8**.
ამ შემთხვევაში **Default encoding** სიაში საჭიროა **КОИ-8**-ს რუსული ვერსიის განმსაზღვრელი **Кириллица** (**KOI8-R**) პუნქტის არჩევა.

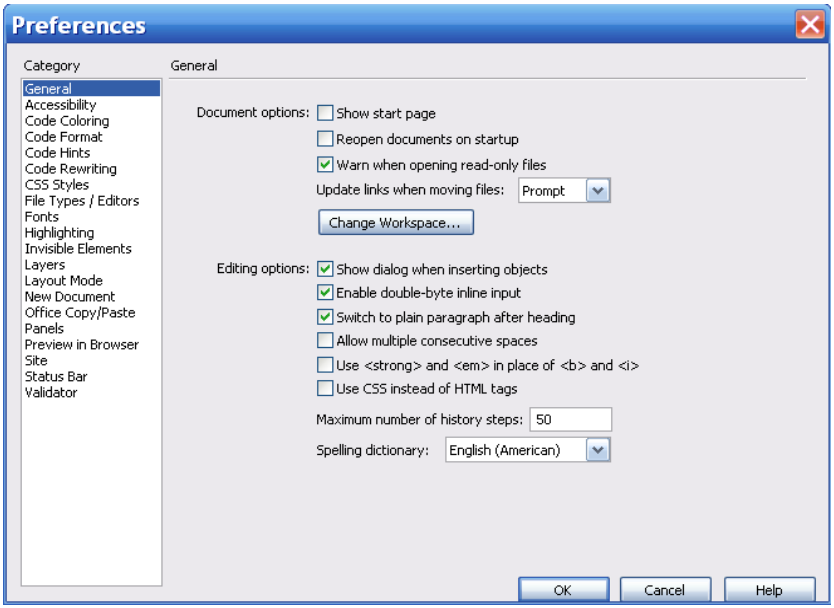


ნახ.3.1. დიალოგური ფანჯარა Preferences-ის კატეგორია New Document

შენიშვნა: *Dreamweaver-ი იძლევა ყოველი ვებ-გვერდის კოდირების ცალცალკე განსაზღვრის შესაძლებლობას. თუ როგორ ხდება ეს, ჩვენ შემდგომში გავარკვევთ.*

ამის მერე, ჩვენ აუცილებლად უნდა ჩავრთოთ ალამი **Use when opening existing files that don't specify an encoding**, თუ ის წინასწარ ჩართული არ იყო. ამის შემდეგ **DreamweaverMX**-ი ავტომატურად გამოიყენებს **Default encoding** სიაში არჩეულ კოდირებას, ყველა გასახსნელი ვებ-გვერდების მიმართ, თუ მათ **HTML**-კოდებში არ არის კოდირების განმსაზღვრელი <META> ტეგი.

კოდირების განსაზღვრის შემდეგ, **Category** სიაში ავირჩიოთ პუნქტი **General**. ფანჯარა **Preferences** მიიღებს სახეს, რომელიც ნაჩვენებია ნახ.3.2-ზე.
ალამი **Use and in place of <b> and <i>** ჩართავს ან გამორთავს ტექსტის ფიზიკური ფორმატირების **<B>** და **<I>** ტეგების ნაცვლად, ლოგიკური ფორმატირების **<STRONG>** და **<EM>** ტეგების გამოყენების შესაძლებლობას. როგორც წესი ის ჩართულია წინასწარ; თუ არადა ის უნდა ჩაირთოს. ჩვენ კი გვსურს ვებ-გვერდების შექმნა უკანასკნელი სტანდარტების შესაბამისად.



ნახ.3.2. დიალოგური ფანჯარა Preferences-ის კატეგორია General

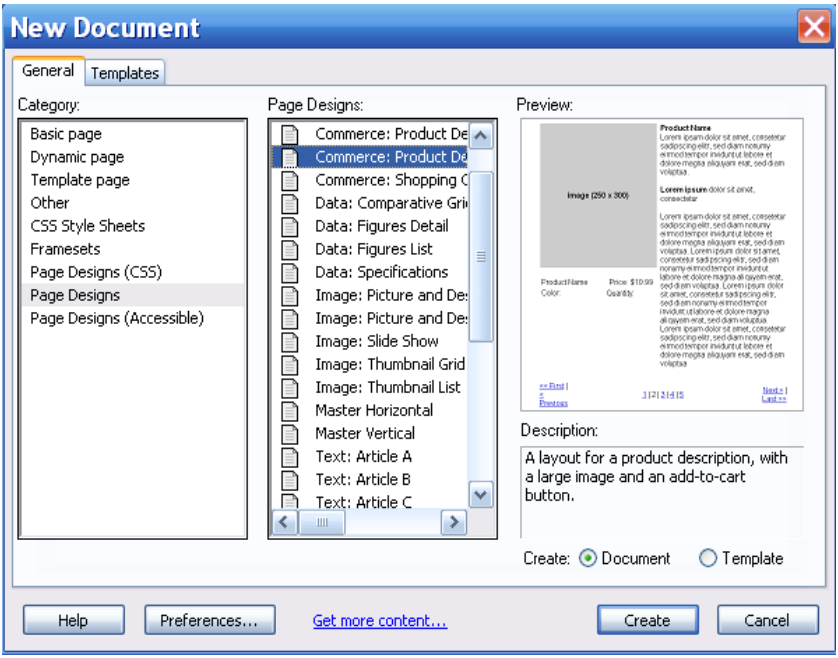
ალამი **Use CSS instead of HTML tags**, ტექსტის ფორმატირებისათვის, **HTML** ტეგების ნაცვლად ჩართავს, ან გამორთავს, **CSS** სტილების გამოყენების შესაძლებლობას. წესით ისიც წინასწარ ჩართული უნდა იყოს, და თუ არა, უნდა ჩაირთოს.
DreamweaverMX-ის გაწყობის დასრულებისთანავე დავაჭიროთ ღილაკს **OK**, რათა დავიმახსოვროთ განხორციელებული ცვლილებები. ამის შემდეგ შეიძლება შევუდგეთ მუშაობას.

DreamweaverMX-ში მუშაობის საფუძვლები

ახლა ჩვენ დავიწყებთ მუშაობას ჩვენს პირველ ვებ-საიტზე. მის შემადგენლობაში შესატანი ყველა ფაილისათვის (ვებ-გვერდები, სტილების ცხრილი, გრაფიკული გამოსახულებები) შევქმნათ ცალკე საქაღალდე, დავარქვათ მას ვთქვათ **Site1** და დავიწყოთ.
Windows სისტემაში პროგრამების გაშვება ყველამ იცის. დავაჭიროთ ჩვენთვის კარგად ნაცნობ ღილაკს **Start-Пуск-გაშვება** *(აქ და შემდგომშიც მართვის ელემენტების დასახელებები მოცემული იქნება რუსულ და ინგლისურ ენებზეც, ვინაიდან DreamweaveMX შეიძლება იყოს, როგორც რუსულ-, ასევე ინგლისურ-ენოვანი)*, და გახსნილ მენიუში ავირჩიოთ პუნქტი **Programs-Программы-პროგრამები**, შემდეგ დავაჭიროთ პუნქტს **Macromedia** და გახსნილ ქვემენიუში ავირჩიოთ პუნქტი **Macromedia DreamweaveMX**. გარკვეული დროის შემდეგ (**DreamweaverMX**-ის გახსნას დრო სჭირდება) ეკრანზე მისი **მთავარი ფანჯარა** გაჩნდება, რომელშიც ჩვენ უნდა ვიმუშაოთ.

ახალი ვებ-გვერდის შექმნა

ახალ ვებ-გვერდზე მუშაობა იწყება მისი შექმნით. იმისათვის, რომ ახალი "ცარიელი" გვერდი შევქმნათ, მენიუ **File**-ში ავირჩიოთ პუნქტი **New**, ან დავაჭიროთ კლავიშების კომბინაციას **<Ctrl>+<N>**. ეკრანზრ გაჩნდება დიალოგური ფანჯარა **New Document**, რომელიც ნაჩვენებია ნახ.3.3.-ზე.



ნახ.3.3.დიალოგური ფანჯარა New Document

იმისათვის, რომ მივიღოთ "სუფთა ფურცელი" ჩვენი ვებ-შემოქმედებისათის, სიაში **Category** ავირჩიოთ **Basic page**, ხოლო სიაში, რომელიც მარჯვნივ მდებარეობს-პუნქტი **HTML** და დავაჭიროთ ღილაკს **Create**. ეკრანზე გაჩნდება დოკუმენტის ფანჯარა, რომელშიც გაიხსნება **DreamweaverMX**-ის მიერ ჩვენთვის შექმნილი ვებ-გვერდი.

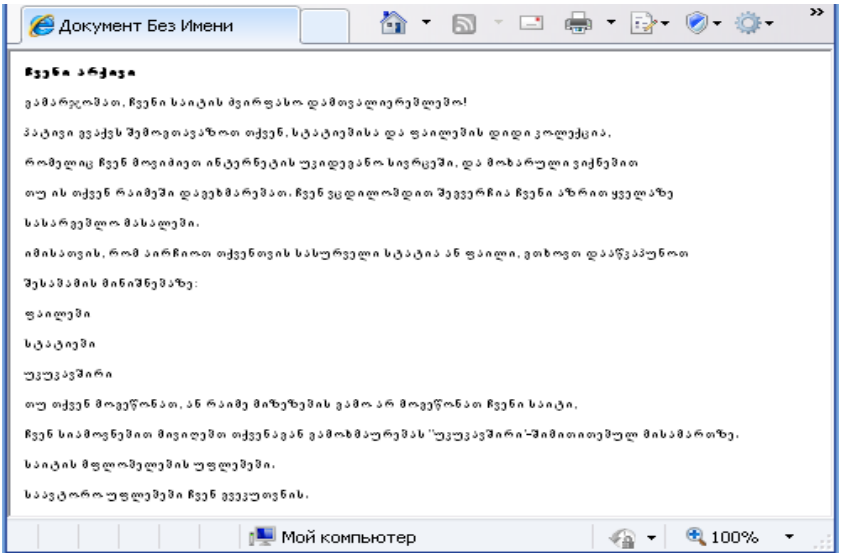
აქ ცოტა რამ მაინც უნდა ითქვას, დოკუმენტის ფანჯრების შესახებ. **DreamweaverMX**-ი რამოდენიმე ვებ-გვერდის ერთდროულად გახსნის საშუალებასაც იძლევა; ამავე დროს ყოველი გვერდი იხსნება თავის საკუთარ, ცალკე ფანჯარაში. ეს ძალიან მოხერხებულია საიტის შექმნის პროცესში: საიტის ერთ გვერდზე მუშაობისას, სხვა გვერდებიც თვალწინ გვაქვს.

შენიშვნა: *Dreamweaver-ი იძლევა ვებ-გვერდების ეგრეთ წოდებული შაბლონების (გვერდების თავისებური ნიმუშები, რომლებზეც წინასწარ არის ზოგიერთი ელემენტები განთავსებული) საფუძველზე*

შექმნის შესაძლებლობას. New Document ფანჯრის სიაში Category ხდება შაბლონების კატეგორიის არჩევა, ხოლო სიაში, რომელიც იქვე მარჯვნივ მდებარეობს-თვით არჩეულ კატეგორიაში შემავალი შაბლონის არჩევა. წინასწარი დათვალიერების არეში ჩვენ შეგვიძლია დავინახოთ, თუ რას წარმოადგენს არჩეული შაბლონი.

ტექსტის აკრეფა

აი დადგა აღსანიშნავი მომენტიც! ახლა ჩვენ ავკრიფავთ ჩვენს პირველ ტექსტს **DreamweaverMX**-ის დოკუმენტების ფანჯარაში (ნახ.3.4) და ამგვარად შევქმნით ჩვენი საიტის მთავარ გვერდს.



ნახ.3.4. ჩვენი პირველი ტექსტი

არ შევუდგებით ახლა მის დაფორმატებას-ჯერ მხოლოდ ავკრიფავთ. ეს ხორციელდება ზუსტად ისე, როგორც ნებისმიერ ტექსურ რედაქტორში-"ვაკაკუნებთ კლავიატურაზე", ვუყურებთ შედეგს და ჩასწორებებისათვის ვიყენებთ ისრიან კლავიშებს **<Backspace>** და **<Del>**. ბოლოს დავიმახსოვრებთ მიღებულ ვებ-გვერდს.

ვებ-გვერდის დასამახსოვრებლად საჭიროა მენიუ **Файл-File-ფაილ-ში** პუნქტ **Сохранить-Save-შენახვა**-ს გამოყენება ან კლავიშების კომბინაციის **<Ctrl>+<C>** დაჭერა. თუ აქამდე ჩვენ არ შეგვინახავს

ჩვენი ვებ-გვერდი (ჩვენ კი ის ჯერ არ შეგვინახავს), ეკრანზე გაჩნდება **Windows**-ის ფაილების შენახვის სტანდარტული დიალოგური ფანჯარა. ავირჩიოთ მასში ადრე შექმნილი საქაღალდე **Site1** და ვიფიქროთ იმაზე, თუ რა დავარქვათ ჩვენი პირველი ვებ-გვერდის ფაილს.

ჩვენ უკვე ვიცით, რომ ვებ-გვერდებიდან ერთ-ერთი ვებ-სერვერის მიერ აღიქმება, როგორც წინასწარ განსაზღვრული საწყისი გვერდი. აგრეთვე ჩვენ ვიცით, რომ ასეთი ვებ-გვერდს ენიჭება სახელი **default** ან **index** (შესაბამისი გაფართოებით **htm** ან **html**). მოდით ჩვენ საწყის ვებ-გვერდს დავარქვათ **default.htm.** შევიტანოთ ეს სახელი ფაილის შენახვის დიალოგური ფანჯრის, ფაილის სახელის შეტანის ველში და დავაჭიროთ შენახვის ღილაკს.

მაშ ასე, ჩვენ შევქმენით ჩვენი პირველი ვებ-გვერდი... მაგრამ, სულ არ გაგვხსენებია მისი დასახელება! სწორედ ის დასახელება, რომელიც **<TITLE>** ტეგით განისაზღვრება და ვებ-დამთვალიერებელში არ აისახება, მაგრამ სათაურში გამოიტანება. მოდით ის ახლავე განვსაზღვროთ, რომ ისევ არ დაგვავიწყდეს. მითუმეტეს რომ, ეს ძალიან მარტივად კეთდება: შესატანი ველი რომელშიც ის აიკრიფება, მდებარეობს პატარა რუხ პანელზე, რომელიც დოკუმენტის ფანჯრის ზედა ნაწილის გასწვრივ არის განთავსებული (ნახ.3.5.). (ამ პანელს ასევე უწოდებენ **დოკუმენტების ინსტრუმენტების პანელს**, ვინაიდან მასზე განთავსებულია ზოგიერთი მართვის ელემენტები, რომლებიც ვებ-გვერდზე სხვადასხვა მოქმედებების განხორციელების საშუალებას იძლევიან).

ნახ.3.5. დოკუმენტების პანელზე ვებ-გვერდის სახელის შესატანი ველი

შევიტანოთ ამ ველში ტექსტი **Cveni arqivi**, ჩვენი საიტის მთავარი გვერდის სახელწოდება და ხელახლა შევინახოთ იგი.

მთელი ამ დროის განმავლობაში, ჩვენ ვმუშაობდით **DreamweaverMX**-ში ვებ-გვერდების წარმოდგენის **WYSIWIG** რეჟიმში. ადრე ნათქვამი იყო, რომ **DreamweaverMX**-ი მომხმარებელს აძლევს აგრეთვე **HTML**-კოდის რედაქტირების საშუალებას.

იმისათვის, რომ მივიღოთ დაშვება **HTML**-კოდთან, ჩვენ უნდა გადავერთოთ კოდის წარმოჩენის რეჟიმში. ეს ხორციელდება სპეციალური ღილაკების საშუალებით, რომლებიც დოკუმენტების ინსტრუმენტალური პანელის მარცხენა ნაწილში მდებარეობენ, როგორც ეს ნახ.3.6-ზეა ნაჩვენები:

ნახ.3.6. ვებ-გვერდების წარმოდგენის რეჟიმების გადამრთველი ღილაკები

მოდით ჩამოვთვალოდ ეს ღილაკები მარცხნიდან მარჯვნისაკენ მიმართულებით:

Код(Code)-HTML კოდის წარმოჩენის რეჟიმი;
Раздельно(Split)-HTML კოდისა და ვებ-გვერდის ერთდროულად წარმოჩენის რეჟიმი. ამ დროს, დოკუმენტის ფანჯარა ორ ნაწილად იქნება დაყოფილი; ზედა ნაწილში წარმოდგენილი იქნება HTML კოდი, ქვედაში-თვითონ ვებ-გვერდი;

Дизайн(Design)-ვებ-გვერდის წარმოჩენის რეჟიმი.

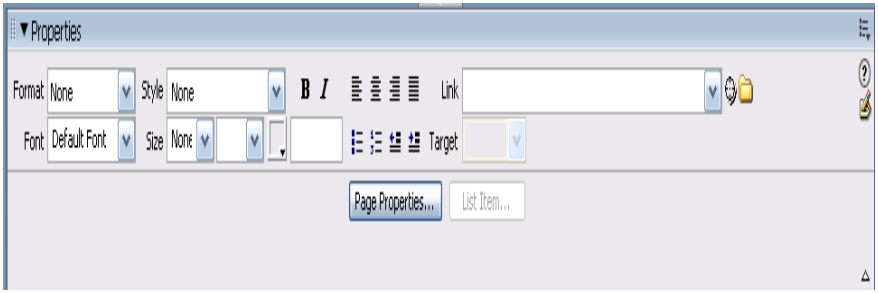
ნახ.3.6.-ზე ჩანს, რომ ამ ღილაკებიდან ერთ-ერთი, კერძოდ **Design,** დაჭერილია, და ჩართულია ვებ-გვერდის წარმოჩენის რეჟიმი. თუ ახლა დავაჭერთ **Code** ღილაკს, ჩაირთვება **HTML**-კოდის წარმოჩენის რეჟიმი; **Code** ღილაკის დაჭერა გამოიწვევს **Design** ღილაკის გამორთვას. ასეთ ღილაკები, რომლებიც ჯგუფებად არიან გაერთიანებული და მათგან მხოლოდ ერთი-ერთი შეიძლება იყოს ჩართული, მუშაობენ როგორც გადამრთველები და მათ **ღილაკ-გადამრთველები** ეწოდებათ.
HTML-კოდის წარმოდგენის რეჟიმში კარგად ჩანს, რომ **DreamweaveMX**-მა ჩვენს მიერ შეტანილი ყოველი აბზაცი მოათავსა **<P>** ტეგის შიგნით. ასევე მან ავტომატურად მოახდინა სათაურის სექციის ფორმირება და მოათავსა მასში გვერდის დასახელება **<META>** ტეგი, რომელიც მიუთითებს გამოყენებულ კოდირებაზე. მაშ ასე, ჩვენ მივიღეთ სწორად გაფორმებული ვებ-გვერდი ისე, რომ **HTML**-კოდის არც ერთი სტრიქონი არ დაგვიწერია.

აწი მოდით შევუდგეთ მის გაფორმებას.

ტექსტის ფრაგმენტების ფორმატირება

მაშ ასე, მოდით გამოვყოთ ჩვენს მიერ აკრეფილი ტექსტის ზოგიერთი ფრაგმენტი მუქი ფერით და კურსივით. ამავდროულად შევისწავლოთ **DreamweaverMX**-ის ინსტრუმენტები, რომლებიც ვებ-გვერდის სხვადასხვა ელემენტებისათვის პარამეტრების განსაზღვრის საშუალებას იძლევიან.

დავიწყოთ იმით, რომ გავხადოთ სტრიქონის-**გამარჯობათ ჩვენი საიტის ძვირფასო მომხმარებლებო!**-ს შრიფტი მუქი ფერის. ამისათვის გამოვყოთ იგი ისევე, როგორც ნებისმიერ რედაქტორში კეთდება. ამის შემდეგ შევხედოთ პროგრამის მთავარი ფანჯრის ქვედა ნაწილს. იქ განთავსებულია პატარა ფანჯარა, რომელიც ნაჩვენებია ნახ.3.7.-ზე.



ნახ.3.7. თვისებების რედაქტორი.

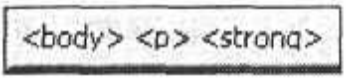
ამ ფანჯარას (უფრო სწორად, პანელს-**DreamweaverMX**-ის მცირე ფანჯარას, რომელზეც განთავსებულია მხოლოდ მართვის ელემენტები, და რომლის დანიშნულება სხვადასხვა პარამეტრების განსაზღვრაში მდგომარეობს) **თვისებების რედაქტორი ეწოდება**. მისი საშუალებით ხდება მუშაობა ვებ-გვერდის შიგთავსთან.

შენიშვნა: *თუ მთავარი გვერდის ქვედა ნაწილში თვისებების რედაქტორის მაგივრად ჩანს მხოლოდ მუქი ზოლი წარწერით Свойства-Properties-თვისებები), საჭიროა ზუსტად ამ წარწერაზე "მაუსით" დაწკაპუნება-და თვისებების რედაქტორი გაჩნდება*

ეკრანზე. თუ იქ ჩანს მხოლოდ ვიწრო მუქი ფერის ხაზი ბრტყელი ღილაკით, უნდა დავაწკაპუნოთ ამ ღილაკზე.

თვისებების რედაქტორის შიგთავსი დამოკიდებულია იმაზე, თუ რას მოვნიშნავთ ჩვენ დოკუმენტის ფანჯარაში. თუ, როგორც მოცემულ შემთხვევაში, ჩვენ მოვნიშნავთ ტექსტს, მაშინ მასში გაჩნდებიან ტექსტის ფორმატირებისათვის საჭირო მართვის ელემენტები.
მონიშნული ტექსტის ფრაგმენტი რომ გავხადოთ მუქი, ამისათვის საჭიროა თვისებების რედაქტორში **B** ღილაკზე დაჭერა, რის შემდეგაც ის დარჩება დაჭერილი. თუ მოგვინდა მუქი შრიფტის გადაქცევა პირიქით ჩვეულებრივ შრიფტად, საკმარისი იქნება ამავე ღილაკის ამორთვა. ასეთ ღილაკებს, რომელთაც ორი მდგომარეობა გააჩნიათ, **ღილაკ-გამომრთველებს** უწოდებენ. ასევე შეიძლება გამოყენებულ იქნას კლავიშების კომბინაცია **<Ctrl>+<B>**-იგი ისევე მუშაობს, როგორც ზემოთ აღწერილი ღილაკი.
ამის შემდეგ, ჩვენ თუ გადავერთვებით **HTML**-კოდის წარმოჩენის რეჟიმში, დავინახავთ, რომ **DreamweaverMX**-მა სტრიქონი-**მოგესალმებით, ჩვენი საიტის ძვირფასო მომხმარებლებო!,** **<STRONG>** ტეგების შიგნით მოათავსა. პრინციპში ჩვენ მისგან ამას არ მოველოდით.
მოდით, ახლა მთლიანად ბოლო აბზაცი, რომელშიც ჩვენი საავტორო უფლებებია აღწერილი, მოვნიშნოთ და მისი შრიფტი გავხადოთ კურსივი. ამავდროულად ვისწავლოთ ტექსტის ფრაგმენტის გამოყოფის საკმაოდ საინტერესო ხერხი, რომელიც მხოლოდ **DreamweaverMX**-ს გააჩნია.
შევხედოთ დოკუმენტების ფანჯრის ქვედა ნაწილს (არა მთავარი ფანჯრის!). იქ ვიწრო მუქი ზოლია განლაგებული, რომელიც ვებ-გვერდის შესახებ სხვადასხვა ცნობებს შეიცავს-მას **სტატუსის ზოლი** ეწოდება. მსგავს ზოლს ჩვენ მრავალი სხვა პროგრამების ფანჯრებშიც შეიძლება შევხვდეთ და ყველგან, ის სხვადასხვა ნაირად იქცევა.
მაშ ასე, **DreamweaverMX**-ის დოკუმენტის სტატუსის ზოლის უმეტეს ნაწილში მოთავსებულია **ტეგების სექცია** (ნახ.3.8). ის შეიცავს მცირე ზომის ღილაკებს, რომლებიც იმ ტექსტის ფრაგმენტის შემცველ ტეგებს შეესაბამებიან, რომელშიც მოცემულ მომენტში ტექსტური კურსორია მოთავსებული. ყველაზე მარჯვენა ღილაკი (****-ნახ.3.8.) შეესაბამება ტექსტს, რომელიც ამ ტეგის შიგთავსს

წარმოადგენს. ღილაკი, რომელიც ოდნავ მარცხნივ მდებარეობს (**<p>**), შეესაბამება მშობელ-ტეგს და ა.შ. ყველაზე მარცხენა ღილაკი-<**body**>- შეესაბამება ვებ-გვერდის სხეულის სექციას, ანუ დოკუმენტის ფანჯრის მთელ შიგთავსს.



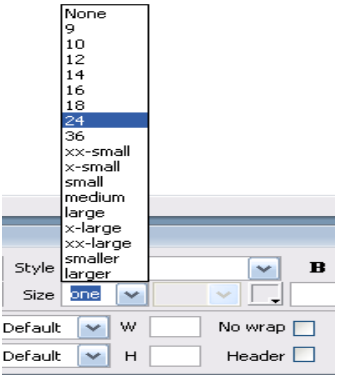
ნახ.3.8. სტატუსის ზოლში განთავსებული ტეგების ზოლი

დავუშვათ, რომ ჩვენ გვინდა **<STRONG >** ტეგით მონიშნული ტექსტის გამოყოფა (სტრიქონის **მოგესალმებით, ჩვენი საიტის ძვირფასო მომხმარებლებო!**, რომლის შრიფტიც ჩვენ ახლახან გავამუქეთ). ამისათვის ტექსტური კურსორი მოვათავსოთ ამ სტრიქონის რომელიმე ადგილას (რის შემდეგაც ტეგების სექცია მიიღებს ნახ.3.8.-ზე ნაჩვენებ სახეს) და დავაჭიროთ ამ სექციის **<STRONG>** ღილაკს, რის მერეც მთელი სტრიქონი გამოყოფილი იქნება (თუმცა ასეთივე წარმატებით ჩვენ შეგვეძლო **<p>** ღილაკის დაწკაპუნებაც). ხოლო ღილაკი <body> მონიშნავს მთლიან ვებ-გვერდს (ანუ **<BODY>** ტეგის მთლიან შიგთავსს). მოდით ტექსტური კურსორი მოვათავსოთ ბოლო აბზაცზე და სტატუსის სტრიქონის ტეგების სექციაში დავაწკაპუნოთ ღილაკს **<p>**. ამის შედეგად მთელი უკანასკნელი აბზაცი გამოყოფილი იქნება. იმისათვის, რომ შრიფტი, რომლითაც აკრეფილია ტექსტის მონიშნული ფრაგმენტი, გადავაქციოთ კურსივად, უნდა დავაჭიროთ ღილაკ-გადამრთველს  , რომელიც თვისებების რედაქტორში მდებარეობს. ასევე შეიძლება გამოყენებული იქნას კლავიშების კომბინაცია **<Ctrl>+<L>**.

ტეგების სექცია ჩვენ იმ შემთხვევაშიც დაგვეხმარება, თუ გვინდა ვებ-გვერდის რომელიმე ელემენტის წაშლა. ამ შემთხვევაში საკმარისია ჯერ იმ ღილაკზე დაჭერა, რომელიც საჭირო ტეგს შეესაბამება,ხოლო შემდეგ კლავიშზე-**<Del>.**

მოდით კიდევ რამე მანიპულაცია მოვახდინოთ ჩვენს ტექსტზე. მაგალითად შევცვალოთ შრიფტი, რომლითაც აკრეფილია სტრიქონი **მოგესალმებით, ჩვენი საიტის ძვირფასო მომხმარებლებო!**, ცოტათი გავზარდოთ მისი ზომა და გავაწითლოთ იგი. ცხადია რომ, ჯერ ჩვენ

უნდა მოვნიშნოთ აღნიშნული ტექსტი, რისთვისაც ტეგების სექციას გამოვიყენებთ.
შრიფტის შესაცვლელად გამოიყენება ნახ.3.9.-ზე ნაჩვენები ჩამოშლადი სია **Font**, ხოლო შრიფტის ზომის შესაცვლელად ნახ.3.10.-ზე ნაჩვენები სია Size.
ორივე ეს სია მდებარეობს იმავე თვისებების რედაქტორში. მოდით ვანახოთ, თუ რა არჩევანს გვთავაზობენ ისინი:



ნახ.3.9. ჩამოსაშლელი სია Font

ნახ.3.10. ჩამოსაშლელი სია Size

ორივე მდებარეობს თვისებების
რედაქტორში

ჩამოსაშლელი სია **Font** შეიცავს რამოდენიმე პუნქტს, რომლებიც წარმოადგენენ ეგრეთ წოდებულ **HTML**-ის **სტანდარტულ შრიფტებს.** მათი გამოყენება შესაძლებელია **Web**-გვერდებზე. იგულისხმება, რომ ეს შრიფტები დაყენებულია ნებისმიერ კლიენტურ კომპიუტერზე და ამიტომ ვებ-გვერდებზე ტექსტის წარმოჩენასთან დაკავშირებით აპრიორი არ შეიძლება რაიმე პრობლემა შეიქმნას. სტანდარტული შრიფტების რაოდენობა დიდი არ არის, მაგრამ ვებ-დიზაინში მეტი არც არის საჭირო.

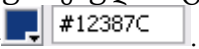
შენიშვნა: *პრინციპში, HTML-ის და CSS-ის სტანდარტები არ კრძალავენ ტექსტისათვის ნებისმიერი შრიფტის გამოყენებას. მაგრამ ამ შემთხვევაში ჩვენ თვითონ მოგვიწევს ზრუნვა იმაზე, რომ ჩვენი*

საიტის დამთვალიერებლებს ქონდეთ ამ შრიფტის მიღების შესაძლებლობა.

ამ სიის პუნქტი **Default Font** მონიშნული ტექსტისათვის განსაზღვრავს იმავე შრიფტს, რაც მშობელ ტეგს გააჩნია.
HTML-ის და **CSS**-ის სტანდარტები აგრეთვე განსაზღვრავენ რამოდენიმე სტანდარტულ სიდიდეებს შრიფტების ზომებისათვის. მათ შეესაბამება პუნქტები **xx-small**, **x-small**, **small**, **medium**, **large**, **x-large** და **xx-large**, რომლებიც ჩამოთვლილია ჩამოშლად სიაში-Size. ამ სიაში ჩამოთვლილია ყველა სტანდარტული ზომა, ყველაზე მცირედან, ყველაზე დიდის მიმართულებით. პუნქტები **larger** და **small** შესაბამისად განსაზღვრავენ შრიფტის ზომას, რომელიც ერთი საფეხურით ან მაღლაა ან დაბლა მშობელი ტეგის შრიფთტან მიმართებაში.
რიცხვითი პუნქტები შრიფტის ზომას განსაზღვრავენ: ასევე შესაძლებელა ჩვენთვის სასურველი ზომის ხელით შეტანა პირდაპირ სიაში. ხოლო პუნქტი **None** შრიფტის ზომას გადააქცევს ისეთად, როგორიც მშობელ ტეგს გააჩნია.
მოდით ჩვენს მიერ მონიშნული სტრიქონის ტექსტისათვის გავნსაზღვროთ შრიფტი **Veranda**, **Arial**, **Helvetica**, **San-serif**, **Sylfaen**, რისთვისაც სიაში **Font**, ავირჩიოთ შესაბამისი პუნქტები. შემდეგ ერთი საფეხურით გავზარდოთ ამ შრიფტის ზომა სიაში **Size**, **larger** პუნქტის არჩევით. ამის შემდეგ მოდით....
თუმცა შევჩერდეთ და გავარკვიოთ, თუ რა არის ეს? ჩამოსაშლელ სიას-Style-ში, რომელიც თვისებების იმავე უცვლელ რედაქტორში მდებარეობს, გაჩნდა პუნქტი **style1**. შრიფტი, რომლითაც აკრეფილია ეს პუნქტი, ზუსტად ემთხვევა ჩვენ მიერ მონიშნული სტრიქონის შრიფტს. სია **Style**-ის დანიშნულება, თუ ლოგიკას მივყვებით, მდგომარეობს სტილური კლასის არჩევაში, რომელიც მიერთებული იქნება მონიშნულ ტექსტთან. გამოდის, რომ **DreamweaverMX**-მა ავტომატურად შექმნა სტილური კლასი და მიუერთა ის ამ სტრიქონს? მოდით ვნახოთ!
გადავერთოთ **HTML**-კოდის წარმოჩენის რეჟიმში და დავაკვირდეთ **Web**-გვერდის სათაურის სექციას. ასეც არის-იქ მდებარეობს ახლადშექმნილი სტილების ცხრილი ერთადერთი სტილით **style1** (ციფრი დასახელიბის შემდეგაც შეიძლება იყოს, ან შესაძლოა შეიცვალოს შემდგომში-ეს ნორმალური მოვლენაა).
მისი განსაზღვრება გამოიყურება ასე:

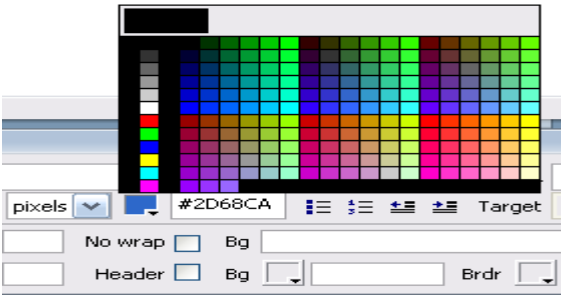
```
.style1 { font-family:Verdana, Arial, Helvetica, Sans-serif,Sylfaen;
          font-weight: bold;
          font-size: larger; }
```

ესეგი ის განსაზღვრავს შრიფტს (ატრიბუტი **font-family**), შრიფტის ზომას (ჩვენთვის უკვე ცნობილი ატრიბუტი **font-size**) და შრიფტის სიმუქეს. **DreamweaverMX**-მა უკუაგდო ტეგი **<STRONG>** და სტილის განსაზღვრებაში შეიტანა ატრიბუტი font-weight, რომელიც სწორედ ამ სიმუქეს განსაზღვრავს (მნიშვნელობა **bold**). ეს პროგრამა უკეთესი გამოდგა, ვიდრე ჩვენ მოველოდით.

ისღა დაგვრჩენია, რომ ჩვენს მიერ არჩეული სტრიქონი ვაიძულოთ "გაწითლდეს". ამისათვის ჩვენ უნდა გადავერთოთ უკან, ვებ-გვერდის წარმოჩენის რეჟიმში და მივმართოთ ფერების არჩევაზე პასუხისმგებელ **DreamweaverMX**-ის განსაკუთრებულ ინსტრუმენტს-**ფერების სელექტორს.** ის ასე გამოიყურება-  .

ჩვენ უკვე ვიცით, როგორ განვსაზღვროთ ფერი **CSS**-ში;უნდა შევიტანოთ მისი ინგლისური სახელწოდება: **green** (მწვანე), **red**(წითელი) და ა.შ. მაგრამ ამ ხერხით შეიძლება განისაზღვროს არა ყველა ფერი, არამედ მხოლოდ ნაწილი. ფერების უმრავლესობა განისაზღვრება ეგრეთ წოდებული **RGB** (**RED,GREEN,BLUE**-წითელი, მწვანე, ლურჯი) კოდით აი ასეთ ფორმატში: **#RRGGBB**. აქ **RR**-თექვსმეტობითი რიცხვია 0-დან **FF**-დე, რომელიც განსაზღვრავს საბოლოო ფერში წითელი ფერის წვლილს, **GG**–მწვანესას, ხოლო **BB**-ლურჯისას. **RGB**-კოდში ფერის განსაზღვრის მაგალითია-**#336699** (ეს მკრთალი-ცისფერი ფერია).

იგივე **RGB** კოდი ჩვენ შეგვიძლია შევიტანოთ სელექტორის მარჯვენა ნაწილში მდებარე შესატან ველში და დავაჭიროთ კლავიშა **<ENTER>**-ს. მაგრამ ჩვენ ხომ არ ვიცით ჩვენთვის სასურველი ფერის კოდო! ამიტომ ჯობია დავაწკაპუნოთ სელექტორის მარცხენა მხარეში მდებარე პატარა კვადრატულ ღილაკს. ეკრანზე გაჩნდება მცირე ზომის, ფერის ასარჩევი ფანჯარა, რომელიც ნაჩვენებია ნახ.3.11-ზე.



ნახ.3.11. ფერის არჩევის გახსნილი ფანჯარა

ამ ფანჯრის უმეტესი ნაწილი უკავია პალიტრას-სხვადასხვა ფერის მცირე ზომის კვადრატებს. საჭირო ფერის ასარჩევად საკმარისია დავაწკაპუნოთ შესაბამის კვადრატს. ამის შემდეგ ფერის არჩევის ფანჯარა დაიხურება, ხოლო არჩეული ფერი იმწამსვე იქნება გამოყენებული. თუ გვინდა ამ ფანჯრის დახურვა ფერის არჩევის გარეშე, საკმარისია დავაჭიროთ კლავიშას **<ESC>.**

მოდით ფანჯარაში ავირჩიოთ წითელი ფერი და მაშინათვე გადავერთოთ **HTML**-კოდის წარმოჩენის რეჟიმში, რათა შევხედოთ სტილების ცხრილს.

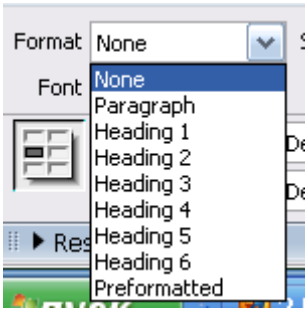
დავინახავთ, რომ **DreamweaverMX**-მა არსებულ ერთად-ერთ სტილში **style1** დაამატა ფერის განსაზღვრის ატრიბუტი **color**, რომელსაც მინიჭებული აქვს არჩეული ფერის შესაბამისი **RGB** კოდი.

აქ კიდევ რამოდენიმე სიტყვა უნდა ითქვას ვებ-დიზაინში გამოსაყენებელ ფერებზე. საქმე იმაშია, რომ სხვადასხვა კომპიუტერულ პლატფორმებს-თუნდაც სხვდასხვა კომპიუტერებს გააჩნიათ ვიდეოსისტემის სხვადასხვა პარამეტრები. ზოგიერთს შეუძლია მხოლოდ თექვსმეტი ფერის წარმოჩენა, სხვებს კი ყველა 16,7 მილიონის, რაც მთლიანად ფარავს ადამიანის თვალის ფერების გარჩევადობის შესაძლებლობებს. ცხადია, კომპიუტერული პლატფორმების ასეთი სიმრავლის პირობებში, ვებ-დიზაინერი ვერ იქნება დარწმუნებული იმაში, რომ ყველა მისი ფერი ყველგან სწორად იქნება წარმოჩენილი. ამიტომაც, **HTML**-სტანდარტი განსაზღვრავს ეგრეთწოდებულ **ფერების უსაფრთხო პალიტრას,** რომელიც გარანტირებულად უზრუნველყოფს ნებისმიერი ფერის სწორად წარმოჩენას ყველა პროგრამაში და ყველა კომპიუტერში. ვებ-დიზაინერებისათვის რეკომენდირებულია აღნიშნული

უსაფრთხო პალიტრის გამოყენება, თუმცა არც მის იგნორირებას არავინ არ კრძალავს.
DreamweaverMX-ის ფერების ასარჩევ პანელში გამოტანილი ფერებიც, აგრეთვე მიეკუთვნებიან ფერების უსაფრთხო პალიტრას.
ახლა დავიმახსოვროთ ჩვენი ერთადერთი ვებ-გვერდი და შევუდგეთ მუშაობას ტექსტის მთლიან აბზაცებთან.

აბზაცების ფორმატირება

როდესაც შეგვქონდა ვებ-გვერდის ტექსტი, ჩვენ გავითვალისწინეთ სამი სტრიქონი, რომელთა გადაქცევას ვაპირებდით სათაურად. მოდით შევასრულოთ ეს. დავიწყოთ პირველი სტრიქონით- **ჩვენი არქივი**. ეს იქნება ჩვენი საიტის დასახელება. მოდით გადავაქციოთ იგი პირველი დონის სათაურად. დავაყენოთ მასზე ტექსტური კურსორი (ამ შემთხვევაში მთელი აბზაცის მონიშვნა სავალდებულო არ არის) და შევხედოთ თვისებების რედაქტორს. რითი შეიძლება ის ჩვენ დაგვეხმაროს?
იქ არის ჩამოსაშლელი სია **Format**. თუ გავხსნით მას ჩვენ დავინახავთ იმას, რაც ნაჩვენებია ნახ. 3.12.-ზე.



ნახ.3.12. თვისებების რედაქტორის
ჩამოსაშლელი სია Format.

აქ ყველაფერი მარტივად არის. ამ სიის პუნქტი **Paragraph** ახდენს ტექსტის, როგორც ჩვეულებრივი აბზაცის (**<p>** ტეგით მონიშნული ტექსტის) ფორმირებას. პუნქტები **Heading1**, **.....,Heading6**, საშუალებას გვაძლევენ გადავაქციოთ იგი შესაბამისად პირველი, მეორე,….,მეექვსე დონის სათაურებად. ამიტომ, ზედმეტი საუბრის გარეშე ავირჩიოთ **Heading1**-და საიტის დასახელებაც მზად გვექნება.

ახლა შევქმნათ უფრო დაბალი დონის სათაურები. დავაყენოთ ტექსტური კურსორი სტრიქონზე ***უკუკავშირი***-**Contact- Обратная связь** და სიაში-**Format**, ავირჩიოთ პუნქტი **Heading2**. იგივე გავაკეთოთ სტრიქონთან ***საიტის მფლობელების უფლებები*** მიმართებაში. აი ახლა კი ყველა სათაური შექმნილია.

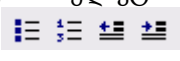
ყველაფერი რაც გავაკეთეთ კარგია, მაგრამ ჩვენი საიტის დასახელება რატომღაც, არცთუ ისე კოხტად გამოიყურება. მოდით განვათავსოთ იგი ცენტრში. თვისებების რედაქტორს სპეციალურად ამ მიზნებისათვის გააჩნია ღილაკების ანაკრები- . ეს ღილაკები განსაზღვრავენ ტექსტის მდებარეობას შესაბამისად (ჩამოთვლის მიმართულებაა მარცხნიდან მარჯვნივ):

მარცხნივ;
ცენტრში;
მარჯვნივ;
სიგანის მიხედვით.

ტექსტური კურსორი ისევ განვათავსოთ პირველ სტრიქონზე, რომელიც ჩვენ უკვე გადავაქციეთ საიტის დასახელებად და ზემოთ აღწერილი ღილაკების ანაკრებში დავაჭიროთ მარცხნიდან მეორე ღილაკს. მორჩა, საიტის დასახელება ჩვენ უკვე ცენტრში მოვაქციეთ.

საინტერესო თავისებურება: მას შემდეგ, რაც ჩვენ დავაჭირეთ ზემოთ დასახელებული ღილაკების ანაკრების მეორე ღილაკს, ის დაჭერილ (ჩართულ) მდგომარეობაში დარჩა. თუ ჩვენ ამის შემდეგ დავაჭერთ სხვა ღილაკს, მაგალითად მესამეს (რომელიც განსაზღვრავს ტექსტის განთავსებას მარჯვენა მხარეს), მაშინ მესამე ღილაკი დარჩება დაჭერილი (ჩართული), ხოლო მეორე "ამოირთვება". ღილაკების ასეთ ანაკრებებს, რომლებშიც მხოლოდ ერთ-ერთი შეიძლება იმყოფებოდეს დაჭერილ მდგომარეობაში, უწოდებენ **ღილაკ-გადამრთველებს**.

გადავერთოთ **HTML**-კოდის წარმოჩენის რეჟიმში და შევხედოთ, რა ჩასვა იქ **DreamweaverMX**-მა. მაშ ასე ტეგები **<H1>** და **<H2>** დასმულია ისე, როგორც უნდა იყოს. მაგრამ, იმის ნაცვლად, რომ შექმნილიყო ახალი სტილი, მან **<H1>**-ტეგში ჩასვა ატრიბუტი **ALIGNE**, **center** მნიშვნელობით, რომელიც განსაზღვრავს ცენტრში განთავსებას. ეს პრინციპში მთლად ის არ არის, რაც ჩვენ გვინდოდა-გამოყენებულია

ფიზიკური ფორმატირების ატრიბუტი, მაგრამ არაუშავს, ჯერჯერობით დავტოვოთ ასე. ჩვენ მას შემდგომში გავასწორებთ.
ახლა შევხედოთ სტრიქონებს, რომლებშიც ჩამოთვლილია ჩვენი საიტის განყოფილებები: ფაილები და სტატიები. არც თუ კარგად გამოიყურებიან ისინი.... მოდით გადავაქციოთ ისინი **HTML**-სიად, რისთვისაც გამოვიყენოთ თვისებების რედაქტორში განთავსებული ღილაკ-გადამრთველების ანაკრები  .
HTML-სია გამოიყურება, როგორც სტრიქონების სიის-პუნქტების ერთობლივობა, რომელთაგან ყოველი განსაკუთრებული ნიშნით არის გამოყოფილი ან დანომრილია. გარკვეული ნიშნებით (**მარკერებით**) გამოყოფილი პუნქტების სიებს **მარკირებულ სიებს** უწოდებენ, ხოლო დანომრილი პუნქტების სიებს-**დანომრილ სიებს**. ზევით აღწერილი ღილაკების ანაკრების მარცხენა ღილაკი მონიშნულ სტრიქონებს გადააქცევს მარკირებული სიის პუნქტებად, ხოლო მარჯვენა-დანომრილი სიის პუნქტებად.

- ფაილები
- სტატიები

ნახ.3.13. მარკირებული სია

1. ფაილები
2. სტატიები

ნახ.3.14. დანომრილი სია

მოვნიშნოთ ფაილები და სტრიქონები და დავაჭიროთ ანაკრების მარცხენა ღილაკს. ჩვენს მიერ შექმნილ მარკირებულ სიას ექნება ნახ.3.13.-ზე ნაჩვენები სახე.
თუმცა შესაძლებელია, რომ უფრო მოხერხებული იყოს დანომრილი სიის შექმნა. ამისათვის საკმარისი იქნება, რომ მოვნიშნოთ ეს სტრიქონები და დავაჭიროთ მარჯვენა ღილაკს. მივიღებთ იმას, რაც ნაჩვენებია ნახ.3.14-ზე. დავტოვოთ ის როგორც არის.
თუ საჭიროა რაიმე სიის პუნქტების გარდაქმნა ჩვეულებრივ აბზაცებად, მაშინ უნდა გამოვყოთ ისინი, და კიდევ ერთხელ დავაჭიროთ მოცემულ მომენტში უკვე დაჭერილ (ჩართულ) მდგომარეობაში მყოფ ღილაკს, რათა ამოვრთოთ ის.
გადავერთოთ **HTML**-კოდის წარმოჩენის რეჟიმში და შევხედოთ, თუ რისგან შედგება **HTML**-სია:

<OL>
    <LI>ფაილები</LI>
        <LI>სტატიები</LI>
</OL>

ჩანს, რომ დანომრილი სია წარმოადგენს წყვილ ტეგს **<OL>**, რომელშიც მოთავსებულია მოცემული სიის პუნქტები. ყოველი პუნქტი, თავის მხრივ, წარმოადგენს მეორე ტეგის-**<LI>** შიგთავსს. როგორც ვხედავთ, არავითარი სირთულე ამაში არ არის.
მარკირებული სიის შესაქმნელად ტეგი- **<OL>** -ის ნაცვლად, გამოიყენება ტეგი **<UL>**. სიის პუნქტები ამ შემთხვევაშიც იქმნებიან იმავე **<LI>** ტეგის საშუალებით.
ეს არის ის, რისი თქმაც შეიძლებოდა ტექსტის აბზაცების ფორმატირებასთან დაკავშირებით. ცხადია, რომ თუ ჩვენ რაიმე, ამაზე მეტის გაგება მოგვინდება, ყოველთვის შეგვიძლია ჩავიხედოთ **DreamweaverMX**-ის ინტერაქტიულ ცნობარში, აგრეთვე **HTML** და **CSS** ენების ცნობარებში, რომლებიც **DreamweaverMX**-ის კომპლექტაციაში შედიან (მათ შესახებ ამ თავის ბოლოშიც ვისაუბრებთ).
ახლა კი მოდით გავარკვიოთ, როგორ ჩავსვათ ვებ-გვერდის ტექსტში, ზოგიერთი სპეციალური სიმბოლოები, კერძოდ "კოპირაიტის" (საავტორო უფლების) ნიშანი.

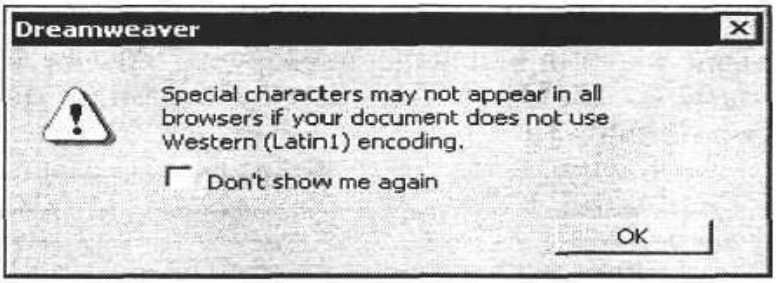
სპეციალური სიმბოლოები და არატექსტური ელემენტები

შევხედოთ ჩვენი ტექსტის ბოლო სტრიქონს, რომელიც შეიცავს ცნობებს საავტორო უფლებების შესახებ. ზოგადად, აღნიშნული მიზნისათვის, აღიარებულ სტანდარტად იქცა კოპირაიტის © ნიშნის გამოყენება. ჩვენ კი გვაქვს გრძელი, არასიმპათიური ტექსტი. მოდით შევცვალოთ იგი მოკლე და შესამჩნევი ნიშნით.
ჯერ მოვნიშნოთ სიტყვები **საავტორო უფლებები ჩვენ გვეკუთვნის.,** (წერტილიც არ უნდა გამოგვრჩეს), რომელთა შეცვლასაც ვაპირებთ ნიშნით ©, და ამოვშალოთ ისინი. შემდეგ ტექსტური კურსორი დავაყენოთ ტექსტის იმ ადგილას, სადაც ჩვენ გვინდა ამ ნიშნის მოთავსება. ავირჩიოთ ქვემენიუ **Special Character-Специальные символы-სპეციალური სიმბოლოები**-ს პუნქტი **Copyrigt -Авторское право-საავტორო უფლებები**, რომელიც თავის მხრივ მდებარეობს მენიუ **Insert-Вставка-ჩასმა**-ს ქვემენიუ **HTML**-ში. **DreamweaverMX**-მა

შესაძლოა ეკრანზე გამოგვიტანოს გაფრთხილება (ნახ.3.15), რომელიც გვამცნობს იმის შესახებ, რომ აღნიშნული სიმბოლო შესაძლოა არ გამოისახოს კორექტულად **DreamweaverMX**-ის პარამეტრების განსაზღვრისას, ჩვენს მიერ მითითებული კოდირების წესის გამო. დავხუროთ ეს ფანჯარა **OK**-ზე დაჭერით, ხოლო იმისათვის, რომ ის მომავალშიც არ გამოვიდეს ეკრანზე, ზემოაღნიშნული ფანჯრის დახურვამდე ჩავრთოთ ალამი **Don't Show me again-Больше не показывать-მომავალში აღარ მაჩვენო.**

ნახ.3.16-ზე ნაჩვენებია ჩვენს მიერ ტექსტში ახლახან ჩასმული სიმბოლო-©.

(არ დაგვავიწყდეს აგრეთვე ბოლო სტრიქონის გასწორება).



ნახ.3,15. გაფრთხილება სპეციალური სიმბოლოს შესაძლო არაკორექტულად წარმოჩენის შესახებ

© ჩვენ, საიტის მფლობელები, 2011

ნახ.3.16. სიმბოლო ვებ-გვერდის ტექსტში

აქ ჩვენ წავაწყდით **HTML**-ის ეგრეთ წოდებულ **სპეციალურ სიმბოლოებს.**

თავსებადობის მოსაზრებებიდან გამომდინარე, არ შეიძლება ამ სიმბოლოების ასე მარტივად ჩასმა **HTML** კოდში. როგორც წესი, ხდება მათი ჩანაცვლება, ან რიცხვითი კოდებით ან მნემონური აღნიშვნებით. კერძოდ სიმბოლო © , **HTML**-კოდში, აღინიშნება როგორც **& copy;**, აი ასე:

<P><EM>© ჩვენ საიტის მფლობელები, 2011</EM></P>

ორმაგი ბრჭყალების (”) სიმბოლო აღინიშნება, როგორც **"** ”ნაკლებობის” სიმბოლ (<)-როგორც **<** ხოლო მეტობის სიმბოლ (>)-როგორც **>** (ბოლოში წერტილ-მძიმის ნიშანი აუცილებელია, **HTML**-კოდის ჩასწორებისას ისინი უნდა გავითვალისწინოთ).

კიდევ ერთი მეტად სასარგებლო ნიშანია-უწყვეტი ჰარის (Space-გამოტოვება) ნიშანი. რაც არ უნდა მოხდეს, ვებ-დამთვალიერებელი არასოდეს არ გადაიტანს სტრიქონს ასეთი ჰარის მიხედვით. ტექსტში რომ მსგავსი ჰარი ჩავსვათ, ტექსტური კურსორი უნდა დავსვათ საჭირო ადგილას, წავშალოთ ჩვეულებრივი ჰარი, თუ ის უკვე არსებობს, და დავაჭიროთ კლავიშების კომბინაციას **<Ctrl>+<Shift>+<ჰარი>**. **HTML** კოდში უწყვეტი ჰარი გამოისახება ასე-

ხანდახან პირიქით, საჭიროა აბზაცის სტრიქონის გახლეჩა ორ აბზაცად, ისე, რომ ეს გახლეჩა დარჩეს სამუდამოდ. ამისათვის საჭიროა, ტექსტური კურსორი დავაყენოთ საჭირო ადგილას, და დავაჭიროთ კლავიშების კომბინაციას **<Shift>+<Enter>**. ამასთან, **HTML** კოდში ჩასმული იქნება განსაკუთრებული, ერთეულოვანი ტეგი **
-სტრიქონის გაწყვეტის** ტეგი.

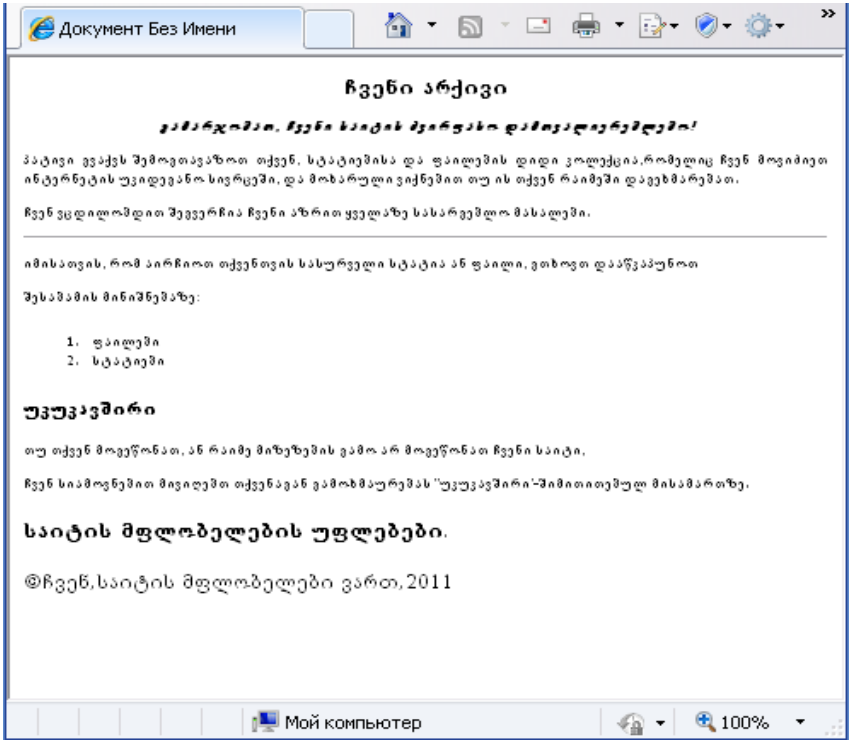
უწყვეტი ჰარით და სხვა სპეციალური სიმბოლოებით სტრიქონების გახლეჩის ექსპერიმენტებით დაღლილები, მოდით შევუდგეთ საქმეს. არ გეჩვენებათ, რომ ჩვენი გვერდის მესამე და მეოთხე აბზაცები, ძალიან ახლოს მდებარეობენ ერთმანეთთან? როგორ დავაშოროთ ისინი ერთმანეთს?

რა თქმა უნდა შეიძლება ცარიელი აბზაცის ჩასმა, თუ ტექსტურ კურსორს მოვათავსებთ მესამე აბზაცის ბოლოს და დავაჭერთ კლავიშას **<Enter>**, მაგრამ ის ლამაზი არ გამოვა. მოდით სხვანაირად გავაკეთოთ, სახელდობრ, გავყოთ ეს აბზაცები **ჰორიზონტალური ხაზით.**

ტექსტური კურსორი მოვათავსოთ მეოთხე აბზაცის დასაწყისში და ავირჩიოთ მენიუ **Insert**-ის, **HTML** ქვემენიუს, პუნქტი-**Horizontal Rule**.

ჩვენს მიერ შექმნილი ჰორიზონტალური ხაზი ნაჩვენებია ნახ. 3.17.

ნახ.3.17. ჰორიზონტალური ხაზი



ნახ.3.18. Site1 საიტის დასრულებული ვებ-გვერდი default.htm

რომ წავშალოთ უადგილო ადგილას ჩასმული ჰორიზონტალური ხაზი, საჭიროა მისი მონიშვნა "მაუსის" საშუალებით და კლავიშა **<Del>**-ზე დაჭერა.

ჰორიზონტალური ხაზი იქმნება ერთეულოვანი ტეგით **<HR>** და მიეკუთვნება ეგრეთ წოდებულ **არატექსტურ ელემენტებს.** არატექსტური ელემენტები განისაზღვრებიან თვით **HTML** კოდში, მაგრამ არ მიეკუთვნებიან ტექსტს. ჰორიზონტალური ხაზების გარდა, არატექსტურ ელემენტებს მიეკუთვნებიან აგრეთვე ცხრილები, რომლებთანაც ჩვენ ახლა დავიწყებთ მუშაობას.

ჯერ-ჯერობით დავიმახსოვროთ ჩვენს მიერ შექმნილი ვებ-გვერდი **default.htm** და შევამოწმოთ ყველაფერი სწორად გავაკეთეთ თუ არა. დასრულებული გვერდი უნდა გამოიყურებოდეს ისე, როგორც ნაჩვენებია ნახ.3.18-ზე. ამის შემდეგ დავხუროთ დასრულებული ვებ-გვერდი, იმ დოკუმენტის ფანჯრის დახურვის ღილაკზე დაჭერით, რომელშიც ეს გვერდია მოთავსებული, ან მენიუ **File**-ის პუნქტ **Close**-ის, ან კიდევ კლავიშების კომბინაციის **<Ctrl>+<W>** არჩევით.

ამით, ჩვენი საიტის მთავარი გვერდი პრინციპში მზად არის. შევუდგეთ სხვა გვერდების შექმნას.

მუშაობა ცხრილებთან

თუ საჭიროა ვებ-გვერდის შეზღუდულ სივრცეში მრავალი რიცხვითი (და არა მხოლოდ რიცხვითი) მონაცემების მოთავსება, უკეთესი საშუალება, ვიდრე ცხრილებია არ არსებობს. თუ გვინდა ლამაზი სიების შექმნა, აქაც ცხრილები დაგვეხმარება. ცხრილებმა დაიპყრეს ვებ-დოკუმენტები. გასაკვირიც არ არის: რამოდენიმე, არც თუ ისე მნიშვნელოვან ნაკლოვანებებთან ერთად, მათ უამრავი ღირსებები გააჩნიათ.

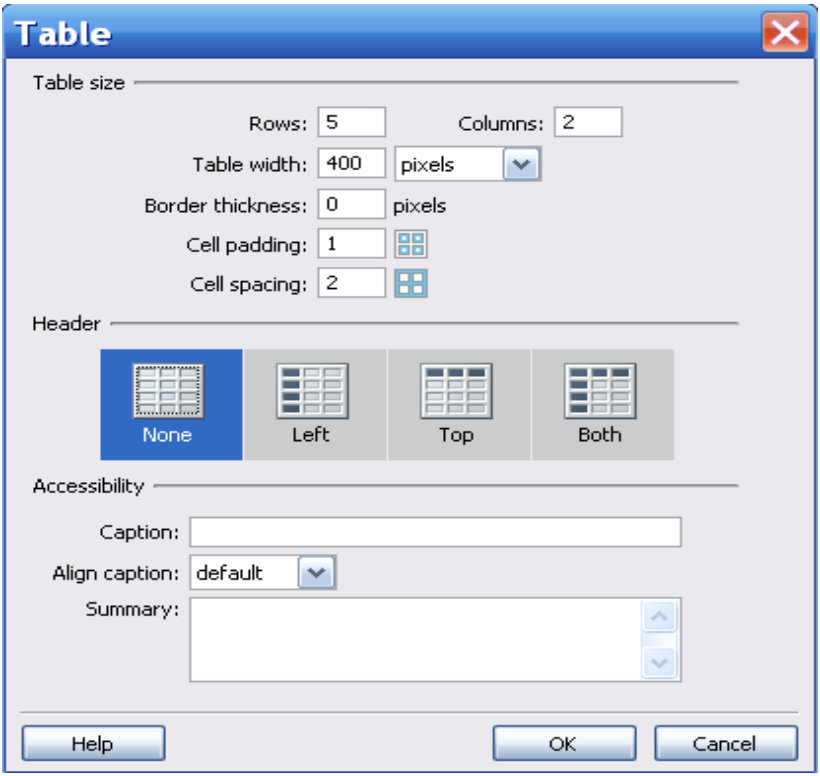
ამიტომ, ჩვენს საკუთარ ვებ-გვერდებზე კატეგორიების, თვით ფაილების და სტატიების სიების წარმოსადგენად, სწორედ ცხრილებს გამოვიყენებთ. ეს მართლაც ძალიან მოხერხებული რამ არის-ამაში ეჭვი ნუ შეგეპარებათ.

მაგრამ ჯერ მოდით შევქმნათ ახალი ვებ-გვერდი. ეს იყოს სტატიების კატეგორიების გვერდი. მივანიჭოთ მას სახელი ***სტატიები,*** ჩავწეროთ მასში რაიმე შესავალი ტექსტი, დავიმახსოვროთ **Site1** საქაღალდეში **Articles.htm** სახელით, და მოვემზადოთ ცხრილებთან გასაცნობად.

ცხრილების შექმნა

დავაყენოთ ტექსტური კურსორი ყველაზე უკანასკნელი აბზაცის ბოლოს და დავაჭიროთ კლავიშას **<Enter>**, რათა შევქმნათ ცარიელი

აბზაცი. (თუ ის უკვე არსებობს, ხელახლა შექმნა არ არის საჭირო). სწორედ აქ შევქნით ჩვენ ჩვენს პირველ ცხრილს.
ცარიელი ცხრილი იქმნება მენიუ **Insert**-ის პუნქტ-Table-ის არჩევით, ან კლავიშების კომბინაციაზე **<Ctrl>+<Alt>+<T>** დაჭერით. ეკრანზე გაჩნდება დიალოგური ფანჯარა Table, რომელიც ნაჩვენებია ნახ.3.19-ზე. ამ ფანჯრის შესატან ველებში **Rows** და **Columns,** შეიტანება შესაბამისად შესაქმნელი ცხრილის სტრიქონებისა და სვეტების რაოდენობა. შევიტანოთ ველში **Rows** რიცხვი 5-ჯერჯერობით ჩვენს ცხრილს ხუთი სტრიქონი ექნება, ხოლო ველში **Columns** შევიტანოთ რიცხვი 2.



ნახ.3.19. დიალოგური ფანჯარა Table

შესატან ველში **Table width** განისაზღვრება ცხრილის სიგანე პიქსელებში ან პროცენტებში გვერდის სიგანის მიმართ.

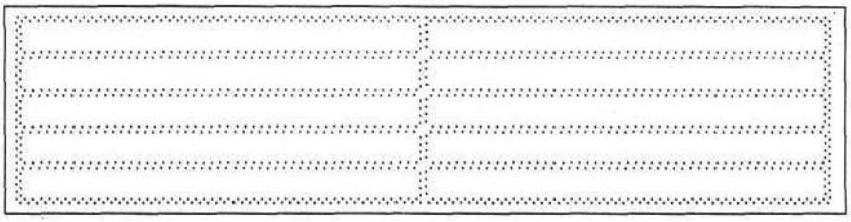
ჩამოსაშლელ სიაში, რომელიც ამ შესატანი ველიდან მარჯვენა მხარეს მდებარეობს, შესაბამისად **pixels** ან **percent** უნდა ავირჩიოთ. მოდით განვსაზღვროთ 400 პიქსელის ტოლი ცხრილის სიგანე-ეს საკმარისი იქნება კატეგორიების სიის შესატანად.

შესატან ველში **Border thickness** მოიცემა ცხრილის საზღვრების სიგანე პიქსელებში. მისი წინასწარ განსაზღვრული (გაჩუმებითი) მნიშვნელობა 1-ის ტოლია. ჩვენ შეგვიძლია 0-ის შეტანა, რომ საერთოდ მოვაშოროთ საზღვრები; მოდით ასეც გავაკეთოთ.

შესატან ველში **Cell padding** განისაზღვრება მანძილი ცხრილის უჯრედის საზღვარსა და მის შიგთავსს შორის პიქსელებში. მისი წინასწარ განსაზღვრული (default,გაჩუმებითი) მნიშვნელობა არის 1-ის ტოლი და დავტოვოთ ის, როგორც არის.

ანალოგიურად, შესატანი ველი **Cell spacing,** გამოიყენება ცხრილის ცალკეული უჯრედების საზღვრებს შორის მანძილის განსაზღვრისათვის.

მისი გაჩუმებითი მნიშვნელობა 2-ის ტოლია და დავტოვოთ იგი, როგორც არის. მართვის დანარჩენი ელემენტები ჩვენ ჯერ-ჯერობით არ დაგვჭირდება. ამიტომ ახლავე დავაჭიროთ **OK**-ღილაკზე. შედეგად ჩვენ მივიღებთ იმას, რაც ნაჩვენებია ნახ.3.20-ზე.



ნახ.3.20. ჩვენი პირველი ცხრილი

მიუხედავად იმისა, რომ ადრე ჩვენ განვსაზღვრეთ ცხრილის საზღვრების ნულოვანი სიგანე (დიალოგური ფანჯარა **Table**-ის, **Border thickness**-შესატან ველში), **DreamweaverMX**-მა მაინც გამოაჩინა **უხილავი საზღვრები,** რომლებიც შედგებიან ვიწრო წერტილოვანი ხაზებისაგან. ვებ-დამთვალიერებელში ეს ხაზები აღარ გამოჩნდება. ჩვენ რომ განგვესაზღვრა ცხრილის **ხილული საზღვრების** არანულოვანი სიგანე, **DreamweaverMX**-ი წარმოაჩენდა მათ უწყვეტი ხაზის სახით.

ახლა დავაყენოთ ტექსტური კურსორი ცხრილის პირველ (ზედა მარცხენა) უჯრაში და დავიწყოთ კატეგორიების სიის აკრეფა. პირველი უჯრის შიგთავსის აკრეფის შემდეგ, ვაჭერთ კლავიშას **<Tab>**, რათა გადავაადგილოთ ტექსტური კურსორი ცხრილის მეორე უჯრაში და გავაგრძელოთ სიის შექმნა.

ჩვენ ამაში **DreamweaverMX**-ი დაგვეხმარება. როდესაც ჩვენ ავკრიფავთ ცხრილის უკანასკნელი უჯრედის შიგთავსს და დავაჭერთ კლავიშას **<Tab>**, ის დაამატებს ცხრილში კიდევ ერთ სტრიქონს. ასე, რომ ჩვენ არ მოგვიწევს ამის კეთება ხელით. მაგრამ, თუ ჩვენ დაგვჭირდება ცხრილის ნებისმიერ ადგილას ახალი სტრიქონის დამატება, ტექსტური კურსორი უნდა მოვათავსოთ იმ სტრიქონის ნებისმიერ უჯრედში, რომლის ზევითაც უნდა ჩაისვას ახალი სტრიქონი, და ავირჩიოთ მენიუ **Modify**-ის, ქვემენიუ **Table**-ის პუნქტი-**Insert Row** ან დავაჭიროთ კლავიშების კომბინაციას **<Ctrl>+<M>.**

ყოველივე აღნიშნულის შედეგად, ჩვენ უნდა მივიღოთ ნახ.3.21.-ზე ნაჩვენები ცხრილის მსგავსი რამ. ჯერ ჩვენ არ შევუდგებით სიის ტექსტის გაფორმებას. ამ ეტაპზე ჯობია, რომ ჩავატაროთ გარკვეული ექსპერიმენტები ჩვენს ახალ ცხრილზე.

ინტერნეტი	გახსნა
ოფისი	გახსნა
მულტიმედია	გახსნა
სისტემა	გახსნა
პროგრამირება	გახსნა
გართობა	გახსნა
სხვადასხვა	გახსნა

ნახ.3.21. ცხრილი-კატეგორიების სია, შევსებული უჯრედებით.

ცხრილებთან მუშაობა

ჩვენ უკვე ვიცით, თუ როგორ უნდა ჩავსვათ ცხრილში ახალი სტრიქონი. ამისათვის საჭიროა ტექსტური კურსორის განთავსება იმ სტრიქონის უჯრედში, რომლის ზემოთაც გვსურს, რომ დაემატოს ახალი სტრიქონი, და ავირჩიოთ მენიუ **Modify**-ის, ქვემენიუ **Table**-ის პუნქტი-**Insert Row** ან დავაჭირთ კლავიშების კომბინაციას **<Ctrl>+<M>**.

ანალოგიურად შეიძლება ჩაისვას ცხრილში ახალი სვეტიც. ვათავსებთ ტექსტურ კურსორს უჯრაში, რომლის მარჯვენა მხარესაც გვინდა, რომ ჩაისვას ახალი სვეტი და ვირჩევთ მენიუ **Modify**-ის, ქვემენიუ **Table**-ის პუნქტ-**Insert Column**-ს ან ვაჭერთ კლავიშების კომბინაციას **<Ctrl>+<Shift>+<A>.**

ასევე მარტივია ნებისმიერი სტრიქონის ან სვეტის წაშლა. სტრიქონის წასაშლელად ტექსტურ კურსორს ვათავსებთ წასაშლელი სტრიქონის ნებისმიერ უჯრაში და ვირჩევთ ან მენიუ **Modify**-ს, ქვემენიუ **Table**-ის პუნქტს-**Delete Row** ან ვაჭერთ კლავიშების კომბინაციას **<Ctrl>+<Sift>+<M>.** ხოლო სვეტის წასაშლელად, რომლის რომელიმე უჯრაში მოთავსებულია ტექსტური კურსორი, ვირჩევთ მენიუ **Modify**-ს, ქვემენიუ **Table**-ის, კონტექსტური მენიუს პუნქტს-Delete **Column** ან ვაჭერთ კლავიშების კომბინაციას **<Ctrl>+<Shift>+<->.**

თვით ცხრილის წაშლა ყველაზე მარტივია, ტეგების სექციის გამოყენებით. ტექსტურ კურსორს ვათავსებთ ცხრილის ნებისმიერ უჯრაში, ვაწკაპუნებთ ტეგების სექციის ღილაკზე **<table>** და ვაჭერთ კლავიშა **<Del>**-ს.

როდესაც ჩვენ შეგვყავდა ტექსტი ცხრილის უჯრედებში, უჯრედების ზომები იცვლებოდნენ ავტომატურად, რათა შიგთავსი ჩატეულიყო მათში. რა თქმა უნდა ეს სასარგებლო რამ არის, მაგრამ ხშირად ეს ზომები იცვლებიან სრულიად გაუგებარი წესით. იმისათვის, რომ ასეთი რამ არ მოხდეს, მოდით თვითონ განვსაზღვროთ უჯრედების ზომები, რათა ვებ-დამთვალიერებელი თვითნებურად არ მოიქცეს.

მოვათავსოთ "მაუსის" კურსორი უჯრედების გამყოფ ვერტიკალურ საზღვარზე და გადმოვქაჩოთ ის ისეთნაირად, რომ მარჯვენა სვეტი იყოს, რაც შეიძლება ვიწრო, რომ მასში ჩაეტიოს მხოლოდ სიტყვა **გახსნა**. ამის შემდეგ ჩვენი ცხრილის ორივე სვეტის სიგანე იქნება მკაცრად დაფიქსირებული. ზუსტად ასევე, ჩვენ შეგვიძლია სტრიქონების სიმაღლის შეცვლა-საჭირო უჯრედებს შორის არსებული საზღვრის გადაქაჩვით "მაუსის" საშუალებით. და ბოლო შტრიხი-თვით ცხრილის სიგანისა და სიმაღლის შეცვლა მისი მარჯვენა ან ქვედა საზღვრის გადაქაჩვით.

ცხრილის, მისი სტრიქონებისა და სვეტების ზომების შეცვლასთან დაკავშირებული ექსპერიმენტებით კმაყოფილებმა, მოდით დავიმახსოვროთ იგი. ახლ კი ნაირფეროვნებისთვის გადავერთოთ **HTML** კოდის წარმოჩენის რეჟიმში, რათა შევხედოთ, თუ რა ტეგების საშუალებით ხდება ჩვენს მიერ შექმნილი ცხრილის ფორმირება. ჩვენ

იქ დავინახავთ ტეგების ისეთ კორიანტელს, რომ შეიძლება შეგვეშინდეს კიდევაც.

როგორ ხდება ცხრილების ფორმატირება

სინამდვილეში არაფელი საშინელი აქ არ არის. საკმარისია გავიგოთ პრინციპი, რომლის მიხედვითაც ხდება **HTML** ცხრილის ფორმირება, რათა გაგვიიოლდეს მის კოდში ორიენტირება. მოდით ნაწილ-ნაწილ განვიხილოთ **HTML** კოდი, რომლის მიხედვითაც ხდება ცხრილის ფორმირება. ჯერ ჩვენ უნდა შევქმნათ თვით ცხრილი. ეს ხდება წყვილი ტეგის **<TABLE>** საშუალებით:

<TABLE WIDTH=”400” BORDER=”0” CELLSPACING=”2” CELLPADDING=”1”> </TABLE>

ატრიბუტი **WIDTH** განსაზღვრავს ცხრილის სიგანეს, ატრიბუტი **BORDER**-ხილული საზღვრის სისქეს, ატრიბუტები **CELLSPACING** და **CELLPADDING**-შესაბამისად მანძილს მეზობელი უჯრედების საზღვრებს შორის და უჯრის საზღვარსა და მის შიგთავსს შორის.
ყველა ეს პარამეტრი ჩვენ განვსაზღვრეთ დიალოგურ ფანჯარაში **<TABLE>** (იხ.ნახ.3.19.).

შემდეგ, წყვილი ტეგის **<TR>** საშუალებით ჩვენ ვახდენთ ცხრილის სტრიქონების ფორმირებას:

<TABLE WIDTH=”400” BORDER=”0” CELLSPACING=”2” CELLPADDING=”1”>
<TR>
</TR>
</TABLE>

აქ ჩვენ მოვახდინეთ მხოლოდ ერთი სტრიქონის ფორმირება-მაგალითისათვის ესეც საკმარისია. შევნიშნოთ, რომ ტეგი <TR> შესაძლებელია იმყოფებოდეს მხოლოდ ტეგის-<TABLE> შიგნით. წინააღმდეგ შემთხვევაში ვებ-დამთვალიერებელი მას არასწორად დაამუშავებს.
შემდეგი ნაბიჯი-ცხრილის უჯრედების ფორმირება წყვილი ტეგის <TD> საშუალებით:

<TD>:
<TABLE WIDTH=”400” BORDER=”0” CELLSPACING=”2”
CELLPADDING=”1”>
<TR>
<TD WIDTH=”327”>ინტერნეტი</TD>
<TD WIDTH=”63”>გახსნა</TD>
</TR>
</TABLE>

აქ ჩვენ სტრიქონში ორი უჯრა მოვათავსეთ, რომლებიც შეიცავენ ტექსტებს ***ინტერნეტი*** და ***გახსნა***. ატრიბუტი **WIDTH** ამ შემთხვევაშიც განსაზღვრავს უჯრედის სიგანეს. ამ შემთხვევაშიც ტეგი **<TD>** მოთავსებული უნდა იყოს მხოლოდ **<TR>** ტეგის შიგნით. წინააღმდეგ შემთხვევაში ვებ-დამთვალიერებელი ვერ დაამუშავებს მას.

ყოველივე ეს შეიძლება ძალიან რთულად მოგვეჩვენოს. მაგრამ სინამდვილეში არავითარი სირთულე აქ არ არის, ყველაფერი ძალზე მარტივია, თუ გავიგებთ პრინციპს. უფრო მეტიც, ცხრილების ფორმირების მსგავსი ხერხი მეტად მოქნილია და სწორედ მოქნილობას შეეწირა ცხრილის **HTML**-კოდის კომპაქტურობა.

ზოგადად **HTML**-ცხრილები-ძალზე მძლავრი რამ არის! ცხრილის უჯრებში შეგვიძლია მოვათავსოთ ყველაფერი რასაც მოვისურვებთ: განუზომელი სიდიდის ტექსტი, გრაფიკული გამოსახულება და თუნდაც სხვა ცხრილი. საჭირო **HTML**-კოდი უბრალოდ შესაბამის **<TD>** ტეგში თავსდება. ერთადერთ საშიშროება არსებობს: თვითონ არ უნდა გავებათ **HTML**-კოდის ხლართებში და არ გადავრიოთ ვებ-დამთვალიერებელი, რომელსაც მოუწევს ამის ყველაფრის ეკრანზე გამოტანა.

უფრო რთული ცხრილები

მოდით დავიმახსოვროთ და დავხუროთ ჩვენი ვებ-გვერდი **Articles.htm.** ახლა ჩვენ შევუდგებით ინტერნეტის თემაზე სტატიების სიის შემცველი ვებ-გვერდის შექმნას. შევქმნათ ახალი ცარიელი ვებ-გვერდი და დავარქვათ მას-**სტატიები-ინტერნეტი**, შევიტანოთ რაიმე საწყისი (შესავალი) ტექსტი და შევინახოთ იგი სახელწოდებით-**Articles_internet.htm**. ზოგადად უნდა ვეცადოთ, რომ ჩვენი გვერდების ფაილებს მივანიჭოთ ”მეტყველი” სახელები-ეს ჩვენ გაგვიადვილებს საქმეს მომავალში. სტატიების სია ისევ ცხრილში

მოვათავსოთ. ეს ცხრილი სამი სვეტისაგან იქნება შემდგარი: ავტორის სახელი, სტატიის დასახელება და "გახსნა"-ს ტიპის ჰიპერმინიშნება, ისევე, როგორც კატეგორიების სიაში. დავსვათ ტექსტური კურსორი შესავალი ტექსტის შემდეგ და ავირჩიოთ მენიუ **Insert**-ის პუნქტი **Table** ან დავაჭიროთ კლავიშების კომბინაციას **<Ctrl>+<Alt>+<T>**. ეკრანზე გაჩნდება დიალოგური ფანჯარა **Table** (იხ.ნახ.3.19).

ვთქვათ ჩვენი ცხრილი შეიცავს ორ სტრიქონს (შესატანი ველი **Row**) და როგორც ჩვენ წინასწარ მოვილაპარაკეთ, სამ სვეტს (შესატანი ველი **Columns**). ვთქვათ, რომ სიგანით ის მთელ ვებ-გვერდს იკავებს, ამიტომ შეტანის ველში **Table width** შევიტანოთ მნიშვნელობა 100, ხოლო ჩამოსაშლელ სიაში მარჯვნივ, ავირჩიოთ პუნქტი **percent**.

ჩვენს რიგით მეორე ცხრილს, **Border thickness**-ის შეტანის ველში ერთიანის ჩაწერით, განუსაზღვროთ 1 პიქსელის ტოლი ხილული საზღვრები. შეტანის ველების **CELLPADDING** და **CELLSPACING** მნიშვნელობები დავტოვოთ შესაბამისად 1-ის და 2-ის ტოლი.

მოდით, ჩვენს ცხრილს გავუკეთოთ ნორმალური "ქუდი". ამისათვის გამოვიყენოთ გადამრთველების ანაკრები **Header**, რომელიც ფანჯარა **Table**-ის ქვედა ნახევარშია განთავსებული. ამ გადამრთველთაგან ნებისმიერს გააჩნია "მეტყველი" გამოსახულება, ამიტომაც ცხადია, რომ ჩვენ უნდა ჩავრთოთ გადამრთველი **Top**. ჩვენ სწორედ ასეც მოვიქცევით.

დაგვრჩა მხოლოდ **OK** ღილაკზე დაჭერა, რომ **DreamweaverMX**-მა შექმნას ჩვენთვის სასურველი ცხრილი. ბუნებრივია, რომ შექმნილი ცხრილის უჯრედები უნდა შეივსოს შიგთავსით. ჩვენ უნდა მივიღოთ ასეთი სიმპათიური ცხრლილი-იხ.ნახ.3.22.

author	name	
მანჯგალაძე ე.	დაპროგრამების საკითხები	გახსნა
თარგამაძე მ.	მონაცემთა ბაზების საფუძვლები	გახსნა
აფრიდონიძე გ.	ოპერაციული სისტემების შესახებ	გახსნა

ნახ.3.22. სტატიების სიის ცხრილი

მოდით ყურადრებით დავაკვირდეთ ამ ნახატს. რა არის მასში საინტერესო?
პირველ რიგში, ჩვენს ცხრილს, ვიწრო უწყვეტი "სამგანზომილებიანი" ხაზის სახით, გაუჩნდა ლამაზი ხილული საზღვრები. უფრო მეტიც, ეს ხაზები გარს ერტყმიან არა მხოლოდ ცხრილს, არამედ მის ცალკეულ უჯრედებსაც, ასე რომ საზღვრები ფაქტიურად ორმაგი გამოდის. ის ნამდვილად ეფექტურად გამოიყურება; ისე ჩანს, თითქოს უჯრედები ჩაზნექილი იყოს ვებ-გვერდის ფონში.
მეორე რიგში, ცხრილის პირველ სტრიქონში მოთავსებული ტექსტი, აკრეფილია მუქი ფერით, თუმცა ეს ჩვენ არ გაგვიკეთებია. რატომ? საქმე იმაშია რომ, **DreamweaverMX**-მა პირველი სტრიქონის უჯრედების ფორმირება ცოტა სხვანაირად მოახდინა: არა **<TD>** ტეგის, არამედ **<TH>** ტეგის საშუალებით. ხოლო **<TH>** ტეგის შიგთავსი ყოველთვის გამოყოფილია მუქი ფერით.
ცხრილის პირველი სტრიქონის აღმწერი **HTML** კოდის ფრაგმენტი ასე გამოიყურება:

```
<TABLE WIDTH="100%" BORDER="1" CELLSPACING="2" CELLPADDING="1">
<TR>
<TH SCOPE="col">ავტორი</TH>
<TH SCOPE="col">დასახელება</TH>
<TH SCOPE="col"> </TH>
</TR>
. . .
```

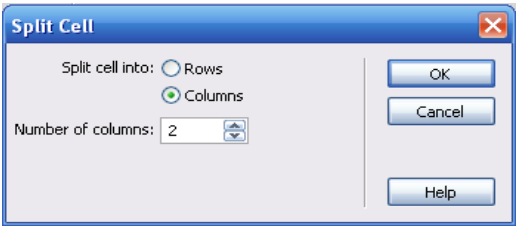
ატრიბუტი **SCOPE**, **col** მნიშვნელობით **<TH>** უჯრედს განსაზღვრავს, როგორც სათაურს ცხრილის მთელი სვეტისათვის. პრინციპში, ის არასავალდებულოა და თანამედროვე **Web**-დამთვალიერებლების მიერ სრულყოფილადაც არ აღიქმება, მაგრამ **DreamweaverMX**-მა ეტყობა თავის დაზღვევა გადაწყვიტა. ჩვენ არ მივაქციოთ ამ ატრიბუტს მნიშვნელობა.
უკანასკნელ ცარიელ უჯრედში **DreamweaverMX**-მა მოათავსა უწყვეტი ჰარის (გამოტოვების) სიმბოლო ** **. ეს აუცილებელია იმისათვის, რომ ცხრილი სწორად იქნას წარმოჩენილი ვებ-

დამთვალიერებლების ძველ პროგრამებში. უკეთესია, თუ ამ სიმბოლოს დავტოვებთ.
თითქოს ყველაფერი კარგად არის. მხოლოდ, პირველ სტრიქონში რიგით მესამე უჯრა დარჩა ცარიელი და გამოიყურება არც თუ ისე ლამაზად. მოდით გავაერთიანოთ ის მეორე უჯრასთან და ორი უჯრა ერთ უჯრად ვაქციოთ.
ჯერ ჩვენ უნდა მოვნიშნოთ ორი გასაერთიანებელი უჯრა. "მაუსის" კურსორი მოვათავსოთ მეორე უჯრაში, დავაჭიროთ მის მარცხენა ღილაკს და ხელის აუშვებლად გადავათრიოთ მარჯვნივ, სანამ ორივე გასაერთიანებელი უჯრა არ გახდება მონიშნული შავი სქელი ხაზით. ამის შემდეგ დავაჭიროთ ღილაკს  , რომელიც მდებარეობს თვისებების რედაქტორში. შედეგი ნაჩვენებია ნახ. 3.23-ზე.

author	name	
მანჯგალაძე ე.	დაპროგრამების საკითხები	გახსნა
თარგამაძე მ.	მონაცემთა ბაზების საფუძვლები	გახსნა
აფრიდონიძე გ.	ოპერაციული სისტემების შესახებ	გახსნა

ნახ.3.23. უჯრედების გაერთიანების შედეგი

ზუსტად ასევე შეიძლება უჯრედების გაერთიანება ვერტიკალშიც. არის შესაძლებელი მრავალი უჯრის გადაქცევა ერთ უჯრად-**HTML**-ის სტანდარტები არ ზღუდავენ მათ რაოდენობას. თუ საჭიროა გაერთიანებული უჯრდის რამდენიმე უჯრედად დაყოფა, მაშინ უნდა დავაყენოთ მასზე ტექსტური კურსორი და დავაჭიროთ ღილაკს  , რომელიც აგრეთვე თვისებების რედაქტორში მდებარეობს. ამის შემდეგ ეკრანზე გაჩნდება დიალოგური ფანჯარა **Split Cell**, რომელიც ნაჩვენებია ნახ.3.24-ზე.



ნახ.3.24. დიალოგური ფანჯარა Split Cell

გადამრთველების ჯგუფი **Split cell into** განსაზღვრავს, თუ როგორ გაიყოფა უჯრედი. კერძოდ გადამრთველი **Rows** განსაზღვრავს უჯრედის ჰორიზონტალურად გაყოფას რამოდენიმე სტრიქონად, რომელთა რაოდენობა დგინდება მთვლელის ველში **Number of rows**. თუ კი არჩეულია გადამრთველი **Columns**, უჯრა გაიყოფა ვერტიკალური მიმართულებით, რამოდენიმე სვეტად, რომელთა რაოდენობა დგინდება მთვლელის ველში **Number of columns**. **OK** ღილაკის დაჭერის მერე გაერთიანებული უჯრა ისევ გაიყოფა.
ახლა შევხედოთ ჩვენი ცხრილის **HTML** კოდს. მას ასეთი სახე ექნება:

```
<TABLE WIDTH=”100%” BORDER=”1” CELLSPACING=”2” CELLPADDING=”1”>
<TR>
<TH SCOPE=”col”>ავტორი</TH>
<TH COLSPAN=”2” SCOPE=”col”>დასახელება</TH>
</TR>
. . .
```

აქ დაემატა ატრიბუტი **COLSPAN**, რომელიც განსაზღვრავს ჰორიზონტალურად გასაერთიანებელი უჯრედების რაოდენობას. ჩანს, რომ იგი მოთავსებულია გასაერთიანებელი უჯრედებიდან ყველაზე მარცხენა უჯრედის ტეგში **<TH>**. რაც შეეხება მესამე უჯრედს, ის **HTML** კოდში არ არის აღწერილი, ვინაიდან ფაქტიურად მეორე გასაერთიანებელი უჯრედის ნაწილი გახდა.
უჯრედების გაერთიანება ვერტიკალურად, ხორციელდება ანალოგიური ატრიბუტის, **ROWSPAN** საშუალებით. ქვემოთ მოყვანილია ცხრილის **HTML** კოდი, რომელშიც უჯრედები გაერთიანებულია ვერტიკალურად:

```
<TABLE WIDTH=”100%” BORDER=”1” CELLSPACING=”2” CELLPADDING=”1”>
  <TR>
      <TD ROWSPAN=”2”> ჯაფარიძე ზ.,თარგამაძე მ. </TD>
      <TD> კომპიუტერის საიმედოობა  </TD>
      <TD>  გახსნა  </TD>
</TR>
<TR>
     <TD> დაპროგრამების საკითხები </TD>
     <TD> გახსნა </TD>
 </TR>
 <TR>
      <TD> თარგამაძე მ. </TD>
       <TD>მონაცემთა ბაზების საფუძვლები</TD>
      <TD> გახსნა </TD>
  </TR>
</TABLE>
```

ამ ცხრილის პირველი და მეორე სტრიქონის პირველი უჯრედები გაერთიანებულია ერთში-ამაზე მეტყველებს ატრიბუტი **ROWSPAN** , რომლის მნიშვნელობაც 2-ის ტოლია. ამასთან მეორე სტრიქონის პირველი უჯრედი ასევე არ არის განსაზღვრული, ვინაიდან ის გაერთიანებული უჯრედის ნაწილი ხდება.
ამით ჩვენ მოვრჩით ცხრილებთან მუშაობას. ახლა მოდით ვისაუბროთ იმაზე, თუ როგორ მოვათავსოთ ვებ-გვერდზე გრაფიკული გამოსახულება **DreamweaverMX**-ში.

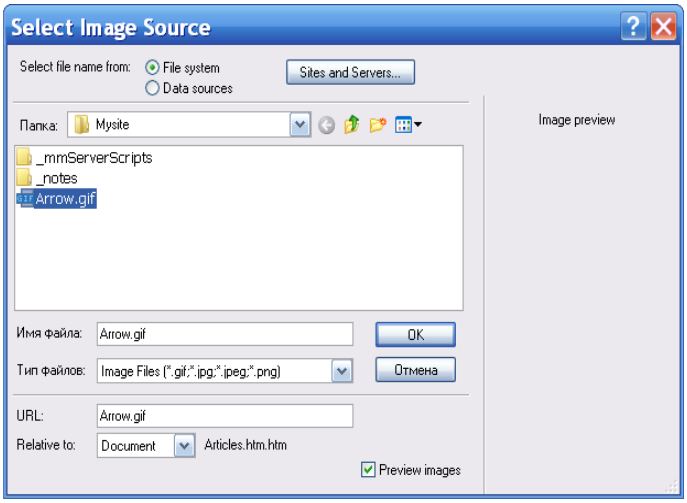
გრაფიკული გამოსახულებების ჩასმა

კიდევ ერთხელ შევხედოთ გვერდს **Articles_internet.htm**. რა შეიძლება კიდევ მას გაუკეთდეს? კერძოდ, ცხრილის მესამე სვეტში მოთავსებულია მომავალი ჰიპერმინიშნებები. მართალი რომ ვთქვათ, არც თუ ისე სახარბიელოდ გამოიყურებიან ისინი. მოდით შევცვალოთ რაიმე უფრო მისაღებით.
გავხსნათ ნებისმიერი გრაფიკული რედაქტორი, რომელიც იძლევა გამოსახულების **GIF** ფორმატში შენახვის საშუალებას (აქ ჩვენ სავსებით დაგვაკმაყოფილებს **Windows**-ის კომპლექტაციაში შემავალი უმარტივესი რედაქტორი **Paint**-იც). დავხატოთ მისი

საშუალებით პატარა გამოსახულება, რომელსაც მარჯვნივ მიმართული ისრის სახე ექნება და დავიმახსოვროთ იგი ფაილში **Arrow.gif.** ამ გამოსახულებით შევცვალოთ ჩვენი ტექსტური მინიშნებები.

პირველყოვლისა, მესამე სვეტის უჯრიდან ამოვშალოთ ტექსტი ***გახსნა***. ამის შემდეგ შევამოწმოთ დაყენებულია თუ არა ამ უჯრაში ტექსტური კურსორი და მენიუ **Insert**-ში ავირჩიოთ პუნქტი **Image**, ან დავაჭიროთ კლავიშების კომბინაციას **<Ctrl>+<Alt>+<I>.** ეკრანზე გაჩნდება დიალოგური ფანჯარა Select Image Source, რომელიც ნაჩვენებია ნახ. 3.25.-ზე.

საქაღალდეებისა და ფაილების ჩამოსაშლელი სია, საშუალებას გვაძლევს ავირჩიოთ სასურველი საქაღალდე და ფაილი. შესატან ველში ***ფაილის დასახელება*** გაჩნდება ჩვენს მიერ არჩეული ფაილის სახელი (შესაძლებელია მისი ხელით შეტანაც). ეს ყველაფერი ჩვენთვის ცნობილია **Windows**-ის ფაილების გახსნისა და დახურვის სტანდარტული დიალოგური ფანჯრების გამოყენების პრაქტიკიდან. ერთადერთი სხვაობა-ეს არის ის, რომ ფანჯრის მარჯვენა მხარეს მდებარეობს წინასწარი დათვალიერების პანელი, სადაც მოცემულ მომენტში ჩანს ჩვენს მიერ დახატული ისარი. შეიძლება ამ პანელის გაქრობა, თუ გამოვრთავთ ალამს **Preview image**.



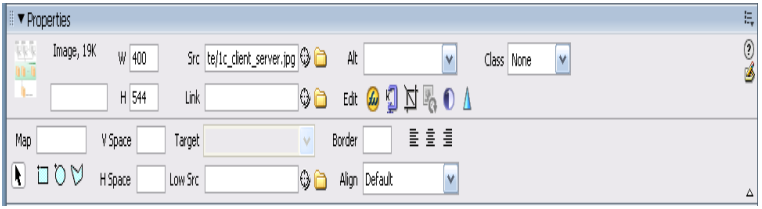
ნახ.3.25. დიალოგური ფანჯარა Select Image Source

მაშ ასე ჩვენ ავირჩიეთ ფაილი, სადაც ინახება ჩვენი ისრის გამოსახულება. დაგვრჩა მხოლოდ OK ღილაკზე დაჭერა. ამის შემდეგ ჩვენს მიერ არჩეული გამოსახულება მოთავსებული იქნება ვებ-გვერდზე.

ჩასმისთანავე **DreamweaverMX**-ი მონიშნავს ამ გამოსახულებას. (თუ ის მონიშნული არ არის, უნდა დავაწკაპუნოთ მასზე "მაუსით"). მონიშნული გამოსახულება შემოფარგლულია შავი ვიწრო ჩარჩოთი, რომლის მარჯვენა და ქვედა საზღვრებზე განთავსებულია პაწაწინა შავი კვადრატები. ისინი ეგრეთ წოდებულ **ზომების შეცვლის მარკერებს** წარმოადგენენ. ჩვენ შეგვიძლია "მაუსით" "ჩავავლოთ" ნებისმიერ მარკერს და გადავათრიოთ ის სხვა ადგილას, რითაც ჩვენ შევცვლით გამოსახულების ჰორიზონტალურ და ვერტიკალურ ზომებს. თუ კი ჩვენ გვინდა ორივე ზომის პროპორციულად შეცვლა, **<Shift>** კლავიშაზე ხელის აუღებლივ გადავათრიოთ მარკერი, რომელიც ჩარჩოს მარჯვენა ქვედა კუთხეში მდებარეობს.

ჩვენი ისარი ძალიან პატარაა-მომხმარებელს გაუჭირდება მისი დაჭერა. ამიტომ, მოდით ცოტათი "გავწელოთ" იგი ჰორიზონტალური მიმართულებით, რომ მან ცხრილის მესამე უჯრა მთლიანად შეავსოს.

გამოსახულების ზომების შეცვლა შესაძლებელია სხვანაირადაც. თუ მოვნიშნავთ გამოსახულებას, თვისებების რედაქტორი წარმოაჩენს მართვის ელემენტების ანაკრებს (ნახ.3.26), რომელთა დანიშნულება გამოსახულებების სხვადასხვა პარამეტრების განსაზღვრაში მდგომარეობს. მათ შორის ამ პანელზე გამოჩნდება შესატანი ველები **W** და **H**, რომლებშიც განისაზღვრება შესაბამისად მონიშნული გამოსახულების სიგანე და სიმაღლე.



ნახ.3.26. თვისებების რედაქტორის სახე, მონიშნული გრაფიკული გამოსახულების შემთხვევაში.

თვისებების რედაქტორში ასევე განთავსებულია ჩამოსაშლელი სია **Alt,** რომლის დანიშნულებაც ალტერნატიული ტექსტის შეტანაა- ტექსტისა, რომელსაც ვებ-დამთვალიერებელი გამოიყვანს ეკრანზე, თუ რაიმე მიზესის გამო ვერ მოახერხებს გამოსახულების ფაილის ჩატვირთვას.

საჭირო ტექსტი შეიტანება პირდაპირ ამ სიაში, როგორც შესატან ველში.

რომ წავშალოთ გამოსახულება, რომელიც აღარ გვჭირდება ან არასწორად არის ჩასმული, საკმარისია მოვნიშნოთ იგი "მაუსზე" დაწკაპუნებით და დავაჭიროთ კლავიშა <Del>-ს.

მოდით, ახლა მოვათავსოთ ასეთივე ისარი, ცხრილის მესამე სვეტის დანარჩენ უჯრებშიც, რისთვისაც:

1."მაუსის" დაწკაპუნებით მოვნიშნოთ ისრის გამოსახულება, თუ ის ჯერ კიდევ მონიშნული არ არის;

2.დავაკოპიროთ იგი **Windows**-ის გაცვლის ბუფერში, კლავიშების კომბინაციაზე-**<Ctrl>+<C>** დაჭერით. (შეგვიძლია აგრეთვე ავირჩიოთ მენიუ **Edit**-ის პუნქტი **Copy**, მაგრამ ამას ცოტათი მეტი დრო ჭირდება) ;

3.წავშალოთ ტექსტი ***გახსნა*** მესამე სვეტის მეორე უჯრიდან და დავაყენოთ მასზე კურსორი;

4.ჩავსვათ უჯრაში გამოსახულება გაცვლის ბუფერიდან, კლავიშების კომბინაციაზე **<Ctrl>+<P>** დაჭერით (ან მენიუ **Edit**-ის პუნქტ **Paste**-ის არჩევით);

5.გავიმეოროთ პუნქტები 3 და 4, ცხრილის მესამე სვეტის მესამე უჯრისათვის.

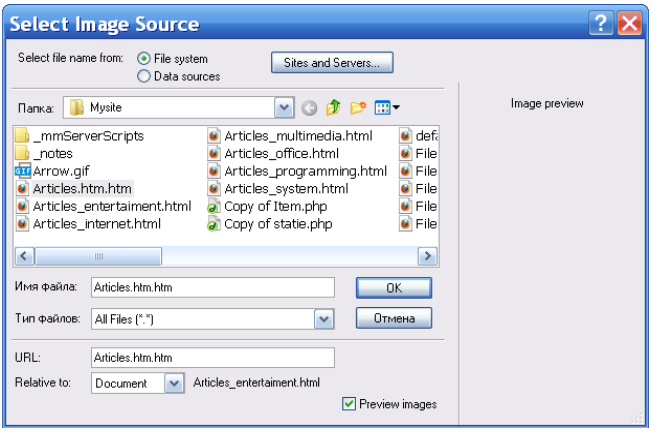
დავიმახსოვროთ გვერდი **Articles_internet.htm** და დავხუროთ იგი. ამის შემდეგ გავხსნათ გვერდი **default.htm.** ამისათვის ავირჩიოთ მენიუ **Edit**-ის პუნქტი **Open** ან დავაჭიროთ კლავიშების კომბინაციას **<Ctrl>+<O>.** ეკრანზე გაჩნდება **Windows**-ის ფაილის გახსნის სტანდარტული დიალოგური ფანჯარა; ავირჩიოთ მასში საჭირო ფაილი და დავაჭიროთ გახსნის ღილაკს.

ჯერ-ჯერობით გადავდოთ ჩვენი საიტის დანარჩენი გვერდების შექმნა. ამ ეტაპზე ჯობია გავიგოთ, თუ როგორ ხდება ჰიპერმინიშნებების შექმნა **DreamweaverMX**-ში.

ჰიპერმინიშნებების შექმნა

რადგან სტატიების კატეგორიების სიის შემცველი ვებ-გვერდი **Articles.htm**, ჩვენ უკვე მზად გვაქვს, მოდით **Default.htm** გვერდზე

შევქმნათ ჰიპერმინიშნება, რომელიც ამ გვერდზე გადაგვიყვანს. ცხადია, რომ ჰიპერმინიშნებად გამოვიყენებთ წარწერას ***სტატიები***, ამიტომ მოვნიშნოთ ის. **DreamweaverMX**-ში ჰიპერმინიშნების შესაქმნელად რამოდენიმე ხერხი არსებობს. განვიხილოთ დამწყები ვებ-დიზაინერებისათვის ყველაზე მარტივი. ავირჩიოთ მენიუ **Modify**-ს პუნქტი **Make Link**, ან დავაჭიროთ ღილაკების კომბინაციას **<Ctrl>+<L>**. ეკრანზე გამოჩნდება დიალოგური ფანჯარა **Select File**, რომელიც ნაჩვენებია ნახ.3.27.-ზე. დაგვრჩა მხოლოდ ფაილების სიიდან ჩვენთვის საჭირო ფაილის არჩევა და **OK** ღილაკზე დაჭერა.



ნახ.3.27. დიალოგური ფანჯარა Select File

ადრე დასახელებული ხერხი, მხოლოდ ფარდობითი ინტერნეტ-მისამართის განსაზღვრის საშუალებას იძლევა, რომელიც მიუთითებს ფაილის მდებარეობაზე მიმდინარე ვებ-გვერდის მიმართ. თუ გვჭირდება აბსოლუტური მისამართის მქონე ჰიპერმინიშნების შექმნა, რომელიც სერვერის მისამართს შეიცავს, მაშინ საჭირო მისამართი უნდა ჩავწეროთ **Select File** დიალოგური ფანჯრის **URL**-ის შესატან ველში და ცხადია დავაჭიროთ ღილაკს OK.

თუ ჩვენ ზუსტად ვიცით ვებ-გვერდის მისამართი, რომელზეც ჰიპერმინიშნებამ უნდ გადაგვიყვანოს, ჩვენ კიდევ უფრო მარტივად შეგვიძლია მოვიქცეთ. თვისებების რედაქტორში (იხ. ნახ.3.7.) არის

ჩამოსაშლელი სია **Link**-ი, რომელშიც, როგორც შესატან ველში, შეიტანება ბილიკი საჭირო ფაილამდე და მისი სახელი. (თუ ეს სახელი ადრე იყო შეტანილი, ჩვენ ასევე შეგვიძლია მისი არჩევა სიიდან). ფაილის სახელის შეტანის შემდეგ უნდა დავაჭიროთ კლავიშას **<Enter>-Dreamweaver**-ი შექმნის ჩვენთვის ჰიპერმინიშნებას. ამავე ჩამოსაშლელი სიის დახმარებით შესაძლებელია ჰიპერმინიშნების ინტერნეტ-მისამართის შეცვლაც. ჰიპერმინიშნების წაშლა ყველფერზე ადვილია. ტეგების სექციაში **<a>** ღილაკზე დაჭერით ვაყენებთ მასზე ტექსტურ კურსორს, და ვაჭერთ **<Del>**-ს.

ჩამოსაშლელი სიის **Link**-ის ქვემოთ მდებარეობს მეორე ჩამოსაშლელი სია-Target. ამ სიის საშუალებით განისაზღვრება ჰიპერმინიშნების მიზანი. პუნქტი**_blank**, ვებ-დამთვალიერებელს მიანიშნებს იმაზე, რომ მან ვებ-გვერდი, რომელზედაც ჰიპერმინიშნება მიუთითებს, გახსნას ახალ ფანჯარაში, ხოლო პუნქტი**_self** ან მნიშვნელობის არარსებობა-გახსნას ვებ-გვერდი იმავე ფანჯარაში.

ჰიპერმინიშნება, რომელიც მიუთითებს ელექტრონული ფოსტის მისამართზე **(საფოსტო ჰიპერმინიშნება),** ოდნავ სხვანაირად იქმნება. მოვნიშნოთ სიტყვები **ამ მისამართზე** (по этому адресу) და მენიუ **Insert**-ში ავირჩიოთ პუნქტი **Email Link**. ეკრანზე გაჩნდება მცირე ზომის დიალოგური ფანჯარა **Email Link**, რომელიც ნაჩვენებია ნახ.3.28.

ნახ.3.28. დიალოგური ფანჯარა Email Link

ველში **Text** თვითონ **DreamveaverMX**-ი ჩასვამს ტექსტს, რომელიც მონიშნულია დოკუმენტის ფანჯარაში (ჩვენს შემთხვევაში-სიტყვები **по этому адресу-ამ მისამართზე**). ჩვენ კი **E-Mail** ველში შევიტანოთ

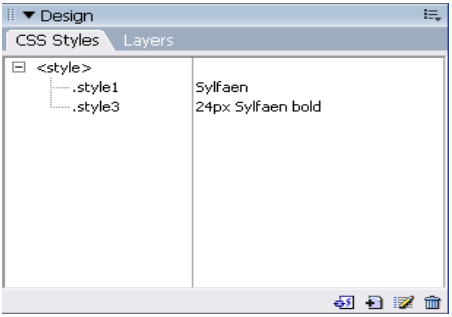
ჩვენთვის საჭირო საფოსტო მისამართი. ამის შემდეგ დაგვრჩება მხოლოდ **OK** ღილაკზე დაჭერა.
თუ ამის შემდეგ ჩვენ გადავერთვებით **HTML** კოდის წარმოჩენის რეჟიმში, დავინახავთ, რომ ჩვენს მიერ შექმნილი საფოსტო ჰიპერმინიშნების ინტერნეტ-მისამართს აქვს შემდეგი სახე:

mailto:admin@mail.somesite.ru

ესეიგი ჩვეულებრივი ჰიპერმინიშნებებისაგან, რომლებიც ფაილებზე მიუთითებენ, საფოსტო ჰიპერმინიშნება განსხვავდება იმით, რომ მას საფოსტო მისამართის წინ გააჩნია სიმბოლო **mailto:** ამასთან მათ შორის არ უნდა იყოს გამოტოვებული ადგილი. რათქმა უნდა ჩვენ შეგვიძლია საფოსტო მისამართის თვისებების რედაქტორის **Link** ჩამოსაშლელ სიაში პირდაპირ შეტანაც ადრე აღნიშნული ხერხით. სხვათაშორის ეს უფრო მარტივი და სწრაფად განხორციელებადია.
ჰიპერმინიშნებად შეიძლება ვაქციოთ არა მხოლოდ ტექსტი, არამედ გრაფიკული გამოსახულებაც. ასე მაგალითად, ჩვენ შეგვიძლია შევქმნათ ჰიპერმინიშნება, რომლის შიგთავსი იქნება **Articles_internet.htm** ვებ-გვერდზე, ცხრილის მესამე სვეტის უჯრედებში, ჩვენს მიერ მოთავსებული ისრებიდან ერთ-ერთი. ეს ზუსტად ისე კეთდება, როგორც ადრე იყო აღწერილი: მოვნიშნავთ ისარს, შევიყვანთ სასურველ ინტერნეტ-მისამართი თვისებების რედაქტორის ჩამოსაშლელ **Link** სიაში და დავაჭერთ კლავიშას **<Enter>**.
უკეთესი იქნება, თუ ჰიპერმინიშნების მიზნის სახით, ჩვენ განვსაზღვრავთ ვებ-დამთვალიერებლის ახალ ფანჯარას, თვისებების რედაქტორის ჩამოშლად **Target** სიაში პუნქტ **_blank**-ის არჩევით. ყველა ჩვენი სტატია გაიხსნას ახალ ფანჯარაში.
და კიდევ ერთი. ფაილების და სტატიების კატეგორიების სიების შემცველ ყველა გვერდზე, უნდა შეიქმნას ჰიპერმინიშნებები, რომლებიც მთავარ გვერდთან იქნებიან დაკავშირებულნი. ხოლო გვერდებზე, რომლებიც შეიცავენ მინიშნებებს თვით ფაილებზე და სტატიებზე, აგრეთვე უნდა მოთავსდეს ჰიპერმინიშნებები შესაბამისი კატეგორიების სიების შემცველ გვერდებზე. ამისთვის მომხმარებლები მხოლო მადლობას გვეტყვიან.

CSS სტილებთან მუშაობა

მაშ ასე, ჩვენი საიტის ყველა გვერდი მზად არის. (მათი შექმნის პროცესს ჩვენ აქ აღარ აღვწერთ, ვინაიდან ის საკმაოდ რთულია). ამის შემდეგ, შეიძლება ვიფიქროთ მათ გალამაზებაზე. კერძოდ- ჩვენი ვებგვერდების გაფორმებაზე **CSS** სტილების საშუალებით.
სტილებთან სამუშაოდ ჩვენ დაგვჭირდება **CSS Styles** პანელი. (პანელი-როგორც ჩვენ უკვე ვიცით, არის **DreamweaverMX**-ის მცირე ზომის ფანჯარა, რომელიც სხვადასხვა სასარგებლო ინსტრუმენტებს შეიცავს). დავაკვირდეთ, გამოტანილია თუ არა ის ეკრანზე; თუ არ არის, მაშინ მენიუ **Windows**-ში, ჩავრთოთ პუნქტი-ჩამრთველი-**CSS Styles** და დავაჭიროთ კლავიშების კომბინაციას **<Shift>+<F11>**. თვით ეს პანელი ნაჩვენებია ნახ.3.29-ზე.

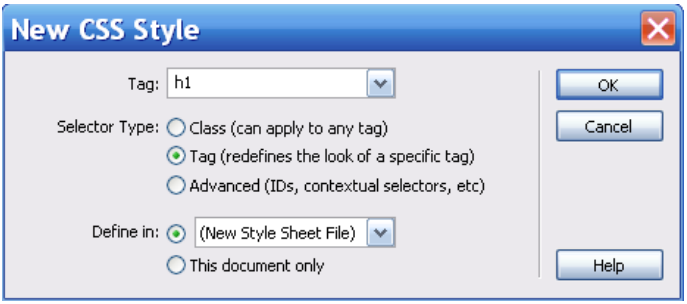


ნახ.3.29. პანელი CSS Styles

ამ პანელის უმეტესი ნაწილი უკავია, მოცემულ მომენტში რედაქტირებად ვებ-გვერდზე (როგორც ამბობენ გამოცდილი მომხმარებლები, აქტიურ დოკუმენტში), მოქმედი სტილების იერარქიულ სიას, რომელშიც მითითებულია, თუ სად რა სტილია გამოყენებული. ასე მაგალითად, ნახ.3.29-ზე ჩანს, რომ **<style>** ”ფესვიდან” ამოსულია ხე, რომელსაც ერთადერთი ”შტოთი” გააჩნია- **.style2**. ეს ნიშნავს, რომ ვებ-გვერდის სტილების შიდა ცხრლიში (სწორად ის აღინიშნება პუნქტით **<style>**) ერთადერთი სტილი **.style2**-ია განსაზღვრული. როგორც გვახსოვს, ეს სტილი ცხრილთან ერთად, **default.htm** ვებ-გვერდზე, რომელიც ამჟამად გვაქვს გახსნილი, თვით **Dreamweaver**-მა შექმნა. ვთქვათ, **default.htm** გვერდი ჩვენ უკვე გავხსენით. ახლა გავაკეთოთ ისე, რომ პირველი დონის ყველა სათაურები **(ტეგი H1**) სწორდებოდნენ ცენტრის

მიმართ, ჩვენი ყოველგვარი ჩარევის გარეშე. ამისათვის ჩვენ დაგვჭირდება **<H1>** ტეგის ხელახლა განსაზღვრის სტილის შექმნა და მისი განთავსება სტილების გარე ცხრილში, ხოლო შემდეგ ამ სტილის მიბმა ჩვენი საიტის ყველა გვერდებთან.

დავაჭიროთ პატარა ღილაკს  , რომელიც **CSS Styles** პანელის ქვედა ნაწილშია განთავსებული. ეკრანზე გამოჩნდება დიალოგური ფანჯარა **New CSS Styles**, რომელიც ნახ.3.30-ზეა ნაჩვენები.



ნახ.3.30. დიალოგური ფანჯარა New CSS Style

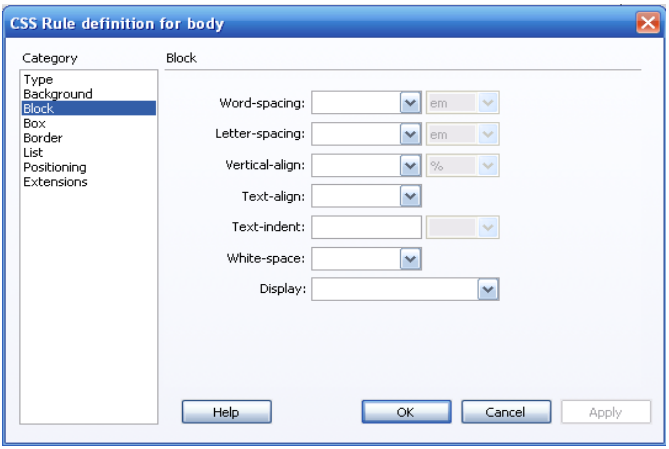
აქ ჩვენ დაგვჭირდება **Selector Type** ჯგუფში შემავალი **Tag** გადამრთველის ჩართვა, რომელიც განსაზღვრავს ტეგის ხელახლა განსაზღვრის სტილის შექმნას. ამის შემდეგ ჩამოსაშლელ სიაში **Tag**, ჩვენ შეგვეძლება ჩვენთვის საჭირო ტეგის, კერძოდ **<H1>** არჩევა. **Selector Type** ჯგუფის გადამრთველი **Class**, აიძულებს **Dreamweaver**-ს შექმნას ჩვენთვის სტილური კლასი, ხოლო გადამრთველი Advanced-კომბინირებული სტილი.

ამის შემდეგ ჩავრთოთ **Define in** ჯგუფის ზედა გადამრთველი და ყურადღება მივაქციოთ იმას, რომ, მარჯვნივ განთავსებულ ჩამოსაშლელ სიაში არჩეულ იქნას პუნქტი (**New Style Sheet File**). ამგვარად, ჩვენ ვამცნობთ **Dreamweaver**-ს, რომ გვინდა შექმნილი სტილის განთავსება სტილების გარე ცხრილში, რომელიც აგრეთვე უნდა შეიქმნას. **Define in** ჯგუფის გადამრთველი **This document only, Dreamweaver**-ს მიუთითებს იმაზე, რომ მოათავსოს შექმნილი სტილი, სტილების შიდა ცხრილში, რაც ჩვენ სრულებით არ გვჭირდება.

დავაჭიროთ ღილაკს **OK**. ეკრანზე გამოჩნდება **Windows**-ის ანალოგიური ფანჯრის მსგავსი **Dreamweaver**-ის ფაილის შესანახი

დიალოგური ფანჯარა. ფაილის სახელის ველში შევიტანოთ **styles.css**-სწორედ ასე ერქმევა ჩვენს სტილების გარე ცხრილს-და დავაჭიროთ შენახვის ღილაკს. ამით, ახალი, ჯერ კიდევ ცარიელი სტილების ცხრილი შექმნილია.

ახლა ჩვენს წინაშეა დიალოგური ფანჯარა **CSS Style Definition**. მის მარცხენა ნაწილში მდებარეობს სია-**Category**, რომელშიც ხდება სტილების ატრიბუტების კატეგორიების არჩევა, ხოლო თვით ამ ატრიბუტების გაწყობა (მნიშვნელობების მინიჭება პარამეტრებისათვის), ხდება ამავე ფანჯრის მარჯვენა ნაწილში განთავსებული მართვის ელემენტების საშუალებით.
ავირჩიოთ სიაში-**Category**, პუნქტი **Block**-ეს კატეგორია შეიცავს ატრიბუტებს, რომლებიც ტექსტის აბზაცების ფორმატს ეხება. ფანჯარა **CSS Style Definition** მიიღებს ასეთ სახეს-ნახ.3.31



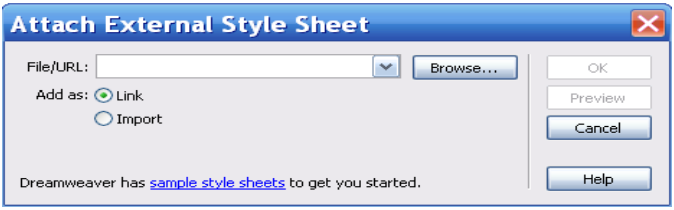
ნახ.3.31.დიალოგური ფამჯარა CSS Style Definition-ის კატეგორია Block.

აბზაცის გასწორება ხდება ჩამოსაშლელი სიის **Text align** საშუალებით. ავირჩიოთ მასში პუნქტი **center** და დავაჭიროთ ღილაკს **OK**. **CSS Style** პანელის სიაში გაჩნდება ახალი "**ხე**" "**ფესვით**" **styles.css** (ჩვენი სტილების გარე ცხრილის ფაილის სახელი) და შტოთი **h1** (ჩვენს მიერ ახლახანს შექმნილი **<H1>** ტეგის ხელახლა განსაზღვრის სტილი).

დაგვრჩა მხოლოდ დოკუმენტის ფანჯრის არჩევა, რომელშიც **default.htm** ვებ-გვერდია გახსნილია, **HTML** კოდის წარმოჩენის რეჟიმში გადართვა და **<H1>** ტეგის ატრიბუტის **ALIGN** წაშლა თავის მნიშვნელობასთან ერთად. საოცარია! ჩვენს მიერ შექმნილი სტილი მუშაობს-საიტის სახელწოდება მაინც გასწორდა ცენტრის მიმართ!

შენიშვნა: *მას შემდეგ, რაც სტილების გარე ცხრილი შეიქმნება, **DreamweaverMX**-ი გახსნის მას დოკუმენტის ახალ ფანჯარაში. არ უნდა დაგვავიწყდეს მისი დროდადრო დამახსოვრება, წინააღმდეგ შემთხვევაში ყველა ჩვენი სტილი შეიძლება დაიკარგოს, მაგალითად დენის მოულოდნელი გამორთვისას. (იგივე ეხება ვებ-გვერდებსაც).*

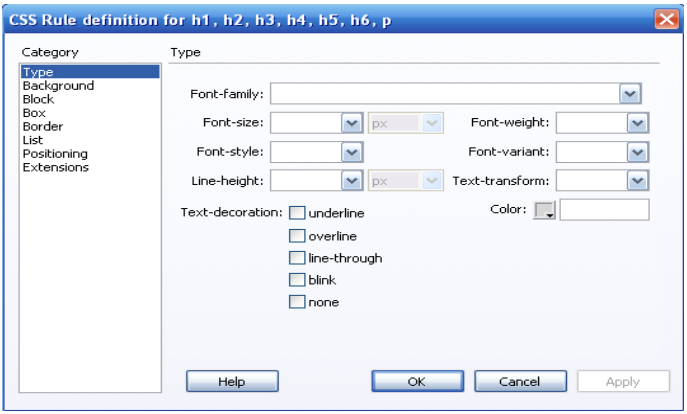
ახლა მოდით დავხუროთ ვებ-გვერდი **default.htm** და სტილების ცხრილი **styles.css**, ამასთან არ უნდა დაგვავიწყდეს მათი წინასწარ დამახსოვრება. ამის მერე გავხსნათ ვებ-გვერდი **Articles.htm**. ჩვენ ახლა მივაბამთ მას ჩვენს მიერ შექმნილ სტილების ცხრილს.
CSS Style პანელში სტილების სია ამჟამად ცარიელია, ვინაიდან სტილების არცერთი ცხრილი ამ გვერდთან ჯერ-ჯერობით არ არის მიბმული. არც სტილების შიდა ცხრილს არ შეიცავს იგი. დავაჭიროთ ღილაკს  , რომელიც **CSS Style** პანელის ქვედა ნაწილშია განთავსებული. ეკრანზე გამოჩნდება მცირე ზომის ფანჯარა **Attach External Style Sheet**, რომელიც ნაჩვენებია ნახ.3.32.



ნახ.3.32. დიალოგური ფანჯარა Attach External Style Sheet

დავაჭიროთ ამ ფანჯრის ღილაკს **Browse**, დიალოგურ ფანჯარაში, **Select File**, რომელიც გამოჩნდება ეკრანზე, ავირჩიოთ ჩვენი სტილების ცხრილის ფაილი და დავაჭიროთ გახსნის ღილაკს. ამის შემდეგ არჩეული ფაილის სახელი გაჩნდება შესატან ველში **File/URL**. ჩავრთოთ გადამრთველი **Link**, რათა სტილების ეს ცხრილი მივაბათ ვებ-გვერდს და დავაჭიროთ ღილაკს **OK**.

(გადამრთველი **Import** აიძულებს **DreamweaverMX**-ს ყველა გარე ცხრლიში განსაზღვრული სტილები განათავსოს შიდა ცხრილში, ეს კი ჩვენ არ გვჭირდება).
ისევ ავირჩიოთ დოკუმენტის ფანჯარა, რომელშიც გახსნილია ვებ-გვერდი **Articles.htm**, გადავერთოთ **HTML** კოდის წარმოჩენის რეჟიმში და მოვაშოროთ **<H1>** ტეგს ატრიბუტი **ALIGN** მის მნიშვნელობასთან ერთად. მორჩა, გარე ცხრილის მოქმედება გავრცელდა ამ გვერდზეც. დაგვრჩა იმავე პროცედურის განმეორება ჩვენი საიტის ყველა გვერდისათვის.
ამის შემდეგ შევქმნათ სტილი საავტორო უფლებების შესახებ ცნობებისათვის. დავიმახსოვროთ ყველა გახსნილი ვებ-გვერდები, დავხუროთ ისინი და გავხსნათ გვერდი **default.htm**. ისევ დავაჭიროთ **CSS Styles** პანელის ღილაკს  . **New CSS** Style ფანჯრის, **Selector Type** ჯგუფში ჩავრთოთ გადამრთველი **Class**, ჩამოსაშლელ სიაში **Name** შევიტანოთ ტექსტი .copyright, შევამოწმოთ, ჩართულია თუ არა **Define in** ჯგუფის ზედა გადამრთველი და არჩეულია თუ არა მის მარჯვნივ მდებარე ჩამოსაშლელ სიაში პუნქტი **style.css.**. (ჩვენ გვჭირდება **copyright** სტილის ჩასმა სტილების **style.css** გარე ცხრილში. ხომ ასეა?). დავაჭიროთ ღილაკს **OK** და მოვხვდებით დიალოგურ ფანჯარაში **CSS Style Definition**.
იმისათვის, რომ განვსაზღვროთ შრიფტის პარამეტრები, ჩვენ დაგვჭირდება **Type** კატეგორიის მართვის ელემენტები, ასე, რომ ავირჩიოთ პუნქტი **Type** სიაში **Category**, და დავინახავთ იმას, რაც ნაჩვენებია ნახ.3.33-ზე. ჩამოსაშლელ სიაში **Size** ავირჩიოთ პუნქტი **smaller**, ხოლო ჩამოსაშლელ სიაში **Style**-პუნქტი **italic**. ამრიგად **copyright**-სტილური კლასისათვის ჩვენ განვსაზღვრავთ დაპატარავებულ კურსივულ შრიფტს. დავაჭიროთ ღილაკს **OK**.



ნახ.3.33. CSS Style Definition დიალოგური ფანჯრის კატეგორია Type.

ახლა ავირჩიოთ დოკუმენტის ფანჯარა, რომელშიც გახსნილია ვებ-გვერდი **default.htm**. ჩვენ უნდა ამოვშალოთ ტეგი **<EM>**, რომელიც განსაზღვრავს საავტორო უფლებების შესახებ ტექსტის კურსივულ მოხაზულობას, რათა მან ხელი არ შეგვიშალოს, და მივაბათ ამ ტექსტს (უფრო სწორად ტეგს-**<P>**) ახლახანს შექმნილი სტილური კლასი. ამის შემდეგ დავაყენოთ ტექსტური კურსორი ტექსტზე ***ჩვენ, საიტის მფლობელები***, "მაუსის" მარჯვენა ღილაკით დავაწკაპუნოთ ტეგების სექციის **** ღილაკზე და ეკრანზე გამოსულ კონტექსტურ მენიუში ავირჩიოთ პუნქტი **Remove Tag**. ამგვარად ჩვენ წავშლით ტეგს **<EM>**, მაგრამ შევინარჩუნებთ მის შიგთავსს. (საჭიროების შემთხვევაში **<Del>** კლავიშაზე დაჭერით, წაიშლება როგორც ტეგი, ასევე მისი შიგთავსიც). ახლა დაგვრჩა მხოლოდ ტეგების სექციაში **<P>** ტეგზე დაწკაპუნება, რათა მოვნიშნოთ მთლიანი აბზაცი და თვისებების რედაქტორის ჩამოსაშლელ სიაში **Style**, ავირჩიოთ საჭირო სტილური კლასის შესაბამისი **copyright** პუნქტი.

სტილების ცხრილების საშუალებით ბავრი რამის გაკეთება არის შესაძლებელი... მაგრამ მოდით ჯერ მაინც შევჩერდეთ. ბოლოს და ბოლოს ვებ-დიზაინის შესახებ ბევრი სხვა წიგნიც მოგვითხრობს. ეს კი ეძღვნება ინტერნეტ-პროგრამირებას. მოდით ვნახოთ, კიდევ რისი შემოთავაზება შეუძლია **DreamweaverMX**-ს. ამისათვის ჯერ დავიმახსოვროთ ყველაფერი, რისი გაკეთებაც უკვე მოვასწარით.

ვებ-გვერდების წინასწარი დათვალიერება.

მიუხედავად იმისა, რომ ვებ-გვერდების წარმოჩენის რეჟიმში, **DreamweaverMX**-ი მათ წარმოგვიდგენს თითქმის ისეთნირად, როგორც ისინი ნაჩვენები იქნებიან ვებ-დამთვალიერებლებში, მაინც ხშირად ჩნდება მათი უშუალოდ ვებ-დამთვალიერებელში წინასწარი დათვალიერების აუცილებლობა. საქმის არსი სიტყვა "თითქმის"-ში მდგომარეობს: **DreamweaverMX**-ს მაინც არ შეუძლია კონკრეტული ვებ-დამთვალიერებელი პროგრამის ბევრი წვრილმანის გათვალისწინება. ამიტომ მასში ასთი შესაძლებლობაა ჩადებული: დოკუმენტის ფანჯრის დაუხურავად ჩვენ შეგვიძლია გამოვიძახოთ ვებ-თამთვალიერებელი და შევაფასოთ ჩვენი ქმნილების საბოლოო სახე "მშობლიურ გარემოში". ამისათვის საკმარისია მენიუ **File**-ის, ქვემენიუ **Preview in Browser** -ში, პუნქტ **iexplore**-ის არჩევა ან რაც უფრო მარტივად და სწრაფად კეთდება, კლავიშა **<F>**-ზე დაჭერა.

ცნობარის გამოძახება

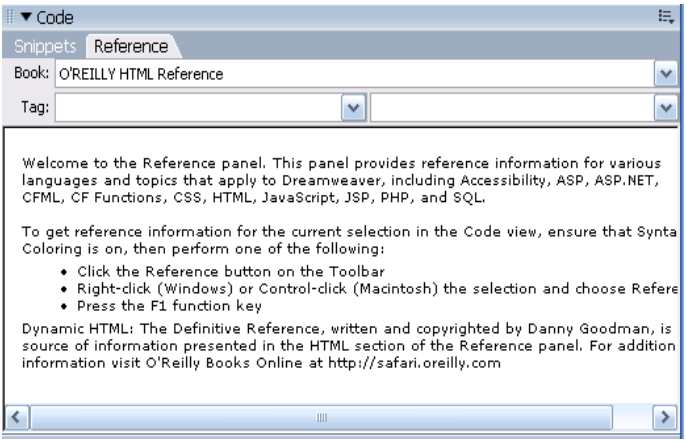
მოცემული წიგნი აღწერს **DreamweaverMX**-ის შესაძლებლობების მხოლოდ ძირითად და აუცილებელ ნაწილს. ამიტომ ამ პროგრამასთან მუშაობისას ჩვენ შეიძლება დაგვჭირდეს-და ნამდვილად დაგვჭირდება-დახმარება. როგორც **Windows**-ის ყველა სერიოზულ გამოყენებით პროგრამებს, **DreamweaverMX**-საც გააჩნია მძლავრი საცნობარო სისტემა. მისი გამოძახებისათვის საჭიროა უბრალოდ <F1> კლავიშაზე დაჭერა ან მენიუ **Help**-ში, **Using Dreamweaver** პუნქტის არჩევა. ამის შემდეგ ეკრანზე გამოჩნდება ცნობარის ფანჯარა, რომელიც ნაჩვენებია ნახ.3.34-ზე.

ამ ფანჯრის მარცხენა ნაწილში, ჩანართში **Содержание-შინაარსი**, განთავსებულია თემების იერარქიული სია. თვით სტატიების შინაარსი აისახება მარჯვენა მხარეს. იმისათვის, რომ წავიკითხოთ სტატია, საკმარისია გავშალოთ სიის სასურველი "ხე" და დავაწკაპუნოთ შესაბამის "შტოზე".

Dreamweaver-ის საცნობარო სისტემა, ასევე შეიცავს საგნობრივ მიმთითებელსაც. მის გამოსაყენებლად უნდა გადავერთოთ ჩანართში **Указатель-მიმანიშნებელი**, ტერმინების სიაში ავირჩიოთ ჩვენთვის სასურველი ტერმინი და ორჯერ დავაწკაპუნოთ მასზე. ცნობარის შესაბამისი სტატია გამოჩნდება მარჯვენა მხარეს. იმისათვის, რომ სწრაფად მოვძებნოთ საჭირო ტერმინი, ჩვენ ასევე შეგვიძლია მისი დასახელებიდან პირველი რამოდენიმე სიმბოლო

შევიტანოთ შესატან ველში, რომელიც სიის ზედა ნაწილშია მოთავსებული და მყისიერად, პირველივე შესაბამისი ტერმინი გამოჩნდება სიაში.

ნახ.3.34. Dreamweaver-ის ცნობარის ფანჯარა.



ნახ.3.35. პანელი Reference

DreamweaverMX-ის შემადგენლობაში ასევე გვაწვდიან **HTML**-ის და **CSS**-ის სრულ სახელმძღვანელოებს გამომცემლობა **O'Reilly**-საგან. მათი შინაარსი აისახება განსაკუთრებულ პანელში **Reference**

(ნახ.3.35). იმისათვის, რომ გამოვიტანოთ იგი ეკრანზე, საჭიროა ჩაირთოს მენიუ **Window**-ის პუნქტი-გადამრთველი Reference ან დავაჭიროთ კლავიშების კომბინაციას **<Shift>+<R1>**.

HTML ცნობარის დასათვალიერებლად, საკმარისია ჩამოსაშლელ სიაში Book, პუნქტ **O'REILLY HTML Reference**-ის არჩევა. ამის შემდეგ ჩამოსაშლელ სიაში Tag, უნდა ავირჩიოთ საჭირო ტეგის დასახელება-პანელში აისახება მისი აღწერა.

ჩამოსაშლელი სიის **Tag**, მარჯვენა მხარეს განთავსებულია მეორე ჩამოსაშლელი სია. მოცემულ მომენტში (იხ.ნახ. 3.35) იქ არჩეულია პუნქტი **Description**-ეს იმის მაჩვენებელია, რომ პანელში **Reference** გამოისახება ცნობები თვით ტეგის შესახებ. თუ საჭიროა ცნობის მიღება **Tag** სიაში არჩეული ტეგის რომელიმე ატრიბუტის შესახებ, საკმარისია უბრალოდ ავირჩიოთ ეს ატრიბუტი მარჯვენა ჩამოსაშლელ სიაში.

CSS-ის ცნობარი გამოიძახება ჩამოსაშლელი სიის **Book**, **O'REILLY CSS Reference** პუნქტის არჩევით. შემდეგ ჩამოსაშლელ სიაში **Style** აირჩევა საჭირო ატრიბუტის დასახელება-**Reference** პანელში გაჩნდება მისი აღწერა.

რა იქნება ამის მერე?

მაშ ასე, დასაწყისისათვის ვფიქრობ საკმარისია. რათქმა უნდა ჩვენ შემდეგაც გავაგრძელებთ **Dreamweaver**-ის გამოყენებას, ანუ შევისწავლით მის სხვა შესაძლებლობებსაც. მაგრამ, ეს იქნება მერე, შემდეგ თავებში.

ვინაიდან ახლა ჩვენი საიტი უკვე მზად არის, დროა ის გამოვაქვეყნოთ ვებ-სერვერზე. შემდეგ თავში ჩვენ სწორედ საიტის პუბლიკაციის პროცესს განვიხილავთ.

თავი 4

ვებ-საიტთან მუშაობა DreamweaverMX-ში

მაშ ასე, ჩვენი საიტი მზად არის. ჩვენ შევქმენით ყველა ვებ-გვერდი, დავაკავშირეთ ისინი ჰიპერკავშირებით და გავაფორმეთ კიდევაც **CSS** სტილების გამოყენებით. დადგა დრო, როდესაც უნდა გამოვაქვეყნოთ იგი ვებ-სერვერზე. სწარედ ამით დავკავდებით ახლა. კი მაგრამ სად ვიპოვოთ შესაფერისი ვებ-სერვერი? მოდით პირველ რიგში ეს საკითხი გადავწყვითოთ ჩვენთვის.

-შესაძლებელია საიტის დარეგისტრირება რომელიმე ისეთ ვებ-სერვერზე, რომელიც საკუთარ მომხმარებლებს საიტების განსათავსებლად უფასოდ უთმობს ადგილს. ასეთი სერვერები ახლა ბევრია და შესაფერისის არჩევა არ გაგვიჭირდება. რათქმა უნდა უზარმაზარი პორტალის განთავსება მათზე შეუძლებელია, მაგრამ ჩვენი ფაილების და სტატიების მცირე არქივისათვის ასეთი ვარიანტი სრულიად მისაღებია.

-შეგვიძლია ვებ-სერვერის პროგრამის დაყენება ჩვენ საკუთარ კომპიუტერზე. ეს ძალიან სწრაფად კეთდება, ათ წუთში-ისიც თუ ძალიან არ ვჩქარობთ. მაგალითად, არსებობს პოპულარული უფასო ვებ-სერვერი **Apache**, რომლის დაყენება თანმხლები ინსტრუქციის მიხედვით ძალზე მარტივია. რათქმა უნდა შეიძლება სხვა ვებ-სერვერის გამოყენებაც, მაგალითად **Microsoft Personal Web-Server** ან **Internet Information Server**, თუ გაგვაჩნია მათთან მუშაობის რაიმე გამოცდილება.

ფასიანი ხოსტინგის ვარიანტს ჩვენ არ განვიხილავთ-ამაზე ახლა ზედმეტი იქნებ ფულის დახარჯვა. ასევე არ განვიხილავთ რომელიმე ორგანიზაციის ვებ-სერვერზე საიტის პუბლიკაციის ვარიანტს. ასე, რომ ჩვენ მხოლოდ ზემოთ ჩამოთვლილი ორი ვარიანტი გაგვაჩნია.
მოდით, ჩვენ მაინც საკუთარ კომპიუტერზე დავაყენოთ ვებ-სერვერი-ის ჩვენ მომავალში ძალიან დაგვეხმარება. საიტის შექმნის პროცესში ჩვენ მუდმივად მოგვიწევს მისი ტესტირება, შეცდომების აღმოჩენა, მათი გასწორება და ისევ ტესტირება. ხოლო ასეთი ქმედებების განხორციელება უმჯობესია საკუთარ სერვერზე: ამ შემთხვევაში არ დაგვჭირდება ინტერნეტთან კავშირის ყოველწუთიერად დამყარება და გარდა ამისა გაწყობის (გამართვის) შესაძლებლობებიც მეტი გვექნება. ასე რომ ვისარგებლოთ შესაბამისი ინსტრუქციების რეკომენდაციებით სანამ ვებ-სერვერს არ ავამუშავებთ.
მაგრამ, როგორ ხორციელდება თვით საიტის პუბლიკაციის პროცესი? ჩვენ ახლა სწორად ამაზე ვისაუბრებთ.

საიტის მომზადება პუბლიკაციისათვის.

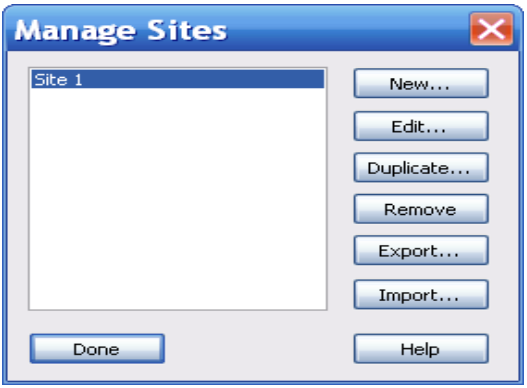
აქ პატარა პრობლემას ვაწყდებით. საქმე იმაშია, რომ ჩვენს მიერ შექმნილ ვებ-გვერდებს **DreamweaverMX**-ი არ განიხილავს როგორც ვებ-საიტს. მისთვის ეს მხოლოდ ცალკეული გვერდების

ერთობლივობაა, რომლებიც ერთმანეთთან, გარდა ჰიპერმინიშნებებისა, არაფრით არ არიან დაკავშირებულნი. ფაქტიურად **DreamweaverMX**-მა არც კი იცის, რომ ჩვენ საიტი შევქმენით.

ასე რომ, პირველი, რაც ჩვენ უნდა გავაკეთოთ-ეს არის ჩვენი საიტის დარეგისტრირება **DreamweaverMX**-ში. ვინაიდან ჩვენ ყველ ვებ-გვერდი ერთ საქაღალდეში-**Site1**-შევინახეთ-ეს პროცესი ბევრ დროს არ წაგვართმევს.

საიტის რეგისტრაცია DreamweaverMX-ში

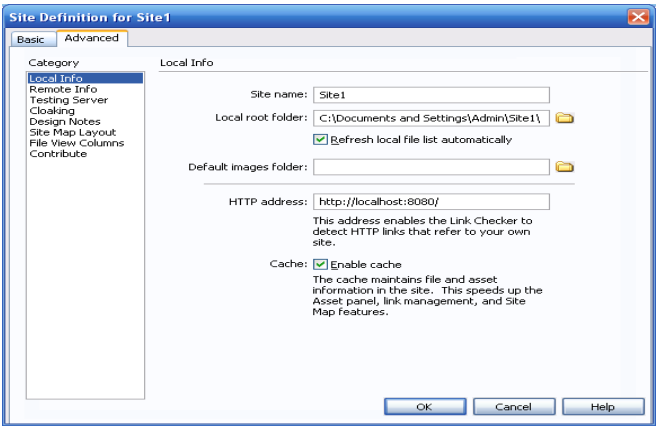
საიტის რეგისტრაცია Dreamweaver-ში იწყება მენიუ **Site**-ში, **Manage Site** პუნქტის არჩევით. ეკრანზე გამოჩნდება დიალოგური ფანჯარა **Manage Site**, რომელიც ნაჩვენებია ნახ.4.1.-ზე. ამ ფანჯრის უმეტესი ნაწილი, უკავია უკვე შექმნილი საიტების ნუსხას. იმისათვის, რომ ამ სიაში ჩვენი საიტიც დავუმატოთ, დავაჭიროთ ღილაკს **New** და იმ მცირე ზომის მენიუში, რომელიც გამოჩნდება ეკრანზე ავირჩიოთ პუნქტი **Site1**.



ნახ.4.1 დიალოგური ფანჯარა Manage Sites

ამის შემდეგ ეკრანზე გამოჩნდება დიალოგური ფანჯარა **Site Definition**, რომელიც ორი ჩანართისაგან შედგება. თუ ის გახსნილია ჩანართზე **Basic**, გადავერთოთ ჩანართზე **Advanced**-ვინაიდან ის უფრო მეტ შესაძლებლობებს იძლევა ჩვენი საიტის გასაწყობად. ამის შემდეგ შევამოწმოთ, არჩეულია თუ არა სიაში **Category** პუნქტი **Local**

Info. აღნიშნული მოქმედებების შემდეგ ფანჯარას **Site Definition**, ასეთი სახე უნდა ჰქონდეს-ნახ.4.2.



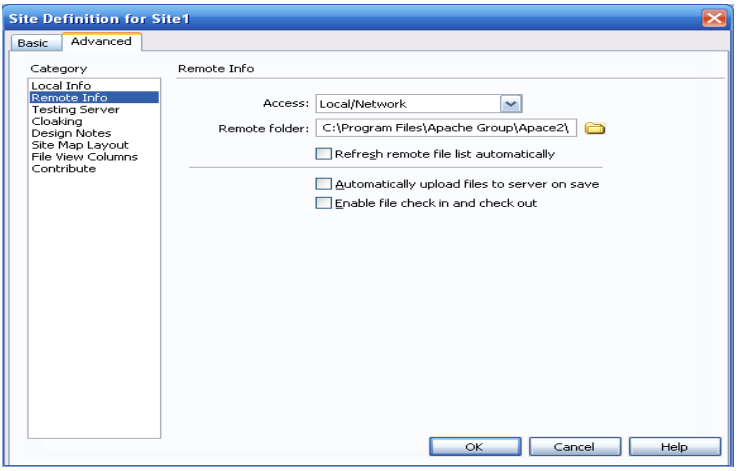
ნახ.4.2. დიალოგური ფანჯარა Site Definition (კატეგორია Local Info).

კატეგორია **Local Info**-ს მართვის ელემენტები საშუალებას გვაძლევენ განვსაზღვროთ ცნობები ჩვენი საიტის იმ ფაილების შესახებ, რომლებთანაც მოცემულ მომენტში ვმუშაობთ. დავარქვათ მათ **საიტის ლოკალური ასლი.** პირიქით, ფაილებს, რომლებიც უკვე გამოქვეყნებულია ვებ-სერვერზე (არა აქვს მნიშვნელობა ლოკალურ თუ დაშორებულ სერვერზე), ვუწოდოთ **საიტის დაშორებული ასლი.**
შესატან ველში **Site name** შევიტანოთ საიტის სახელწოდება, რომელიც აისახება **Manage Site** ფანჯრის საიტების სიაში. შევიტანოთ იქ **Site1**, როგორც ნაჩვენებია ნახ.4.2.
შესატან ველში **Local root folder** მიეთითება გზა (ბილიკი) საიტის ლოკალური ასლის საქაღალდემდე. მისი მოცემა ყველაზე მარტივია იმ საქაღალდეს პიქტოგრამაზე დაჭერით, რომელიც შესატანი ველის მარჯვნივ მდებარეობს. ამ დროს ეკრანზე გამოჩნდება **Windows**-ის სტანდარტული დიალოგური ფანჯარა, რომელშიც უნდა ავირჩიოთ ჩვენთვის სასურველი საქაღალდე.

შესატან ველში **Default images folder**, როგორც წესი შეიტანება იმ საქაღალდეს სახელი, რომელშიც წინასწარ არის მოთავსებული ვებ-საიტზეა გამოყენებული ყველა გრაფიკული გამოსახულება. ვინაიდან ჩვენს შემთხვევაში **root-корневой-ძირეულ** საქაღალდეში

ერთად-ერთი გამოსახულება **Arrow.gif**-ია შენახული, ჩვენ არაფელს არ შევიტანთ ამ ველში.

შესატან ველში **HTTP address** შეიტანება ჩვენი საიტის ინტერნეტ-მისამართი. პრინციპში მისი შეტანა აუცილებელი არ არის, მაგრამ მაშინ **DreamweaverMX**-ი ვერ შეამოწმებს ჰიპერმინიშნებების სისწორეს. მოდით, შევიტანოთ მასში განსაკუთრებული ინტერნეტ-მისამართი-**http://localhost:8080**, რომელიც ჩვენს ლოკალურ კომპიუტერს **(ლოკალურ ჰოსტს)** აღნიშნავს.

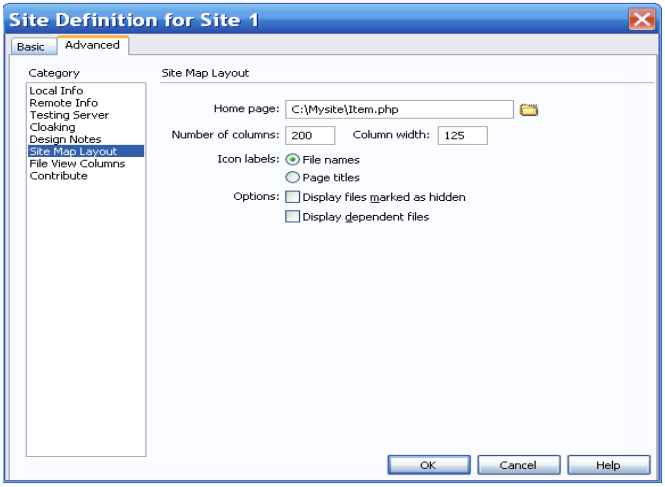


ნახ.4.3. დიალოგური ფანჯარა Site Definition (კატეგორია Remote Info)

ყურადღება! აქ ნაგულისხმევია, რომ ვებ-სერვერი იყენებს პორტს 8080. თუ კი ის სხვა პორტს იყენებს, ინტერნეტ მისამართში იმ პორტის ნომერი უნდა მიეთითოს.

ახლა, სიაში **Category** ავირჩიოთ პუნქტი **Remote Info**, სადაც საიტის დაშორებული ასლის პარამეტრები განისაზღვრება. ფანჯარა **Site Definition** ასეთ სახეს მიიღებს-ნახ.4.3.

პირველყოვლისა, ჩამოსაშლელ სიაში **Access** ავირჩიოთ პუნქტი **Local/Network**. ამითი ჩვენ ვამცნობთ **Dreamweaver**-ს, რომ ვებ-სერვერი, რომელზეც უნდა იქნას გამოქვეყნებული საიტის დაშორებული ასლი, მდებარეობს იმავე კომპიუტერზე, რომელზეც მისი ლოკალური ასლია განთავსებული.



ნახ.4.4. დიალოგური ფანჯარა Site Definition
(კატეგორია Site Map Layout).

მაშინათვე **Access** სიის ქვემოთ გამოჩნდება შესატანი ველი **Remote folder**.

აქ შეიტანება გზა (ბილიკი) საიტის დაშორებული ასლის ძირეულ საქაღალდემდე. ამ მიზნით, რათქმა უნდა ყველაზე მარტივია, შესატანი ველის მარჯვნივ მდებარე საქაღალდეს ნიშანზე დაწკაპუნება და ეკრამზე გამოსული **Windows**-ის სტანდარტულ დიალოგურ ფანჯარაში სასურველი საქაღალდის არჩევა. ვებ-სერვერ **Apache**-ის შემთხვევაში წინასწარ განსაზღვრული ძირეული საქაღალდე-ეს არის საქაღალდე, რომელშიც მოთავსებულია **Apache>\...\htdocs.**

დაგვრჩა მხოლოდ Category სიაში, პუნქტ **Site Map Layout**-ის არჩევა (ნახ.4.4), რაღაც-რაღაცეების შემოწმება და ზოგიერთი დამატებითი პარამეტრების განსაზღვრა.

პირველყოვლისა, შევამოწმოთ, ჩასვა თუ არა **Dreamweaver**-მა შესატან ველში **Home page** ვებ-გვერდის სახელი, როგორც გაჩუმებითი (default,წინასწარ განსაზღვრული,ნაგულისხმები). თუ არა, შევიტანოთ ის, ან დავაწკაპუნოთ შესატანი ველის მარჯვნივ მდებარე საქაღალდეს ნიშნაკზე და გამოსულ **Select Files**-ის მსგავს დიალოგურ ფანჯარაში-**Choose Home Page**, ავირჩიოთ საჭირო ფაილი.

საიტის პარამეტრების შეტანის დასრულებისთანავე დავაჭიროთ **Site Definition** დიალოგური ფანჯრის **OK** ღილაკს. ამის შემდეგ Dreamweaver-ი განახორციელებს ფაილების სიის სკანირებას და მეხსიერებაში შექმნის მათ ჩამონათვალს. შემდეგ დავაჭიროთ დიალოგური ფანჯრის **Manage Sites**-ის ღილაკს **Done**. ამით საიტის რეგისტრაცია დასრულებული იქნება.

საიტის ფაილებთან მუშაობა. პანელი Files

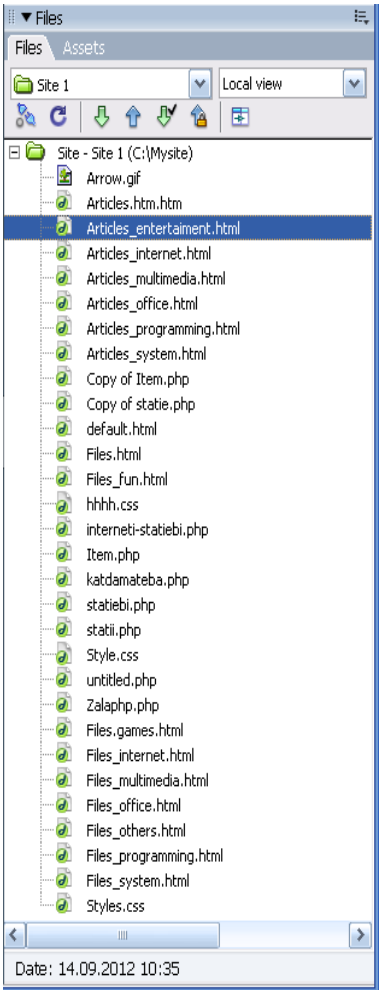
ახლა ჩვენ გავეცნობით **DreamweaverMX**-ის ერთ ძალზე სასარგებლო და მოხერხებულ ინსტრუმენტს, რომლის დანიშნულებას, ვებ-გვერდების შემადგენელ ფაილებთან მუშაობა წარმოადგენს. ეს არის პანელი **Files**. მისი ეკრანზე გამოსატანად საჭიროა, მენიუ **Windows**-ის პუნქტ **Files**-ის ჩართვა ან კლავიშა **<F8>**-ზე დაჭერა. თვით ეს პანელი ნაჩვენებია ნახ.4.5-ზე.

Files პანელის უმეტესი ნაწილი დაკავებულია საიტის შემადგენელი იმ ფაილებისა და საქაღალდეების იერარქიული სიით, რომელთა ძირებიც, საიტის ძირეულ საქაღალდეში მდებარეობენ. პანელ **Files**-ს თავისი საკუთარი, მართვის ელემენტებიანი ინსტრუმენტარიები გააჩნია, რომლებიც უზრუნველყოფენ ყველაზე ხშირად გამოსაყენებელ ბრძანებებთან და სტატუსის სტრიქონთან სწრაფ წვდომას.

Files პანელის ფაილებისა და საქაღალდეების სია წააგავს **Windows**-ის ფაილების **გამცილებლის-проводник** ანალოგიურ სიას: შესაძლებელია მისი დაკეცვა და განშლა, აგრეთვე სასურველი ფაილის ან საქაღალდეს მონიშვნა "მაუსზე" დაწკაპუნებით. თუ საჭიროა რამოდენიმე ფაილის ერთდროულად გამოყოფა, საკმარისია მათზე დაწკაპუნება, კლავიშა **<Ctrl>**-ზე ხელის აუღებლად. თუ გვინდა სიის ყველა ფაილის ერთდროულად არჩევა, უნდა დავაჭიროთ კლავიშების კომბინაციას **<Ctrl>+<A>** ან დავაწკაპუნოთ პიქტოგრამაზე  , რომელიც პანელის ზედა მარჯვენა კუთხეში მდებარეობს და ამ დროს ეკრანზე გამოსული დამატებითი მენიუს ქვემენიუ **Edit**-ში ავირჩიოთ პუნქტი **Select All**.

პანელ **Files**-ის მარცხენა მხარეს მდებარე ჩამოსაშლელი სია, საშუალებას გვაძლევს პანელზე გამოსაყვანად, სწრაფად ავირჩიოთ **DreamweaverMX**-ში დარეგისტრირებული საიტებიდან ნებისმიერი. ეს ნიშნავს იმას, რომ ჩვენ **DreamweaverMX**-ში შეგვიძლია

დავარეგისტრიროთ საიტების ნებისმიერი რაოდენობა და მარტივად გადავიდეთ ერთიდან მეორეზე.



ნახ.4.5. პანელი Files

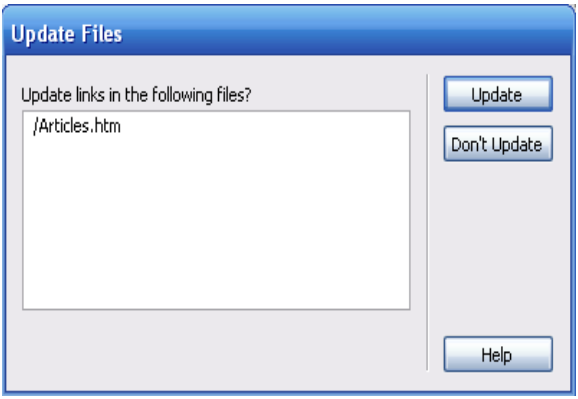
ის ჩამოსაშლელი სია, რომელიც პანელის მარჯვენა მხარეს მდებარეობს, საშუალებას გვაძლევს ავირჩიოთ საიტის ასლი-**ლოკალური** ან **დაშორებული**-რომლის შიგთავსის დანახვაც გვსურს

Files პანელში. აღნიშნული პუნქტის გაჩუმებითი(ნაგულისხმევი) მნიშვნელობა-**Local view**, რომელიც წინასწარ არის არჩეული, განსაზღვრავს ლოკალური ასლის ფაილების, ხოლო **Remout view**-მათი დაშორებული ასლების ასახვას პანელში **Files**.
ადრე ნათქვამი იყო, რომ **Files** პანელის ფაილების სია წააგავს **Windows**-ის გამცილებლის ანალოგიურ სიას, იმ გამონაკლისის გათვალისწინებით, რომ მასში აისახება, როგორც ფაილები, ასევე საქაღალდეები. ისევე, როგორც გამცილებელი, პანელი **Files** გვაძლევს ამ ფაილებისა და საქაღალდეების მართვის საშუალებას.
ასე მაგალითად, რომ წავშალოთ სიაში მონიშნული ფაილი ან საქაღალდე, საჭიროა დამატებითი მენიუს, ქვემენიუ **File**-ის, პუნქტ **Delete**-ის არჩევა ან **<Del>** ღილაკზე დაჭერა. ამის შემდეგ **DreamweaverMX**-ი შეგვეკითხება, მართლა გვსურს თუ არა ამ ფაილის (საქაღალდეს) წაშლა; ამის შემდეგ უნდა დავაჭიროთ **Yes** ან **No** ღილაკს, რათა შესაბამისად დავადასტუროთ ან უარვყოთ წაშლის ოპერაცია.

ყურადღება! ***DreamweaverMX****-ი წაშლილ ფაილებს ან ფაილების შემცველ საქაღალდეებს ათავსებს* ***Windows****-ის კალათაში. მაგრამ ცარიელ საქაღალდეებს შლის უკვალოდ.*

იმისათვის, რომ სახელი გადავარქვათ მონიშნულ ფაილს საჭიროა, კიდევ ერთხელ დავაწკაპუნოთ მასზე "მაუსით" ან დავაჭიროთ კლავიშა **<Del>**-ს. შეიძლება ასევე ავირჩიოთ დამატებითი (კონტექსტური) მენიუს, ქვემენიუ **File**-ის პუნქტი **Rename**. ამის შემდეგ მოცემული ფაილის სახელის ადგილას გაჩნდება პატარა შესატანი ველი, სადაც ჩასმული იქნება ფაილის ძველი დასახელება. მოვახდინოთ მისი რედაქტირება ან შევიტანოთ ახალი სახელი, რის შემდეგაც ახალი სახელის დასამახსოვრებლად დავაჭიროთ კლავიშა **<Enter>**-ს ან უარსაყოფად კლავიშა **<Esc>**-ს.
თუ სხვა ვებ-გვერდებზე არსებობენ ჰიპერმინიშნებები სახელგარდაქმნილ ფაილზე, **DreamweaverMX**-ი შემოგვთავაზებს მათ შესწორებას. ეკრანზე გაჩნდება დიალოგური ფანჯარა **Update Files**, რომელიც ნაჩვენებია ნახ.4.6.-ზე. სიაში **Update links in following files?** მოთავსებულია ის ვებ-გვერდები, რომლებზეც აღმოჩენილი იყო სახელგარდაქმნილი ფაილისაკენ მიმართული ჰიპერმინიშნებები. დავაჭიროთ ღილაკს **Update**, რომ განვაახლოთ

ისინი ან **Don't Update**, რათა უარვყოთ განახლება; უკანასკნელ შემთხვევაში ჩვენ ამის გაკეთება ხელით მოგვიწევს.



ნახ.4.6. დიალოგური ფანჯარა Update Files

თუ რომელიმე ფაილზე ორჯერ დავაწკაპუნებთ ან მოვნიშნავთ მას და დავაჭერთ **<Enter>** კლავიშას, **DreamweaverMX**-ი გახსნის ამ ფაილს. ასე ხდება ვებ-გვერდებზე და სტილების ცხრილებზეც, მაგრამ თუ ჩვენ კომპიუტერზე დაყენებული გვაქვს ვებ-გრაფიკის პროგრამა **Macromedia Fireworks**, მაშინ გრაფიკული ფაილების გახსნაც ამ ხერხით შეგვიძლია.

პანელი **Files**-ის საშუალებით შეგვიძლია ასევე გავხსნათ ახალი ვებ-გვერდი, თუ ავირჩევთ დამატებითი მენიუს **File** ქვემენიუს პუნქტ **New File**-ს ან დავაჭერთ კლავიშების კომბინაციას, **Ctrl>+<Shift>+<N>**. ამის შემდეგ სიის ყველაზე ქვედა ნაწილში გაჩნდება ახალი ფაილი სახელწოდებით **untitled.htm**, რომელიც ჩვენ მაშინათვე შეგვიძლია შევცვალოთ (უნდა შევცვალოთ კიდევაც). სახელის შეცვლის შემდეგ, ვაჭერთ კლავიშა **<Enter>**-ს, **DreamweaverMX**-ი გახსნის ახალ ცარიელ გვერდს და ჩვენ შეგვიძლია მასში შიგთავსის შეტანა.

ახალი საქაღალდეც ანალოგიურად იქმნება. ამისათვის საჭიროა პუნქტის **New Folder** არჩევა ან კლავიშების კომბინაციის **<Ctrl>+<Shift>+<Alt>+<N>** დაჭერა. აღნიშნულის შემდეგ, სიის ყველაზე ქვედა ნაწილში გაჩნდება ახალი საქაღალდე სახელწოდებით **untitled**, რომელიც ჩვენ უნდა შევცვალოთ და დავაჭიროთ კლავიშას **<Enter>**. შემდეგ შეგვიძლია "მაუსის" საშუალებით ჩვენთვის საჭირო ფაილების (სხვა საქაღალდეებისაც)

გადათრევა ამ საქაღალდეში, ხოლო **DreamweaverMX**-ი ჩვენთვის უკვე ნაცნობი **Update Files** ფანჯრის ეკრანზე გამოყვანით, შემოგვთავაზებს მათზე მიმთითებელი ჰიპერმინიშნებების შეცვლას (იხ. ნახ.4.6.). თუკი ფაილების გადათრევის დროს **<Ctrl>** კლავიშა დაჭერილი იქნება, **DreamweaverMX**-ი გადატანის ნაცვლად, დააკოპირებს ამ ფაილებს.

დავუშვათ, რომ საიტის **DreamweaverMX**-ში რეგისტრაციის დროს შეგვეშალა გვერდის ფაილის გამოცხადება საწყის **(default ან index)** ფაილად (ან თუ თვით **DreamweaverMX**-ი შეცდა-სწორედ ის ცდილობს ამის განსაზღვრას ავტომატურად). რომ გამოვასწოროთ ეს შეცდომა, მოვნიშნოთ ვებ-გვერდი, რომელიც უნდა გამოცხადდეს საწყის გვერდად (საშინაო გვერდი, თავფურცელი) და ავირჩიოთ დამატებითი მენიუს, **Site** ქვემენიუს პუნქტი **Set as Home Page** (გამოვაცხადოთ, როგორც საწყისი გვერდი). სწრაფიც არის და მარტივიც.

ვებ-გვერდების შემოწმება

დადგა თუ არა ჩვენი საიტის პუბლიკაციის დრო? არა, ჯერ კიდევ ადრეა. გვმართებს მოთმინება, სულ ცოტა დაგვრჩა! მოდით ჯერ შევამოწმოთ **HTML**-კოდისა და ჰიპერმინიშნებების კორექტულობა და შემდეგ გამოვაქვეყნოთ ჩვენი ქმნილება ქსელში.

HTML-კოდის კორექტულობის შემოწმება

ვებ-გვერდების შექმნის პროცესში არ არის გამორიცხული შეცდომების დაშვება. ასეთი შეცდომები შეიძლება უკავშირდებოდეს მაგალითად, ჰიპერმინიშნებების ინტერნეტ-მისამართების არასწორად განსაზღვრას ან ვებ-გვერდების დასახელების გამორჩენას (ტეგი **<TITLE>**). და თუ ეს უკანასკნელი ასე თუ ისე მოსათმენია, პირველი საერთოდ დაუშვებელია.

მაშ ასე, შევამოწმოთ **HTML**-კოდის სისწორე. ამისათვის უნდა დავხუროთ დოკუმენტების ყველა გახსნილი ფანჯარა, რომ მათ ხელი არ შეგვიშალონ მუშაობაში. პროგრამის მთავარი ფანჯრის მენიუ **Site**-ში ავირჩიოთ პუნქტი **Reports** ან იმავე სახელწოდების პუნქტი, **File** პანელის დამატებითი მენიუს, ქვემენიუ **Site**-ში. ეკრანზე გამოჩნდება დიალოგური ფანჯარა **Reports**, რომელიც ნაჩვენებია ნახ. 4.7.-ზე.

ამ ფანჯრის უმეტესი ნაწილი უკავია ორი "შტოსაგან" შემდგარ იერარქიულ სიას **Select reports**, რომელიც ანგარიშში ჩასართველი მონაცემების არჩევის საშუალებას გვაძლევს. ჩვენთვის ინტერესს წარმოადგენს მეორე "შტო"-**HTML Reports**, რომელიც შემდეგ პუნქტებს შეიცავს:

-**Combinable Nested Font Tags**-ფიზიკური ფორმატირების ჩალაგებული **<FONT>** ტეგების მოძებნა, რომელთა გაერთიანება უმტკივნეულოდ არის შესაძლებელი;

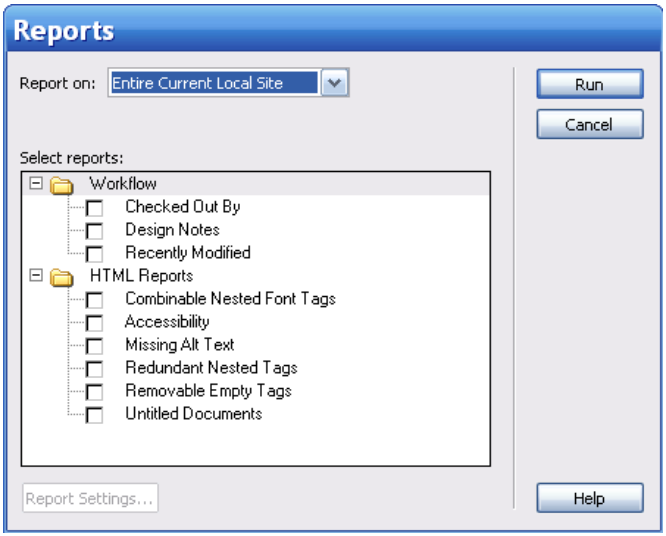
-**Accessibility**-ვებ-გვერდების სხვადასხვა ელემენტების განსაზღვრის კორექტულობის შემოწმება (მაგ. ყველა გვერდისთვის არის თუ არა განსაზღვრული კოდირება და ა.შ.);

-**Missing Alt Text-<IMG>** ტეგების შეუვსებელი **ALT** ატრიბუტების (ალტერნატიული ტექსტი) მოძებნა;

-**Redundant Nested Tags**–გამოუყენებელი ჩალაგებული ტეგების მოძებნა;

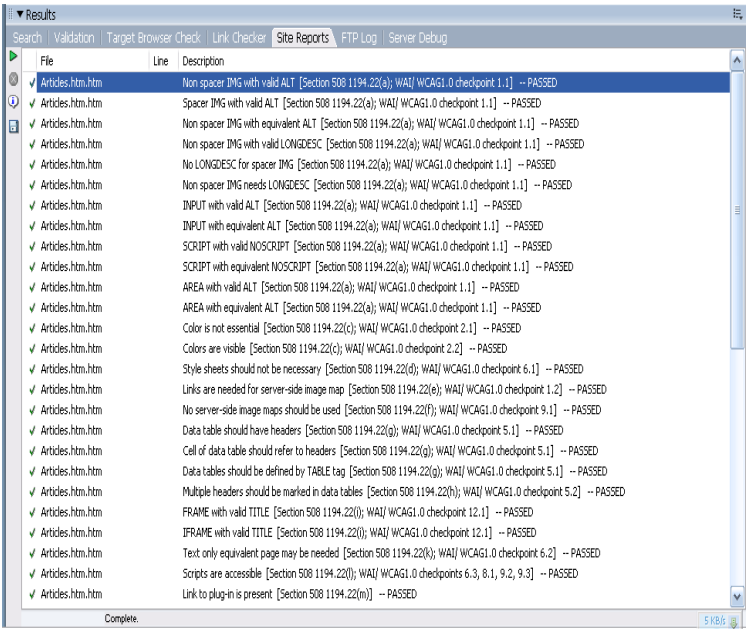
-**Removable Empty Tags**-ცარიელი ტეგების მოძებნა, რომელთა წაშლა უმტკივნეულოდ არის შესაძლებელი;

-**Untitled Documents**-დასახელების არ მქონე (რომელთაც არ გააჩნიათ ტეგი **<TITLE>**) ვებ-გვერდების მოძებნა.



ნახ.4.7. დიალოგური ფანჯარა Reports

თითოეული პუნქტი წარმოადგენს ალამს, რომელთა ჩართვა ან გამორთვა შეგვიძლია. მოდით ყველა მათგანი ჩავრთოთ.
შემდეგ შევამოწმოთ, ჩამოსაშლელ სიაში **Report on** არჩეულია თუ არა პუნქტი **Entire Current Local Site**, რომელიც **DreamweaverMX**-ს ავალებს ვებ-საიტის ყველა გვერდის შემოწმებას. (**Current Document** პუნქტის არჩევა იწვევს მხოლოდ აქტიურ დოკუმენტში გახსნილი ვებ-გვერდის შემოწმებას). ამის შემდეგ დავაჭიროთ ღილაკს **Run**. გარკვეული დროის შემდეგ **DreamweaverMX**-ი გამოიყვანს პანელს **Site Reports**, რომელიც ნაჩვენებია ნახ.4.8-ზე.
ამ პანელში განთავსებულ სიაში, ჩამოთვლილია ჩვენს **HTML**-კოდში მოძიებული შეცდომები და ნაკლოვანებები. თუ ნებისმიერ პუნქტზე ორჯერ დავაწკაპუნებთ, ჩვენ გავხსნით ფაილს, რომელშიც შეცდომაა ნაპოვნი; ამასთან დოკუმენტის ფანჯარა გაიხსნება **HTML**-კოდის წარმოჩენის რეჟიმში, ხოლო შეცდომის შემცველი ფრაგმენტი მონიშნული იქნება. იმისათვის, რომ ნაპოვნი შეცდომების შესახებ უფრო დაწვრილებითი ცნობები მივიღოთ, საჭიროა **More Info** ღილაკზე (ძახილის ნიშნის გამოსახულებით) დაჭერა, რის შემდეგაც გაიხსნება **DreamweaverMX**-ის ცნობარის ფანჯარა, ჩვენთვის სასურველი ცნობებით.



ნახ.4.8. პანელი Site Reports

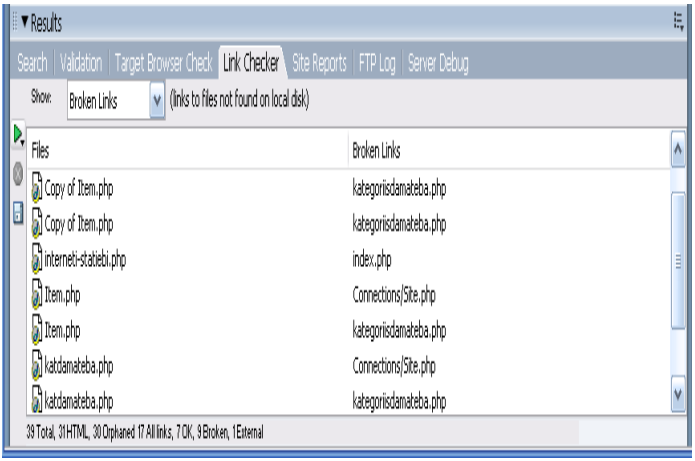
პრინციპში, ის რაც იპოვა **DreamweaverMX**-მა ჩვენს კოდში, არც თუ ისე საშიში შეცდომებია, ვებ-დამთვალიერებლების არც ერთი პროგრამა არ მიაქცევს მათ ყურადღებას. მაგრამ იქ თუ შეგვხვდება რაიმე ნამდვილად სერიოზული, უმჯობესია ის გასწორდეს.
ვინაიდან პანელი **Site Reports** ჩვენ ამის შემდეგ აღარ დაგვჭირდება, შეიძლება მისი დახურვა. ამისათვის საკმარისია მის დამატებით მენიუში პუნქტ **Close Panel Group** არჩევა.

ჰიპერმინიშნებების შემოწმება

ვთქვათ, რომ ჩვენ სახელი გადავარქვით ჩვენი საიტის რომელიმე ვებ-გვერდს და არ შევასწორეთ შესაბამისი ჰიპერმინიშნება, რომელიც მასზე მიუთითებს, და არც **DreamweaverMX**-მა გააკეთა ეს ჩვენს მაგივრად. ასეთ ჰიპერმინიშნებებს, რომლებიც მიუთითებენ არარსებულ ფაილზე, "**გაწყვეტილ**" ან "**მკვდარ**" ჰიპერმინიშნებებს უწოდებენ.
თუ ჩვენი საიტის დამთვალიერებელი დააწკაპუნებს ასეთ ჰიპერმინიშნებაზე, ვებ-სერვერი ვერ მოძებნის არარსებულ გვერდს

და ვებ-დამთვალიერებელს გაუგზავნის შეტყობინებას შეცდომის შესახებ კოდით 404. სწორედ ეს არის ის, ავად სახსენებელი **შეცდომა 404,** რომელსაც ჩვენ ხშირად ვაწყდებით, როცა ვმოგზაურობთ ინტერნეტ-ქსელის უკიდეგანო სივრცეებში.

რათქმა უნდა **DreamweaverMX**-ი ყველაფერს აკეთებს, რომ არ დაუშვას "გაწყვეტილი" ჰიპერმინიშნებების არსებობა. მაგრამ, დაზღვეულები მაინც ვერ ვიქნებით. ამიტომ, საიტის პუბლიკაციის წინ საჭიროა ჰიპერმინიშნებების კორექტულობის შემოწმება. კორექტულობის შემოწმება შესაძლებელია როგორც ცალკეულ გვერდებზე, ასევე მთელ საიტზეც. ცალკეულ გვერდზე ჰიპერმინიშნებების შესამოწმებლად, პირველ რიგში საჭიროა მისი გახსნა. ამის შემდეგ ავირჩიოთ მთავარი ფანჯრის მენიუ **File**-ის, ქვემენიუ **Check Page**-ის, პუნქტი **Check Links** ან პანელ **File**-ის დამატებითი მენიუს, ქვემენიუ **File**-ის იმავე სახელწოდების პუნქტი, ან კიდევ დავაჭიროთ კლავიშების კომბინაციას **<Shift>+<F8>**. თუ ჩვენ გვსურს ჰიპერმინიშნებების შემოწმება საიტის ყველა გვერდზე, უნდა გამოვიყენოთ პანელ **File**-ის დამატებითი მენიუს, ქვემენიუ **Site**-ის, პუნქტ **Check Links Sitewide** ან დავაჭიროთ კლავიშების კომბინაციას **<Ctrl>+<F8>.**



ნახ.4.9. პანელი Link Checker

ამის შემდეგ ნებისმიერ შემთხვევაში ეკრანზე გამოჩნდება პანელი **Link Checker**, რომელიც ნახ. 4.9.-ზეა ნაჩვენები.

ამ პანელის ზედა ნაწილში მდებარეობს ჩამოსაშლელი სია **Show**, სადაც შეიძლება ავირჩიოთ, თუ რა უნდა აისახოს პანელზე წარმოდგენილ სიაში. ამ სიაში ხელმისაწვდომია სამი პუნქტი:

-**Broken Links**-"გაწყვეტილი" ჰიპერმინიშნებები;
-**External Links**-"გარე" ანუ სხვა საიტებზე მიმთითებელი ჰიპერმინიშნებები;
-**Orphaned Files**-"ობოლი" ფაილები, ანუ ფაილები, რომლებზეც არ მიუთითებს არც ერთი ჰიპერმინიშნება.

პანელ **Files**-ის უმეტესი ნაწილი დაკავებული იქნება ნაპოვნი არასწორი ჰიპერმინიშნებების სიით. ამ სიის ყოველი პუნქტი შეიცავს ვებ-გვერდის მისამართს, სადაც ნაპოვნი იყო ჰიპერმინიშნება და ამ მინიშნების ინტერნეტ-მისამართს. სიის ნებისმიერ პოზიციაზე დაჭერით ჩვენ გავხსნით შესაბამის ვებ-გვერდს დოკუმენტის ფანჯარაში, ამასთან **DreamweaverMX**-ი თვითონ მონიშნავს არასწორ მინიშნებებას.
ახლა მოდით ვნახოთ, არის თუ არა ყველაფერი რიგზე ჰიპერმინიშნებებთან დაკავშირებით. მაშ ასე, "გაწყვეტილი" მინიშნებები არ გაგვაჩნია, ხოლო გარე მინიშნება მხოლოდ ერთია-ელექტრონული ფოსტის მისამართი. "ობოლი" ფაილები ასევე არა გვაქვს. ესეიგი ყველაფერი ნორმალურად არის.

პანელ Files-ის და დოკუმენტების ფანჯრის ურთიერთქმედება.

დავუშვათ, რომ ჩვენ მაინც მოგვივიდა შეცდომა ჰიპერმინიშნებებში. გარდა ამისა, ჩვენ დაგვავიწყდა ფაილებისა და სტატიების გვერდებზე შეგვექმნა ჰიპერმინიშნებები, რომლებიც შესაბამისი კატეგორიების სიებზე მიუთითებენ. ეს კი უწესრიგობაა-ჩვენი საიტის დამთვალიერებელს უნდა ჰქონდეს შესაძლებლობა უპრობლემოდ იმოგზაუროს გვერდიდან გვერდზე. მოდით გავასწოროთ შეცდომები.
ჩვენთვის ჰიპერმინიშნებების შექმნის ორი ხერხია ცნობილი და ორივე მათგანი ძალზე მარტივია. მაგრამ **DreamweaverMX**-ი ჩვენი საიტის გვერდების ერთმანეთთან დაკავშირების კიდევ ორ შესაძლებლობას გვაძლევს, რომლებიც ხელმისაწვდომი ხდებიან იმ შემთხვევაში, თუ ჩვენ დავარეგისტრირებთ ჩვენს საიტს (საიტის **DreamweaverMX**-ში დარეგისტრირების შესახებ იხილეთ ამ თავის დასაწყისი). ჩვენ ახლა მათ განვიხილავთ.

ჯერ შევამოწმოთ, გახსნილი გვაქვს თუ არა პანელი **Files**; თუ არა, ჩავრთოთ პუნქტი **Files**-ის გადამრთველი **Windows** მენიუში ან დავაჭიროთ კლავიშას **<F8>**. შემდგომ პანელ Files-ში, საიტის ლოკალური ასლის ფაილების სიაში, ფაილის დასახელებაზე ორჯერ დაწკაპუნებით გავხსნათ ვებ-გვერდი **Articles_internet.htm**. ამის შემდეგ ტექსტური კურსორი დავაყენოთ სტატიების სიის ქვემოთ, ავკრიფოთ ტექსტი **კატეგორიების სიაზე** და მოვნიშნოთ იგი. სწორედ ამ ტექსტს გადავაქცევთ ჩვენ ჰიპერმინიშნებად, რომელიც მიუთითებს ვებ-გვერდზე **Articles_internet.htm**.

ახლა ცოტათი გავიმარტივოთ სიცოცხლე. როგორც წესი **DreamweaverMX**-ი პირველივე გაშვებისას ხსნის უამრავ პანელს, რითაც მთლიანად მიუვალს ხდის თავისი ფანჯრის მარჯვენა ნაწილს. რომ გამოვანთავისუფლოთ იგი, მოდით დავხუროთ ყველა პანელი, გარდა **Files**-ისა და თვისებების რედაქტორისა, რისთვისაც საკმარისია დამატებითი მენიუს **Close Panel Group** პუნქტის არჩევა. შემდგომ მოვათავსოთ "მაუსის" კურსორი პანელ **Files**-ის ფანჯრის ლურჯი სათაურის მარცხენა კიდეზე, იქ სადაც მდებარეობს თავისებური "სახელურიც", დავაჭიროთ "მაუსის" მარცხენა ღილაკს და გადავათრიოთ პანელი ეკრანის შუაში. ასეთნაირად ჩვენ გამოვანთავისუფლებთ თვისებების ფანჯრის მარჯვენა ნაწილს.

კი მაგრამ რა ხდება იქ ასეთი საინტერესო? მოდით შევხედოთ თვისებების რედაქტორს, კერძოდ შესატან ველს **Link**, სადაც მოცემულია ჰიპერმინიშნების ინტერნეტ-მისამართი. მისგან მარჯვნივ მდებარეობს პატარა ნიშანი  . ასე რომ, ჰიპერმინიშნების შესაქმნელად, საკმარისია მისი გადათრევა პირდაპირ საჭირო ფაილზე, პანელ **Files**-ის სიაში. ასე მაგალითად, თუ ჩვენ გადავათრევთ ამ ნიშანს ფაილების სიაში მდებარე **Articles.htm** ფაილზე, და "დავაგდებთ" მას, ჩვენს მიერ მონიშნული ტექსტი ***კატეგორიების სიაზე***, გარდაიქმნება ჰიპერმინიშნებად. სწორად ეს არის **Files**-პანელის საშუალებით ჰიპერმინიშნების შექმნის პირველი ხერხი.

მეორე ხერხიც ასევე მარტივია. ჰიპერმინიშნების შესაქმნელად, ჩვენ შეგვილია **Files**-პანელის სიიდან, საჭირო ფაილის გადათრევა პირდაპირ თვისებების რედაქტორის შესატან ველში **Link**. როდესაც ჩვენ დავიწყებთ კატეგორიების სიაზე მიმთითებელი ჰიპერმინიშნებების შექმნას სხვა გვერდებზეც, შეგვეძლება ამ ხერხის მოსინჯვაც.

ამით დასრულდ? დიახ. ახლა ჩვენ ნამდვილად შეგვიძლია საიტის გამოქვეყნება საკუთარ ვებ-სერვერზე. პერსპექტივაში-ქსელშიც.

საიტის პუბლიკაცია

მაშ ასე, დადგა ის სანუკვარი მომენტი, როდესაც ჩვენი საიტის შექმნა მთლიანად დასრულებულია! ჩვენ შევქმენით ყველა გვერდი, გავასწორეთ შეცდომები **HTML**-კოდში და ჰიპერმინიშნებების ინტერნეტ-მისამართების კორექტულობაც კი შევამოწმეთ. ფაქტიურად ჩვენ გავაკეთეთ ყველაფერი ის, რაც ხშირად საკმაოდ გამოცდილ ვებ-დიზაინერებსაც კი ავიწყდებათ. ამის შემდეგ შეიძლება ჩვენი საიტის პუბლიკაცია ვებ-სერვერზე.

ჯერ ჩვენ განვიხილავთ საკითხს, თუ როგორ ხდება საიტის პუბლიკაცია ლოკალურ ვებ-სერვერზე, სწორედ იმ ვებ-სერვერზე, რომელიც ჩვენ დავაყენეთ საკუთარ კომპიუტერზე ამ თავის დასაწყისში. ხოლო შემდეგ, როდესაც ყველაფერს დაწვრილებით შევამოწმებთ, გადავალთ ჩვენი საიტის პუბლიკაციაზე ინტერნეტ-ქსელში.

საიტის პუბლიკაცია ლოკალურ Web -სერვერზე.

ჩვენი საიტის გამოქვეყნების უმარტივესი ხერხი-ეს არის პანელ **File**-ში, საიტის ლოკალური ასლის ფაილების სიაში, ძირეული საქაღალდეს (იერარქიული სიის "ფესვი") მონიშვნა და ინსტრუმენტების პანელში **Put File(s)** ღილაკზე () დაჭერა. მოცემული ღილაკით გაიშვება ფაილების კოპირების პროცესი ვებ-სერვერის ფესვურ საქაღალდეში. ასევე ჩვენ შეგვიძლია ავირჩიოთ პანელის დამატებითი მენიუს **Site** ქვემენიუს პუნქტი **Put** ან დავაჭიროთ კლავიშების კომბინაციას **<Ctrl>+<Shift>+<U>.**

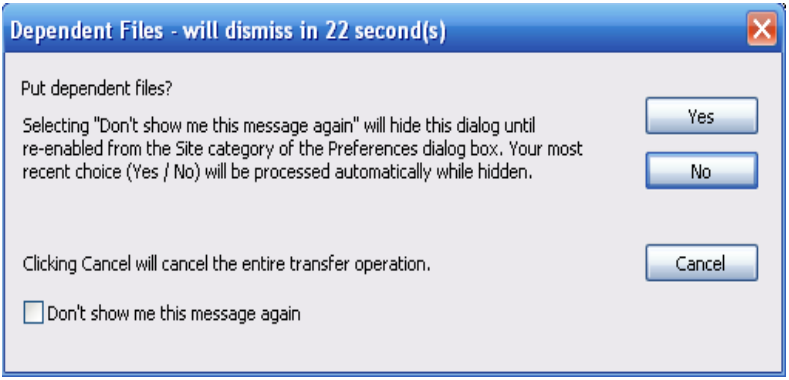
რადგანაც ჩვენ ავირჩიეთ ძირეული საქაღალდე, ანუ ფაქტიურად მთელი საიტი, **DreamweaverMX**-ი შეგვეკითხება, მართლა გვინდა თუ არა მთელი საიტის კოპირება სერვერზე. პასუხად ვაჭერთ ღილაკს OK. თვალის დახამხამებაში ჩვენი საიტი გამოქვეყნებული იქნება ვებ-სერვერზე.

მოდით ვნახოთ, გააკეთა კი ყველაფერი სწორად **DreamweaverMX**-მა. პანელ **Files**-ის ინსტრუმენტების მარჯვენა ჩამოსაშლელ სიაში ავირჩიოთ პუნქტი **Remote View**. ამ პანელის სია მაშინათვე დაგვანახებს საიტის დაშორებული ასლის ყველა ფაილს. რომ

გავაგრძელოთ მუშაობა ლოკალურ ასლთან, საკმარისია ამავე სიაში ავირჩიოთ პუნქტი **Local View**.
ამის შემდეგ შეიძლება გაიშვას ვებ-დამთვალიერებელი, მისამართის სტრიქონში აიკრიფოს **http://localhost/** და ხელმეორედ დავაკვირდეთ ჩვენ საიტს.

ყურადღება! *თუ ჩვენი სერვერი იყენებს პორტს, რომელიც განსხვავდება მე-80-საგან, ჩვენ მოგვიწევს მისამართის სტრიქონში* ***http://localhost****:<პორტის ნომერი> აკრეფა.*

DreamweaverMX-ი ვებ-სერვერზე ცალკეული ფაილების გამოყენების საშუალებასაც იძლევა. ამისათვის საკმარისია პანელ **Files**-ის სიაში საჭირო ფაილების მონიშვნა და იმავე ღილაკის **Put File(s)** დაჭერა. თუ ჩვენ რომელიმე ვებ-გვერდები გვაქვს გახსნილი, შეცვლილი და დაუმახსოვრებელი, **DreamweaverMX**-ი შემოგვთავაზებს მათ დამახსოვრებას. ამ შემთხვევაში უნდა დავაჭიროთ ღილაკს **Yes**, შესაბამისი ფაილის დასამახსოვრებლად, ხოლო **No**-დამახსოვრების უარსაყოფად ან ღილაკს **Cancel**-მისი პუბლიკაციის უარსაყოფად.
ასევე, ეკრანზე **Dependent Files** დიალოგური ფანჯრის (ნახ.4.10) გამოტანით, **DreamweaverMX**-ი შეიძლება შეგვეკითხოს, გამოაქვეყნოს თუ არა სერვერზე ფაილები, რომლებთანაც მოცემული გვერდი დაკავშირებულია ჰიპერმინიშნებებით. თუ დავაჭერთ ღილაკს **Yes**, ეს ფაილები გამოქვეყნდება, თუ დავაჭერთ **No**-გამოქვეყნდება მხოლოდ ის ფაილები, რომლებიც ჩვენ მოვნიშნეთ პანელ **Files**-ის სიაში, თუ **Cancel**-სერვერზე ფაილების პუბლიკაცია შეწყდება.



ნახ.4.10. დიალოგური ფანჯარა Dependent Files

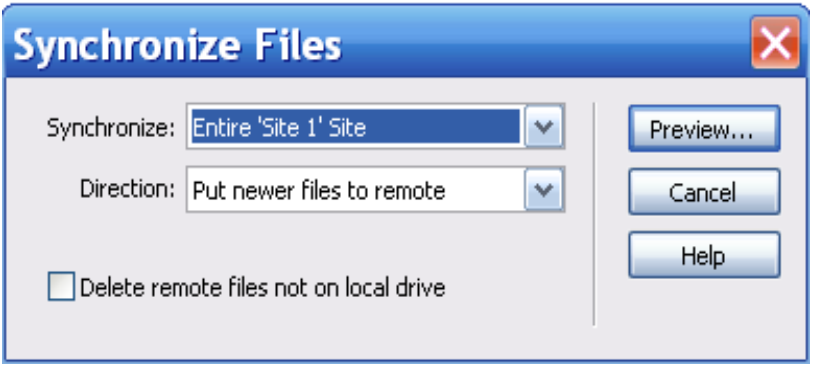
დიახ, ყველაფერი ეს მარტივია, თუ ჩვენ ზუსტად ვიცით რომელი ფაილების გამოქვეყნება გვინდა. მაგრამ, რა ვქნათ მაშინ, თუ ჩვენ დაგვავიწყდა, თუ რომელი ფაილები შევცვალეთ (ჩვეულებრივი სიტუაცია)? ხელახლა უნდა მოვახდინოთ მთელი საიტის პუბლიკაცია? არა, უკეთესი გზაც არსებობს!

საქმე იმაშია, რომ ოპერაციული სისტემა **Windows** (როგორც მრავალი სხვა ოპერაციული სისტემა) ინახავს ყოველი ფაილის უკანასკნელი ცვლილების თარიღს და დროს. ორი თარიღის შედარებით, შეგვიძლია გავარკვიოთ, რომელი ფაილია უფრო ახალი. თუ ფაილს, რომელიც ლოკალურ ასლს მიეკუთვნება, გააჩნია უკანასკნელი ცვლილების უფრო გვიანდელი თარიღი, ვიდრე დაშორებული ასლის იმავე ფაილს, ეს იმას ნიშნავს რომ იგი შეცვლილი იყო **DreamweaverMX**-ში, მაგრამ ჯერ გამოქვეყნებული არ არის.

შესაბამისად უნდა მოხდეს მისი კოპირება სერვერზე, რათა შენარჩუნებულ იქნას საიტის წაშლილი ასლის აქტუალურობა.

სწორად ასეთ პრინციპზეა დაფუძნებული საიტის ასლების **სინქრონიზაციის** მექანიზმი. **DreamweaverMX**-ი ამოწმებს ფაილების სხვადასხვა ასლების თარიღებს და იღებს გადაწყვეტილებას, თუ რომელი მათგანის კოპირება უნდა მოხდეს სერვერზე. ეს პროცედურა მარტივი და საიმედოა, თუ რათქმა უნდა ჩვენი კომპიუტერის ჩაშენებული საათი და კალენდარი სწორად არიან დაყენებული.

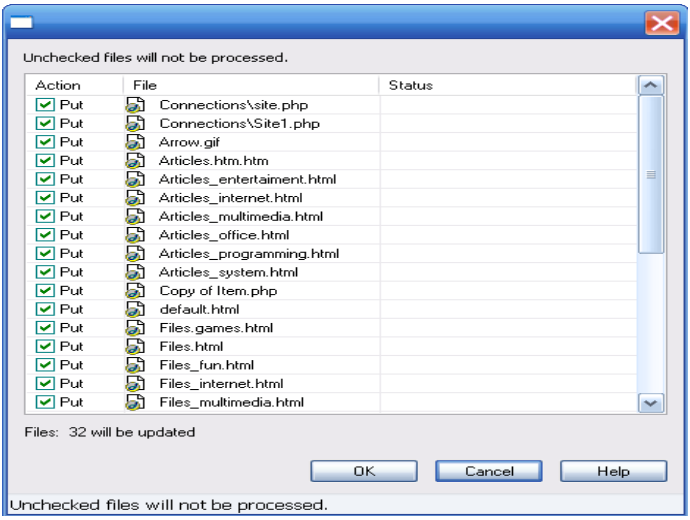
იმისათვის, რომ ჩავრთოთ ფაილების სინქრონიზაციის პროცესი, ავირჩიოთ პანელ **Files**-ის დამატებითი მენიუს, ქვემენიუ **Site**-ის პუნქტი **Synchronize Files**. ეკრანზე გამოჩნდება დიალოგური ფანჯარა **Synchronize Files**, რომელიც ნაჩვენებია ნახ.4.11.



ნახ.4.11. დიალოგური ფანჯარა Synchronize Files

ჩამოსაშლელი სია **Synchronize** იმ ფაილების განსაზღვრის საშუალებას გვაძლევს, რომელთა სინქრონიზებაც გვჭირდება. პუნქტი **Selected Local File Only(მხოლოდ არჩეული ლოკალური ფაილები)** საშუალებით შეგვიძლია მხოლოდ სიაში მონიშნული ფაილების სინქრონიზირება. ხოლო პუნქტი **Entire (მთლიანად)** <ჩვენი საიტის დასახელება> **Site**, მთლიანად მთელი საიტის სინქრონიზაციის საშუალებას გვაძლევს. ასე რომ მოდით სწორედ ეს უკანასკნელი ავირჩიოთ.
ჩამოსაშლელ სიაში **Direction**, ჩვენ დაგვჭირდება პუნქტ **Put newer files to remote(უახლესი ფაილების გაგზავნა დაშორებულ სერვერზე)** არჩევა, რათა **DreamweaverMX**-მა მოახდინოს ახალი ფაილების კოპირება სერვერზე და ამით მოახდინოს საიტის დაშორებული ასლის მოძველებული ფაილების ჩანაცვლება. თუ ჩვენ წაშლილი გვაქვს რომელიმე ფაილები, მაშინ დაგვჭირდება ასევე **Delete remote files not on local drive (ლოკალურ სერვერზე არარსებული ფაილების წაშლა დაშორებულ სერვერზე)** ალმის ჩართვა.

თვით სინქრონიზაციის პროცესის გაშვება ხდება **Preview** ღილაკზე დაჭერით. ამის შემდეგ ეკრანზე გამოჩნდება სინქრონიზაციას დაქვემდებარებული ფაილების სიის ფანჯარა, რომელიც ნაჩვენებია ნახ. 4.12.-ზე. აქ ჩვენ დამატებით შეგვიძლია მივუთითოთ, თუ რომელი ფაილების სინქრონიზებაა საჭირო და რომელთა-არა;



ნახ.4.12. სინქრონიზაციას დაქვემდებარებული ფაილების სიის ფანჯარა.

ამ ფანჯრის უმეტესი ნაწილი დაკავებულია სინქრონიზაციას დაქვემდებარებული ფაილების სიით. ის ორგანიზებულია სამ სვეტიანი ცხრილის სახით, რომელთაც ჩვენ ახლა განვიხილავთ.
სვეტი **Action** შეიცავს იმ მოქმედების დასახელებას, რომელიც ფაილის მიმართ იქნება გამოყენებული. ჩვენს შემთხვევაში ასეთი მოქმედება შეიძლება იყოს ორი:

-**Put**-კოპირება სერვერზე,
-**Delete**-წაშლა.

მოქმედების დასახელების მარჯვენა მხარეს მდებარეობს ალამი, რომელიც წინასწარ არის ჩართული.
მისი გამორთვით, ჩვენ შეგვიძლია ამ ფაილზე მოქმედების (შესაბამისად კოპირების ან წაშლის) გაუქმება.
სვეტი **File** შეიცავს ფაილის სახელს, ხოლო სვეტი **Status**-ტექსტს, რომელიც აღნიშნავს სინქრონიზაციის მდგომარეობას და გამოყვანილი იქნება მისი დასრულებისას.
სინქრონიზაციის პროცესის გაშვება ხდება **OK** ღილაკზე დაჭერით. თუ რომელიმე ფაილები სინქრონიზაციის შედეგად უნდა

წაიშალონ, **DreamweaverMX**-ი გაგვაფრთხილებს ამის შესახებ; **OK** ღილაკზე დაჭერა ახორციელებს ამ ფაილების წაშლას, ხოლო ღილაკის **Cancel** დაჭერა-უარყოფს მას.
სინქრონიზაციის დასრულებისას **DreamweaverMX**-ი იმავე ფანჯარაში გამოიყვანს სინქრონიზაციის შედეგებს და შეავსებს სვეტს **Status**. ამის შემდეგ საჭიროა ღილაკზე **Close** დაჭერით ამ ფანჯრის დახურვა.

შენიშვნა: *იმისათვის, რომ გამოვაქვეყნოთ ჩვენი საიტი, ასევე შეიძლება Windows-ის გამცილებლის გახსნა (ან ფაილების მართვის ანალოგიური ფანჯრის) და ლოკალური ასლის ფაილების უბრალოდ სერვერის "ძირეულ" საქაღალდეში კოპირება. მაგრამ, ბუნებრივია **DreamweaverMX**-ში საიტის გამოქვეყნება ხორციელდება უფრო მარტივად და თვალსაჩინოდ.*

Web -საიტის პუბლიკაცია დაშორებულ სერვერზე

ლოკალურ **Web**-სერვერზე საიტის პუბლიკაციასთან დაკავშირებული საკითხების განხილვის შემდეგ, მოდით გადავიდეთ დაშორებულ სერვერზე ანუ ინტერნეტში მისი პუბლიკაციის პროცესზე. დავუშვათ, რომ ჩვენ ყველაფერი შევამოწმეთ და გავასწორეთ ყველა შეცდომა, ასე, რომ ჩვენი საიტი მზად არის ექსპლუატაციისათვის. მაგრამ, როგორ განვახორციელოთ ეს?

Web -საიტის პუბლიკაციისათვის FTP პროტოკოლის გამოყენება

დღეისათვის დაშორებულ სერვერზე საიტის პუბლიკაციისათვის პრაქტიკულად ყოველთვის **FTP** პროტოკოლი გამოიყენება. პირველ თავში ჩვენ ავღნიშნეთ, რომ ამ პროტოკოლის გამოყენება შესაძლებელია, როგორც **FTP**-კლიენტის მიერ **FTP**-სერვერიდან ფაილების გადმოსატვირთად, ასევე ფაილების ჩასატვირთად სერვერზე. ამგვარად, ჩვენ შეგვიძლია გავუშვათ **FTP**-კლიენტის პროგრამა (**CuteFTP**, **FileZilla** ან ნებისმიერი სხვა) და უბრალოდ გადავწეროთ ჩვენი საიტის ფაილები ვებ-სერვერის ძირეულ საქაღალდეში. უფრო მეტიც, ჩვენ ამისათვის არც კი დაგვჭირდება **FTP**-კლიენტი. თვით **DreamweaverMX**-ს გააჩნია **FTP**-სერვერზე საიტების პუბლიკაციის საშუალებები.

ზემოთ ნათქვამიდან გამომდინარეობს, რომ სერვერულ კომპიუტერზე, რომელზეც მუშაობს უფასო ვებ-სერვერის პროგრამა, ასევე უნდა იყოს დაყენებული **FTP**-სერვერი (ტექნიკურად ამის გაკეთება რთული არ არის). ორივე ეს სერვერული პროგრამა ისე უნდა იყოს გამართული, რომ ყოველმა მათმა მომხმარებელმა მიიღოს თავის განკარგულებაში თავისი ძირეული საქაღალდე (ესეც საკმაოდ მარტივად კეთდება). ამასთან, რათქმა უნდა სერვერულ კომპიუტერზე უნდა მუშაობდეს ახალი მომხმარებლების დამრეგისტრირებელი პროგრამა, რომელიც კლიენტების შესახებ ინფორმაციას შეიტანს ვებ და **FTP** სერვერების მომხმარებელთა სიაში და ყოველი ახალი მომხმარებლისათვის შექმნის ძირეულ საქაღალდეს.
მოდით დაწვრილებით განვიხილოთ ვებ-სერვერზე **FTP**-პროტოკოლის საშუალებით საიტის პუბლიკაციასთან დაკავშირებული, ჩვენი ქმედებების თანმიმდევრულობა.

1. ჩვენ ვრეგისტრირდებით უფასო სერვერზე, რისთვისაც ვაცნობებთ მას ჩვენს **მომხმარებლის სახელს** (ან ლოგინს, ინგლისურიდან-**login**) და **პაროლს**. მაშინათვე მომხმარებლის სახელი და პაროლი შეიტანება **Web** და **FTP** სერვერების მომხმარებელთა სიებში, ხოლო სერვერული კომპიუტერის დისკოზე ავტომატურად იქმნება ჩვენი საიტის ძირეული საქაღალდე. ასევე ავტომატურად იქმნება ჩვენი საიტის ინტერნეტ-მისამართი და მის შესახებ გვეგზავნება შეტყობინება.

2. ჩვენი სახელისა და პაროლს შეტანის შემდეგ, **FTP**-კლიენტ პროგრამის საშუალებით ან პირდაპირ **DreamweaverMX**-იდან ვერთვებით **FTP** სერვერში. **FTP** სერვერი ამოწმებს, არსებობს თუ არა მის სიებში ასეთი სახელი და პაროლი და თუ არსებობს, დაგვაკავშირებს პირდაპირ ჩვენს ჯერ კიდევ ცარიელ ძირეულ საქაღალდესთან. გზა (ბილიკი) სერვერული კომპიუტერის ფაილური სისტემის საქაღალდემდე, რომელიც ჩვენი საიტის ძირეულ საქაღალდეს შეესაბამება, ასევე ინახება **FTP** სერვერის მომხმარებელთა სიაში.

3. ვაკოპირებთ ჩვენი საიტის ფაილებს ჩვენს ძირეულ საქაღალდეში.

4. გამოვეთიშოთ **FTP** სერვერს. ეს აუცილებლად უნდ გაკეთდეს, ვინაიდან **FTP** სერვერის პროგრამა, კლიენტთან კავშირის შესანარჩუნებლად, სერვერული კომპიუტერის მეხსიერებას იყენებს, რომელიც უსასრულო არ არის.
5. ვუშვებთ ვებ-დამთვალიერებელს, შეგვაქვს პირველ პუნქტში მიღებული ჩვენი საიტის ინტერნეტ-მისამართი და ვათვალიერებთ საიტს.

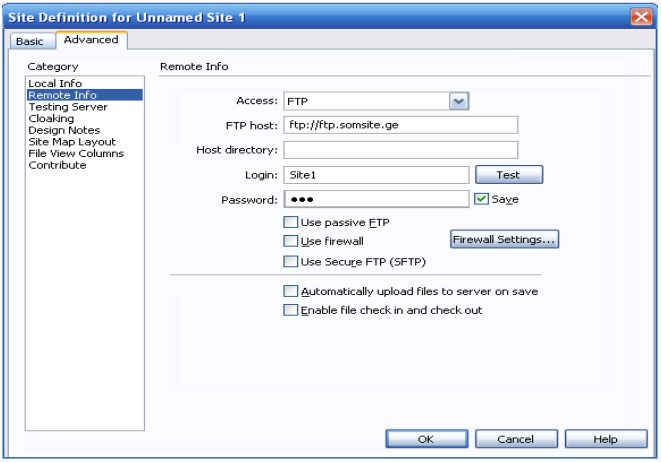
პრინციპში არავითარი სირთულე ამაში არ არის-**Web** და **FTP** სერვერების გამართვასთან დაკავშირებულ ყველა ამოცანას, თავის თავზე იღებს განსაკუთრებული პროგრამა, რომელიც სერვერულ კომპიუტერზე მუშაობს. ჩვენ მხოლოდ გვევალება-არ დაგვავიწყდეს სახელი, პაროლი და ჩვენი საიტის ინტერნეტ-მისამართი.

შენიშვნა: *ზოგიერთი უფასო ვებ-სერვერი მომხმარებელს მისი საიტის ფაილების ჩატვირთვის საშუალებას აძლევს მხოლოდ ვებ-დამთვალიერებლიდან. ეს ხერხი გამოსადეგია მხოლოდ დამწყები მომხმარებლებისათვის, რომლებმაც არ იციან **FTP**-კლიენტებთან მუშაობა და ძალზე მოუხერხებელია.*

DreamweaverMX-ის გამართვა საიტის პუბლიკაციისათვის FTP-ს საშუალებით

ცხადია, საიტის ვებ-სერვერზე პუბლიკაციისათვის **FTP**-ს საშუალებით, შესაძლებელია **FTP**-კლიენტების ჩვეულებრივი პროგრამების გამოყენება. მაგრამ ჩვენ ხომ **DreamweaverMX**-ში ვმუშაობთ, რომელსაც თვითონ შეუძლია ყველაფრის ამის გაკეთება-საჭიროა მხოლოდ დავალების მიცემა. მაშ ასე, დავავალოთ!

ჯერ, ჩვენ დაგვჭირდება დარეგისტრირებული საიტის პარამეტრების მნიშვნელობებში ზოგიერთი ცვლილებების შეტანა. მთავარი ფანჯრის მენიუ **Site**-ში ავირჩიოთ პუნქტი **Manage Site**. ეკრანზე გამოსულ **Manage/Site** დიალოგურ ფანჯარში მოცემულ საიტების სიაში, ავირჩიოთ ჩვენი საიტი და დავაჭიროთ ღილაკს **Edit**. ეკრანზე გაჩნდება დიალოგური ფანჯარა **Site Definition**; ავირჩიოთ პუნქტი **Remote Info** სიაში-**Category**. ამის შემდეგ ჩამოსაშლელ სიაში **Access** ავირჩიოთ პუნქტი **FTP**, რითაც **DreamweaverMX**-ს შევატყობინებთ, რომ ჩვენ ვიწყებთ ფაილების გაგზავნას სერვერზე **FTP** პროტოკოლის საშუალებით (ნახ.4.13).



ნახ.4.13. დიალოგური ფანჯარა Site Definition (კატეგორია Remote Info, არჩეულია ფაილების FTP-პროტოკოლით გაგზავნის რეჟიმი).

ამის მერე ჩვენ დაგვჭირდება შემდეგი პარამეტრების შეტანა **Site Definition** ფანჯრის შესაბამის შესატან ველებში:

-**FTP host**- ვებ-საიტის პუბლიკაციაზე პასუხისმგებელი **FTP**-სერვერის მისამართი;
-**Host directory**-ბილიკი, ჩვენი საიტის ძირეულ საქაღალდემდე. რამდენადაც FTP-სერვერი, რიგორც წესი, მაშინათვე გვაკავშირებს ამ საქაღალდესთან, დავტოვოთ ეს ველი ცარიელი;

-**Login**-მომხმარებლის სახელი, რომლითაც ჩვენ დავრეგისტრირდით სერვერზე;
-**Password**-ჩვენი პაროლი (მისი შეტანის დროს შესატან ველში გამოისახება ან ვარსკვლავები ან წერტილები, იმის მიხედვით, თუ რომელი ვერსიის Windows-ი აყენია ჩვენს კომპიუტერზე).

წინასწარი დათქმით **DreamweaverMX**-ი იმახსოვრებს ჩვენს მიერ შეტანილ პაროლს და მიერთებისას ავტომატურად უგზავნის მას **FTP**-სერვერს. თუ კი ჩვენ, უსაფრთხოების მიზნებიდან გამომდინარე არ გვსურს ამ პაროლის დამახსოვრება და მის ხელახლა შეტანას

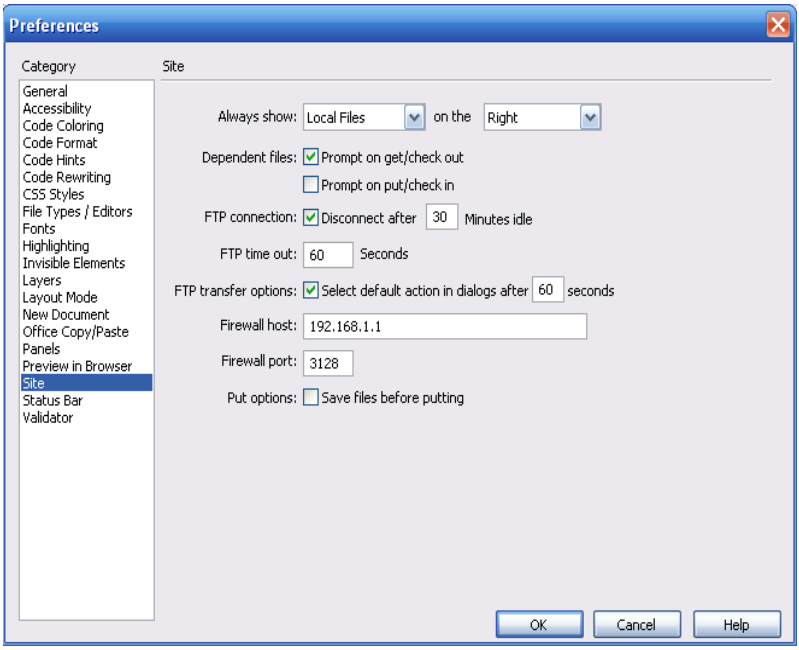
ვაპირებთ ყოველი ჩართვის დროს, მაშინ უნდა გამოვრთოთ ალამი **Save**.

ღილაკზე **Test** დაჭერით, შეგვიძლია ჩვენს მიერ შეტანილი პარამეტრების სისწორის შემოწმება. **DreamweaverMX**-ი შეეცდება **FTP**-სერვერთან დაკავშირებას და წარმატების ან წარუმატებლობის შესაბამისად გამოიტანს გაფრთხილების ფანჯარას სათანადო ტექსტით.

თუ **FTP**-სერვერი, რომლის საშუალებითაც ჩვენ საიტის გამოქვეყნებას ვაპირებთ, მხოლოდ **პასიურ FTP** პროტოკოლს იყენებს (**FTP**-პროტოკოლების განსაკუთრებული ნაირსახეობა, რომელიც გამოიყენება უსაფრთხოების თვალსაზრისიდან გამომდინარე), ჩვენ დაგვჭირდება ალმის **Use passive FTP** ჩართვა. ზოგადად უმჯობესია, რომ ის ყოველთვის ჩართული იყოს.

შემდეგ ჩავრთოთ ალამი **Use Secure FTP (SFTP)**, თუ **FTP**-სერვერზე შესაღწევად გამოიყენება **FTP**-პროტოკოლის **SFTP** დაცული ვერსია (**Secure FTP**).

თუ ჩვენ ქსელში გავდივართ **პროქსი-სერვერის** ან **ბრანდმაუერის** (განსაკუთრებული პროგრამები, რომლებიც იცავენ ორგანიზაციის ქსელს ინტერნეტიდან შემოტევებისაგან) საშუალებით, მაშინ **DreamweaverMX**-ი შესაბამისად უნდ გავმართოთ. ამისათვის ჩავრთოთ ალამი Use **firewall** და დავაჭიროთ ღილაკს **Firewall Settings**. ეკრანზე გამოჩნდება ჩვენთვის მესამე თავიდან უკვე ნაცნობი დიალოგური ფანჯარა **Preferences**, რომელიც გადართული იქნება Site კატეგორიაზე (ნახ.4.14.).



ნახ. 4.14. დიალოგური ფანჯარა Preferences (კატეგორია Site).

აქ ჩვენ მოცემულ მომენტში გვაინტერესებს მხოლოდ შესატანი ველები **Firewall host** (პროქსი-სერვერის მისამართი) და **Firewall port** (პროქსი სერვერის პორტი). ორივე ეს პარამეტრი შეიძლება გავიგოთ პროქსი-სერვერის ადმინისტრატორისაგან. მათი განსაზღვრის შემდეგ რიგრიგობით ვაჭერთ **Preferences** და **Site Definition** დიალოგური ფანჯრების **OK**-ღილაკებს და **Manage Sites** ფანჯრის ღილაკს-**Done**.

შენიშვნა: *თუ ქსელში გასვლა ხორციელდება პროქსი-სერვერით და ბრანდმაუერით, მაშინ დიალოგურ ფანჯარაში* ***Site Definition****, ზედმეტი არ იქნებოდა ასევე* ***Use passive FTP*** *ალმის ჩართვა.*

საიტის პუბლიკაცია FTP პროტოკოლის საშუალებით.

მაშ ასე, მოსამზადებელი სამუშაოები დასრულებულია. დადგა მომენტი, როდესაც ჩვენი ვებ-საიტი უნდა მოვათავსოთ მსოფლიო ქსელში.

პირველი მოქმედება, რომელიც ჩვენ უნდა შევასრულოთ-ეს არის ის, რომ დავამყაროთ კავშირი **FTP**-სერვერთან. ამისათვის ეკრანზე გამოვიტანოთ პანელი **Files** და დავაჭიროთ ღილაკს **Connect to Remote Host** (), რომელიც ამ პანელის ინსტრუმენტებს შორის არის მოთავსებული. თუ ჩვენ გამოვრთავთ, დიალოგური ფანჯრის **Site Definition**-ის, კატეგორია **Remote Info**-ს, ალამს **Save**, რითაც ვუარყოფთ ჩვენს მიერ შეტანილი პაროლის დამახსოვრებას, **DreamweaverMX**-ი გამოიტანს მცირე ზომის დიალოგურ ფანჯარას შესატანი ველით (სადაც უნდა შევიტანოთ პაროლი) და ღილაკებით OK (ჩართვა) და **Cancel** (ჩართვის უარყოფა).

სერვერთან კავშირის დამყარების შემდეგ, ჩვენ შეგვიძლია ჩვენი საიტის გამოქვეყნება ზემოთ აღწერილი ხერხებიდან, რომელიმე ერთ-ერთის გამოყენებით. დიახ, დაშორებულ სერვერზე, **FTP**-სერვერის საშუალებით საიტის პუბლიკაცია, ზუსტად ისევე ხორციელდება, როგორც პუბლიკაცია ლოკალურ სერვერზე.

რათქმა უნდა, ამ შემთხვევაში უფრო მეტხანს მოგვიწევს ლოდინი, ვინაიდან ინტერნეტში შეღწევის არხი, შედარებით სწრაფიც კი, გაცილებით ნელა მუშაობს, ვიდრე მყარი დისკო.

DreamweaverMX-ი საიტის სერვერზე გაგზავნის პროცესის მიმდინარეობას გვანახებს სპეციალურ, მცირე ზომის დიალოგურ ფანჯარაში. ჩვენ შრგვიძლია შევწყვიტოთ საიტის გაგზავნის ოპერაცია, თუ ამავე ფანჯრის ღილაკს Cancel დავაჭერთ.

მნიშვნელოვანია: *საიტის დაშორებულ ასლთან მუშაობის დასრულებისთანავე, აუცილებლად უნდა გავწყვიტოთ კავშირი **FTP**-სერვერთან, რისთვისაც საკმარისია კიდევ ერთხელ **Connect to Remote Host** ღილაკზე დაჭერა.*

ყურადღება! *აუცილებლად უნდა გავწყვიტოთ კავშირი **FTP**-სერვერთან, საიტის დაშორებულ ასლთან მუშაობის დასრულებისთანავე! კლიენტთან ყოველი კავშირის მხარდასაჭერად **FTP**-სერვერი, სერვერული კომპიუტერის მეხსიერებას იყენებს და შეიძლება დადგეს ისეთი მომენტი, როდესაც სხვა კლიენტებისათვის საკმარისი რესურსები არ აღმოჩნდეს. პატივი ვცეთ სხვა კლიენტებსაც.*

გილოცავთ! თქვენ უკვე ნამდვილი ვებ-დიზაინერები გახდით! თქვენ უკვე გაქვთ თქვენი ვებ-საიტი ინტერნეტში! თქვენ გყავთ დამთვალიერებლები და თაყვანისმცემლებიც!

რა იქნება ამის მერე ?

ყოველივე აღნიშნული კარგი საბაბია მცირედი შესვენებისათვის. ამის შემდეგ კი-წინ ინტერნეტ-პროგრამირების მიუწვდომელი ციხესიმაგრის ასაღებად! (საბრძოლო მოქმედებების განვითარებას იხილავთ ამ წიგნის მეორე ნაწილში).

ნაწილი 2

ჩვენი პირველი სერვერული პროგრამები

თავი 5. შესავალი ვებ-პროგრამირებაში
თავი 6. მონაცემთა ბაზები
თავი 7.PHP-სერვერული გამოყენებითი პროგრამების დაწერის ტექნოლოგია
თავი 8. უმარტივესი სერვერული ვებ-გვერდები
თავი 9. მონაცემების შეტანისა და ჩასწორების რეალიზაცია

პირველი ვებ-საიტის შექმნა არცთუ ისე რთული იყო. მითუმეტეს, რომ ამაში ჩვენ **Micromedia DreamweaverMX**-ი გვეხმარებოდა.

დროა გადავდგათ მეორე ნაბიჯი თანამედროვე ინტერნეტ-ტექნოლოგიების სამყაროში-დავიწყოთ ინტერნეტ-პროგრამირება. ჩვენი მეორე საიტი წარმოდგენილი იქნება, როგორც ბაზებიდან მონაცემების ამომკრეფი და მათი ჩვეულებრივ ვებ-გვერდებად გარდამქმნელი ნამდვილი პროგრამა. ასეთი პროგრამის დაწერაში ჩვენ ისევ უბადლო **DreamweaverMX**-ი დაგვეხმარება.

თავი 5

შესავალი ვებ-დაპროგრამებაში

ჩვენ შევქმენით ჩვენი პირველი ვებ-საიტი. ის მშვენივრად მუშაობს, მას საკმაოდ ბევრი დამთვალიერებელი ჰყავს, მას თაყვანისმცემლებიც კი გაუჩნდა, რაც ძალიან კარგია ჩვენი შესაძლებლობების შემოწმებისათვის. მაშ, კიდევ რა გვინდა ჩვენ?

მართალია, ჯერ ჩვენი საიტი არცთუ ისე პრეზენტაბელურია გარეგნულად, მაგრამ ეს გამოსწორებადია. თუ უკეთესად შევისწავლოთ **DreamweaverMX**-ს, **HTML**-ს და **CSS**-ს, შევძლებთ უფრო ლამაზი ვებ-გვერდების შექმნას, შევარჩევთ ფერებს, შრიფტებს, ყველაფერ ამას გავაფორმებთ ისეთნაირად, რომ მომხმარებლები კმაყოფილები დარჩნენ-და ყველაფერი მშვენივრად იქნება. ვებ-გვერდების შექმნის ძირითადი პრინციპები ჩვენთვის უკვე ცნობილია, ხოლო დანარჩენს მოევლება.

მაგრამ, როგორც კი ჩვენ დავიწყებთ სტატიებისა და ფაილების სიებში ახალი პოზიციების დამატებას, როგორც კი შევეცდებით ჩვენი საიტის გაფართოებას, ვთქვათ, მასში ახლად დამატებული პოზიციების სიების შეტანას ან მოძიებას, მაშინათვე წავაწყდებით გადაულახავ წინააღმდეგობას. **HTML** ენა არ გვთავაზობს ძიების ორგანიზების არანაირ ინსტრუმენტს, ესეიგი ჩვენ მოგვიწევს სპეციალური საძიებო პროგრამების დაწერა. მაგრამ, როგორ უზრუნველვყოთ მომხმარებლისათვის სტატიების სიის არა დასახელებების, არამედ ავტორების მიხედვით დახარისხების შესაძლებლობა. თუ მხოლოდ **HTML** ენის საშუალებებით ვისარგებლებთ-არანაირად.

სწორედ აქ ჩვენ შევდივართ ვებ-პროგრამირების სფეროში. რისთვის და როგორ-ეს არის ის ორი კითხვა, რასაც პასუხი გაეცემა ამ თავში. მაგრამ, პასუხის გასაცემად შორიდან მოგვიწევს დაწყება.

სტატიკური ვებ-გვერდების ნაკლოვანებები და მათი დაძლევა.

ჯერ ჩვენ ვისაუბრებთ იმის შესახებ, თუ როგორ ინახება ინფორმაცია, რომლის წარმოდგენასაც ვაპირებთ ჩვენს საიტზე და როგორ მიეწოდება იგი მომხმარებელს, კერძოდ, მონაცემებზე და მათ წარმოდგენაზე. აღნიშულში გარკვევის შემდეგ, ჩვენ ადვილად გავიგებთ ყველაფერ დანარჩენსაც.

მონაცემები და მათი წარმოდგენა

მოდით გავხსნათ ჩვენს მიერ ადრე შექნილი სტატიებისა და ფაილების სიების ნებისმიერი ვებ-გვერდი, ნებისმიერ ტექსტურ რედაქტორში ან **HTML**-კოდის წარმოჩენის რეჟიმში გადართულ **DreamweaverMX**-ში. (შეგვიძლია გავხსნათ კატეგორიების სიის გვერდიც-ეს არც თუ ისე მნიშვნელოვანია. მთავარია არ გავხსნათ თავფურცელი-ვინაიდან არავითარ რეალურ ინფორმაციას,

რომლისთვისაც საიტი იქმნებოდა, ის არ შეიცავს). ყურადღებით დავაკვირდეთ გახსნილ ვებ-გვერდს. რას დავინახავთ? თუ ყურადღებას არ მივაქცევთ **HTML** -კოდის ხლართებს და კარგად დავფიქრდებით, შეიძლება ნებისმიერი ვებ-გვერდის ორი ნაწილი გამოვყოთ. ეს არის თვით ის ინფორმაცია, რომლისთვისაც საიტი შეიქმნა და ამ ინფორმაციის წარდგენა ადამიანებისთვის-მომხმარებლებისთვის, დამთვალიერებლებისთვის, თვალსაჩინო სახით. მოდით უფრო დაწვრილებით ვისაუბროთ მათ შესახებ.

ინფორმაცია-ეს თვით მონაცემებია, ნებისმიერი ვებ-გვერდის "სული და გული". როგორც წესი, იგი ექვემდებარება მისი სტრუქტურის განმსაზღვრელ მკაცრ წესებს ან სხვანაირად რომ ვთქვათ, **სტრუქტურირებულია**. მაშ ასე, თუ ჩვენ შევხედავთ სტატიების სიას, მაშინ დავინახავთ, რომ ის დაყოფილია კონკრეტული სტატიების აღმწერ ცალკეულ სტრიქონებად, ხოლო ყოველი სტრიქონი-დაყოფილია სტატიის დასახელებისა და ავტორის სახელის შემცველ სვეტებად. (სტატიებზე ჰიპერმინიშნებების შემცველი მესამე სვეტი არავითარ რეალურ ინფორმაციას არ შეიცავს და ამიტომ მას არ განვიხილავთ). ასე რომ, შეიძლება ითქვას, რომ ჩვენი ინფორმაცია სტრუქტურირებულია ცხრილის სახით.

უნდა აღინიშნოს, რომ ცხრილის სახით სტრუქტურირებული ინფორმაცია ვებ-გვერდებზე ძალიან ხშირად გვხვდება, ვინაიდან ის დამთვალიერებლების მიერ უკეთესად აღიქმება. მართლაც, ბევრად უფრო მოხერხებულია ცხრილის დათვალიერება (მითუმეტეს თუ ინფორმაცია დახარისხებული იქნება რომელიმე სვეტის მიხედვით), ვიდრე უწყვეტი ტექსტის ჯურღმულებში გარკვევა. გარდა ამისა, ცხრილები მეტად კომპაქტურნი არიან-ისინი ცოტა ადგილს იკავებენ ვებ-გვერდებზე, რაც ასევე მნიშვნელოვანია.

წინსწრებით უნდა ავღნიშნოთ, რომ სხვადასხვა პროგრამების მიერ ცხრილების დამუშავება გაცილებით მარტივად ხორციელდება. ცხადია, ამ შემთხვევაში საუბარია არა თვით ვებ-გვერდებზე, არამედ ამ პროგრამების მიერ დასამუშავებელ ფაილებში, ინფორმაციის შენახვის ხერხებზე. კლასიკური მაგალითის სახით, შეიძლება დავასახელოთ მონაცემთა რელაციური ბაზები, რომელთა შესახებ დაწვრილებით მე-6 თავში ვისაუბრებთ.

შენიშვნა: *ინფორმაცა აგრეთვე შეიძლება იყოს არასტრუქტურირებული. ტიპიური მაგალითია-ჩვენი საიტის*

მთავარი გვერდი, რომელიც ჩვეულებრივ ტექსტს შეიცავს. ინფორმაცია, რომელსაც ის ატარებს არ ექვემდებარება არავითარ სტრუქტურირებას.

მაშ ასე, ინფორმაციასთან დაკავშირებით ყველაფერი გავარკვიეთ, მაგრამ წარმოდგენა რაღა არის?

წარმოდგენა-ეს არის წესების ერთობლივობა, რომელთა შესაბამისადაც, ინფორმაცია მიეწოდება მის საბოლოო მომხმარებელს, ჩვენს შემთხვევაში-ვებ-საიტის დამთვალიერებელს. წარმოდგენა აღწერს, თუ როგორ ხდება ცხრილის ყოველი სვეტის ფორმირება, სად გამოიტანება იგი, როგორია მისი რიგითი ნომერი, რომელი სვეტის მიხედვით ხდება ცხრილის დახარისხება და ა.შ.. შეიძლება ითქვას, რომ წარმოდგენა ახორციელებს "ნედლი" ინფორმაციის გარდაქმნას საბოლოო პროდუქტად-იმად, რასაც მომხმარებელი დაინახავს თავის ეკრანზე.
HTML-კოდში, წარმოდგენასთან დაკავშირებულ ზოგიერთ რამეს, შეუიარაღებელი თვალითაც კარგად დავინახავთ. ჯერ-ერთი, ეს არის ჩვენი ვებ-გვერდის სახელწოდების სექცია, რომელიც ვებ-დამთვალიერებლისათვის აუცილებელ დამხმარე ინფორმაციას შეიცავს. მეორე, ეს არის თვით სტატიების ტექსტებზე გადამყვანი ჰიპერმინიშნებები. მესამე-რათქმა უნდა არის **CSS**-სტილების ცხრილები. მიუხედავად იმისა, რომ ისინი არ შედიან **HTML**-კოდის შემადგენლობაში-წარმოდგენების ქრესტომატიულ მაგალითებად შეგვიძლია განვიხილოთ.
ახლა, კიდევ ერთხელ გადავავლოთ თვალი **HTML**-კოდს, რათა გულდასმით განვაცალკევოთ ინფორმაცია წარმოდგენისაგან. ამასთან, გავიგოთ ერთი მეტად არასასიამოვნო, თუმცა ზოგიერთისთვის საინტერესო რამ...

სტატიკური ვებ-გვერდების ნაკლოვანებები

კერძოდ-ჩვენს ვებ-გვერდებზე ინფორმაცია ფაქტიურად განუყოფლად არის დაკავშირებული წარმოდგენასთან.

მართლაც, **HTML**-კოდი წარმოადგენს "ნედლი" ინფორმაციისა და ვებ-დამთვალიერებელში მისი წარმოდგენის წესების უცნაურ კომბინაციას. ასე მაგალითად, სტრიქონებისა და სვეტების თანმიმდევრობა მკაცრად არის განსაზღვრული თვით კოდში; თუ

ჩვენ გვსურს მისი შეცვლა, ვთქვათ პირველ ადგილზე სტატიის დასახელების განთავსება, მაშინ მთელი კოდის გადაკეთება მოგვიწევს, რაც დაკავშირებული იქნება ყოველი სტრიქონის პირველი და მეორე უჯრედების ურთიერთშენაცვლებასთან. ეს საკმაოდ რთული და შრომატევადი სამუშაოა.

შემდეგ, ვცადოთ სიაში ახალი სტატიის ჩამატება. თუ ჩვენ ახალ სტრიქონს მოვათავსებთ ცხრილის ბოლოში, ის ვებ-დამთვალიერებელშიც ასევე აისახება-ცხრილის ბოლოში. ამ შემთხვევაში ცხრილი უკვე აღარ იქნება დახარისხებული სტატიების სახელწოდებების მიხედვით. ჩვენ მოგვიწევს ახალი სტრიქონის აღმწერი **HTML**-კოდის ჩამატება ცხრილის ზუსტად განსაზღვრულ ადგილას, მეზობელი სტრიქონების ერთმანეთისაგან დაშორების გზით.

თუ ჩვენ გვსურს სტატიების წარმოდგენა არა ცხრილის, არამედ სიის სახით? თუ ჩვენ გვინდა დამატების თარიღის მიხედვით დახარისხებული სტატიების სიის გამოტანა? ფაქტიურად ჩვენ მოგვიწევს ვებ-გვერდის მთლიანად გადაკეთება. ეს კი საკმაოდ რთული საქმეა.

კიდევ ერთი. სტატიები და ფაილები ჩვენს საიტზე დაყოფილია კატეგორიებად. ყოველი ასეთი კატეგორიისათვის, ჩვენ მოგვიწია ცალკე ვებ-გვერდს შექმნა თავისი საკუთარი სიით. აღნიშნულის შედეგად, ჩვენი საიტი შედგება აბსოლუტურად ერთი და იგივე ტიპის მრავალი ვებ-გვერდისაგან, რომლებიც ერთმანეთისაგან მხოლოდ ინფორმაციით განსხვავდებიან.

მაშ ასე, ჩვენ გავარკვიეთ, რომ ჩვენს საიტზე ინფორმაციის წარმოდგენა გაერთიანებულია თვით ინფორმაციასთან და ისინი მჭიდროდ არიან ერთმანეთთან დაკავშირებულნი. ხშირად, სწორედ ამიტომ უწოდებენ **HTML**-ენაზე დაწერილ კლასიკურ ვებ-გვერდებს-**სტატუკურ ვებ-გვერდებს.**

იბადება კითხვა: არსებობს თუ არა ხერხი, რომელიც მოგვცემს საშუალებას ინფორმაცია გამოვყოთ წარმოდგენისაგან? სამწუხაროდ **HTML**-ენის საშუალებებით ამის გაკეთება არ გამოვა. თუმცა...

ინფორმაციის, წარმოდგენისაგან გამოყოფის პირველი წარმატებული მცდელობა-**CSS** სტილების ცხრილები იყო. ჩვენ უკვე ვიცით მე-2 თავიდან, რომ ვებ-გვერდებზე ტექსტების გასაფორმებლად, ადრე ფიზიკური ფორმატირების ტეგებს იყენებდნენ. ისინი "თავხედურად" ძვრებოდნენ პირდაპირ ინფორმაციაში და შეჰქონდათ მასში წარმოდგენის წესები, გარდა ამისა-ძალიან

ჩახლართულს ხდიდნენ **HTML**-კოდს. სტილების ცხრილებმა შესაძლებელი გახადეს წარმოდგენის წესების ნაწილის, თვით ინფორმაციის გარეთ გამოტანა -სათაურის სექციაში, ხანდახან კი ცალკეულ ფაილებშიც.

სამწუხაროდ, სტილების ცხრილები-პრობლემის მხოლოდ ნაწილობრივი გადაწყვეტაა. ისინი მხოლოდ ინფორმაციის გაფორმებას განსაზღვრავენ: ტექსტის შრიფტი და ფერი, განლაგება, აბზაცების დაშორების სიდიდეები და სხვა. მათი საშუალებით შეიძლება ტექსტის აბზაცის ან გამოსახულების გარშემო ჩარჩოების შექმნა, მისი ზომების განსაზღვრა და თუ ეს საჭირო იქნება, კურსორის ფორმის შეცვლაც აბზაცზე მისი განთავსებისას. (თუ როგორ კეთდება ყოველივე ეს, აღწერილია **CSS**-ის ინტერაქტიულ ცნობარში, რომელიც **DreamweaverMX**-ის კომპლექტაციაში შედის). მაგრამ ცხრილის მონაცემების დახარისხება და მისი სიად გარდაქმნა მათ არ შეუძლიათ.

სტილების ცხრილები იყო რადიკალური, თუმცა ცალმხრივი გადაწყვეტა. ისინი იქცნენ **HTML**-ენის თანდაყოლილი ნაკლოვანებების აღმომფხვრელ დანამატებად. ზემოთ აღწერილი პრობლემის სრულად გადაწყვეტისათვის, საჭირო იყო კიდევ ერთი რადიკალური გადაწყვეტა და ის ნაპოვნი იქნა!

სერვერული პროგრამები-ინფორმაციის, წარმოდგენებისაგან გამოყოფის რადიკალური ხერხი.

მოდით ვებ-დიზაინიდან ყურადღება პროგრამებზე გადავიტანოთ. ჩვეულებრივი პროგრამები, რომლებთანაც ჩვენ საქმე გვაქვს ყოველდღიურად: ტექსტური რედაქტორები, ელექტრონული ცხრილები, გრაფიკული პაკეტები და სხვა. რა დახმარების გაწევა შეუძლიათ მათ ჩვენთვის და რას გვკარნახობენ ისინი.

ავიღოთ იგივე **Microsoft Word**-დღეისათვის ერთ-ერთი ყველაზე პოპულარული ტექსტური რედაქტორი. მასში აკრეფილი დოკუმენტები ინახება ფაილებში .doc გაფართოებით. კონკრეტულად, თუ როგორ ინახება მასში მთელი ინფორმაცია-ტექსტი, გრაფიკა, ცნობები ფორმატირების შესახებ და ა.შ.-ჩვენთვის უცნობია. თუ ასეთ ფაილს გავხსნით ჩვეულებრივ ტექსტურ რედაქტორში, რომელიც **.txt** ფაილებით ოპერირებს (იგივე **NotePad**), ჩვენ დავინახავთ გაუგებარი სიმბოლოების ერთობლივობას.

თუ ჩვენ გავხსნით .doc ფაილს **Microsoft Word**-ში (ან ნებისმიერ სხვა პროგრამაში, რომლისთვისაც გასაგებია ეს ფორმატი), ეკრანზე

დავინახავთ დოკუმენტის დაფორმატებულ ტექსტს, რომელიც შესაძლოა შეიცავდეს გრაფიკასაც და ცხრილებსაც. ესეიგი **Word**-მა თვითონ "იცის," როგორ დააფორმატოს და წარმოადგინოს დოკუმენტი, რომელიც ფაილში ინახება.

მაშინ რა გამოდის? გამოდის, რომ **Microsoft Word**-ის .doc გაფართოების მქონე ფაილი-ეს ინფორმაციაა. ხოლო თვით **Microsoft Word**-არის წარმოდგენა. ჩვენ არ ვიცით, კონკრეტულად, თუ როგორ ინახება ინფორმაცია **.doc** ფაილებში-მას წაკითხვისათვის და ჩასწორებისათვის მისაღები ფორმით **Microsoft Word**-ი წარმოგვიდგენს.

აი თურმე სად ყოფილა პრობლემის გადაწყვეტა! გადაწყვეტა ყოფილა ისეთი პროგრამის არსებობა, რომელიც მუშაობს იმავე კომპიუტერში, რაც ვებ-სერვერი, კითხულობს ფაილებიდან ინფორამაციას და მის მიმართ იყენებს წარმოდგენის წესებს, რომელთა ზემოქმედების შედეგად საიტის დამთვალიერებელს, აღნიშნული ინფორმაცია წარედგინება მისაღები ფორმით. თუ მომხმარებელს სურს, რაიმე მოქმედების განხორციელება ინფორმაციაზე, ის ამ პროგრამას უგზავნის შესაბამის მოთხოვნას, რომლის პასუხადაც პროგრამა იყენებს ამავე ინფორმაციის მიმართ, წარმოდგენის სხვა წესებს (მაგალითად ცვლის სტრიქონების დახარისხების თანმიმდევრობას, ან გამოყავს იგი არა ცხრილის, არამედ სიის სახით).

ახლა დავუშვათ, რომ ჩვენ გვინდა დამთვალიერებელში ამ ინფორმაციის წარმოდგენის ხერხის შეცვლა. ამისათვის ჩვენ უნდა გადმოვიწეროთ ეს პროგრამა, მასში ახალი წესების დამატებით, და მოვათავსოთ იგი სერვერულ კომპიუტერზე. ამით მორჩება-აღარ მოგვიწევს თვით ინფორმაციის გადაკეთება.

აქ ორი საინტერესო მომენტი ჩნდება, რომლებზეც ჩვენ აუცილებლად უნდა გავამახვილოთ ყურადღება. უკავშირდებიან ეს მომენტები იმას, რომ მომხმარებელი, რომელმაც ეს ინფორმაცია უნდა მიიღოს, მუშაობს მხოლოდ ვებ-დამთვალიერებლის საშუალებით. ჩვენ ხომ ვებ-საიტს ვაკეთებთ ბოლოს და ბოლოს!

პირველი მომენტი. სერვერული პროგრამა, რომელიც ინფორმაციას წარმოდგენის წესების შესაბამისად ამუშავებს, უნდა მუშაობდეს ვებ-სერვერთან ერთად. ყველა ვებ-დამთვალიერებელს მუშაობა შეუძლია მხოლოდ ვებ-სერვერთან; სხვა პროგრამებთან მუშაობა მათ არ "იციან". ეს იმას ნიშნავს, რომ ვებ-სერვერი კლიენტისაგან იღებს მოთხოვნას, გადაუგზავნის მას დასამუშავებლად შესაბამის

პროგრამას, მიიღებს მისგან ინფორმაციის დამუშავების შედეგებს და გაუგზავნის მათ კლიენტს.
მეორე, პროგრამის მუშაობის შედეგები წარმოადგენენ ჩვეულებრივ ვებ-გვერდებს, რომლებიც მომხმარებლის ვებ-დამთვალიერებელში უნდა აისახონ. ვებ-გვერდების (და ჩვეულებრივი ტექსტური ფაილების) გარდა, ვებ-დამთვალიერებელი არ ცნობს არანაირ სხვა დოკუმენტებს. ასე, რომ ამას უნდა შევეგუოთ.
ყოველივე ზემოაღნიშნულიდან გამომდინარე, მოდით ჩამოვწეროთ, სერვერულ კომპიუტერზე მომუშავე, განსაზღვრული წარმოდგენის წესების გამომყენებელი პროგრამისაგან, ცალკეულ ფაილებში შენახული ინფორმაციის მიღების თანმიმდევრობა.

1.მომხმარებელი ვებ-დამთვალიერებლის საშუალებით მოითხოვს რაიმე ინფორმაციას;
2.ვებ-დამთვალიერებელი ამყარებს კავშირს ვებ-სერვერთან და უგზავნის მას კლიენტის მოთხოვნას;
3.ვებ-სერვერი იღებს მოთხოვნას და ახდენს მის გაშიფვრას;
4.ვებ-სერვერი გაუშვებს საჭირო პროგრამას და საჭიროების შემთხვევაში გადასცემს მას მოთხოვნის დამატებით პარამეტრებს;
5.პროგრამა ამოკრიბავს ფაილში ან ფაილებში შენახულ საჭირო ინფორმაციას, მის მიმართ გამოიყენებს განსაზღვრულ წარმოდგენებს და გარდაქმნის ვებ-გვერდად, რომელსაც უგზავნის ვებ-სერვერს.
6.ვებ-სერვერი, პროგრამის მიერ ფორმირებულ ვებ-გვერდს უგზავნის ვებ-დამთვალიერებელს;
7.ვებ-დამთვალიერებელი იღებს ვებ-გვერდს და გამოაქვს იგი ეკრანზე, რის შემდეგაც წყვეტს კავშირს სერვერთან.

დაგვრჩა რამოდენიმე ახალი ტერმინის შემოღება, რომლებიც დაგვეხმარებიან იმაში, რომ შემდგომში არ დავიბნეთ.
პროგრამას, რომელიც ვებ-სერვერთან ერთად მუშაობს და ახორციელებს ინფორმაციის დამუშავებას პროგრამისტის მიერ განსაზღვრული წარმოდგენის წესების შესაბამისად, ვუწოდოთ **სერვერული პროგრამა.** ხოლო, მის მიერ ფორმირებულ ვებ-გვერდს, რომელიც უკვე გადამუშავებულ ინფორმაციას შეიცავს, ვუწოდოთ **დინამიური ვებ-გვერდი.**

ჩვენთვის უკვე ნაცნობ "კლიენტ-სერვერულ" არქიტექტურას დაემატა შუალედური რგოლი-სერვერული პროგრამა. იგი სამრგოლიანი გახდა - "კლიენტი-სერვერული პროგრამა-მონაცემები".

უნდა აღინიშნოს, რომ სერვერული პროგრამები დაგვეხმარებიან კიდევ ერთი პრობლემის გადაწყვეტაში, რომელიც ვერ წყდებოდა **HTML**-ის საშუალებებით, კერძოდ, ვებ-გვერდებიდან საიტების მომხმარებლების მიერ გამოგზავნილი ინფორმაციის გადამუშავების ორგანიზებაში. ვებ-გვერდებზე საკმაოდ ხშირად გვხვდებიან ეგრეთ წოდებული **ფორმები**-მომხმარებლის მიერ რაიმე მონაცემების (სახელი და პაროლი, ჩვეულებრივი და ელექტრონული ფოსტის მისამართები, ანკეტები და სხვა) შეტანისათვის განკუთვნილი მართვის ელემენტების ერთობლივობა. ასე, რომ მათი საშუალებით გაგზავნილი მონაცემების დამუშავება შესაძლებელი ხდება სწორედ სერვერული პროგრამების გამოყენებით. თუ როგორ უნდა დაიწეროს ასეთი პროგრამები და როგორ შევქმნათ ფორმები, ჩვენ გავიგებთ მე-9 თავში.

ჯერ-ჯერობით გავაგრძელოთ საუბარი სერვერულ პროგრამებზე და მათ ნაირსახეობებზე. გავიგოთ, თუ რის გაკეთებას ვაპირებთ მთელი ამ წიგნის მანძილზე.

სერვერული პროგრამების შექმნის ტექნოლოგიები

ცხადია, რომ ჩვეულებრივი, სტატიკური ვებ-გვერდების და სერვერული პროგრამების შექმნა, ერთმანეთისაგან სრულიად განსხვავებულად ხება. უფრო მეტიც, ყოველი მათგანისათვის სრულიად განსხვავებული ენები გამოიყენება. თვით მათი შექმნის პრინციპებიც მნიშვნელოვნად განსხვავდებიან ერთმანეთისაგან. სწორედ ამ პრინციპებზე, ენებზე და განსხვავებებზე გვექნება ახლა საუბარი.

აქტიური სერვერული ვებ-გვერდები

როდესაც საუბარი იყო სერვერულ პროგრამებზე და მათი მუშაობის პრინციპებზე, გვეჩვენებოდა, რომ დროდადრო ისინი ძალიან გვახსენებდნენ ვებ-გვერდებს. მართლაც, მუშაობენ ისინი სერვერთან ერთად, რომელიც კლიენტების მოთხოვნების შესაბამისად უშვებს შესრულებაზე ამა თუ იმ პროგრამას, გადასცემს მათ გარკვეულ დამატებით პარამეტრებს და მათგან იღებს დამუშავების შედეგებს.

სერვერული პროგრამის ინტერნეტ-მისამართიც კი იწერება ისევე, როგორც ვებ-გვერდის მისამართი, მცირეოდენი გამონაკლისით:

http://www.somesite.ge/progs/prog.exe

აქ ჩვენ მივმართეთ სერვერულ პროგრამას prog1.exe, რომლის ფაილიც ინახება **http://www.somesite.ge** სერვერის საქაღალდეში progs. პასუხად ჩვენ მივიღებთ ამ პროგრამის მიერ, აღნიშნულ სერვერზე შენახული მონაცემების ბაზაზე ფორმირებულ ვებ-გვერდს. შედეგი თითქმის ისეთივეა, როგორსაც მივიღებდით, თუ მივმართავდით ჩვეულებრივ, სტატიკურ ვებ-გვერდს.

რამდენადაც სერვერული პროგრამა, თავისი მუშაობის შედეგის სახით, ქმნის ჩვეულებრივ ვებ-გვერდს, ჭკვიანმა ადამიანებმა ასეთი რამე მოიფიქრეს. რატომ არ შეიძლება, რომ ჩვეულებრივ სტატიკურ ვებ-გვერდში ჩავსვათ პროგრამული კოდის პატარ-პატარა ფრაგმენტები (ეგრეთ წოდებული **სცენარები**), რომელიმე მათთვის გასაგებ პროგრამირების ენაზე? ასეთი ფრაგმენტის დანახვისას, ვებ-სერვერი დაამუშავებს მას და მუშაობის შედეგს-თუნდაც ეს იყოს სტატიების სიის შემცველი ცხრილი-პირდაპირ ჩასვამს ვებ-გვერდის იმავე ადგილას. ცხადია სერვერი ვებ-გვერდიდან წინასწარ ამოჭრის კოდის ამ ფრაგმენტს, ვინაიდან ვებ-დამთვალიერებელს ის აღარაფრისთვის აღარ დასჭირდება.

ასეთი **აქტიური სერვერული ვებ-გვერდების** შექმნის იდეა, რომლებშიც **HTML**-კოდთან ერთად, მონაცემების დამუშავების პროგრამული კოდის ფრაგმენტებიც იქნებოდა მოთავსებული, ეტყობა საკმაო ხნის მანძილზე არსებობდა. მხოლოდ ამით შეიძლება აიხსნას ის გარემოება, რომ პრაქტიკულად ერთდროულად გაჩნდა ასეთი გვერდების და შესაბამისი პროგრამული უზრუნველყოფის შექმნის რამოდენიმე ტექნოლოგია. ამ ტექნოლოგიების(ეგრეთ წოდებული **გადამამუშავებლების**) დანიშნულება მდგომარეობდა იმაში, რომ "ესწავლებინა" ვებ-სერვერებისათვის, გვერდებზე გამოყენებული პროგრამული კოდის აღქმა. ყველაზე მეტი პოპულაროზა დაიმსახურა ხუთმა ტექნოლოგიამ, რომელთაც ჩვენ ახლა განვიხილავთ.

აქტიური სერვერული გვერდების შექმნის ყველაზე პირველი ტექნოლოგია, რომელსაც ჩვენ აქ განვიხილავთ-ეს არის **PHP**(**Personal Home Page**, პერსონალური საშინაო გვერდი). იგი შექმნილი იყო ერთ-ერთი ნორვეგიული სტუდენტის მიერ, ჯერ კიდევ გასული

საუკუნის 90-იანი წლების შუახანებში და დღეისათვის, როგორც ჩანს ყველაზე პოპულარული გახდა. ძირითადში მისი პოპულარობა განპირობებულია იმით, რომ **PHP**-ს გვერდების დამუშავების პროგრამული უზრუნველყოფა ხელმისაწვდომია უფასოდ და ღიაა მისი კოდი ცვლილებებისა და დამატებებისათვის. გარდა ამისა გადამამუშავებელი უზრუნველყოფს სამ ათეულამდე პოპულარული და ნაკლებადცნობილი ვებ-სერვერების მხარდაჭერას, რაც იმას ნიშნავს, რომ მისი გამოყენება შესაძლებელია პრაქტიკულად ყველგან.

ვებ-გვერდებზე **PHP**-ს სცენარების დასაწერად გამოიყენება პროგრამირების განსაკუთრებული ენა, რომელიც პოპულარულ ენა **C**-ზე არის დაფუძნებული. ამ ენას ასეც ეწოდება-**PHP.** ის ძალზე მარტივია შესასწავლად და საკმარისად განვითარებულია რთული პროგრამების შესაქმნელად. რაც შეეხება თვით **PHP**-კოდების შემცველ ვებ-გვერდებს, მათი შენახვა უნდა მოხდეს **php** გაფართოების მქონე ფაილებში-ეს აუცილებელია, სხვანაირად გადამამუშავებელი მათ ჩათვლის სტატიკურ ვებ-გვერდებად.

დღეისათვის **PHP** გვერდების შემცველი საიტები-ჩვეულებრივი მოვლენაა ქსელში. **PHP**-გადამამუშავებლები დაყენებულია უფასო ვებ-სერვერების უმრავლესობაზე.

ძალიან ადვილია მათი დაყენება ლოკალურ ვებ-სერვერზეც; თუ როგორ კეთდება ეს, აღწერილია ამ წიგნის **მე-3** დანართში.

დაახლოებით იმავე პერიოდში (90-იანი წლების შუახანი) ფირმა **Microsoft**-ის მიერ შემუშავებული იქნა სერვერული გვერდების ტექნოლოგია **ASP** (**Active Server Pages**, აქტიური სერვერული გვერდები). თავდაპირველად ის გამოიყენებოდა ვებ-სერვერში **Microsoft Internet Information Server**, მაგრამ შემდგომში გაჩნდნენ მესამე ფირმების გადაწყვეტილებები, რომლებიც უზრუნველყოფდნენ გვერდების **ASP**-გადამამუშავებლების დამატებას სხვა ვებ-სერვერებშიც. **ASP**-გვერდებზე სცენარების დაწერა ხდება **VBScript** ენის საშუალებით, თუმცა შესაძლებელია აგრეთვე **IavaScript**-ის და ზოგიერთი სხვა ენის გამოყენებაც. თვითონ გვერდებს უნდა ჰქონდეთ გაფართოება asp.

ASP.NET-ეს არის **ASP** ტექნოლოგიის გაფართოება, რომელიც ასევე **Microsoft**-ის მიერ არის შემუშავებული 21-საუკუნის დასაწყისში. **ASP.NET**-ს უფრო მაღალი წარმადობა და საიმედოობა გააჩნია ვიდრე **ASP**-ს და ამავე დროს ბევრი დამატებითი შესაძლებლობებიც

აქვს. დღეისათვს **ASP.NET**-ის მხარდაჭერა მხოლოდ **Microsoft Internet Information Server**-ს შეუძლია.

JSP (**Java Server Pages, JavaScript**-ზე დაწერილი სერვერული გვერდები)- ეს თავისებური "ადექვატური პასუხია" **ASP**-ზე, რომელიც ფირმა **Netscape**-ის მიერ არის შემუშავებული ასევე მე-20 საუკუნის 90-წლების შუახანებში თავისი საკუთარი ვებ-სერვერისათვის-**Netscape Web Server**. **JSP** პრაქტიკულად არ განსხვავდება **ASP**-საგან იმ ერთად-ერთი გამონაკლისის გარდა, რომ სცენარების დასაწერად გამოიყენება პროგრამირების ენა **JavaScript**. დღეისათვის **JSP**-ტექნოლოგია საკმაოდ პოპულარულია; ვებ-სერვერების დიდი რაოდენობისათვის, მათ შორის უფასოებისთვისაც, არსებობენ JSP-გადამამუშავებლები.

უკანასკნელი ტექნოლოგია, რომელიც აქ მოხსენიების ღირსია-არის ფირმა Macromedia-ს მიერ იმავე "ოქრის ხანაში-90-იანი წლების შუა პერიოდში)" შემუშავებული **ColdFusion**. ამ ტექნოლოგიას ზემოთ ჩამოთვლილებთან შედარებით ძირეული განსხვავებები გააჩნია-სცენარების ნაცვლად მასში გამოიყენებიან განსაკუთრებული, HTML-ტეგების მსგავსი ტეგები, რომელთა დანიშნულება გვერდების გადამუშავებაში მდგომარეობს. ასეთი ტეგის დანახვისას, გადამამუშავებელი ახორციელებს მონაცემების შერჩევას, დამუშავებას და შედეგებს ათავსებს HTML-კოდის იმ ადგილას, სადაც ეს კოდი შეხვდა. ColdFusion ტექნოლოგია მხარდაჭერილია მრავალი ვებ-სერვერის მიერ, მონაცემთა დიდ მასივებზე მანიპულირების საშუალებას იძლევა, მაგრამ ჯერ-ჯერობით ვერ ჰპოვა ფართო გავრცელება (ეტყობა გადამამუშავებლის მაღალი ფასის გამო).

არსებობს კიდევ რამოდენიმე სხვა, ნაკლებად პოპულარული აქტიური სერვერული გვერდების ტექნოლოგია. როგორც წესი, მათი მხარდაჭერა ხდება მხოლოდ რომელიმე ერთი სერვერის მიერ და ამიტომ ვერ მოხდა მათი გავრცელება. ჩვენ მათ არ განვიხილავთ.

სერვერული პროგრამირების სხვა ტექნოლოგიები

ბოლოს მოკლედ შევეხოთ სერვერული პროგრამების შექმნის დღეისათვის გამოყენებად სხვა ტექნოლოგიებსაც. მათი რაოდენობა სულ სამია.

ტექნოლოგია CGI (Common Geteway Interface, ურთიერთ გაცვლის საერთო ინტერფეისი)-ყველაზე ძველი, მაგრამ ჯერ კიდევ

დაუბერებელი ტექნოლოგიაა. **CGI პროგრამები**-ეს ჩვეულებრივი შესრულებადი exe ფაილები ან Perl, Python და სხვა ენებზე დაწერილი პროგრამებია. კლიენტის მოთხოვნის საპასუხოდ ვებ-სერვერი შესრულებაზე უშვებს ასეთი პროგრამის ასლს, გადასცემს მას მონაცემებს, იღებს შედეგებს და ბოლოს აჩერებს პროგრამის შესრულებას. **CGI** პროგრამების შექმნა და გაწყობა ძალიან მარტივია, მაგრამ თუ ვებ-სერვერი ძალიან ბევრ კლიენტურ მოთხოვნას მიიღებს და შესაბამისად ბევრ ასეთ პროგრამას გაუშვებს, მაშინ შესაძლოა სერვერული კომპიუტერის რესურსები არ აღმოჩნდეს საკმარისი და ის "ჩამოეკიდება".

ტექნოლოგია ISAPI (**Internet Server Application Programming Interfase**, ინტერნეტ-სერვერული გამოყენებითი პროგრამების დაპროგრამების ინტერფეისი), რომელიც ფირმა Microsoft-ის მიერ არის შემუშავებული, თავისუფალია ამ სახის ნაკლოვანებებისაგან. ISAPI პროგრამა-ეს არის დინამიური ბიბლიოთეკა Windows DLL, რომელიც მუდმივად ვებ-სერვერთან ერთად მუშაობს და ყველა შემოსულ კლიენტურ მოთხოვნებს ერთდროულად ამუშავებს. ასეთი პროგრამები ნაკლებ კომპიუტერულ რესურსებს მოითხოვენ, მაგრამ ძალზე რთულია მათი შექმნა და გაწყობა. არსებობს აგრეთვე ISAPI-ის ანალოგიური ტექნოლოგია **NSAPI** (**Netscape Server Application Programming Interface, Netscape** სერვერის გამოყენებითი პროგრამების პროგრამირების ინტერფეისი), რომელიც ფირმა Netscape-ის მიერ არის შემუშავებული.

ISAPI და **NSAPI** პროგრამებთან ახლოს არიან ეგრეთ წოდებული **ვებ-სერვერების გაფართოებები.** ისინი გაიშვებიან ცალკეულ ეგზემპლიარებად და კლიენტების მოთხოვნების შესასრულებლად მუდმივად ვებ-სერვერებთან ერთად მუშაობენ. ერთად-ერთი განსხვავება: **ISAPI** და **NSAPI** ტექნოლოგიები სტანდარტიზებულია და მათ შეუძლიათ ბევრ სხვადასხვა სერვერებთან მუშაობა, ხოლო გაფართოებები იწერებიან რომელიმე ერთი სერვერისათვის. დღეისათვის ყველაზე პოპულარულია ვებ-სერვერ **Apache**-ის გაფართოებები და ეს არც არის გასაკვირი, ვინაიდან ის უპოპულარესი ვებ-სერვერია.

ისღა დაგვრჩა სათქმელი, რომ ეს ტექნოლოგიები გამოიყენებიან თვით აქტიური სერვერული გვერდების გადასამუშავებელი პროგრამების დასაწერად. ასე მაგალითად, **PHP**-გადამამუშავებელი

არსებობს როგორც **ISAPI** პროგრამა, როგორც ვებ-დამთვალიერებლის გაფართოება და როგორც **CGI** გამოყენებითი პროგრამა. ასე, რომ შესაძლებელია მისი "მორგება" პრაქტიკულად ნებისმიერ მეტნაკლებად სერიოზულ ვებ-სერვერზე.

Web -საიტზე აქტიური სერვერული გვერდების გამოყენება.

დიახ სწორედ ასე იქნება! დროა ვიგრძნოთ სერვერული დაპროგრამების მთელი სიმძლავრე.

კატეგორიების სიებს, **სტატიებს** და **ფაილებს**, რომელთა გამოტანასაც ვაპირებთ საიტზე, ჩვენ შევინახავთ მონაცემების ფაილში. ეს ფაილი შენახული იქნება ვებ-სერვერზე, სადმე ძირეული საქაღალდეს გარეთ, რათა ვერცერთმა დამთვალიერებელმა ვერ შეძლოს მისი გადმოტვირთვა და დათვალიერება. ხოლო მონაცემების ამოსაკრეფად ჩვენ გამოვიყენებთ ორ სერვერულ გვერდს. ორად ორ სერვერულ გვერდს!

-**Categories.php**-მომხმარებლისათვის გამოიყვანს კატეგორიების სიას. იმისდა მიხედვით, მთავარი გვერდის თუ რომელ ჰიპერმინიშნებაზე დააწკაპუნა მომხმარებელმა, ეს იქნება სტატიების ან ფაილების კატეგორიების სია.

-**Items.php**-უჩვენებს მომხმარებელს, მის მიერ არჩეულ კატეგორიას მიკუთვნებული სტატიების ან ფაილების სიას.

მთავარი გვერდი-**default.htm**-ძველი დაგვრჩება. მასზე არსებული ინფორმაცია ხშირად არ იცვლება და ამიტომ რაიმე სტრუქტურირებას არ საჭიროებს. (ერთადერთი-მერე ჩვენ მოგვიწევს ინტერნეტ-მისამართების შეცვლა ჰიპერმინიშნებებში ***ფაილები*** და ***სტატიები***, მაგრამ ეს წუთიერი საქმეა).

ცხადია, სერვერული გვერდების შესაქმნელად ჩვენ ავირჩევთ **PHP** ტექნოლოგიას. ის ფართოდ არის გავრცელებული, მხარდაჭერილია ძალიან ბევრი უფასო ვებ-სერვერების მიერ, გამორჩეულად მარტივად ყენდება და გაეწყობა.

როდესაც ჩვენ დავიწყებთ ჩვენი ახალი საიტის ტესტირებას, ჩვენს კომპიუტერზე დავაყენებთ **PHP** გვერდების გადამამუშავებელს და მივუერთებთ მას ჩვენს ლოკალურ ვებ-სერვერს. ჩვენ მშვენიერ საიტს მივიღებთ.

რა იქნება ამის მერე?

მაშ ასე, ჩვენ გადავდგით პირველი ნაბიჯი სერვერული ინტერნეტ-დაპროგრამების სამყაროში. მოკლე და მორიდებული, მაგრამ მაინც ნაბიჯი.

ჩვენ სულ ვსაუბრობდით ინფორმაციის დამუშავების, წარმოდგენის წესების გამოყენების, სერვერული პროგრამების და მათი შექმნის ტექნოლოგიების შესახებ, მაგრამ ერთი სიტყვითაც არ გვიხსენებია, სად და როგორ ინახება თვით ინფორმაცია, რომლის დამუშავებასაც ჩვენ ვაპირებთ. დროა შევავსოთ აღნიშნული ხარვეზი ჩვენს განათლებაში! მომდევნო თავი სწორედ ამას მიეძღვნება.

თავი 6.

მონაცემთა ბაზები

მანამ, სანამ დავიწყებდეთ ინფორმაციის დამუშავებას, ის სადმე უნდა მოვიძიოთ და როგორმე უნდა შევინახოთ. ინფორმაციის მოძიებისა და შეგროვების საკირხებს ჩვენ აქ არ განვიხილავთ. ჯობია ვისაუბროთ იმაზე, თუ როგორ შევინახოთ შეგროვებული ინფორმაცია, რომ ის არსად დაიკარგოს და ყოველთვის ხელქვეშ გვქონდეს.

თუ ვისაუბრებთ კომპიუტერებთან მიმართებაში, მაშინ მათ დისკოებზე ინფორმაცია ინახება ფაილებში. ეს ყველასათვის ცნობილია, ვისაც ერთხელ მაინც შეუნახავს **Microsoft Word**-ში აკრეფილი დოკუმენტი. ფაილებს შეუძლიათ პროგრამული კოდების, კონფიგურაციების შესახებ მონაცემების, ტექსტების, გრაფიკების, ბგერების, ვიდეო რგოლების, ცხრილების და სხვათა შენახვა. ხოლო მთელი ამ სიმდიდრის გადამუშავებას ახორციელებენ სპეციალური პროგრამები, რომლებმაც იციან, თუ რა უყონ ამ იმფორმაციას.

რაც შეეხება სტრუქტურირებულ მონაცემებს, რომელთა შესახებაც ჩვენ მე-5 თავში გვქონდა საუბარი, მათი შენახვისათვის იყენებენ სრულიად განსხვავებული და ძალზე შესამჩნევი ტიპის ფაილებს. ასეთ ფაილებს **მონაცემთა ბაზებს** უწოდებენ.

შესავალი მონაცემთა რელაციურ ბაზებში

მაშ ასე, **მონაცემტა ბაზა**-ეს არის ფაილი ან ფაილების ერთობლივობა, რომლებშიც ინახავენ გარკვეული წესით სტრუქტურირებულ ინფორმაციას. ასეთი ინფორმაციის დასამუშავებლად იყენებენ განსაკუთრებულ პროგრამებს, რომელთაც **მონაცემთა ბაზების მართვის სისტემებს** (მბმს) უწოდებენ. ასეთი მბმს-ის მაგალითია-უპოპულარესი პროგრამა **Microsoft Access**, რომელიც მუშაობს .mdb ტიპის მონაცემებთან.

მათში მოთავსებული ინფორმაციის სტრუქტურირების წესების მიხედვით, მონაცემთა ბაზები იყოფიან რამოდენიმე სახეობად. კომპიუტერულ ინდუსტრიაში ყველაზე ფართო გავრცელება ჰპოვეს ეგრეთ წოდებულმა **რელაციურმა მონაცემთა ბაზებმა**, რომელთა შესახებაც ახლა ვიწყებთ საუბარს.

რა არის რელაციური მონაცემთა ბაზები.

რელაციური მონაცემთა ბაზები გამოიყენებიან ცხრილების სახით სტრუქტურირებული და ერთმანეთთან დაკავშირებული ინფორმაციის შესანახად. თუ უფრო მოკლედ და მარტივად ვიტყვით, მონაცემთა რელაციური ბაზა შეიცავს ერთმანეთთან დაკავშირებული ცხრილების ერთობლივობას. სწორედ მანაცემთა ბაზების ეს სახეობაა დღეისათვის ყველაზე პოპულარული.

საქმე იმაშია, რომ კომპიუტერების საშუალებით დასამუშავებელი ინფორმაციის ძალიან დიდი კლასი, შეიძლება წარმოდგენილი იქნას ცხრილების ანაკრების სახით. ესენია სხვადასხვა სიები, ჟურნალები, კატალოგები, საბუღალტრო წიგნები, ცნობარები და მრავალი სხვა; ფაქტიურად სწორედ ესენი წარმოადგენენ ცხრილებს. მსგავსი ინფორმაცია თავისთავად მოითხოვს, რომ ის იყოს წარმოდგენილი მონაცემთა ბაზის სახით. გარდა ამასა, მონაცემთა რელაციური ბაზები საშუალებას იძლევიან მკაცრად იქნას განსაზღვრული ინფორმაციის სისწორის და მთლინობის წესები, რაც ასევე არ არის ნაკლებად მნიშვნელოვანი. ამასთან ძალიან სწრაფად ხდება მათი დამუშავება შესაბამისი რელაციური მბმს-ით (ამის შესახებ ჩვენ ვსაუბრობდით მე-5 თავში).

შენიშვნა: *არსებობენ არარელაციური ბაზებიც, რომლებშიც ინფორმაციის სტრუქტურირება ხდება სხვ წესებით: ქსელური,*

ობიექტური და სხვა. მათი გამოყენება იშვიათად ხდება და მხოლოდ სპეციალურ შემთხვევებში, ამიტომ ჩვენ მათ არ განვიხილავთ

რელაციური მბმს-ების მაგალითებია: უკვე ნახსენები **Microsoft Access, Corel Paradox, Borland dBase, Microsoft FoxPro.** ყველა მათგნი **სამომხმარებლო** ან **სამაგიდე** მბმს-ია და ისინი განკუთვნილი არიან ჩვეულებრივი მომხმარებლებისათვის. ისინი თავიანთ ფაილებს ინახავენ მომხმარებლების კომპიუტერების ლოკალურ დისკოებზე ან ფაილური სერვერების დისკოებზე და პირდაპირ ამ ფაილებთან მუშაობენ.

მონაცემთა დიდი მასივების დასამუშავებლად, რომელთაც მომხმარებლების დიდი რაოდენობა ერთდროულად უერთდება, უფრო მძლავრი პროგრამები გამოიყენება: **Borland InterBase, MySQL**, **Microsoft SQL Server**, **Informix, Sybaze, Oracle**. ისინი სხვა პრინციპით ფუნქციონერებენ, კერძოდ, თუ როგორ-ჩვენ მოგვიანებით გავიგებთ. ყველა მათგანი მუშაობს ფაილების თავის საკუთარ ფორმატთან, თუმცა სამაგიდო მბმს-ებს, როგორც წესი, შეუძლიათ გახსნან "სხვისი" ფაილებიც. ერთ მონაცემთა ბაზას შეუძლია დაიკავოს როგორც ერთი დიდი ფაილი (**Microsoft Access, Borland InterBase**) ასევე მრავალი უფრო მცირე ფაილები (**Corel Paradox, Borland dBase, Oracle**); უკანასკნელ შემთხვევაში ეს ფაილები მოთავსებულები უნდა იყვნენ ერთ საქაღალდეში).
ახლა მოდით უფრო ახლოს გავეცნოთ მონაცემთა რელაციურ ბაზებს. ჩვენ შემდგომში სწორედ მათთან მოგვიწევს მუშაობა.

მანაცემთა რელაციური ბაზების

შემადგენელი ნაწილები.

რელაციური მონაცემთა ბაზა შედგება სამი ძირითადი ნაწილისაგან, რომელთაც ჩვენ დაწვრილებით განვიხილავთ ამ თავის მომდევნო ნაწილებში.

ცხრილები, ველები, ჩანაწერები

ცხრილი-ეს არის სტრუქტურირებული მონაცემების ერთობლივობა. ცხრილის მაგალითი ნაჩვენებია ნახ.6.1-ზე.

date	author	name
12.08.2011	მანჯგალაძე ე.	დაპროგრამების საკითხები
24.09.2011	თარგამაძე მ.	მონაცემთა ბაზების საფუძვლები
30.10.2011	აფრიდონიძე გ.	ოპერაციული სისტემების შესახებ
10.07.2011	ჯაფარიძე ზ.	კომპიუტერის საიმედოობა

ნახ.6.1. ცხრილი-სტატიების სია

ეს ცხრილი შეიცავს სამი სვეტისაგან შემდგარ სტატიების სიას: სიაში ***სტატიის ჩამატების თარიღი, სტატიის ავტორი*** და ***სტატიის დასახელება.*** არ არის საჭირო რაიმე აბსტრაქტული მონაცემების მოფიქრება, თუ ჩვენ უკვე გაგვაჩნია მზა მონაცემები.

ცხრილი მონაცემთა ბაზის ფაილში პრაქტიკულად იმავე სახით ჩაიწერება: უჯრედები შეადგენენ სტრიქონს, ხოლო სტრიქონები-ცხრილს. ამ ფაილის გახსნისთანავე მბმს გაარკვევს, თუ რასთან აქვს საქმე და ცხრილს გამოიტანს ეკრანზე მასთან მუშაობისათვის მისაღები ფორმით.

მონაცემთა ბაზაში შენახულ ყოველ ცხრილს უნდა ჰქონდეს ამ ბაზის ფარგლებში უნიკალური სახელი. ეს იმისთვის არის საჭირო, რომ მბმს-ამ (და ჩვენც) შევძლოთ ამ ცხრილის მოძებნა.

მოდით ახლა, შემოვიღოთ რამოდენიმე ახალი ტერმინი, რომლებითაც ოპერირებენ მონაცემთა ბაზებთან მომუშავე მომხმარებლები და პროგრამისტები.

დასაწყისისათვის ცხრილის ცალკეულ, რეალური მონაცემების შემცველ სტრიქონს, ვუწოდოთ **ჩანაწერი**. (ნახ.6.1.-ზე მუქი ფერით გამოყოფილი სტრიქონი ვერ იქნება ჩანაწერი, ვინაიდან შეიცავს არა რომელიმე სტატიის აღმწერ რეალურ მონაცემებს, არამედ-სვეტების დასახელებებს). სიაში შეტანილ ყოველ სტატიას დაეთმობა ერთი ჩანაწერი.

შემდეგ, ცალკეული სტრიქონის-ჩანაწერის ყოველ უჯრას ვუწოდოთ **ველი**. ყოველ ველს უნდა ჰქონდეს ცხრილის ფარგლებში უნიკალური დასახელება; ცხრილის დასახელების სტრიქონში, სწორედ ველების დასახელებებია მოცემული. შეიძლება ითქვას, რომ ველი-ეს არის ჩანაწერის შემადგენელი მონაცემების ერთი ულუფა.

თვით ველებში მოთავსებულ მონაცემებს უწოდებენ მათ **მნიშვნელობებს**.

ჩანაწერები ველებად იყოფიან იმისათვის, რომ **მბმს**-ებს გაუადვილდეთ ცხრილების დამუშავება. ასე მაგალითად, თუ ჩვენ ცალკე **date** სახელწოდების მქონე ველად გამოვყოფთ (რათქმა უნდა ამ ველს შეიძლება სხვანაირადაც ეწოდოს), სტატიის დამატების თარიღს, ჩვენ მოგვეცემა ჩანაწერების, ამ ველის მიხედვით დახარისხების შესაძლებლობა. მბმს ჩვენი მოთხოვნის პასუხად კითხულობს ყველა სტრიქონის **date** ველის მნიშვნელობებს და ცვლის მათ თანმიმდევრულობას, რათა დაალაგოს ისინი თარიღის ზრდადობის ან კლიბადობის მიხედვით. ასევე შეგვიძლია დასათვალიერებლად შევარჩიოთ მხოლოდ ის ჩანაწერები, როლებშიც **date** ველის მნიშვნელობები, მოთავსებული იქნება გარკვეულ დიაპაზონში-**მბმს** უბრალოდ არ გამოიტანს იმ ჩანაწერებს, რომლებიც არ აკმაყოფილებენ ჩვენს მიერ განსაზღვრულ პირობებს.

ყოველ ველს შეუძლია შეინახოს რომელიმე ერთი ტიპის მონაცემები: სტრიქონი, რიცხვი, თარიღი და სხვა. ველში შესანახი მონაცემების ტიპი განისაზღვრება ველის შექმნის დროს (შეიძლება მისი შეცვლა, თუ არასწორად იქნა განსაზღვრული). **მბმს** უფლებას არ მოგვცემს ჩავწეროთ ვთქვათ თარიღი, ველში, რომელიც რიცხვებისათვის არის განკუთვნილი. მონაცემთა ბაზების უმრავლესობის ფორმატით მხარდაჭერილი მონაცემების ტიპები მოცემულია ცხრილში- 6.1.

ცხრილი 6.1. მონაცემთა ტიპები, რომლებიც მხარდაჭერილია მონაცემთა ბაზების უმეტესობის ფორმატებით.

დასახელება	განმარტება
სტრიქონული	ნებისმიერი სიმბოლოების: ასოები, ციფრები, სასვენი ნიშნები, ჰარები და სხვა, შემცველი ფიქსირებული სიგრძის სტრიქონები
მთელრიცხვებიანი	მთელი რიცხვები
მცოცავი მძიმით	წილადი რიცხვები
ლოგიკური	True(ჭეშმარიტი) და Folse

	(მცდარი) ტიპის მნიშვნელობები
თარიღი	თარიღის მნიშვნელობები
თარიღი და დრო	თარიღისა და დროის გაერთიანებული მნიშვნელობა
მემო	ნებისმიერი სიგრძის ტექსტი (Memo)
მთვლელი	თანდათანობით მზარდი და ცხრილის ფარგლებში არაგანმეორებადი მთელი რიცხვები. ასეთი ტიპის ველები გამოიყენებიან სპეციალური დანიშნულებით

შენიშვნა: *მთელი და მცოცავი მძიმით რიცხვითი ტიპების მქონე მონაცემების რამდენიმე ნაირსახეობა არსებობს, რომლებიც ერთმანეთისაგან განსხვავდებიან რიცხვების სიდიდით, რომელთა ჩაწერაც შესაძლებელია მოცემული ტიპის ველში. ჩვენ მათ შესახებ მოგვიანებით ვისაუბრებთ.*

ველში შესანახი მონაცემების ტიპების ხისტად განსაზღვრა, ასევე ემსახურება **მბმს** მიერ მონაცემების გადამუშავების გაიოლებას. ასე მაგალითად, მთელი რიცხვების და მცოცავ მძიმიანი რიცხვების ველების მნიშვნელობები შეიძლება მონაწილეობდნენ არითმეტიკულ გამოთვლებში. ლოგიკური ველები ძალიან მცირე ადგილს იკავებენ და სწრაფად ხდება მათი დამუშავება. ხოლო, **memo** ველების დამუშავება ძალიან ნელა ხდება, სამაგიეროდ მათ შეუძლიათ ნებისმიერი ზომის ტექსტის შენახვა, რაც ხშირად ძალზე აუცილებელია.

მოდით, კიდევ ერთხელ შევხედოთ ნახ.6.1.-ს. იქ წარმოდგენილ ცხრილს სამი ველი გააჩნია. რა ტიპის არიან ისინი? მოდით ვიფიქროთ.
ველი date ჩვენ შევიტანეთ იმისათვის, რომ გვქონოდა ამ ველის მიხედვით დახარისხების და ყველაზე "ახალი" სტატიების მოძიების შესაძლებლობა. მაშასადამე, დამუშავების დასაჩქარებლად მას უნდა მიენიჭოს თარიღის ტიპი. ხოლო ველები **author** და **name** შეიცავენ

ჩვეულებრივ ტექსტს, ამიტომ ისინი უნდა იყვნენ სტრიქონული ტიპის.
ისღა დაგვრჩა სათქმელი მხოლოდ, რომ ველების ერთობლივობას, მათი სახელებისა და მონაცემთა ტიპების ჩათვლით უწოდებენ ცხრილის **სტრუქტურას** ან **მეტამონაცემებს**. თვით რეალური მონაცემები-ველებისა და ჩანაწერების შიგთავსი-სტრუქტურაში არ შედიან.

წესები

ადრე ჩვენ ვთქვით, რომ ცხრილის ველს გააჩნია ორი აუცილებელი პარამეტრი: სახელი და მონაცემების ტიპი. ამ პარამეტრების განსაზღვრა ხდება ცხრილის შექმნის დროს და შემდგომში შესაძლებელია მათი შეცვლა, მაგალითად, იმ შემთხვევაში, თუ აღმოჩნდება, რომ არასწორად იყო განსაზღვრული ან აუცილებელი გახდება ცხრილის სტრუქტურის შეცვლა.
მონაცემთა ჩანაწერების უმრავლესობის ფორმატს გააჩნია აგრეთვე, ველის კიდევ ერთი პარამეტრი-**ველის წესები.** წესები განსაზღვრავენ იმ პირობებს, რომლებსაც უნადა აკმაყოფილებდნენ ამ ველში ჩასაწერი მონაცემები. ასეთი პირობები შეიძლება იყოს:

-ველში რაიმე მნიშვნელობის არსებობის აუცილებლობა (**სავალდებულო ველი**);
-**დიაპაზონი**, რომელშიც უნდა მოთავსდნენ რიცხვითი მნიშვნელობები ან თარიღები;
-მოცემული ველის მნიშვნელობის **ურთიერთკავშირი** იმავე ჩანაწერის სხვა ველების მნიშვნელობებთან;
-მნიშვნელობა, რომელიც უნდა მოთავსდეს ველში ახალი ჩანაწერეის შესრულებისას (ველის წინასწარ განსაზღვრული **-გაჩუმებითი მნიშვნელობა).**
ველის წესებიდან ყველაზე ხშირად გამოიყენება სავალდებულო ველები და გაჩუმებითი მნიშვნელობები-ისინი მხარდაჭერილია მონაცემთა ბაზების ფორმატების უმრავლესობით. შედარებით ნაკლებად გამოიყენება დიაპაზონის წესები, კიდევ უფრო იშვიათად-მოცემული ველის და ცხრილის სხვა ველებს მნიშვნელობებს შორის ურთიერთკავშირები.
მაგალითისათვის ისევ ჩვენს ცხრლის მივმართოთ, რომელიც ნახ.6.1.-ზეა ნაჩვენები. ველი **data** შეიცავს სიაში სტატიის დამატების თარიღს და სრულიად ლოგიკურია, რომ მისი მნიშვნელობა

იწერებოდეს ახალი ჩანაწერის შესრულებისთანავე. ეს ნიშნავს იმას, რომ ჩვენ ამ ველისათვის შეგვიძლია შევქმნათ წესი, რომლის მიხედვითაც ამ ველის გაჩუმებითი მნიშვნელობა, მიმდინარე თარიღის ტოლი იქნება.
წესების გამოყენება შესაძლებელია აგრეთვე არა მხოლოდ ცალკეული ველის, არამედ მთლიანი ცხრილის მიმართაც **(ცხრილის წესები).** ასეთ წესებს მიეკუთვნება:

-ცხრილის ფარგლებში რომელიმე ველის მნიშვნელობათა უნიკალურობის მოთხოვნა **(უნიკალური ველი);**
-ცხრილის რომელიმე ველის მნიშვნელობის ურთიერთკავშირი, ამავე ცხრილის ნებისმიერი ჩანაწერის, ნებისმიერი სხვა ველის მნიშვნელობებთან.

ცხრილების წესებიდან ყველაზე ხშირად გამოიყენება უნიკალური ველების წესი. ჩვეულებრივ, უნიკალური ველები წარმოადგენენ გასაღებ ველებს (გასაღები ველების შესახებ უფრო დაწვრილებით მოგვიანებით ვისაუბრებთ).

ურთიერთკავშირები აქაც გაცილებით იშვიათად გამოიყენება, ვინაიდან მონაცემთა ბაზების ფორმატებიდან, მხოლოდ ზოგიერთების მიერ ხდება მათი მხარდაჭერა, ამავე დროს ძალზე იშვიათად ჩნდება მათი გამოყენების აუცილებლობა.

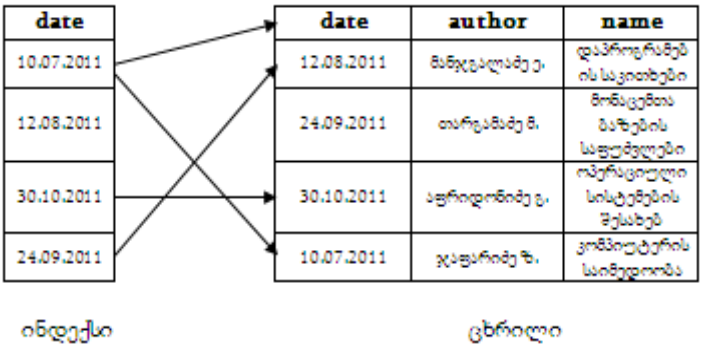
ინდექსები და გასაღებები

დავუშვათ,რომ ჩვენ უკვე შევქმენით ცხრილი, რომელიც ნახ.6.1-ზეა გამოსახული, და ვავსებთ მას მონაცემებით. დავუშვათ, რომ ჩვენ ასევე უკვე შევქმენით საკმაოდ ბევრი ჩანაწერი და გაგვიჩნდა მათი ეკრანზე გამოტანის სურვილი, რაიმე ნიშნის ან ველის მნიშვნელობის მიხედვით დახარისხებული სახით. აქ ჩვენ წავაწყდებით პრობლემას, რომლის შესახებაც ღირს უფრო დაწვრილებითი საუბარი.
საქმე იმაშია, რომ მანაცემთა ბაზებთან სამუშაო ყველა პროგრამა, ცხრილში მონაცემებს ამატებს იმავე თანმიმდევრობით, რომლითაც ხდება მათი შეტანა. შედეგად ვღებულობთ დაუხარისხებელ ჩანაწერებს. უფრო სწორად, დახარისხებულს იმ რიგის მიხედვით, რა რიგითაც მოხდა მათი შეტანა. თუკი ჩვენ გვინდა მათი დახარისხება

სტატიების დამატების თარიღის მიხედვით (**data** ველის მნიშვნელობების მიხევით)? როგორ მოვიქცეთ ამ შემთხვევაში?
ჩვეულებრივ, კარგ მბმს-ს უნდა შეეძლოს ჩანაწერების ეკრანზე გამოტანისას, მათი დახარისხება ჩვენს მიერ განსაზღვრული ნებისმიერი წესის მიხედვით. მგრამ ის ამას გააკეთებს ძალიან ნელა. მართლაც, მბმს-მ უნდა გააკეთოს შემდეგი:

1.წაიკითხოს ფაილიდან ყველა ჩანაწერი, რომელიც ჩვენ გვჭირდება;
2. მოათავსოს ისინი ოპერატიულ მეხსიერებაში;
3.დაახარისხოს ისინი მოცემული რიგის მიხედვით;
4.გამოიყვანოს დახარისხებული ჩანაწერები ეკრანზე.

თუ ჩანაწერების რაოდენობა დიდი არ არის, რათქმა უნდა ის ბევრ დროს (მეხსიერებასაც) არ მოითხოვს. თუკი ბევრია? (სწორედ ამიტომ არის, რომ **მბმს**-ები, მუშაობისას მოითხოვენ დიდი მოცულობის ოპერატიულ მეხსიერებას-მათ დამუშავებაში მონაწილე ჩანაწერები სადმე უნდა შეინახონ).
დავუშვათ, რომ ჩვენ პირდაპირ მონაცემთა ბაზაში შევქმენით ველების მნიშვნელობათა განსაკუთრებული ბლოკი, რომლის მიხედვითაც უნდა ვაწარმოოთ ჩანაწერების დახარისხება. ამასთან ამ ბლოკში მნიშვნელობები განთავსებულია იმ თანმიმდევრობით, რა თანმიმდევრობითაც უნდა მოხდეს ჩანაწერების დახარისხება. გარდა ამისა, ეს მასივი შეიცავს მინიშნებებს ჩანაწერებზე, რომლებსაც შეესაბამებიან მასში შენახული ველების მნიშვნელობები (ნახ.6.2).



ნახ.6.2. ცხრილი-ინდექსიანი სტატიების სია

ასეთ მასივს დავარქვათ **ინდექსი**. და შევხედოთ, რა მოხდება, თუ **მბმს**-ს ჩვენ მოვთხოვთ ცხრილის ჩანაწერების დახარისხებას იმ ველის მიხედვით, რომლის მნიშვნელობებიც ინდექსშია შენახული **(ინდექსირებული ველის მიხედვით).**

მბმს გახსნის მონაცემთა ბაზების ფაილს, მოძებნის საჭირო ცხრილს და მისი ველისთვის შექმნილ ინდექს, ერთსაც და მეორესაც ჩატვირთავს მეხსიერებაში. იცის რა სასურველი დახარისხებისათვის ჩანაწერების თანმიმდევრულობა-სწორედ ის არის ჩაწერილი ინდექსში-მბმს სწრაფად გადაადგილებს ცხრილის ჩანაწერებს შესაბამისი თანმიმდევრობით და ეკრანზე გამოიტანს დახარისხებულ ცხრილს.

ინდექსების მხარდაჭერა ხდება მონაცემთა ბაზების აბსოლუტურად ყველა ფორმატის მიერ და ხშირად ხდება მათი გამოყენება. მართლაც, ინდექსი იკავებს მცირე ადგილს როგორც დისკოზე, ასევე ოპერატიულ მეხსიერებაშიც, ხოლო დახარისხების ოპერაციას მნიშვნელოვნად აჩქარებს. ერთად ერთი ნაკლოვანება: ნებისმიერი ჩანაწერის დამატების, შეცვლის, ან წაშლის შემთხვევაში მბმს იძულებული იქნება შესაბამისად შეცვალოს ინდექსიც, რასაც გარკვეული დრო ჭირდება. ამიტომ, არ არის სასურველი ძალიან ბევრი ინდექსების შექმნა-მათ განახლებას ბევრი დრო დასჭირდება.

დახარისხების გარდა, ინდექსები გვეხმარება ჩანაწერების ფილტრაციის ოპერაციების შესრულებაშიც-მომხმარებლის მიერ განსაზღვრული გარკვეული წესების ერთობლივობის (**კრიტერიუმების**) შესაბამისად ჩანაწერების არჩევაში. მბმს ტვირთავს ინდექსს მეხსიერებაში, ეძებს განსაზღვრული კრიტერიუმების შესაბამის მნიშვნელობებს და ამოკრიბავს სასურველ ჩანაწერებს ცხრილიდან. ყველაფერი მარტივად და სწრაფად ხდება!

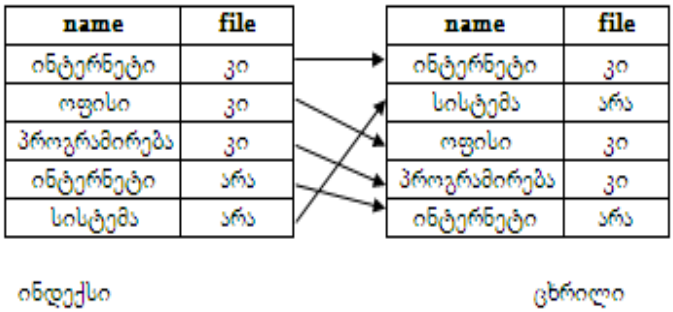
შესაძლებელია, რომ ინდექსები შეიცავდნენ ერთზე მეტ ველს. ასეთ ინდექსებს **რთულ ინდექსებს** უწოდებენ. რთული ინდექსების დამუშავება უფრო დიდხანს ხდება, ვიდრე **მარტივი ინდექსებისა,** ამიტომ მათ შედარებით იშვიათად იყენებენ.

თავდაპირველად, ცხრილის გახსნისას, მბმს არცერთ ინდექსს არ ტვირთავს მეხსიერებაში-მათი ამოქმედება ხდება მხოლოდ დახარისხებისა და ფილტრაციის შემთხვევაში. მაგრამ არსებობს შესაძლებლობა ერთ-ერთი ინდექსი გავხადოთ ცხრილის გახსნისას

ჩატვირთვად ინდექსად; ამასთან ცხრილი თავიდანვე იქნება დახარისხებული ამ ინდექსის შესაბამისად. ასეთ ინდექსს **პირველად გასაღებს** ან **გასაღებ ინდექსს** უწოდებენ, ხოლო ველს, რომლის საფუძველზეც ეს ინდექსია შექმნილი-**გასაღებ ველს** უწოდებენ. გასაღები ინდექსი მთელ ცხრილზე მხოლოდ ერთი შეიძლება იყოს.
გასაღები ველი (ან ველები), რომლებიც ქმნიან გასაღებ ინდექსს უნდა აკმაყოფილებდნენ შემდეგ პირობებს:

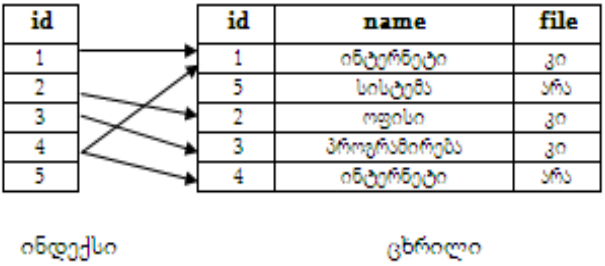
-ისინი აუცილებლად უნდა შეიცავდნენ მნიშვნელობას (ანუ უნდა წარმოადგენდნენ სავალდებულო ველებს);
-მათ უნდა გააჩნდეთ ცხრილის ფარგლებში უნიკალური მნიშვნელობები (წარმოადგენდნენ უნიკალურ ველებს). რთული გასაღები ინდექსის შემთხვევაში ზემოაღნიშნული შეეხება ამ ინდექსში შემავალი ყველა ველის მნიშვნელობათა ერთობლივობას.

შეიძლება ითქვას, რომ გასაღები ინდექსის შექმნა განაპირობებს ცხრილში ორი წესის არსებობას: სავალდებულო ველის არსებობა (ველის წესი) და უნიკალური ველის არსებობა (ცხრილის წესი).
შევხედოთ ნახ.6.3.-ს. იქ ნაჩვენებია კატეგორიების სიის შემცველი ცხრილი. მას ორი ველი გააჩნია: **name**-კატეგორიების დასახელება და **file**-ლოგიკური ველი, რომელიც გვიჩვენებს, აღწერს თუ არა მოცემული ჩანაწერი ფაილის კატეგორიას.
გარდა ამისა, ეს ცხრილი შეიცავს ორივე ამ ველისაგან შემდგარ გასაღებ ინდექსს. ჩვენ რომ ინდექსში მხოლოდ name ველი შეგვეტანა, ჩვენს ინდექსში ორი ერთნაირი ***ინტერნეტი*** მნიშვნელობა გაჩნდებოდა, რომლებიც სხვადასხვა ჩანაწერებს მიეკუთვნებიან. მბმს გამოგვიყვანდა შეტყობინებას შეცდომის შესახებ და არ შექმნიდა ასეთ ინდექსს.



ნახ.6.3.ცხრილი-კატეგორიების სია, რთული გასაღები ველით.

ვინაიდან ჩვენ მივიღეთ რთული ინდექსი, რომელიც ორ ველს შეიცავს, მოდით შევეცადოთ მისგან განთავისუფლებას. შევქმნათ კატეგორიების სიის შემცველ ცხრილში კიდევ ერთი ველი-**id** და განვსაზღვროთ მისი ტიპი, როგორც **მთვლელი.** ამის შემდეგ, მხოლოდ ამ ერთი ველის საფუძველზე შევქმნათ გასაღები ველი- იხილეთ ნახ.6.4. ვინაიდან მთვლელის ტიპის ველი გარანტირებულად შეიცავს უნიკალურ მნიშვნელობებს, ეს ჩვენ მარტივად გამოგვივა და მბმს-ც არ იქნება წინააღმდეგი.



ნახ.6.4. ცხრილი-კატეგორიების სია, სუროგატული გასაღები ინდექსით.

ასეთ გასაღებ ინდექსს, რომელიც შეიცავს არა რეალურ, არამედ "განყენებულ" მონაცემებს, **სუროგატული გასაღები ინდექსი** ეწოდება. პრაქტიკულად ყოველთვის, სუროგატულს წარმოადგენენ მხოლოდ გასღები ინდექსები.

იბადება ლოგიკური კითხვა: რისთვის არის საჭირო ასეთი ინდექსი? ის ხომ რეალურად არაფრის დახარისხებას არ აწარმოებს? საქმე იმაშია, რომ ასეთი გასაღები ინდექსები გამოიყენებიან განსაკუთრებულ შემთხვევებში, კერძოდ ცხრილებს შორის კავშირების დასამყარებლად, რის შესახებაც ჩვენ შემდეგში ვისაუბრებთ. გარდა ამისა, გასაღები ინდექსები (სუროგატული და ჩვეულებრივი) საჭიროა იმისთვის, რომ ცალსახად იქნას იდენტიფიცირებული ცალკეული ჩანაწერი, მაგალითად მისი შეცვლის ან წაშლის მიზნით. მაგრამ ამის შესახებაც მომავალში გვექნება საუბარი, როდესაც ჩვენ დავიწყებთ **მონაცემთა მართვის ენის-SQL** შესაწავლას.

მონაცემთა ბაზების ბევრი ფორმატი მოითხოვს, რომ ყველა ინდექსს, გარდა გასაღები ინდექსებისა, ცხრილის ფარგლებში გააჩნდეს უნიკალური სახელი. გასაღებ ინდექს კი სახელი არ ჭირდება, ვინაიდან ის ერთია მთელს ცხრილში. ზოგიერთი ფორმატი კი სულ არ მოითხოვს ინდექსების დასათაურებას.

კავშირები

დავუბრუნდეთ ცხრილს-სტატიების სიას, რომელიც ნახ.6.1-ზეა წარმოდგენილი. რა გვაკლია ჩვენ აქ? სწორია-ცნობები იმის შესახებ, თუ რომელ კატეგორიას მიეკუთვნება თითოეული სტატია. მოდით დავუმატოთ ისინი, რაც მთავარია კატეგორიების სია ჩვენ უკვე გვაქვს (იხ.ნახ.6.4).

ერთი შეხედვით ეს თითქოს მარტივად კეთდება. ამისთვის მხოლოდ უნდა შევქმნათ ცხრილში-სტატიების სია-კიდევ ერთი სტრიქონული ველი, რომელიც შეიცავს კატეგორიების დასახელებებს, ასეთნაირად-იხ.ნახ.6.5.

date	author	name	category
12.08.2011	მანჯგალაძე ე.	დაპროგრამების საკითხები	ინტერნეტი
24.09.2011	თარგამაძე მ.	მონაცემთა ბაზების საფუძვლები	ინტერნეტი
30.10.2011	აფრიდონიძე გ.	ოპერაციული	სისტემა

		სისტემების შესახებ	
10.07.2011	ჯაფარიძე ზ.	კომპიუტერის საიმედოობა	ინტერნეტი

ნახ.6.5. ცხრილი-სტატიების სია კატეგორიების სახელწოდებებით.

ხომ ყველაფერი მარტივია? მაგრამ ეს ისეთი სიმარტივეა, რომელიც ქურდობაზე უარესია. მოდით გავარკვიოთ რატომ.

-ყველაზე დიდი ველები (ანუ ისეთი ველები, რომლებიც მონაცემთა ბაზების ფაილში ყველაზე დიდ ადგილს იკავებენ)-სტრიქონული (სიმბოლური) და **memo** ველებია. ამასთან, თუ **memo** ველის ზომა ყოველთვის ტოლია მისი მნიშვნელობის ზომისა, სიმბოლური ველის ზომა განისაზღვრება მისი შექმნისას და ყოველთვის მუდმივი რჩება. არავითარი მნიშვნელობა არა აქვს, ჩვენ მასში ჩავწერთ ერთ სიმბოლოს, სტრიქონს, რომელიც მთელს ველს დაიკავებს, თუ საერთოდ დავტოვებთ ცარიელს. ამიტომ სასურველია, რომ სტრიქონული (სიმბოლური) ველების რაოდენობა ცხრილში დაყვანილი იქნას მინიმუმამდე.

-ჩვენს მიერ შექმნილი ველი **category** ფაქტიურად შეიცავს ერთი და იმავე მნიშვნელობების შეზღუდულ ანაკრებს-კატეგორიების დასახელებებს. ეს, არც თუ ისე კორექტულია-ველები შეძლებისდაგვარად უნდა შეიცავდნენ ყოველი ჩანაწერისათვის უნიკალურ მნიშვნელობებს. ასე გვკარნახობენ მონაცემთა რელაციური ბაზების შექმნის წესები.

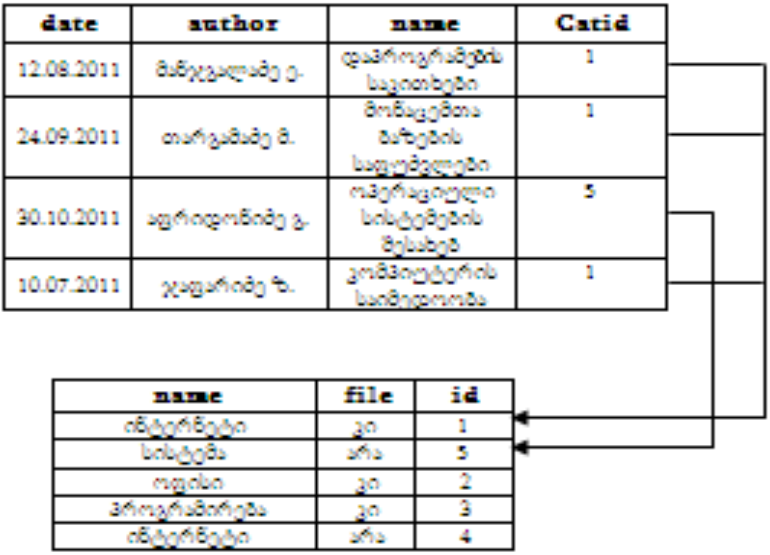
-ახლა, მოდით შევხედოთ ჩვენს ცხრილს პრაქტიკული თვალსაზრისიდან. დავუშვათ, რომ ჩვენ გვჭირდება რომელიმე კატეგორიის დასახელების შეცვლა: შეცდომის შესწორება, სიტყვა ”ინტერნეტი”-ის დაწერა ინგლისურად-„**Internet**” და ა.შ.. ამისათვის ჩვენ მოგვიწევს მთელი ცხრილის დათვალიერება, ყველა იმ ჩანაწერის მოძებნა, რომელთა category ველების მნიშვნელობების შეცვლა გვსურს და შესწორებების შეტანა ყოველ მოძებნილ ჩანაწერში.

-ჩვენ გვაქვს ცხრილი-კატეგორიების სია. ჩვენ ის ტყუილად ხომ არ შეგვიქმნია?

განვიხილოთ საკითხის ასეთი ალტერნატიული გადაწყვეტა. რა მოხდება, თუ ჩვენ ცხრილში-სტატიების სია, შევინახავთ არა თვით კატეგორიის დასახელებას, არამედ მინიშნებას შესაბამის ჩანაწერზე?

მითუმეტეს, რომ კატეგორიების სიას გააჩნია სუროგატული გასაღები ინდექსი, რაც იმას ნიშნავს, რომ ჩვენ შეგვძლია ცალსახად მივუთითოთ ჩვენთვის სასურველ ჩანაწერზე-კატეგორიაზე. შესაძლებელია თუ არა ასეთი მიდგომა?

რათქმა უნდა შესაძლებელია. შევხედოთ ნახ.6.6-ს. იქ გამოსახულია ცხრილი-სტატიების სია, რომელშიც შექმნილია ახალი რიცხვითი ველი **catid** (ველი **category**-წაშლილია). ეს ველი ინახავს ცხრილის "კატეგორიების სია", id ველის მნიშვნელობას, რომელიც შეესაბამება საჭირო ჩანაწერს-კატეგორიას. სტატიების სიის ცხრილის დამუშავებისას, მბმს ამოიღებს ამ მნიშვნელობებს, მოძებნის შესაბამის ჩანაწერებს ცხრილში და გამოიტანს ეკრანზე ამ ჩანაწერების **name** ველების მნიშვნელობებს (კატეგორიების დასახელებებს).



date	author	name	Catid
12.08.2011	მაისურაძე ე.	დაპროგრამების საკითხები	1
24.09.2011	თარგამაძე მ.	მონაცემთა ბაზების საფუძვლები	1
30.10.2011	ფერიდოწიძე გ.	ოპერაციული სისტემების შესახებ	5
10.07.2011	ჯაფარიძე ზ.	კომპიუტერის საიმედოობა	1

name	file	id
ინტერნეტი	კი	1
სისტემა	არა	5
ოფისი	კი	2
პროგრამირება	კი	3
ინტერნეტი	არა	4

ნახ.6.6. ცხრილი-სტატიების სია, რომელიც დაკავშირებულია ცხრილთან-კატეგორიების სია

ფაქტიურად ჩვენ შევქმენით **კავშირი** ჩვენი მონაცემთა ბაზის ცხრილებს შორის, რომელშიც ცხრილი-სტატიების სია, დამატებითი

მონაცემებისათვის მიმართავს ცხრილს-კატეგორიების სია. ამგვარად ჩვენ ყველა კურდღელი ერთდროულად დავიჭირეთ: ჩვენი მონაცემთა ბაზის მოცულობა შემცირდა, ყოველი ჩანაწერის ველი შეიცავს მხოლოდ მოცემული ჩანაწერისათვის გათვალისწინებულ უნიკალურ მნიშვნელობებს, და თვით ცხრილიც-კატეგორიების სია-იქნა გამოყენებული. ხოლო, რომელიმე კატეგორიის სახელის შესაცვლელად, საკმარისია ცხრილის-კატეგორიების სიის-**name** ველის მნიშვნელობების შეცვლა.

მონაცემთა ბაზების პროფესიონალური დამპროექტებლები, **პირველადს** ან **მშობლიურ** ცხრილს უწოდებენ ცხრილს, რომელიც შეიცავს დამატებით მონაცემებს, ხოლო **მეორადს** ან **შვილობილს**-ცხრილს, რომელიც დამატებითი მონაცემებისათვის მიმართავს პირველად ცხრილს. მაშასადამე, ჩვენს შემთხვევაში სტატიების სია-მეორადი ცხრილია, ხოლო კატეგორიების სია-პირველადი. აქვე ავღნიშნოთ, რომ კავშირს, რომელშიც პირველადი ცხრილის ერთ ჩანაწერს მიმართავს მეორადი ცხრილის რამოდენიმე ჩანაწერი, ეწოდება კავშირი "**ერთი-მრავალთან**."

სტატიების სიის ცხრილის **catid** ველს, რომელიც შეიცავს მინიშნებებს პირველადი ცხრილის ჩანაწერებზე, **გარე ინდექსი** ეწოდება. "გარე"-იმიტომ, რომ ეს ველი გარეა პირველად ცხრილთან მიმართებაში, ხოლო "ინდექსი"-იმიტომ, რომ პრაქტიკულად ყოველთვის სწორედ ამ ველის საფუძველზე იქმნება ინდექსი.

მხოლოდ ისღა დაგვრჩა სათქმელი, რომ მონაცემთა რელაციურმა ბაზებმა თავიანთი დასახელება მიიღეს იმის გამო, რომ შედგებიან ურთიერთდაკავშირებული ცხრილების ერთობლივობისაგან (ინგლისურიდან **relation**-კავშირი).

შენიშვნა: *მონაცემთა ბაზების ზოგიერთი ფორმატი, კავშირების (მეტამონაცემები) განსაზღვრის საშუალებას იძლევა პირდაპირ ცხრილის სტრუქტურაში. ეს საკმაოდ სასარგებლო შესაძლებლობებს ქმნის, მაგალითად იმისათვის რომ, შევქმნათ "**ერთი-ერთთან**" კავშირები, როდესაც პირველადი ცხრილის ერთ ჩანაწერზე მიმართვა, შეუძლია მეორადი ცხრილის მხოლოდ ერთ ჩანაწერს ან კიდევ უზრუნველვყოთ მინიშნებათა ერთიანობის მხარდაჭერა. მონაცემთა ბაზების სხვა ფორმატები არ იძლევიან ცხრილებს შორის კავშირების, მათივე სტრუქტურებში განსაზღვრის შესაძლებლობებს; ამის შესახებ ზრუნვა იმ პროგრამისტების საქმეა, რომლებიც წერენ*

მონაცემების დამუშავების პროგრამებს ან იმ მომხმარებლების, ვინც მოითხოვს მონაცემებს მბმს-ებიდან.

ფაქტიურად ჩვენ ყველაფერი ვთქვით რელაციური მონაცემთა ბაზების სამ შემადგენელ ნაწილზე. ამის შემდეგ, მოდით ვისაუბროთ თვით მბმს-ებზე, რომლებიც უზრუნველყოფენ მათ დამუშავებას.

შენიშვნა: *მონაცემთა ბაზების ზოგიერთი ფორმატი, ასევე იძლევა მონაცემთა დამუშავებისთვის განკუთვნილი მცირე პროგრამების, თვით ბაზებში შენახვის საშუალებას. ასეთი პროგრამები ორი სახისა არიან.* ***ტრიგერები****, რომელთა შესრულება ხდება ჩანაწერების დამატების, შეცვლის, წაშლის ოპერაციების დროს და რომელთაც შეუძლიათ მონაცემების შემოწმებასთან და დამუშავებასთან დაკავშირებული, რაიმე დამატებითი ოპერაციების შესრულება.* ***შენახვადი პროცედურების*** *გამოძახება ხდება თვით მომხმარებლის მიერ, და გამოიყენება მონაცემების რთული პირობების შესაბამისად შერჩევისა და დამუშავებისათვის.*

სამაგიდო და სერვერული რელაციური მბმს-ები

ჩვენ უკვე ვიცით, რომ მბმს-ეს არის მონაცემთა ბაზებთან სამუშაო პროგრამა. სწორედ **მბმს**-ის საშუალებით მომხმარებელი და სხვა პროგრამები იღებენ მონაცემთა ბაზებში შენახულ ინფორმაციასთან წვდომის შესაძლებლობას.
როგორც წესი, ნებისმიერი მბმს შედგება ორი ნაწილისაგან. პირველი ნაწილი-ეს ის პროგრამაა, რომელთანაც მომხმარებელი უშუალოდ მუშაობს-**მონაცემების კლიენტი.** მეორე ნაწილი კი უშუალოდ მონაცემთა ბაზებით არის დაკავებული: იღებს მონაცემთა კლიენტისაგან, მონაცემების არჩევასთან ან შეცვლასთან დაკავშირებულ მოთხოვნებს, ასრულებს მათ და უგზავნის კლიენტს. ეს ეგრეთ წოდებული **მონაცემების პროცესორია.** შეიძლება ითქვას, რომ მონაცემთა კლიენტი ახორციელებს მომხმარებლისაგან მოთხოვნების მიღებას და შედეგების გამოტანას, ხოლო პროცესორი-უშუალოდ მონაცემების დამუშავებას.
იმისდა მიხედვით, თუ როგორ არიან რეალიზებული კლიენტი და მონაცემთა პროცესორი, მბმს-ები იყოფიან ორ დიდ ჯგუფად: **სამაგიდე** და **კლიენტ-სერვერული**. ჩვენ ახლა მათ შესახებ ვისაუბრებთ.

სამაგიდე მბმს რეალიზებულია ერთადერთი პროგრამის სახით; კლიენტიც და მონაცემთა პროცესორიც გაერთიანებულია ერთ შესრულებად ფაილში. (რათქმა უნდა სხვანაირადაც ხდება. მაგალითად, **Microsoft Access**-ში მონაცემთა პროცესორი რეალიზებულია **DLL**-ბიბლიოთეკების ანაკრების სახით, მაგრამ მიუხედავად ამისა ფაქტიურად კლიენტთან ერთად, მაინც ერთ მთლიანობას წარმოადგენს. ეს არის პირველი განსხვავება. მეორე განსხვავება: სამაგიდე მბმს მუშაობს უშუალოდ მონაცემთა ბაზების ფაილებთან ზუსტად ისევე, როგორც **Microsoft Word** მუშაობს დოკუმენტების ფაილებთან.

როდესაც მომხმარებელს ჭირდება ბაზიდან მონაცემების მიღება, იგი **მბმს**-ის საშუალებით ხსნის ამ ბაზის შემცველ ფაილს. მბმს კითხულობს დამხმარე ინფორმაციის შემცველი ფაილის დასაწყისს (ეგრეთ წოდებულ **ფაილის სათაურს**), რომელიც ჯერ ტვირთავს მონაცემების პირველ ფრაგმენტს და ამუშავებს მას, შემდეგ-მეორეს, მესამეს და ა.შ., სანამ მომხმარებლისათვის საჭირო ყველა მონაცემი არ იქნება გამოტანილი ეკრანზე. თუ მომხმარებელი ცვლის რაიმე მონაცემებს, მბმს ცვლილებებს წერს ფაილის განსაზღვრულ ადგილას, ცვლის სხვადასხვა დამხმარე სტრუქტურებს და შესაძლოა ფაილის სათაურშიც ჩაწეროს რაიმე. მუშაობის დასრულებისას, მომხმარებელი ხურავს მონაცემთა ბაზის შემცველ ფაილს.

ეს ჩვეულებრივი პრინციპია და როგორც ჩანს, არც თუ ისე ცუდი. ამასთან მბმს-ები ძალზე სწრაფად მუშაობენ, მაგრამ მხოლოდ იმ შემთხვევაში, თუ მონაცემთა ბაზების ფაილები მდებარეობენ იმავე კომპიუტერის დისკოზე, სადაც დაყენებულია თვით მბმს. თუ მომხმარებლისათვის საჭირო მონაცემთა ბაზის ფაილები მდებარეობენ სხვა კომპიუტერზე, მბმს-ის მუშაობის სიჩქარე მკვეთრად ეცემა. ეს გასაგებიც არის, ვინაიდან ცნობილია, რომ ქსელში მონაცემების გადაცემა მნიშვნელოვნად ნელა ხდება, ვიდრე კომპიუტერის შიგნით. თუ კი ერთი და იგივე მონაცემთა ბაზა რამოდენიმე მომხმარებელმა ერთდროულად გახსნა, მუშაობა თითქმის შეუძლებელი ხდება-მომხმარებელს დროის უმეტესი ნაწილის განმავლობაში უწევს ლოდინი, სანამ მბმს, ბაზის ფაილიდან მონაცემების მორიგ ფრაგმენტს მიიღებს.

ამიტომ ჯერ კიდევ 60-იან წლებში შეიქმნა **სერვერული მბმს**-ები (ანუ მონაცემთა სერვერები), რომლებიც მეორე ჯგუფს განეკუთვნებიან. **სერვერული მბმს**-ეს ცალკე პროგრამის სახით გაფორმებული მონაცემთა პროცესორია, რომელიც სპეციალურად ამისთვის

გამოყოფილ სერვერულ კომპიუტერზე მუშაობს. როგორც ნებისმიერი სხვა სერვერი, ის კლიენტებისაგან იღებს მოთხოვნებს, კითხულობს მონაცემებს ბაზების ფაილიდან, ამუშავებს მათ და დამუშავების შედეგებს უგზავნის კლიენტებს.

კი მაგრამ, რომელი პროგრამების გამოყენება შეიძლება სერვერული მბმს-ების კლიენტების სახით? ამასთან დაკავშირებით პრობლემა არ არსებობს. ჯერ-ერთი, ეს სპეციალურად დაწერილი პროგრამებია, რომელთა დანიშნულებაც მდგომარეობს რომელიმე წინასწარ განსაზღვრულ მონაცემთა ბაზებთან მუშაობაში. ჩვეულებრივ, როგორც კლიენტის, ასევე თვით მონაცემთა ბაზის მიწოდება ხდება გარკვეული სახის მონაცემების შენახვისა და დამუშავებისათვის განკუთვნილი ერთიანი პაკეტის სახით: საბუღალტრო მონაცემები, კატალოგები, გაყიდვების შესახებ ანგარიშები და სხვა. მეორე, სამაგიდო მბმს-ების მნიშვნელოვანი უმრავლესობა უზრუნველყოფს მუშაობას მათ სერვერულ "კოლეგებთან".

ვინაიდან სერვერულ მბმს-ებთან მომუშავე პროგრამების რაოდენობა ძალზე დიდია, საჭიროა, რომ "ვასწავლოთ" მათ, თუ როგორ უნდა განახორციელონ ქსელის საშუალებით სერვერებთან მუშაობა. ამისათვის კლიენტურ კომპიუტერებზე, გარდა თვით მონაცემთა კლიენტებისა, აყენებენ აგრეთვე **სერვერების კლიენტურ ნაწილებსაც**-მცირე ზომის პროგრამებს, რომლებიც ასრულებენ შუამავლის როლს მონაცემების კლიენტებსა და სერვერებს შორის.

კლიენტი უგზავნის თავის მოთხოვნებს სერვერის კლიენტურ ნაწილს, ის "ალაგებს" მათ ქსელურ პაკეტებში და უგზავნის სერვერს. სერვერისაგან პასუხის მიღების შემთხვევაში, იგივე კლიენტური ნაწილი "ამოალაგებს" მის შიგთავსს და უგზავნის კლიენტს.

სამაგიდე მბმს-ების **უპირატესობები**: დაყენებისა და გამოყენების სიმარტივე და დამატებითი პროგრამული უზრუნველყოფის მიმართ დაბალი მოთხოვნები (მათ ხომ არ ჭირდებათ მონაცემების სერვერები). **ნაკლოვანებები**: ქსელის საშუალებით მონაცემთა ბაზებისადმი მრავალმომხმარებლიან რეჟიმში წვდომის დროს, შედარებით დაბალი სწრაფქმედება, არასაკმარისი საიმედოობა და დაცულობა. ამიტომ სამაგიდე მბმს-ი გამოიყენებიან პერსონალური მონაცემთა ბაზების მართვისათვის (სატელეფონო წიგნები, ლიტერატურის კატალოგებიდა ა.შ.) და შედარებით მცირე, როგორც წესი არაქსელური მონაცემთა დამუშავების სისტემების შექმნისათვის.

სერვერული მბმს-ების უპირატესობები: მაღალი წარმადობა (ვინაიდან ქსელში გადაიცემიან მხოლოდ მოთხოვნები და პასუხები, რომლთა ზომები ყოველთვის ნაკლებია, ვიდრე ფაილების ფრაგმენტების ზომები), მაღალი საიმედოობა და დაცულობა. **ნაკლოვანებები**: დაყენების, გამართვის და თანხლების სირთულე. მაგრამ, ვინაიდან სერვერული მბმს-ები გამოიყენებიან წარმოებების დონის მონაცემთა დამუშავების დიდი სისტემების შესაქმნელად, ეს ნაკლოვანებები მნიშვნელოვან როლს არ თამაშობენ.

იმ მონაცემების შენახვისათვის, რომლებთანაც მუშაობენ ჩვენს მიერ მე-5 თავში განხილული სერვერული პროგრამები, ძირითადად გამოიყენებიან სერვერული მბმს-ები. მაგრამ, თუ საიტი მცირეა თავისი ზომებით და ხშირადაც არ ხდება მისი დათვალიერება, შეიძლება გამოყენებული იქნას სამაგიდე მბმს-აც. წინამდებარე წიგნში სწორედ ასეთი მიდგომაა განხორციელებული: აქ განხილული საიტი მონაცემების შენახვისათვის იყენებს მონაცემთა ბაზებს **Microsoft Access**-ს.

ახლა კი დაგვრჩა მხოლოდ ის, რომ მოვიყვანოთ სამაგიდე და სერვერული მბმს-ების მაგალითები. მაშ ასე, სამაგიდე მბმს-ებია: ჩვენთვის კარგად ნაცნობი **Microsoft Access, Corel Paradox**, **Borland dBase, Microsoft FoxPro** და სხვა ნაკლებად ცნობილი პროგრამები.

ხოლო, სერვერულ მბმს-ებს მიეკუთვნებიან: **Borland InterBase, MySQL, Microsoft SQL Server, PostgreSQL, Informix, Oracle, Sybase, IBM DB2** და მრავალი სხვა.

შენიშვნა: *არსებობს აგრეთვე მონაცემებთან სამუშაო პროგრამების განსაკუთრებული კლასი-ეგრეთ წოდებული* ***მონაცემების უნივერსალური პროცესორები.*** *ამ პროგრამების დანიშნულება მდგომარეობს იმაში, რომ ისინი უზრუნველყოფენ სამაგიდე მბმს-ების მუშაობის შესაძლებლობას მონაცემთა ბაზების სხვადასხვა ფორმატებთან, როგორც სამაგიდე, ისე სერვერულ ფორმატებთან. მონაცემთა უნივერსალური პროცესორის ტიპიური მაგალითია-ODBC (Open DataBase Connectivity, მონაცემთა ბაზებისადმი ღიად წვდომა), რომელიც შედის Windows-ის სტანდარტულ კომპლექტაციაში და მისი ანალოგი JDBC (Java DataBase Connectivity, Java-ზე დაწერილი პროგრამებისათვის მონაცემთა ბაზებისადმი წვდომა). (ამ წიგნში სწორედ ODBC-ია გამოყენებული საიტის*

შესაქმნელად). პრაქტიკულად ყველა თანამედროვე მბმს იყენებს ODBC-ს; JDBC-ი ნაკლებად არის გავრცელებული.

სერვერულ მბმს-ებს შორის გამორჩეული ადგილი უკავია პროგრამას-**MySQL**. ეს საკმაოდ მძლავრი, ძალიან სწრაფი და რესურსებისადმი ნაკლებად მომთხოვნი მონაცემთა სერვერია. ამავე დროს იგი უფასოა და ვრცელდება ღია ტექსტით. ვებ-საიტების უმრავლესობაში სწორედ მას იყენებენ.

მონაცემთა დამუშავების ენა SQL

მაშ ასე, ჩვენ დავასრულეთ საუბარი მონაცემთა რელაციურ ბაზებზე. ამის შემდეგ საუბარი შეეხება იმას, თუ როგორ ახორციელებენ მონაცემთა კლიენტები თავიანთი მოთხოვნების ფორმირებას სერვერებისადმი, კერძოდ მონაცემთა დამუშავების ენას **SQL**.

რისთვის არის საჭირო SQL.

საკმაოდ ბევრი მბმს არსებობს. ოდნავ ნაკლებია ბაზებში მონაცემთა შენახვის ფორმატების რაოდენობა. მოდით შევხედოთ სხვადასხვა მბმს-ების სიას, რომელიც ზემოთ იყო მოტანილი: ყოველი მბმს, მონაცემებს თავის საკუთარ ფორმატში ინახავს.

სამაგიდე მბმს-ებთან დაკავშირებით ყველაფერი მარტივად არის: ჩვენ ვხსნით მონაცემთა ბაზის ფაილს და პროგრამა თვითონ ერკვევა მათში. თუ ფორმატი, რომლითაც შენახულია ეს ფაილი არ მიეკუთვნება მის მიერ "გასაგებ"-ს, მაშინ მოგვიწევს სხვა უფრო "**ჭკვიანი**" პროგრამის ძებნა. (შესაძლებელია აგრეთვე **ODBC**-ის მსგავსი მონაცემთა უნივერსალური პროცესორის გამოყენება, მაგრამ მაშინ ხელმიუწვდომელი იქნება მონაცემთა ბაზების ამ ფორმატისთვის დამახასიათებელი სხვადასხვა დამატებითი შესაძლებლობები).
სერვერული **მბმს**-ები კი, სწორედ იმისათვის იქმნებოდნენ, რომ, რაიმე მონაცემების მომთხოვნი ნებისმიერი პროგრამისათვის, უზრუნველეყოთ ამ მონაცემებისადმი საიმედო წვდომის შესაძლებლობა. ხშირად ისე ხდება, რომ მომხმარებელმა არც კი იცის, თუ რომელ მონაცემთა სერვერს მიმართავს ის. (ეს არც უნდა აწუხებდეს მას).

გამოდის, რომ საჭიროა მონაცემთა სერვერისათვის მოთხოვნის გაგზავნის და მისგან პასუხის მიღების, რაიმე სტანდარტული ხერხის არსებობა, რაღაცა ისეთისა, რაც მოთხოვნების სტანდარტული ენის მაგავარი იქნებოდა. ასეთი ენა შეიქმნა-**SQL-ის (Structured Query Language, მოთხოვნების სტრუქტურირებული ენა)** სახით.

SQL-ენა ჩანაწერების ამოკრეფის, დამატების, შეცვლის და წაშლის მოთხოვნების, სავსებით გააზრებული წინადადებების სახით ჩაწერის საშუალებას იძლევა; ამ თვალსაზრისით ის უახლოვდება ჩვეულებრივ ინგლისურ ენას. მოთხოვნების ჩასაწერად გამოიყენება **გასაღები სიტყვების** საკმაოდ მცირერიცხოვანი ანაკრები-მონაცემთა სერვერისათვის გასაგები განსაკუთრებული ბრძანებების ერთობლივობა.
მაშ ასე, მონაცემთა კლიენტის, სერვერთან ნორმალურად ურთიერთქმედებისათვის, ორივეს უნდა ესმოდეს **SQL** ენა. წარმოვადგინოთ ასეთი ურთიერთქმედების სქემა:

1.მონაცემთა კლიენტი ახდენს მოთხოვნის ფორმირებას SQL ენაზე, რაიმე სახით. (მას შეუძლია თვითონ გააკეთოს ეს, ან სთხოვოს მომხმარებელს, ხელით შეიტანოს მოთხოვნა. სპეციალიზირებული პროგრამები იყენებენ პირველ მიდგომას, ხოლო მონაცემთა ბაზებისადმი წვდომის შესაძლებლობების მქონე სამაგიდე მბმს-ები-ორივეს).

2.მონაცემთა კლიენტი, ფორმირებულ მოთხოვნას გადასცემს მონაცემების სერვერის კლიენტურ ნაწილს, რომელიც კლიენტურ კომპიუტერზეა დაყენებული.

3.კლიენტური ნაწილი მიღებულ მოთხოვნას "ალაგებს" ქსელურ პაკეტებში და გადასცემს მათ მონაცემთა სერვერს.

4.მონაცემთა სერვერი იღებს მოთხოვნას, ახდენს მის დეშიფრაციას და უშვებს შესრულებაზე, რის შემდეგაც, შედეგებს აგზავნის უკან, სერვერის კლიენტურ ნაწილთან.

5.მონაცემთა სერვერის კლიენტური ნაწილი, იღებს შედეგს, "ამოალაგებს" შიგთავსს და უბრუნებს მონაცემების კლიენტს.

6.მონაცემთა კლიენტი იღებს შედეგს და გამოაქვს იგი ეკრანზე, ან ახორციელებს რაიმე მოქმედებას (მაგალითად, მომხმარებელს აცნობებს შეცდომის შესახებ).
SQL ენის გასაღები სიტყვების ძირითდი ანაკრები, მკაცრად არის სტანდარტიზებული, და იგი, მონაცემთა ყველა სერვერის მიერ არის მხარდაჭერილი. მაგრამ, ყოველ სერვერს, თავისი უნიკალური შესაძლებლობების მხარდასაჭერად, შეუძლია შეიტანოს SQL-ში თავისი გასაღები სიტყვებიც, რომელთა მხარდაჭერა მხოლოდ მის მიერ მოხდება.

აღნიშნული უნიკალური შესაძლებლობები და მათი შესაბამისი გასაღები სიტყვები, აღწერილია მონაცემთა სერვერის თანმხლებ დოკუმენტაციაში.

შენიშვნა: *ODBC-ის მსგავს მონაცემთა უნივერსალურ პროცესორებთან სამუშაოდ, ასევე გამოიყენება SQL ენა.*

ქვემოთ, აღწერილი იქნება მონაცემთა დამუშავებისათვის განკუთვნილი SQL-ის გასაღები სიტყვების ძირითადი ანაკრები.

ცხრილებიდან ჩანაწერების ამორჩევა

სიმარტივიდან გამომდინარე, დავიწყოთ მონაცემების შერჩევის მოთხოვნებით. ამავდროულად შევისწავლოთ ზოგიერთი რამ, რაც მომავალში ჩვენ აუცილებლად დაგვჭირდება.

მონაცემების შერჩევის უმარტივესი მოთხოვნები

ცხრილიდან მონაცემების შერჩევის უმარტივესი SQL მოთხოვნის ფორმატი ასეთია:

SELECT {DISTINCT}*|<მძიმეებით გამოყოფილი ველების სია><ცხრილის დასახელება/>

გასაღები სიტყვის **SELECT** შემდეგ, მოდის იმ ველების ჩამონათვალი, რომელთა ამოკრეფაც არის საჭირო ცხრილიდან. ჩამონათვალი შედგება ერთმანეთისაგან მძიმით გამოყოფილი ველების დასახელებებიდან; ამასთან შედეგში, ველები იმავე თანმიმდევრობით იქნებიან წარმოდგენილი, როგორც მოთხოვნაში.

თუ ველების ჩამონათვალის ნაცვლად ჩავსვამთ ვარსკვლავის ნიშანს (*), ამორჩეული იქნება ყველა ველი.
ველების ჩამონათვალს მოსდევს გასაღები სიტყვა **FROM**, რომლის შემდეგ იწერება იმ ცხრილის სახელი, რომლიდანაც ხდება მონაცემების ამოკრეფა. **SQL** მოთხოვნა მთავრდება წერტილ-მძიმის ნიშნით.

ყურადღება! *ზოგიერთი მბმს, მაგალითად MySQL, მოითხოვს, რომ ყველა SQL მოთხოვნა მთავრდებოდეს წერტილ-მძიმით. სხვებისათვის ეს მოთხოვნა აუცილებელი არ არის. ყველა შემთხვევაში, აუცილებელია მონაცემთა კონკრეტული სერვერის თანმხლები დოკუმენტაციის წაკითხვა.*

მოდით განვიხილოთ ჩვენს მიერ შექმნილი ცხრილებიდან-კატეგორიებისა და სტატიების სიები-მონაცემების შერჩევის SQL-მოთხოვნების რამოდენიმე მაგალითი.

SELECT * FROM items;

ეს მოთხოვნა გამოგვიყვანს **items** (სტატიების სია) ცხრილის ყველა ჩანაწერს. ხოლო მოთხოვნა,

SELECT name FROM categories;

დაგვიბრუნებს მხოლოდ ცხრილის-categories (კატეგორიების სია), name ველის მნიშვნელობებს ასეთი სახით-იხ. ნახ.6.7..

SELECT name FROM categories

name
ინტერნეტი
სისტემა
ოფისი
პროგრამირება
ინტერნეტი

ნახ.6.7. SQL მოთხოვნის მიერ დაბრუნებული categories ცხრილის, name ველის მნიშვნელობების სია.

ჩვენ ჯერ არ განგვიხილავს გასაღები სიტყვა **DISTINCT**, რომელიც შეიძლება შეგვხვდეს **SQL** მოთხოვნაში. თუ ეს სიტყვა მითითებულია, მაშინ მხოლოდ უნიკალური ჩანაწერების დაბრუნება ხდება. მაგალითად მოთხოვნა:

SELECT DISTINCT name FROM categories;

დაგვიბრუნებს შედეგს, რომელიც ნაჩვენებია ნახ.6.8.-ზე. ცხადია, რომ ორი **ინტერნეტი** სტრიქონის ნაცვლად, ჩვენ მივიღებთ ერთს.

SELECT DISTINCT name FROM categories

name
ინტერნეტი
სისტემა
ოფისი
პროგრამირება

ნახ.6.8. SQL მოთხოვნის მიერ დაბრუნებული, categories ცხრილის, name ველის უნიკალური მნიშვნელობების სია

მონაცემების დახარისხება

თავდაპირველად მონაცემების შერჩევის SQL მოთხოვნა, ჩანაწერებს აბრუნებს იმ თანმიმდევრულობით, როგორითაც ისინი დამატებული იყვნენ ცხრილში. დახარისხების რიგითობის განსაზღვრისათვის გამოიყენება დამატებითი გასაღები სიტყვა **ORDER BY**, რომელიც მოთხოვნის ბოლოს იწერება:

. . . ORDER BY <მძიმეებით გამოყოფილი კრიტერიუმების სია>

ხოლო, თვით დახარისხების კრიტერიუმებს აქვთ ასეთი სახე:

<იმ ველის დასახელება, რომლის მიხედვითაც ხორციელდება დახარისხება> {DESC}

მაშ ასე, ველები, რომელთა მიხედვითაც უნდა მოხდეს ჩანაწერების დახარისხება, ერთმანეთისაგან მძიმეებით გამოიყოფიან და მათი ჩამოითვლა ხდება **ORDER BY** გასაღები სიტყვის შემდეგ, რომელიც თავის მხრივ, მოთხოვნის ბოლოს, წერტილ-მძიმის წინ იწერება. ამასთან მონაცემთა სერვერი ჩანაწერების დახარისხებას განახორციელებს შემდეგი წესების მიხედვით:

1.თავდაპირველად ჩანაწერების დახარისხება ხდება იმ ველის მიხედვით, რომელიც სიაში პირველად არის მითითებული.
2.თუ რამოდენიმე სხვადასხვა ჩანაწერისათვის ამ ველის მნიშვნელობები ერთნაირია, მაშინ შემდგომში ჩანაწერების დახარისხება ხდება იმ ველის მიხედვით, რომელიც სიაში მეორედ არის მითითებული.
3.თუ, რამოდენიმე სხვადასხვა ჩანაწერში აღნიშნული ველის მნიშვნელობები ისევ დაემთხვევა ერთმანეთს, მაშინ მათი დახარისხება მოხდება იმ ველის მიხედვით, რომელიც სიაში მითითებულია მესამედ და ა.შ..
გაჩუმებით, ჩანაწერების სორტირება ხდება ველების მნიშვნელობების ზრდადობის მიხედვით. თუ საჭიროა მათი დახარისხება მოცემული ველის მნიშვნელობების კლებადობის მიხედვით, ველის სახელის შემდეგ უნდა დავსვათ გასაღები სიტყვა **DESC**.
მოთხოვნა, რომელიც ქვემოთ არის ნაჩვენები, გამოიტანს **items** ცხრილის ყველა ჩანაწერს, რომლებიც დახარისხებული იქნებიან სტატიების ავტორების სახელების მიხედვით:

SELECT * FROM items ORDER BY author;

ხოლო ეს მოთხოვნა categories ცხრილის ჩანაწერებს დაახარისხებს ჯერ **file** ველის, შემდეგ კი **name** ველის, კლებადობის მიხედვით:

SELECT file, name FROM categories ORDER BY file, name DESC;

SELECT file, name FROM categories SORT BY file, name DESC;

file	name
არა	სისტემა
არა	ინტერნეტი
კი	პროგრამირება
კი	ოფისი
კი	ინტერნეტი

ნახ.6.9. SQL მოთხოვნის მიერ დაბრუნებული, categories ცხრილის, file და name ველების მიხედვით დახარისხებული მნიშვნელობების სია.

შედეგად ჩვენ მივიღებთ იმას, რაც გამოსახულია ნახ.6.9.-ზე. მივაქციოთ ყურადღება-შედეგში ველები ნაჩვენებია ზუსტად იმავე თანმიმდევრობით, რა თანმიმდევრობითაც ჩამოვთვალეთ ისინი მოთხოვნაში.

მონაცემების ფილტრაცია

ჩანაწერების ველების მნიშვნელობების მიხედვით ფილტრაციისათვის, გამოიყენება გასაღები სიტყვა **WHERE**. ეს სიტყვა იწერება გასაღებ სიტყვებს შორის **FROM** და **ORDER BY:**

. . . WHERE <მძიმეებით გამოყოფილი ფილტრაციის კრიტერიუმების სია> . . .

ხოლო, თვით ფილტრაციის კრიტერიუმებს გააჩნიათ ჩვეულებრივი პირობების სახე:

<ველის დასახელება><შედარების ოპერატორი><მოცემული მნიშვნელობა>

შედარების ოპერატორი-ეს **SQL**-ის განსაკუთრებული ბრძანებაა, რომელიც განსაზღვრავს მოცემული მნიშვნელობისა და ველის მნიშვნელობის ტოლობას ან უტოლობას:

Id=3

აქ **id** ველის მნიშვნელობა ედრება მოცემულ მნიშვნელობას 3. თუ **id**-ს მნიშვნელობა 3-ის ტოლი იქნება, მაშინ ჩანაწერი მოხვდება მოთხოვნის შედეგში.
SQL-ის სტანდარტში ხელმისაწვდომი ყველა შედარების ოპერატორი ჩამოთვლილია ცხრილში 6.2.

ცხრილი 6.2. SQL-ის სტანდარტში ხელმისაწვდომი შედარების ოპერატორები.

შედარების ოპერატორი	**განმარტება**
=	ტოლობა
<> ან !=	უტოლობა
<	ნაკლებობა
>	მეტობა
<=	ნაკლებობა ან ტოლობა
>=	მეტობა ან ტოლობა

ამგვარად, იმისათვის, რომ ამოვარჩიოთ **მანჯგალაძე ე**.-ს ყველა სტატია, ჩვენ ასეთი მოთხოვნა უნდა ჩავწეროთ:

SELECT * FROM items WHERE author=”მანჯგალაძე ე.”;

როგორც ვხედავთ, **=** ოპერატორის გამოყენება შესაძლებელია აგრეთვე ორი სტრიქონის ერთმანეთთან შესადარებლადაც. შესაბამისად, ოპერატორი **<>** შეიძლება გამოყენებულ იქნას უტოლობის შესამოწმებლად, როგორც ქვემოთ არის ნაჩვენები:

SELECT * FROM items WHERE author<>თარგამაძე მ.”;

ეს მოთხოვნა დაგვიბრუნებს ჩვენ ყველა სტატიას, გარდა იმ სტატიებისა, რომელთა ავტორიც არის თარგამაძე ე..

ყურადღება: *სტრიქონული (სიმბოლური) სიდიდეები, რომლებიც SQL მოთხოვნების ნაწილს წარმოადგენენ ბრჭყალებში უნდა იყოს ჩასმული.*

მაგრამ როგორ მოვიქცეთ, თუ ჩვენ გვჭირდება როგორც მანჯგალაძე ე., ასევეთარგამაძე მ.-ს სტატიები? ამ შემთხვევაში უნდა გამოვიყენოთ განსაკუთრებული **ლოგიკური ოპერატორი-OR**:

SELECT *FROM items WHERE author=”მანჯგალაძე ე.” OR author=” თარგამაძე მ.” ;

ოპერატორი **OR** მონაცემების სერვერს კარნახობს, რომ უნდა შესრულდეს ”**ან**” პირველი ”**ან**” მეორე პირობა (“**OR**”-ის ინგლისურიდან თარგმანი არის-”**ან**”).
ამ შემთხვევაში ჩანაწერი მოხვდება შედეგში.

მეორე ლოგიკური ოპერატორი-**AND** (ინგლისურიდან თარგმანი-”და”)-მოითხოვს, რომ შესრულებული იყოს, როგორც პირველი, ასევე მეორე პირობაც.

SELECT id FROM categories WHERE name=”ინტერნეტი” AND file=true;

ადრე განხილული მოთხოვნა დაგვიბრუნებს **categories** ცხრილის იმ ჩანაწერის **id** ველის მნიშვნელობას, რომლის **name** ველის მნიშვნელობა ტოლია ინტერნეტ-ის, ხოლო ლოგიკური ველის **file** მნიშვნელობა-**true**(“ჭეშმარიტია”).
ბოლო ლოგიკური ოპერატორის-**NOT**-ის ინგლისურიდან თარგმანი ნიშნავს ”არა”-ს. მაგალითად, მოთხოვნა:

SELECT * FROM categories WHERE NOT id=3;

დააბრუნებს ყველა ჩანაწერს, რომელთა **id** ველის მნიშვნელობა ”**არა**” უდრის 3-ს.
შესაძლებელია ლოგიკური ოპერატორების კომბინირებაც. მაგალითად, განვიხილოთ ასეთი მოთხოვნა:

SELECT * FROM items WHERE NOT (author=”მანჯგალაძე ე.” OR author=”თარგამაძე მ.”) ;

ეს მოთხოვნა დააბრუნებს ყველა სტატიას, რომელთა ავტორები არ არიან არც მანჯგალაძე და არც თარგამაძე. დავაკვირდეთ იმას, რომ,

ჩვენ გამოვიყენეთ ფრჩხილები, რომლებიც მონაცემთა სერვერს კარნახობენ ჯერ მოთხოვნის ნაწილის შესრულებას:

author=”მანჯგალაძე ე.” OR author=”თარგამაძე მ.”

ხოლო შემდეგ მის მიმართ ოპერატორ **NOT**-ის გამოყენებას, ვინაიდან მისი შესრულება ხდება **OR** და **AND** ოპერატორების წინ (როგორც პროფესიონალი პროგრამისტები ამბობენ, გააჩნია უფრო მაღალი ”პრიორიტეტი”). თუ ჩვენ არ გამოვიყენებთ ფრჩხილებს:

SELECT * FROM items WHERE NOT author=”მანჯგალაძე ე.” OR author=”თარგამაძე მ.” ;

მაშინ მონაცემთა სერვერი, ჩვენს მოთხოვნას განიხილავს როგორც ”ყველა სტატია, რომელთა ავტორები არიან ან ”არა” ”მანჯგალაძე,” ან ”თარგამაძე”. ანუ ის ჯერ ასრულებს პირობას:

author=”მანჯგალაძე ე.”

შემდეგ გამოიყენებს მის მიმართ ოპერატორს NOT, და მერე შეასრულებს პირობას

author=”თარგამაძე მ.”

დაბოლოს ორივე ამ პირობების მიმართ გამოიყენებს ოპერატორს OR. ამ მოთხოვნის შედეგი სულ სხვანაირი იქნება.
ჩანაწერების ფილტრაციას ჩვენ კიდევ დავუბრუნდებით. ხოლო, ჯერ-ჯერობით განვაგრძოთ მონაცემების შერჩევის SQL მოთხოვნების შესწავლა.

ცხრილებს შორის კავშირების განსაზღვრა.

ცხრილების ერთმანეთთან დასაკავშირებლად და მათგან მონაცემების მისაღებად, გამოიყენება იგივე გასაღები სიტყვა **WHERE**. მხოლოდ იმ განსხვავებით, რომ ამ სიტყვის შემდგომ ჩაწერილ ფილტრაციის კრიტერიუმს გააჩნია ოდნავ განსხვავებული სახე. თუ როგორი-ამას ჩვენ ახლავე ვნახავთ კონკრეტულ მაგალითზე.

დავუშვათ, რომ ჩვენ გვინდა სტატიების ავტორების გვარებთან ერთად, იმ კატეგორიების დასახელებების მიღება, რომელთაც ეს სტატიები მიეკუთვნებიან. ამისათვის ჩვენ დაგვჭირდება **items** (სტატიები) და **categories** (კატეგორიები) ცხრილების დაკავშირება ერთმანეთთან ისე, როგორც ეს ნაჩვენებია ნახ.6.6.-ზე. დავწეროთ ასეთი **SQL** მოთხოვნა:

SELECT items.author, items.name, categories.name FROM items, categories WHERE items.catid=categories.id;

ჩვენთვის უფრო იოლი იქნება მისი ნაწილ-ნაწილ გარჩევა, მარცხნიდან მარჯვენა მიმართულებით.
-გასაღები სიტყვის **SELECT**-ის შემდეგ, როგორც უკვე ვიცით, იწერება ჩვენთვის საჭირო ველების ჩამონათვალი. მაგრამ აქ პატარა პრობლემას ვაწყდებით: ორივე ცხრილს-**items** და **catrgories**-გააჩნიათ ველი **name**. იმისათვის, რომ მონაცემთა სერვერმა იცოდეს, თუ რომელი ცხრილიდან აიღოს ეს ველი, ველის დასახელების წინ უნდა დავწეროთ ცხრილის სახელი, რომელსაც იგი ეკუთვნის და გამოვყოთ ისინი ერთმანეთისაგან წერტილით ასეთნაირად: **items.name** და **categories.name**.

შენიშვნა: *ზოგადად ველის დასახელების წინ იმ ცხრილის დასახელების ჩაწერა, რომელსაც ეს ველი მიეკუთვნება-SQL მოთხოვნების ჩაწერის კარგ სტილად ითვლება. ამიტომ მომავალშიც ასევე მოვიქცევით.*

გასაღები სიტყვის FROM შემდეგ იწერება იმ ორივე ცხრლის დასახელება, რომლებიდანაც ვიღებთ მონაცემებს: items და categories. ეს დასახელებები ერთმანეთისაგან მძიმით უნდა გამოვყოთ.
-**WHERE** გასაღები სიტყვის შემდეგ იწერება ცხრილების ურთიერთდაკავშირების კრიტერიუმი. მას ისეთივე სახე აქვს, რაც ფილტრაციის კრიტერიუმს-**items.catid=categories.id**-და მონაცემთა სერვერს აიძულებს, რომ **items** ცხრილის ყოველი ჩანაწერისათვის, მოძებნოს categories ცხრილის ისეთი ჩანაწერი, რომლის **id** ველის მნიშვნელობა უდრის იმავე **items** ცხრილის ჩანაწერის **catid** ველის მნიშვნელობას.

ამ მოთხოვნის შესრულებისთანავე, მივიღებთ შედეგს, რომელიც ნაჩვენებია ნახ.6.10.-ზე.

SELECT items.author,items.name,categories.name.FROM items, categories WHERE items.catid=categories.id

author	name	name
მანჯგალაძე ე.	დაპროგრამების საკითხები	ინტერნეტი
თარგამაძე მ.	მონაცემთა ბაზების საფუძვლები	ინტერნეტი
აფრიდონიძე გ.	ოპერაციული სისტემების შესახებ	სისტემა
ჯაფარიძე ზ.	კომპიუტერის საიმედოობა	ინტერნეტი

ნახ.6.10. items ცხრილის author და name, და categories ცხრილის name ველების მნიშვნელობათა ჩამონათვალი

როგორც ვიცით, **items** ცხრილში მოთავსებულია როგორც სტატიები, ასევე ფაილებიც. ჩვენ კი გვინდა მხოლოდ სტატიები. მოდით გავასწოროთ ეს შეცდომა მოთხოვნაში პატარა ფრაგმენტის ჩამატებით (**SQL** კოდის დამატებული ფრაგმენტი გამოყოფილია მუქი ფერით):

SELECT items.author, items.name, categories.name FROM items, categories
WHERE itams.catid=categories.id **AND categories.file=false**
ORDER BY categories.name, items.name;

აქ ჩვენ დავუმატეთ აგრეთვე ფილტრაციის კრიტერიუმიც, რომელიც **categories** ცხრილიდან მხოლოდ იმ ჩანაწერებს ამოარჩევს, რომელთა **file** ველის მნიშვნელობა ტოლია **false**-ის (”მცდარი”). ყურადღება მივაქციოთ იმას, რომ მოცემული კრიტერიუმი დაკავშირებულია ლოგიკური ოპერატორის **AND**-ის საშუალებით, ფილტრაციის

კრიტერიუმთან. ეს იმისთვის არის საჭირო, რომ ერთდროულად მოქმედებდეს ორივე კრიტერიუმი: დაკავშირების და ფილტრაციის.
დაბოლოს, ჩვენ დავახარისხეთ შერჩეული ჩანაწერები ჯერ კატეგორიების დასახელებების მიხედვით (ველი-**categories.name**), ხოლო შემდეგ-სტატიების დასახელებების მიხედვით (ველი-**items.name**).

ველების ფსევდონიმები

მოდით კიდევ ერთხელ შევხედოთ ჩვენს მიერ მიღებული ცხრილების ურთიერთდაკავშირების შედეგს (ნახ.6.10). რა არ არის აქ ისე, როგორც უნდა იყოს?
ჩვენ გვაქვს ორი ველი ერთი და იგივე სახელით-**name**. თუ ჩვენ გამოვიყენებთ ამ შედეგს პროგრამაში (ჩვენ სწორედ ამის გაკეთებას ვაპირებთ!), მაშინ შესაძლოა წავაწყდეთ პრობლემას: როგორ გავაგებინოთ პროგრამას, თუ რომელი ველი გვჭირდება? მოდით, რაიმე მოვუხერხოთ ამ ერთნაირ სახელებს!
სპეციალურად ასეთი შემთხვევებისთვის **SQL** ენას გააჩნია შესაძლებლობა, მიანიჭოს ველს ჩვენს მიერ მოფიქრებული სხვა სახელი-ეგრეთ წოდებული **ველის ფსევდონიმი.** ფსევდონიმი იქმნება გასაღები სიტყვის AS საშუალებით ასეთნაირად:

SELECT . . .<ველის დასახელება> AS <ფსევდონიმი> . . .

ყველაფერი ეს იწერება ველების ჩამონათვალში, რომელიც გასაღები სიტყვის SELECT-ის შემდეგ მდებარეობს.

თუ ყველაფერი გასაგებია, მაშინ, მოდით კიდევ ერთხელ გადავწეროთ ჩვენი წინა მოთხოვნა. ის ასეთ სახეს მიიღებს (დამატებული და შეცვლილი **SQL** კოდი გამოყოფილია მუქი ფერით):

```
SELECT items.author, items.name AS item_name, categories.name AS
cat_name
FROM items, categories
WHERE itams.catid=categories.id AND categories.file=false
ORDER BY categories.name, items.name;
```

აქ ჩვენ ველს **items.name**, მივანიჭეთ ფსევდონიმი **item_name** (ცხრილების ველების დასახელებებში ”ჰარები” დაუშვებელია, ამიტომაც ვიყენებთ ”ქვედა ტირეს” ნიშანს), ხოლო **categories.name** ველის-ფსევდონიმი იქნება **cat_name**. ამის შემდეგ, ჩვენი მოთხოვნის შედეგის ყოველ ველს, ექნება უნიკალური სახელი.

SQL-ის აგრეგატული ფუნქციები

SQL-ენის ყველაზე რთული და განსაკუთრებულად აღსანიშნავი შესაძლებლობა-ეს გახლავთ, **აგრეგატული ფუნქციები,** რომლებიც მრავალ ჩანაწერზე, ერთდროულად, სხვადასხვა მოქმედებების შესრულების შესაძლებლობას იძლევიან. მათი საშუალებით, სტატისტიკის შეგროვებასთან დაკავშირებული სამუშაოს მნიშვნელოვანი ნაწილი, შეგვიძლია გადავაბაროთ თვით მონაცემთა სერვერს.
ვთქვათ, გვსურს სტატიების რაოდენობის განსაზღვრა თითოეულ კატეგორიაში. ამისათვის ჩვენ მოგვიწევს ერთ-ერთი აგრეგატული ფუნქციის გამოყენება, რომელიც დაითვლის ჩანაწერების რაოდენობას. მაგრამ ჯერ ჩვენ დაგვჭირდება ზოგიერთი წინასწარი მოქმედებების შესრულება, კერძოდ-ჩანაწერების დაჯგუფება კატეგორიების მიხედვით.

დაჯგუფება-ეს არის ჩანაწერების გაერთიანება ჯგუფებად, რაიმე კრიტერიუმების მიხედვით, რომელთაც **დაჯგუფების კრიტერიუმებს** უწოდებენ. დაჯგუფება ხორციელდება გასაღები სიტყვის **GROUP BY** საშუალებით, რომლის შემდეგაც იწერება თვით დაჯგუფების კრიტერიუმები:

GROUP BY <მძიმეებით გამოყოფილი ველების დასახელებები, რომელთა მიხედვითაც ხორციელდება ჩანაწერების დაჯგუფება>

გასაღები სიტყვა **GROUP BY**, დაჯგუფების კრიტერიუმებთან ერთად, იწერება გასაღები სიტყვის **ORDER BY**-ის წინ.
მაშ ასე, ჩვენ უნდა მივიღოთ სტატიების რაოდენობა თითოეულ კატეგორიაში. ამისათვის საჭიროა ჩანაწერების დაჯგუფება ჯერ კატეგორიების მიხედვით (კერძოდ **categories.name** ველის მიხედვით), რაშიც ჩვენ დაგვეხმარება ასეთი სახის მოთხოვნა:

```
SELECT items.author, items.name AS item_name, categories.name AS
cat_name
FROM items, categories WHERE items.catid=categories.id AND
categories.file=false
ORDER BY categories.name, items.name;
```

ავღნიშნოთ ორი მნიშვნელოვანი მომენტი, რომელთა გათვალისწინება აუცილებელია დაჯგუფების შემცველი მოთხოვნის შედგენისას, სხვანაირად მონაცემთა სერვერი ვერ შეძლებს მის შესრულებას.

-ველები, რომელთა მიხედვითაც ხორციელდება ჩანაწერების დაჯგუფება, უნდა იყვნენ პირველები გასაღები სიტყვის **SELECT**-ის ველების ჩამონათვალში და განთავსებულნი უნდა იყვნენ იმავე თანმიმდევრობით, რა თანმიმდევრობითაც ისინი არიან ჩამოთვლილი გასაღები სიტყვის **GROUP BY** შემდეგ.

-ველები, რომელთა მიხედვითაც ხდება ჩანაწერების დაჯგუფება, პირველები უნდა იყვნენ გასაღები სიტყვის **ORDER BY** ველების ჩამონათვალში და ისევე, როგორც წინა შემთხვევაში, იმავე თანმიმდევრობით უნდა იყვნენ განლაგებულნი, რა თანმიმდევრობითაც ჩამოთვლილი არიან გასაღები სიტყვის **GROUP BY**-ის შემდეგ.

ჩანაწერების დაჯგუფების შემდეგ, ჩვენ აგრეგატული ფუნქციის **COUNT**(<ველის სახელი, რომლის მიხედვითაც ხდება დათვლა>) გამოყენებით, შეგვიძლია შევასრულოთ ჩანაწერების რაოდენობის დათვლა თითოეულ ჯგუფში:

```
SELECT categories.name,
COUNT (items.name)
AS item_count
FROM items, categories
WHERE items.catid=categories.id
GROUP BY categories.name
ORDER BY categories.name;
```

აქ ჩვენ ფუნქციის პარამეტრის (არგუმენტის) სახით ჩავსვით **items** ცხრილის **name** ველის სახელი, ვინაიდან ჩვენ სწორედ ამ ცხრილის ჩანაწერების დათვლა გვჭირდება. ზოგადად, შეიძლებოდა

ნებისმიერი ველის არჩევა, მაგალითად, **author** და **date**-მოცემულ შემთხვევაში ეს პრინციპიალური არ არის.
დაგვრჩა მხოლოდ სტატიების კატეგორიების შერჩევის კრიტერიუმების განსაზღვრა:

SELECT categories.name,
COUNT (items.name)
AS item_count
FROM items, categories
WHERE items.catid=categories.id AND categories.file=false GROUP BY categories.name
ORDER BY categories.name;

საბოლოო მოთხოვნა ასეთ შედეგს დაგვიბრუნებს-იხ.ნახ.6.11.

SELECT categories.name,COUNT(items.name)AS item_count FROM items, categories WHERE items.catid=categories.id AND Categories.file=false GROUP BY categories.name ORDRR BY Categories.name;

name	Item_count
ინტერნეტი	3
სისტემა	1

ნახ.6.11. categories ცხრილის name ველის მნიშვნელობათა სია და თითოეულ კატეგორიაში სტატიების რაოდენობა.

შევნიშნოთ, რომ ჩვენი მოთხოვნის შედეგი შეიცავს **ინტერნეტი** და **სისტემა** კატეგორიების შესაბამის მხოლოდ ორ ჩანაწერს. რა იქნა დანარჩენი ჩანაწერები? დანარჩენი ჩანაწერები მონაცემთა სერვერმა არ გამოიტანა, ვინაიდან **items** ცხრილში არ არის არც ერთი ჩანაწერი, რომელიც მათზე მინიშნებას შეიცავს. დაჯგუფების ეს თავისებურება მხედველობაშია მისაღები.
SQL სტანდარტი განსაზღვრავს რამოდენიმე აგრეგატულ ფუნქციას, რომელთა გამოყენებაც ჩვენ შეგვიძლია მოთხოვნების ფორმირებისას. ყველა მათგანი ჩამოთვლილია ცხრილში 6.3.. მათი გამოყენება ხდება მხოლოდ რიცხვითი ველების მიმართ, გარდა ჩვენთვის უკვე ნაცნობი ფუნქციისა **COUNT.**

ცხრილი 6.3. SQL ენაში ხელმისაწვდომი აგრეგატული ფუნქციები

აგრეგატული ფუნქცია	განმარტება
COUNT (<ველი>)	ჩანაწერების რაოდენობა
SUM (<ველი>)	მოცემული ველის მნიშვნელობათა ჯამი, ჯგუფში შემავალ ყველა ჩანაწერში
AVG(<ველი>)	მოცემული ველის მნიშვნელობების საშუალო მნიშვნელობა, ჯგუფში შემავალი ყველა ჩანაწერისათვის
MIN(<ველი>)	მოცემული ველის მნიშვნელობებს შორის ყველაზე მინიმალური, ჯგუფში შემავალი ყველა ჩანაწერისათვის
MAX(<ველი>)	მოცემული ველის მნიშვნელობებს შორის ყველაზე მაქსიმალური, ჯგუფში შემავალი ყველა ჩანაწერისათვის

ამით დავასრულოთ საუბარი მონაცემების შერჩევის განმახორციელებელი **SQL** მოთხოვნების შესახებ. ჩვენ განვიხილეთ ყველაფერი ის, რაც შეიძლება დაგვჭირდეს საწყის ეტაპზე. სრულად **SQL** ენა აღწერილია მონაცემთა სერვერის თანმხლებ დოკუმენტაციაში.

ცხრილებში ჩანაწერების შეცვლა

ამის შემდეგ გადავიდეთ მონაცემების შეცვლის-ჩანაწერების დამატების, ჩანაცვლების და წაშლის მოთხოვნებზე.

ჩანაწერის დამატება

ცხრილში ჩანაწერის დამატების SQL მოთხოვნა-მონაცემების შეცვლის მოთხოვნებს შორის ყველაზე მარტივია. ის იქმნება გასაღები სიტყვის **INSERT INTO** საშუალებით:

INSERT INTO <ცხრილის დასახელება> {<მძიმეებით გამოყოფილი ველების დასახელებები>}VALUES {<მძიმეებით გამოყოფილი ველების მნიშვნელობები>} ;

აქ გასაღები სიტყვის **INSERT INTO**-ს შემდეგ იწერება იმ ცხრილის სახელი, რომელშიც ხდება ახალი ჩანაწერის დამატება. შემდეგ ხდება ბრჩხილებში ახალი ჩანაწერის იმ ველების სახელების ჩამოთვლა, რომლებშიც მნიშვნელობები უნდა მოთავსდეს. თვით ამ მნიშვნელობების ჩამოთვლა ხდება ბრჩხილებში, იმავე თანმიმდევრობით, რა თანმიმდევრობითაც ველებია ჩამოთვლილი გასაღები სიტყვის **VALUES** შემდეგ (ეს მნიშვნელოვანია!).
ქვემოთ ნაჩვენებია ცხრილში **items** ახალი ჩანაწერის დამატების მაგალითი:

INSERT INTO items (name, author) VALUES (“პროგრამები, რომლებიც არაფერს არ აკეთებენ”, ”ფილფანი”);

ჩანაწერის ჩანაცვლება

ჩანაწერის ჩანაცვლების მოთხოვნა მონაცემების შეცვლის ყველა მოთხოვნებს შორის ურთულესია. ის აიგება გასაღები სიტყვის **UPDETE** საშუალებით:

UPDATE <ცხრილის დასახელება> SET <1-ველის დასახელება>=<1-ველის ახალი მნიშვნელობა>,<მე-2 ველის დასახელება>=<მე-2 ველის ახალი მნიშვნელობა> . . .
WHERE <შესაცვლელი ჩანაწერის მოსაძებნად საჭირო ფილტრაციის კრიტერიუმი> ;

გასაღები სიტყვის UPDETE შემდეგ იწერება ცხრილის სახელი, რომლის ჩანაწერიც უნდა შეიცვალოს. შემდეგ მოდის გასაღები სიტყვა SET, ხოლო მის მერე-ერთმანეთისაგან მძიმეებით გამოყოფილი შემდეგი სახის წყვილების ანაკრები

”ველის დასახელება=მის ახალ მნიშვნელობას”.

ბოლოს იწერება მონაცემების შერჩევის მოთხოვნებიდან, ჩვენთვის კარგად ნაცნობი გასაღები სიტყვა **WHERE**.

ახლა კი იწყება ყველაზე საინტერესო რამ.
იმისათვის, რომ შევცვალოთ რომელიმე ჩანაწერი, მონაცემთა სერვერმა ის უნდა მოძებნოს. იმისთვის, რომ მოძებნოს, ჩვენ გასაღები სიტყვა **WHERE**-ს შემდეგ, უნდა განვსაზღვროთ ფილტრაციის ისეთი კრიტერიუმი, რომ მან ცალსახად მიგვითითოს ჩვენთვის საჭირო ჩანაწერზე. ასეთი კრიტერიუმის განსაზღვრა ორნაირად არის შესაძლებელი.
ჯერ-ერთი, ჩვენ შეგვიძლია მოვძებნოთ ჩანაწერი მისი ყველა ველის ძველი მნიშვნელობების მიხედვით. ქვემოთ ნაჩვენებია მოთხოვნის მაგალითი, რომელიც ასეთ მიდგომას იყენებს:

```
UPDATE items SET name=”Web-დიზ”
WERE name=”დაპროგრამების საკითხები”
AND author=”მანჯგალაძე ე.” AND date=(12.08.2011) ;
```

ცხადია, რომ ასეთი მოთხოვნა ძალიან გადატვირთულია, ამავე დროს ძალიან ნელა სრულდება, ვინაიდან მონაცემთა სერვერს ძებნის განხორციელება უწევს ცხრილის ყველა ველის მიხედვით. მაგრამ, თუ ცხრილი არ შეიცავს ველებს, რომლებიც ცალსახად განსაზღვრავენ ჩანაწერს, ეს ჩანაწერის შეცვლის მოთხოვნის აგების ერთად-ერთი ხერხია.

შენიშვნა: *თარიღების მნიშვნელობების განსაზღვრა SQL მოთხოვნებში სხვადასხვანაირად შეიძლება მოხდეს და იგი კონკრეტულ სერვერზეა დამოკიდებული. ჩვენ ავირჩიეთ ხერხი, როდესაც თარიღის მნიშვნელობა ჩასმულია ბრჩხილებში.*

მეორე ხერხი-ჩვენ შეგვიძლია ჩანაწერის ძებნის გამოყენება ველის მიხედვით, რომელიც ცალსახად ახდენს მის იდენტიფიცირებას. ასეთ ველს შეიძლება გააჩნდეს მაგალითად, მთვლელის ტიპი (როგორც ჩვენი ცხრილი **categories**-ის შემთხვევაში)

```
UPDATE categories SET name=”Internet” WHERE id=1 ;
```

ასეთი მოთხოვნა გაცილებით კომპაქტურია იმის გამო, რომ გამოიყენება ძებნა მხოლოდ ერთი **id** ველის მიხედვით. გარდა ამისა, ვინაიდან ამ ველის საფუძველზე შექმნილია გასაღები ინდექსი, ძებნა ძალიან სწრაფად ხდება.

ზემოაღნიშნულიდან ერთი ძალიან მნიშვნელოვანი დასკვნა გამომდინარეობს. მონაცემთა სერვერულ ბაზაში ყოველ ცხრილს, შეძლებისდაგვარად, ერთი ველი მაინც უნდა გააჩნდეს, რომელიც ცალსახად მოახდენს ყოველი მისი ჩანაწერის იდენტიფიცირებას. ეს საშუალებას მოგვცემს შევამციროთ **SQL** მოთხოვნის ზომები და დავაჩქაროთ ძებნა.
ჩავინიშნოთ, რომ ცხრილში **items** უნდა დავამატოთ მთვლელის ტიპის მქონე id ველი.

ჩანაწერის წაშლა

ჩანაწერის წაშლის მოთხოვნა, ასევე ძალზე მარტივია. ის იქმნება გასაღები სიტყვის **DELETE FROM** საფუძველზე:

DELETE FROM <ცხრილის დასახელება>
WHERE <წასაშლელი ჩანაწერის მოსაძებნად საჭირო ფილტრაციის კრიტერიუმი> ;

აქ ყველაფერი ისევეა, როგორც ჩანაწერის შეცვლის მოთხოვნაში: უნდა მიეთითოს იმ ცხრილის სახელი, რომლიდანაც ხდება ჩანაწერის წაშლა, და მისი მოძებნისათვის საჭირო ფილტრაციის კრიტერიუმები. ქვემოთ ნაჩვენებია ცხრლი **categories**-იდან ჩანაწერის ამომშლელი მოთხოვნის მაგალითი:

DELETE FROM categories WHERE id=3 ;

SQL-ის სხვა მოთხოვნები

ჩვენ ახლახან განვიხილეთ **SQL**-ის ზოგიერთი მოთხოვნები, რომლებიც მონაცემების შერჩევისა და შეცვლისათვის იყვნენ განკუთვნილნი. მაგრამ, ამით **SQL**-ის შესაძლებლობები არ ამოიწურება. მის შემადგენლობაში არიან ისეთი მოთხოვნებიც, რომლებიც მონაცემებზე სხვა მოქმედებებსაც ახორციელებენ. ჩვენ მათ შემდეგში განვიხილავთ.
ზოგადად, **SQL**-ის მოთხოვნები შეიძლება დაყოფილ იქნან სამ ჯგუფად:

-მონაცემების მართვის მოთხოვნები. აქ შედიან ჩვენს მიერ უკვე განხილული მონაცემების შერჩევის ყველა მოთხოვნები, მათ შორის, ჩანაწერების დამატების, ჩანაცვლების და წაშლის მოთხოვნები.
-მონაცემების განსაზღვრის მოთხოვნები. ეს არის მონაცემთა ბაზების, ცხრილების, ინდექსების, კავშირების და სხვათა შექმნის, შეცვლის და წაშლის მოთხოვნები. ჩვენ მათ არ განვიხილავთ, ვინაიდან ჩვენს ამოცანებში არ შედის ასეთი მოთხოვნების შექმნა.
-დამხმარე მოთხოვნები. ასრულებენ სხვადასხვა ტექნიკურ ამოცანებს: მონაცემთა ბაზების გამოყენების შესახებ სტატისტიკური მონაცემების შეგროვება, სარეზერვო კოპირება და სხვა. დამხმარე მოთხოვნებიდან ზოგიერთს ჩვენ შემდგომში განვიხილავთ ამ წიგნის ფარგლებში.

ჯერ-ჯერობით ამით დავასრულოთ **SQL** ენის განხილვა. ცხადია, ჩვენ ვერ გავეცანით ყველა მის შესაძლებლობას; მათი რაოდენობა ძალიან დიდია, ამასთან ბევრი მათგანი ვერც გამოგვადგება ჩვენ მოცემულ ეტაპზე. ზოგიერთ რამეს ჩვენ დამატებით შევისწავლით მერე, როდესაც დავიწყებთ სერვერული პროგრამების წერას. **SQL** ენის სრული აღწერა ყველა მისთვის დამახასიათებელი თავისებურებების გათვალისწინებით, მოცემულია მონაცემთა სერვერების დოკუმენტაციაში.

წვდომის უფლებების განაწილება. უფლებები.

ახლა ჩვენ გვჭირდება მონაცემთა სერვერებთან (და საერთოდ ყველა სერვერულ პროგრამებთან) დაკავშირებული, კიდევ ერთი მნიშვნელოვანი საკითხის განხილვა. ეს არის ეგრეთ წოდებული **წვდომის (შეღწევის) უფლებების განაწილება:** ანუ სერვერულ პროგრამაში დარეგისტრირებული მომხმარებლების, მათი სახელების და პაროლების, აგრეთვე მომხმარებლებისათვის სხვადასხვა ოპერაციების შესრულებისათვის მისანიჭებელი უფლებების სისტემის განხილვა.
ჩვენთვის უკვე ცნობილია, რომ ყოველ პროგრამა-სერვერს, იქნება ეს **FTP**-სერვერი, **WEB**-სერვერი თუ ელექტრონული ფოსტის სერვერი, გააჩნია იმ მომხმარებლების სია, რომელთაც შეუძლიათ მასთან მუშაობა. (ამას ჩვენ წავაწყდით მაშინ, როდესაც ვტვირთავდით ჩვენს ვებ-საიტს უფასო ვებ-სერვერზე. ჩვენ მაშინ მოგვიწია დარეგისტრირება, შემდეგ კი სახელისა და პაროლის შეტანა). როგორც წესი, მომხმარებლებს რომლებიც ამ სიაში არ არიან

დაფიქსირებულნი, საერთოდ არ შეუძლიათ სერვერთან მიერთება. თუმცა ამ წესიდან გამონაკლისიც არსებობს. მრავალი სერვერული პროგრამა, იგივე **Web**- და **FTP**-სერვერები, იძლევიან ეგრეთ წოდებული **ანონიმური წვდომის** შესაძლებლობას, როდესაც სიაში დაურეგისტრირებელ მომხმარებლებს შეუძლიათ ჩაერთონ სერვერში და იმუშაონ მასთან. კერძოდ, ყველა ვებ-სერვერი უშვებს ანონიმურ წვდომას, სხვა შემთხვევაში ჩვენ ნებისმიერ საიტზე შესასვლელად დაგვჭირდებოდა ჯერ დარეგისტრირება, შემდეგ კი საკუთარი სახელისა და პაროლის შეტანა. რათქმა უნდა დაურეგისტრირებელი მომხმარებლების უფლებები ამ შემთხვევაში ძალიან შეზღუდულია-ძირითადად მხოლოდ დათვალიერება არის ნებადართული.

საფოსტო სერვერები და მონაცემთა სერვერები პირიქით, იძლევიან მხოლოდ ავტორიზებული წვდომის შესაძლებლობას, მომხმარებლების სიაში რეგისტრაციით და სახელისა და პაროლის შეტანით. ეს იმისთვის კეთდება, რომ მნიშვნელოვანი მონაცემები უცხო ადამიანების ხელში არ მოხვდნენ. სწორედ ავტორიზებული წვდომის და მონაცემთა სერვერებში მისი რეალიზაციის საკითხებს მიეძღვნება ჩვენი შემდგომი საუბარი.

მაშ ასე, ყოველ მონაცემთა სერვერს (ისევე, როგორც ნებისმიერ სხვა სერვერს, რომელიც ავტორიზებული წვდომის საშუალებას იძლევა) გააჩნია დარეგისტრირებულ მომხმარებელთა სია, რომელთაც ნებადართული აქვთ მონაცემთა ბაზებთან მიერთება და მათი გამოყენება. ამ სიას აწარმოებს ადამიანი, რომელიც ემსახურება მონაცემთა სერვერს-**სერვერის ადმინისტრატორი:** ის საჭიროების შემთხვევაში ამატებს ახალ მომხმარებლებს, შლის ძველებს, ცვლის ცნობებს მომხმარებლების შესახებ. ბაზის გახსნის ნებისმიერი მცდელობისას, სერვერი მოითხოვს მომხმარებლისაგან მის სახელსა და პაროლს, და მათი მიღების შემთხვევაში ამოწმებს, აქვს თუ არა ამ მომხმარებელს მოცემულ ბაზაში წვდომის უფლება. თუ აქვს, სერვერი აძლევს მას ბაზაში წვდომის შესაძლებლობას, თუ არა-უგზავნის შეტყობინებას არასწორი ჩართვის შესახებ.

მაგრამ, აქ ყველაფერი არც თუ ისე მარტივად ხდება. საქმე იმაშია, რომ მომხმარებლის სახელისა და პაროლის გარდა, მონაცემთა სერვერი ინახავს აგრეთვე მონაცემებს იმის შესახებ, თუ კერძოდ რომელ მონაცემთა ბაზებთან და ცხრილებთან გააჩნია მომხმარებელს წვდომის შესაძლებლობა და თუ, რა მოქმედებების

შესრულება შეუძლია მათზე. ამ ცნობებს უწოდებენ **წვდომის უფლებებს.**
უფლებები, რომლებიც მხარდაჭერილია მონაცემთა სერვერების მიერ, შეიძლება დაიყოს ოთხ ჯგუფად:

-უფლებები ბაზის ცხრილებიდან მონაცემების შერჩევაზე (ე.წ. **უფლებები წაკითხვაზე**);

-უფლებები მონაცემების შეცვლაზე-ჩანაწერების დამატება, შეცვლა, წაშლა **(უფლებები ჩაწერაზე**);

-უფლებები მონაცემთა ბაზების, ცხრილების, ინდექსების, კავშირების და ა.შ. შექმნაზე **(მონაცემთა ბაზების ადმინისტრატორის უფლებები).**

-უფლებები დამატებითი ოპერაციების შესრულებაზე-მონაცემთა ბაზების გამოყენების შესახებ სტატისტიკური მონაცემების მიღება, სარეზერვო კოპირება და სხვა **(სერვერის ადმინისტრატორის უფლებები).**

შეიძლება სხვადასხვა ჯგუფიდან უფლებების ცალცალკე მინიჭება. ასე მაგალითად, შეიძლება მომხმარებელს **user4257**, მიენიჭოს უფლება მხოლოდ ჩაწერაზე, მაგრამ არ მიეცეს წაკითხვის უფლება.
ცალცალკე შეიძლება მინიჭებულ იქნას შემდეგი უფლებები:

-მონაცემთა სერვერის ყველა ბაზაზე;
-ცალკეულ მონაცემთა ბაზაზე;
-ცალკეულ ცხრილზე;
-ცხრილების ცალკეულ ველებზე.

მაგალითად, იმავე მომხმარებელს სახელით **user4257** შეიძლება გააჩნდეს **categories** ცხრილის წაკითხვის უფლება, **items** ცხრილში ჩაწერისა და მისგან წაკითხვის უფლება, ხოლო **items.date** ველში მხოლოდ ჩაწერის უფლება.
ყველაფერი რაზეც ვისაუბრეთ, ჩვენ დაგვჭირდება მაშინ, როდესაც დავიწყებთ ჩვენი ახალი საიტისათვის მონაცემთა ბაზის შექმნას. ახლა კი დავასრულოთ საუბარი ზოგადად მონაცემების სერვერული

ბაზების შესახებ, და უფრო დაწვრილებით გავეცნოთ კონკრეტულ მონაცემთა სერვერს, რომელსაც **MySQL**-ეწოდება.

მონაცემთა სერვერი MySQL და მისი შესაძლებლობები

MySQL-ეს მონაცემთა პოპულარული სერვერია, რომელსაც ვებ-საიტების შექმნისას იყენებენ. ჩვენც მას გამოვიყენებთ. რატომ?

- **MySQL**-ძალიან სწრაფი და კომპიუტერის რესურსებისადმი ნაკლებად მომთხოვნი სერვერია.
- **MySQL**-ის შესაძლებლობები გაცილებით მეტია, ვიდრე ეს საჭიროა ვებ-საიტების შესაქმნელად. რათქმა უნდა კონკურენტი სერვერები უფრო მძლავრნი არიან და დიდი ვებ-პორტალების დამპროექტებლები სწორედ მათ ანიჭებენ უპირატესობას, მაგრამ მცირე ზომის საიტებისათვის სრულებით საკმარისია **MySQL**.
- **MySQL** ვრცელდება უფასოდ, გარდა ამისა-მისი საწყისი კოდები ღიაა შესწავლისა და სრულყოფისათვის.
-დაბოლოს-**MySQL** ბრწყინვალედ მუშაობს აქტიური სერვერული ვებ-გვერდების შექმნის **PHP** ტექნოლოგიასთან, რომელიც ჩვენ **მე-5 თავში** ავირჩიეთ ჩვენი საიტის ასაგებად.

იმისათვის, რომ შევამოწმოთ ჩვენი საიტი, ჩვენს კომპიუტერზე უნდა დავაყენოთ **MySQL**. თუ როგორ უნდა განვახორციელოთ ეს, აღწერილია ამ წიგნის **მე-2 დანართში.**
ჯერ, მოდით, მოკლედ "გადავავლოთ თვალი" MySQL-ის შესაძლებლობებს, რათა ვიცოდეთ, რა შეუძლია მას და რა არა. ამავე დროს "დავაკავშიროთ" მისი შესაძლებლობები იმასთან, რაც უკვე ვისწავლეთ ამ თავში.
პრინციპში, **MySQL**-ის შესაძლებლობები, მონაცემთა სხვა სერვერების შესაძლებლობების ანალოგიურია. MySQL-ის მიერ მხარდაჭერილია **SQL** მოთხოვნები, მონაცემთა ბაზებთან მომხმარებლების წვდომა, ინდექსები, მონაცემთა ტიპების სიმრავლე და სხვა. ეს ყველაფერი საკმარისია ჩვენი საიტის ასაგებად.

MySQL-ის მიერ მხარდაჭერილი ზოგიერთი მონაცემთა ტიპები ჩამოთვლილია ცხრილში 6.4.. მათი რაოდენობა დიდია (ხოლო მხარდაჭერილი მონაცემთა ტიპების სრული სია კიდევ უფრო მეტია).

ცხრილი 6.4. MySQL-ის მიერ მხარდაჭერილი მონაცემთა ზოგიერთი ტიპები.

მონაცემთა ტიპები	აღნიშვნა MySql-ში	შენიშვნა
სტრიქონული	VARCHAR	სტრიქონები, რომელთა სიგრძე იცვლება 1-დან 255-სიმბოლომდე. სტრიქონის სიგრძე განისაზღვრება ველის შექმნისას.
მთელრიცხვა	SMALLINT	მთელი რიცხვები -32768-დან 32767-მდე ან 0-დან 65535-მდე
	MEDIUMINT	მთელი რიცხვები -8388608-დან +8388607-მდე ან 0-დან 16777215-მდე
	INT	მთელი რიცხვები -2147483648-დან +2147483647-მდე ან 0-დან 4294967295-მდე
	BIGINT	მთელი რიცხვები -9223372036854775808-დან +9223372036854775807-მდე ან 0-დან 18446744073709551615-მდე
მცოცავი მძიმით	FLOAT	რიცხვები მცოცავი მძიმით $-3,402823466*10^{38}$-დან $-1,175494351*10^{-38}$-მდე და $+1,175494351*10^{-38}$-დან $+3,402823466*10^{38}$-დან
	DOUBLE	რიცხვები მცოცავი მძიმით

		-1,7976931348623157*10^{308}-დან -2,2250738585072014*10^{-308}-მდე და +2,2250738585072014*10^{-308}-დან +1,7976931348623157*10^{308}-მდე
ლოგიკური	BOOL	ლოგიკური სიდიდე ”true” (ჭეშმარიტი) ან ”false” (მცდარი)
თარიღი	DATE	თარიღის მნიშვნელობა 01.01.1000-დან 31.12.9999-მდე
თარიღი და დრო	DATETIME	თარიღისა და დროის გაერთიანებული მნიშვნელობები 01.01.1000 00:00:00-დან 31.12.9999 23:59:59-მდე
მემო (Memo)	BLOB	ცვლადი სიგრძის სტრიქონები. იტევენ 65535-მდე სიმბოლოებს.
	MEDIUMBLOB	ცვლადი სიგრძის სტრიქონები.იტევენ 16777215-მდე სიმბოლოებს.
	LONGBLOB	ცვლადი სიგრძის სტრიქონები.იტევენ 4294967295-მდე სიმბოლოებს.

მონაცემთა ტიპების გარდა, ველებს გააჩნიათ სხვა დამატებითი პარამეტრებიც, რომელთაც **ველების ატრიბუტებს** უწოდებენ. ასე მაგალითად, ატრიბუტი **UNSIGNED**, რიცხვით ველებს უკრძალავს უარყოფითი მნიშვნელობების მიღებას. ატრიბუტი **AUTO_INCREMENT** ჩვეულებრივ მთელრიცხვა ველს გარდაქმნის

მთვლელის ველად (**MySQL**-ის მთვლელისათვის ცალკე მონაცემთა ტიპი განსაზღვრული არ არის). ხოლო, ატრიბუტი **DEFAULT** საშუალებას გვაძლევს ველისათვის განვსაზღვროთ გაჩუმებითი მნიშვნელობა, ანუ იგი ფაქტიურად ველისათვის გარკვეულ წესს განსაზღვრავს.

აქ კიდევ ერთი რამე უნდა ითქვას. ახალი ჩანაწერის შექმნისას, ყველა ველებს, რომელთათვისაც მნიშვნელობები არ იყო მინიჭებული (არა აქვს მნიშვნელობა, ჩანაწერის დამატების მოთხოვნის საშუალებით თუ **DEFAULT** ატრიბუტით განსაზღვრული გაჩუმებითი მნიშვნელობა), ენიჭებათ განსაკუთრებული მნიშვნელობა **NULL**. ეს იმის მაჩვენებელია, რომ ველი საერთოდ არ შეიცავს არავითარ მონაცემებს. თუ საჭიროა, რომ ეს ველი აუცილებლად შეიცავდეს მნიშვნელობას, ანუ მომხმარებელს მოეთხოვებოდეს ამ ველში ყოველთვის მნიშვნელობის შეტანა, მას უნდა მიენიჭოს ატრიბუტი **NOT NULL**.

MySQL-ის მიერ მხარდაჭერილი სხვადასხვა წესების სიმრავლე ძალზე შთამბეჭდავია-შესაძლებელია უფლებების ცალ-ცალკე მინიჭება სხვადასხვა სახის SQL მოთხოვნების შესრულებაზე. ასე მაგალითად, იმისათვის, რომ მომხმარებელს მიეცეს **SELECT** მოთხოვნის საშუალებით ცხრილიდან მონაცემების ამოკრეფის შესაძლებლობა, საჭიროა **წესების ატრიბუტის** გამოყენება, რომელსაც ასევე-**SELECT** ეწოდება. შესაბამისად, ჩანაწერების დამატების უფლებას იძლევა ატრიბუტი **INSERT**, შეცვლისას-**UPDATE**, ხოლო წაშლისას-**DELETE**. ასევე შეიძლება უფლებების მინიჭება ცხრილების და ინდექსების შექმნაზე, დამხმარე ოპერაციების შესრულებაზე და ა.შ.. ასე, რომ წვდომის უფლებების განაწილებასთან დაკავშირებით **MY SQL**-ში ყველაფერი რიგზეა.

მომხმარებლის სახელისა და პაროლის გარდა, **MY SQL** ასევე იძლევა იმ კომპიუტერის ინტერნეტ-მისამართის განსაზღვრის საშუალებას, რომლიდანაც მოცემულ მომხმარებელს შეუძლია სერვერთან დაკავშირება. ფაქტიურად კომპიუტერის ინტერნეტ-მისამართი **MySQL**-ში წარმოადგენს მომხმარებლის სახელის ნაწილს, რომელიც მოცემულ შემთხვევაში ასე ჩაიწერება:

<მომხმარებლის სახელი>@<კომპიუტერის ინტერნეტ-მისამართი>

ანუ, ისევე როგორც ელექტრონული ფოსტის მისამართი. მაგალითად:

root@localhost

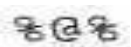
მომხმარებელ **root**-ს აძლევს უფლებას დაუკავშირდეს მონაცემთა სერვერს მხოლოდ ლოკალური კომპიუტერიდან (ინტერნეტ-მისამართია-**localhost**).
წინსწრებით უნდა ავღნიშნოთ, რომ მომხმარებელი **root** ჩვეულებრივად სერვერის ადმინისტრატორს წარმოადგენს და შესაბამისად მას მაქსიმალური უფლებები გააჩნია.
ხოლო მომხმარებელს **remout_user** შეუძლია დაუკავშირდეს სერვერს მხოლოდ **dev.domain.ge** კომპიუტერიდან და სხვა არც ერთიდან (თუნდაც ლოკალურიდან).

remote_user@dev.domain.ru

თუ საჭიროა მომხმარებლისათვის ნებისმიერი კომპიუტერიდან დაკავშირების შესაძლებლობის მიცემა, საჭირო იქნება ინტერნეტ-მისამართის ნაცვლად % შაბლონის ჩასმა, რომელიც განსაზღვრავს ნებისმიერ ინტერნეტ-მისამართს. მაგალითად:

Travelling_user@%

მომხმარებელს **travelling_user**, სერვერთან დაკავშირება შეუძლია ნებისმიერი კომპიუტერიდან-ლოკალურიდანაც და დაშორებულიდანაც.
შაბლონის % გამოყენება შესაძლებელია მომხმარებლის სახელის ნაცვლადაც; მაშინ ის განსაზღვრავს ნებისმიერ მომხმარებელს. ასე მაგალითად, თუ ჩავწერთ **%@localhost**
მაშინ ლოკალური კომპიუტერიდან სერვერთან დაკავშირება შეეძლება ნებისმიერ მომხმარებელს (ნებისმიერი სახელით, თუნდაც ის ცხადად არ იყოს მომხმარებლების სიაში მითითებული). თუ კი ასე ჩავწერთ



მაშინ, სერვერთან დაკავშირება შეეძლება ნებისმიერ მომხმარებელს, ნებისმიერი კომპიუტერიდან. რათქმა უნდა ასეთ მომხმარებელს მინიმალური უფლებები უნდა განესაზღვროს, წინააღმდეგ შემთხვევაში მას შეეძლება მონაცემებზე ნებისმიერი მოქმედების შესრულება, რაც მოესურვება.

MySQL-ის ყველა შესაძლებლობა სრულად არის აღწერილი ელექტრონულ დოკუმენტაციაში, რომლის გადმოტვირთვა შესაძლებელია **MySQL**-ის შემქმნელების საიტიდან- **http://www.mysql.com.**

ჩვენ კი შევუდგეთ ჩვენი საიტისათვის მონაცემთა ბაზის შექმნას. (იგულისხმება, რომ ჩვენ უკვე დავაყენეთ ჩვენს კომპიუტერზე **MySQL** და ყველა თანმხლები პროგრამები. თუ კი არა, მაშინ **MySQL**-ის და სხვა დამხმარე პროგრამების დაყენებისა და გამართვის პროცესები აღწერილია ამ წიგნის მე-2 და მე-4 დანართებში შესაბამისად).

ვქმნით მონაცემთა ბაზას ჩვენი საიტისათვის.

პრინციპში ჩვენი საიტისათვის მონაცემთა ბაზა ჩვენ უკვე პრაქტიკულად შექმნილი გვაქვს. ჩვენ სულ ორი ცხრილი დაგვჭირდება: კატეგორიების ჩამონათვალი **categories** და სტატიებისა და ფაილების სია **items.** კატეგორიების ჩამონათვალი ნაჩვენებია ნახ.6.4.-ზე, ხოლო სტატიებისა და ფაილების სია-ნახ.6.1.-ზე. ჩვენ დაგვრჩა მათში მხოლოდ ზოგიერთი ცვლილებების შეტანა.

პირველი, რასაც ჩვენ გავაკეთებთ-დავამატებთ ახალ ველს სტატიებისა და ფაილების სიის ცხრილში **items**. ამ ველს დავარქვათ **id**, და მივანიჭოთ მას მონაცემთა ტიპი-მთვლელი. ეს ველი საშუალებას მოქვცემს ზუსტად მივუთითოთ ცხრილის იმ ჩანაწერზე, რომელიც გვჭირდება, რაც ძალიან გამოგვადგება ჩანაწერების შეცვლისა და წაშლის დროს.

ახლა შევხედოთ **date** ველს, უფრო ზუსტად, ამ ველის სახელს. ის ემთხვევა გასაღებ სიტყვას **DATE,** რომელიც გამოიყენება თარიღის მონაცემთა ტიპის განსაზღვრისას, ცხრილის შექმნის SQL მოთხოვნაში. **SQL**-ის წესების მიხედვით, ველების, ცხრილების და ინდექსების სახელები არ უნდა ემთხვეოდნენ გასაღებ სიტყვებს. ამიტომ შევცვალოთ ამ ველის სახელი **added**-ით.

შეტანილი ცვლილებების გათვალისწინებით, **items** ცხრილის საბოლოო სტრუქტურა ნაჩვენებია ცხრილში 6.5. იქ აგრეთვე,

ველებისათვის ნაჩვენებია **MySQL**-ის მიერ მხარდაჭერილი მონაცემთა რეალური ტიპები-ასე უფრო მარტივდება ცხრილის შექმნა.

ცხრილი 6.5. items ცხრილის სტრუქტურა

ველის დასახელება	**ველის აღწერა**	**ველის მონაცემების ტიპი**	**მონაცემების ტიპის აღწერა**
id	ჩანაწერის იდენტიფიკატორი	SMALL UNSIGNED	მთელი რიცხვები 0-დან 65535-მდე. სტატიებისა და ფაილების რაოდენობა ცოტა გვექნება, ამიტომ ეს საკმარისია)
author	სტატიის ან ფაილის ავტორი	VARCHAR(30)	30-სიმბოლოიანი ფიქსირებული სიგრძის მქონე სტრიქონი
name	სტატიის ან ფაილის დასახელება	VARCHAR(60)	60-სიმბოლოიანი ფიქსირებული სიგრძის მქონე სტრიქონი
added	სტატიის დამატების თარიღი	DATE	თარიღი
href	ფაილის ან სტატიის ინტერნეტ მისამართი	VARCHAR(255)	255-სიმბოლოიანი ფიქსირებული სიგრძის მქონე

			სტრიქონი (სწორედ ასეთია ინტერნეტ მისამართის სიგრძე)
catid	მინიშნება კატეგორიაზე (categories ცხრილის შესაბამისი ჩანაწერის id ველის მნიშვნელობა)	SMALLINT UNSIGNED	მთელი რიცხვები 0-დან 65535-მდე

ამ ცხრილის **id** ველის საფუძველზე შევქმნათ გასაღები ინდექსი. აგრეთვე შევქმნათ კიდევ ორი ინდექსი: ერთი-**added** ველის საფუძველზე, ხოლო მეორე-**catid** ველის საფუძველზე. დავარქვათ მათ შესაბამისად **added** და **catid**. ეს ინდექსები დაგვეხმარებიან თარიღის მიხედვით დახარისხების დაჩქარებაში და მოცემულ კატეგორიას მიკუთვნებული ჩანაწერების მოძებნაში.
სამწუხაროდ, **MySQL**-ს ჩანაწერის გაჩუმებითი მნიშვნელობის სახით შეუძლია გამოიყენოს მხოლოდ კონკრეტული რიცხვები, სტრიქონები და თარიღების მნიშვნელობები. ასე რომ, ჩანაწერის დამატებისას, added ველში მნიშვნელობების შეტანა მოუწევს ჩვენს მომავალ სერვერულ პროგრამას.

ცხრილში categories ცვლილებები საერთოდ არ გვექნება-იქ არაფერია შესაცვლელი. მისი სტრუქტურა ნაჩვენებია ცხრილში 6.6..
აქ ჩვენ ისევ შევქმნით გასაღებ ინდექსს **id** ველის საფუძველზე. მეორე ინდექსს ჩვენ შევქმნით უკვე ორი დანარჩენი ველის საფუძველზე: **name** და **file**. მას მივანიჭოთ სახელი **namefile**.
ახლა კი-ყურადღება! ინდექსს **namefile**, ჩვენ ვაქცევთ **უნიკალურ ინდექსად**, მისთვის განსაკუთრებული **ინდექსის ატრიბუტის UNIQUE** მინიჭებით. უნიკალური ინდექსი შეიძლება შეიცავდეს მხოლოდ უნიკალურ, მის ფარგლებში განუმეორებად მნიშვნელობებს. თუ კი მომხმარებელი შეეცდება ჩანაწერის ველებში

ისეთი მნიშვნელობების შეტანას, რომლებიც არღვევენ ინდექსის უნიკალურობას, მონაცემთა სერვერი შეატყობინებს მას შეტანის ოპერაციასთან დაკავშირებული შეცდომის შესახებ და უარყოფს შეტანილი მონაცემების ბაზაში დამახსოვრების ოპერაციას.

ცხრილი 6.6. categories ცხრილის სტრუქტურა

ველის დასახელება	ველის აღწერა	ველის მონაცემების ტიპი	მონაცემთა ტიპების აღწერა
id	ჩანაწერის იდენტიფიკატორი	INT UNSIGNED	მთელი რიცხვები 0-დან 65535-მდე. კატეგორიების რაოდენობა ცოტა გვექნება, ასე, რომ ეს საკმარისი იქნება
name	კატეგორიის დასახელება	VARCHAR (20)	20-სიმბოლოიანი ფიქსირებული სიგრძის სტრიქონი
file	თუ "ჭეშმარიტია", მაშინ ჩანაწერი აღნიშნავს ფაილს, წინააღმდეგ შემთხვევაში-სტატიების კატეგორიას	BOOL	ლოგიკური სიდიდეები

დაგვრჩა მხოლოდ თვით მონაცემთა ბაზისათვის სახელის შერჩევა. მოდით ნუ ვიმტვრევთ თავს და ვუწოდოთ მას უბრალოდ-**site**.
ყოველივე ზემოაღნიშნულის შემდეგ, გვრჩება მხოლოდ **MySQL** სერვერის გაშვება და მისი მხარდამჭერი ერთ-ერთი მონაცემთა

კლიენტის გამოყენებით, მაგალითად **MySQL Control Center**, ახალი მონაცემთა ბაზის, ცხრილებისა და ინდექსების შექმნა.
არის თუ არა მზად მონაცემტა ბაზა? არამც და არამც. საჭიროა კიდევ **MySQL** სერვერის მომხმარებლების სიაში ახალი მომხმარებლის დამატება, რომლის სახელითაც ჩვენი მომავალი სერვერული პროგრამა იმუშავებს ახლად შექმნილ მონაცემთა ბაზასთან. დავარქვათ მას იგივე სახელი, რაც ბაზას-**site**. განუსაზღვროთ მას ბაზის ორივე ცხრილში მონაცემების წაკითხვისა და შეცვლის უფლებები. ახლ კი ჩვენ ნამდვილად შეგვიძლია ვიზეიმოთ ჩვენი პირველი მონაცემთა ბაზის დაბადება.
კიდევ ერთი რამ. მოდით ახლადშექმნილი მონაცემთა ბაზის ცხრილებში შევიტანოთ რამოდენიმე ჩანაწერი. ისინი ჩვენ გამოგვადგებიან მაშინ, როდესაც დავიწყებთ ჩვენი პირველი სერვერული ვებ-გვერდების შექმნას, რათა მაშინათვე დავინახოთ სწორად მუშაობენ ისინი თუ არა. ჩვენ ხომ შემდგომში ნებისმიერ დროს შეგვეძლება მათი წაშლა.

რა იქნება ამის მერე?

ამ თავში ჩვენ გავარკვიეთ, თუ რას წარმოადგენენ მონაცემთა ბაზები და როგორ უნდა ვიმუშაოთ მათთან. ჩვენ გავეცანით მონაცემთა სერვერების პროგრამებს, **SQL** ენას და რამოდენიმე მარტივი მოთხოვნაც კი შევადგინეთ. აგრეთვე ჩვენ გავიგეთ წვდომის უფლებებისა და მათი განაწილების შესახებ-ისინი დაგვეხმარებიან ჩვენ, ჩვენთვის ძვირფასი მონაცემების დაცვაში დაუპატიჟებელი თვალებისა და ხელებისაგან.

მაგრამ მხოლოდ მონაცემთა ბაზა საკმარისი არ არის. საჭიროა კიდევ პროგრამა, რომელიც მათ დაამუშავებს და გამოუტანს მომხმარებლებს-ჩვენი საიტის დამთვალიერებლებს. ხოლო, რომ დავწეროთ ასეთი პროგრამა, საჭიროა შესაბამისი პროგრამირების ენის ცოდნა. ამიტომ მომდევნო თავში, ჩვენ დავიწყებთ **PHP** ტექნოლოგიის და აქტიური სერვერული ვებ-გვერდების დაწერის პრინციპების შესწავლას. რეალურად მომუშავე სერვერული პროგრამების შექმნამდე ჯერ კიდევ შორია, ამიტომ მოუხმოთ მოთმინებას.

თავი 7

PHP-სერვერული გამოყენებითი პროგრამების წერის ტექნოლოგია

მონაცემები, მათი დამუშავების პროგრამების გარეშე -არაფელია, მკვდარი კაპიტალია. ასე რომ, ჩვენი შემდგომი ნაბიჯი იქნება-სერვერული პროგრამის დაწერა, რომელიც იმუშავებს ჩვენს მიერ მე-6 თავში შექმნილ მონაცემთა ბაზასთან.

უფრო ადრე, მე-5 თავში, ჩვენ გადავწყვიტეთ, რომ ჩვენი სერვერული პროგრამა წარმოდგენილი იქნებოდა აქტიური სერვერული გვერდების ერთობლივობის სახით-ამ მონაცემების დამმუშავებელი პროგრამული კოდების ფრაგმენტების შემცველი ჩვეულებრივი ვებ-გვერდების სახით.

როგორც კი შეხვდება ასეთ ფრაგმენტს, ვებ-სერვერი მას გადასცემს სერვერული გვერდების დამმუშავებელ სპეციალურ პროგრამას, შემდეგ იღებს მისგან დამუშავების შედეგს ჩვეულებრივი **HTML** კოდის სახით და სვამს მას ზუსტად იმ ადგილას, სადაც თავდაპირველად მდებარეობდა პროგრამული კოდის აღნიშნული ფრაგმენტი.

მაშინვე, ჩვენ ჩვენი საიტის აქტიური სერვერული გვერდების შესაქმნელად, ავირჩიეთ ტექნოლოგია **PHP.** ის საკმაო შესაძლებლობებს იძლევა რთული საიტების შესაქმნელად, ძალიან მწარმოებლურია, უფასოა (მისი საწყისი კოდებიც ღიაა!) და ამასთან მშვენივრად მუშაობს მონაცემთა სერვერთან **MySQL**. (ზოგადად **PHP**-ს მრავალ სხვადასხვა მონაცემთა სერვერთან შეუძლია მუშაობა, მათ შორის **ODBC**-თანაც). ბაზარზე არსებული სხვა ტექნოლოგიები ან ფასიანებია, ან მიბმულები არიან რომელიმე ერთ სერვერთან.

მაშ ასე არჩევანი გაკეთებულია. რას ვაკეთებთ ამის შემდეგ?

PHP-ს ძირითადი ცნებები

დავიწყოთ **PHP**-ს ძირითადი ცნებებისა და მუშაობის პრინციპების შესწავლით. ამავდროულად შევქმნათ რამოდენიმე მარტივი სერვერული გვერდი, რათა მივიღოთ გარკვეული გამოცდილება და თან შევამოწმოთ, თუ როგორ მუშაობს ჩვენი ვებ-სერვერი.

მაგრამ ჯერ ჩვენს კომპიუტერზე უნდა დავაყენოთ თვით **PHP**-ს პროგრამული მოდულები და გავმართოთ ადრე დაყენებული ვებ-სერვერი ისეთნაირად, რომ მან შეძლოს აღნიშნული ტექნოლოგიის

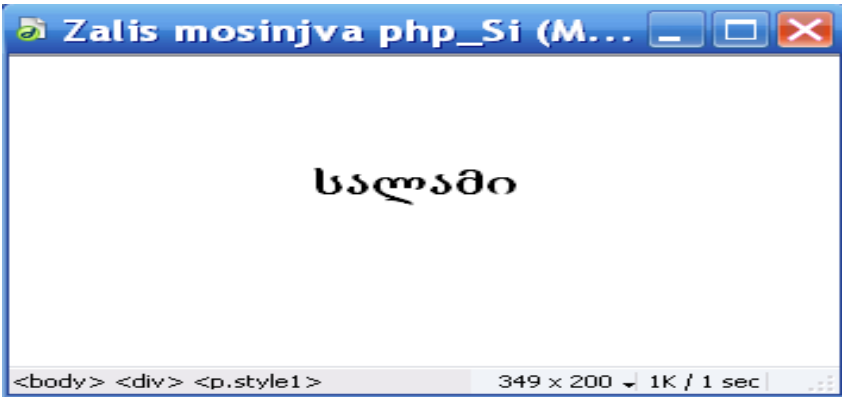
მხარდაჭერა. თუ როგორ უნდა გაკეთდეს ყოველივე აღნიშნული, აღწერილია **მე-3 დანართში.**

PHP სცენარების დაწერა

PHP-ს პროგრამული კოდის ფრაგმენტები (ეგრეთ წოდებული სცენარები) იწერებიან პირდაპირ ვებ-გვერდის **HTML** კოდში. ამასთან ისინი თავსდებიან განსაკუთრებული წყვილი ტეგის შიგნით **<?php.. . ?>.**
მოდით პროგრამა "NotePad"-ში ავკრიფოთ ასეთი **HTML** კოდი, რომელიც შეიცავს მცირე ზომის **PHP** სცენარს (გამოყოფილია მუქი შრიფტით):

```
<HTML>
   <HEAD>
      <TITLE> PHP-ში ძალების მოსინჯვა</TITLE>
   </HEAD>
   <BODY>
      <?php echo <P>"სალამი!"</P> ; ?>
   </BODY>
</HTML>
```

გამოტანის ოპერატორს **echo**-ს, მის შემდეგ მითითებული სიმბოლური მნიშვნელობა, გამოჰყავს ვებ-გვერდის იმ ადგილას, სადაც თვითონ იმყოფება. რაც შეეხება თვით სიმბოლურ გამოსახულებას, ის მოთავსებულია ორმაგ ბრჭყალებში (ეს მნიშვნელოვანია!) და შეიცავს სიტყვა **სალამი**-ს შემცველი ჩვეულებრივი აბზაცის **HTML** კოდს.
დავიმახსოვროთ ეს კოდი ფაილში 7.1.php (**PHP**-ის ყველა აქტიურ სერვერულ გვერდებს აუცილებლად **.php** გაფართოება უნდა ჰქონდეთ, სხვა შემთხვევაში ვებ-სერვერი მათ ჩვეულებრივ ვებ-გვერდებად აღიქვამს და უბრალოდ გადაუგზავნის ვებ-დამთვალიერებელს). ამის შემდეგ დავაკოპიროთ ეს ფაილი ჩვენ ლოკალური ვებ-სერვერზე განთავსებული საიტის ძირეულ საქაღალდეში და გავუშვათ თვით ვებ-სერვერი. ამის მერე თუ გავხსნით ვებ-დამთვალიერებელს და მისამართის სტრიქონში ავკრიფავთ **http://localhost:8080/7.1.php**-ს დავინახავთ იმას, რაც ნაჩვენებია ნახ.7.1.-ზე.



ნახ.7.1. ვებ-დამთვალიერებელში 7.1.php სერვერული გვერდის გახსნის შედეგი

ყურადღება! *ზემოთ ნათქვამი სამართლიანია მხოლოდ იმ შემთხვევისათვის, თუ ჩვენს ვებ-სერვერზე გამოყენებულია პორტი 8080 (როგორც ეს აღწერილია პირველ დანართში). თუ კი ვებ-სერვერზე გამოყენებულია სხვა პორტი, საჭირო იქნება ზემოთ აღნიშნულ ინტერნეტ-მისამართში 8080-ის ნაცვლად ამ პორტის ნომრის ჩასმა. მომავალში ჩვენ ყოველთვის ვიგულისხმებთ, რომ ვებ-სერვერში გამოყენებულია პორტი 8080.*

მოდით ახლა გამოვიტანოთ ეკრანზე **PHP**-ს დამმუშავებლის მიერ ფორმირებული საბოლოო ვებ-გვერდის HTML კოდი. (**Microsoft Internet Explorer**-ში ამის გაკეთება შესაძლებელია პუნქტის **View HTML code-Просмотр HTML-кода- HTML-კოდის დათვალიერება** არჩევით მენიუში **View-Вид-დათვალიერება**). ჩვენ დავინახავთ შემდეგს:

```
<HTML>
    <HEAD>
        <TITLE> ძალების მოსინჯვა :
    </HEAD>
    <BODY>
        “<P>სალამი!</P>
    </BODY>
```

</HTML>

ესეიგი სცენარის ნაცვლად **PHP** დამმუშავებელმა ჩასვა მისი შესრულების შედეგი. ეს იმის მაჩვენებელია, რომ ყველაფერი ნორმალურად მუშაობს.

ჩვენ შეგვიძლია ჩვენი პირველი **PHP** გვერდის კოდის ოდნავ სხვა ნაირად ჩაწერა:

```
<HTML>
    <HEAD>
        <TITLE> PHP-ში ძალების მოსინჯვა</TITLE>
    </HEAD>
    <BODY>
        <P>  <?php echo “სალამი!”; ?> </P>
    </BODY>
</HTML>
```

აქ ჩვენ ”გავიტანეთ” თვით ტეგები **<P>** და**</P>** სიმბოლური მნიშვნელობიდან სცენარში. შევხედოთ, შეძლებს თუ არა **PHP** დამმუშავებელი ამის დამუშავებას. დავაკოპიროთ ახალი გვერდი სერვერის ძირეულ საქაღალდეში და გავხსნათ იგი ვებ-დამთვალიერებელში. შედეგი ნაჩვენებია ნახ.7.1.-ზე.
უფრო მეტიც, ჩვენ შეგვიძლია ასეთი კოდის დაწერა:

```
<P>სალამი,<?php echo “მშვიდობა!” ; ?></P>
```

ან თუნდაც ასეთის:

```
<?php echo “<P>”; ?><?php echo “სალამი,”; ?><?php echo “მშვიდობა!”; ?><?php echo “<P>” ; ?>
```

და **PHP** დამმუშავებელი მის დამუშავებასაც შეძლებს.
თუ **PHP** სცენარის კოდი ერთ სტრიქონზე მეტია, მაშინ მისი ასე ჩაწერა ჯობია:

```
<?php
```

```
    echo “<P>”;
    echo “სალამი, “;
    echo “მშვიდობა!”;
    echo  “</P>”:
?>
```

ანუ ერთი **<?php . . .?>** ტეგის შიგნით. ასეთნაირად ჩვენი სერვერული გვერდის კოდს უფრო კომპაქტურს და წასაკითხად უფრო ადვილს გავხდით.

უნდა აღინიშნოს, რომ **PHP** აქტიური სერვერული ვებ-გვერდები შესაძლებელია სულ არ შეიცავდნენ **HTML** კოდს- და მხოლოდ სცენარს, უფრო ზუსტატ ერთად-ერთ **PHP**-სცენარს წარმოადგენდნენ. ასეთი რამ ხდება და არც თუ ისე იშვიათად. მაგრამ მსგავს გადაგვარებულ შემთხვევებშიც კი სცენარი მოთავსებული უნდა იყოს **<?php . . .?>** ტეგის შიგნით, წინააღმდეგ შემთხვევაში ვებ-სერვერი ვერ გაიგებს, თუ რა უნდა უყოს მას.

ოპერატორები, არგუმენტები და გამოსახულებები

მოდით, ახლა უფრო დეტალურად განვიხილოთ ჩვენს მიერ ახლახანს დაწერილი მცირე ზომის **PHP**-სცენარი. მოვაშოროთ მთელი **HTML** კოდი, რათა მან ხელი არ შეგვიშალოს და ყურადღება **PHP**-კოდზე გავამახვილოთ.

```
echo “<P>სალამი!</P>” ;
```

მაშ ასე, აქ ჩვენ ვხედავთ უკვე ჩვენთვის ნაცნობ გამოტანის ოპერატორს **echo**-ს. ის იღებს ერთ არგუმენტს-სტრიქონს და გამოაქვს იგი ეკრანზე, უფრო სწორად **HTML**-კოდის იმ ადგილას სადაც შეხვდა. (პრინციპში ეს ერთი და იგივეა, ვინაიდან სერვერული პროგრამის მიერ ფორმირებული ვებ-გვერდი მაინც მოხვდება მომხმარებლის ეკრანზე).

ახლა კი დროა განვსაზღვროთ **ოპერატორი** და **არგუმენტი**. **ოპერატორი**-ეს **PHP** ენის ბრძანებაა, რომელიც ახორციელებს რაიმე მოქმედებებს მისთვის გადაცემულ მონაცემებზე (**არგუმენტებზე**) ან თვით სცენარზე.

PHP-ში ოპერატორები მრავალნაირია. ჩვენ უკვე ვიცნობთ ერთ-ერთ მათგანს-გამოტანის ოპერატორს **echo**-ს. არსებობენ აგრეთვე **არითმეტიკული** ოპერატორები, რომლებიც რიცხვებზე

ელემენტარულ მოქმედებებს ასრულებენ: მიმატება, გამოკლება, გამრავლება, გაყოფა. სცენარის მაგალითი, რომელიც იყენებს არითმეტიკულ ოპერატორს ასე გამოიყურება:

Echo 2+2;

აქ ჩვენ ვხედავთ მიმატების ოპერატორს, რომელიც იღებს ორ არგუმენტის-რიცხვით მნიშვნელობებს 2 და 2. მათი შეკრების შემდეგ ის გვიბრუნებს მიღებულ ჯამს (შედეგს), რომელსაც თავის მხრივ, არგუმენტის სახით იღებს ოპერატორი echo. შევნიშნოთ, რომ არ ხდება რიცხვითი მნიშვნელობების ბრჭყალებში ჩასმა.
ადრე განხილული **PHP** სცენარი შედგება ერთი გამოსახულებისაგან. გამოსახულება-ეს არის **PHP**-კოდის განუყოფელი ფრაგმენტი, რომელიც ერთ სრულ მოქმედებას ახორციელებს. ასე მაგალითად:

echo 2+2

ახორციელებს ერთ სრულ მოქმედებას: კრიბავს ორ რიცხვს და შედეგი გამოყავს ეკრანზე. ამასთან იგი განუყოფელია, ანუ არ ხდება მისი დანაწევრება უფრო მცირე ნაწილებად.
ეს სცენარი კი შედგება ოთხი გამოსახულებისაგან:

$a=2;
$b=3
$c=$a+$b;
Echo $c;

ყოველი გამოსახულება აუცილებლად წერტილ-მძიმის ნიშნით უნდა მთავრდებოდეს. წერტილ-მძიმე-ეს არის გამოსახულების დასასრულის ნიშანი; როგორც კი მას წააწყდება **PHP** დამმუშავებელი, ჩათვლის, რომ გამოსახულება დამთავრებულია და იგი უნდა შესრულდეს.
გამოსახულებები-ეს თავისებური მოლეკულებია, რომელთა საშუალებითაც ხდება სცენარების აწყობა. ოპერატორები და მათი არგუმენტები (აგრეთვე ფუნქციები, რომელთაც ჩვენ მერე განვიხილავთ)-თავისი არსით ატომებს, ანუ **PHP**-ენის სტანდარტულ ელემენტებს წარმოადგენენ, რომელთა საშუალებითაც

აიგებიან გამოსახულებები. ყოველივე ეს თავისებური პროგრამისტული ფიზიკაა.

უნდა აღვნიშნოთ, რომ **PHP**-ენის ოპერატორების წარმოსადგენად, ისევე, როგორც **SQL**-ში, იყენებენ განსაკუთრებულ დარეზერვირებულ სიტყვებს, რომელთაც **გასაღებ სიტყვებს** უწოდებენ. ამასთან რეგისტრს, რომლითაც ამ გასაღები სიტყვების შემადგენელი ასოებია აკრეფილი მნიშვნელობა არა აქვს. ასე მაგალითად, შეიძლება აიკრიფოს **echo, Echo** ან **ECHO** და ეს ერთი და იგივე გამოტანის ოპერატორი იქნება.

ცვლადები

მოდით, კიდევ ერთხელ შევხედოთ სამი გამოსახულებისაგან შემდგარ ზემოთ მოყვანილ სცენარს. საერთოდ რისი მაქნისია ის? რას წარმოადგენენ ლათინური ასოები, რომელთა წინაც დოლარის ნიშანი დგას?

საქმე იმაშია, რომ **PHP**-ენა პროგრამისტებს თავიანთ განკარგულებაში აძლევს ეგრეთ წოდებულ **ცვლადებს**. **ცვლადი**-ეს არის კომპიუტერის მეხსიერების უბანი, რომელიც ეძლევა პროგრამისტს თავისი რაიმე მონაცემების შესანახად: არგუმენტების ან ოპერატორების შესრულების შედეგების შესანახად. პროგრამისტს შეუძლია წეროს გამოსახულებები, რომლებიც რაიმე მონაცემებს მოათავსებენ ცვლადებში ან ამოიღებენ მათ იქედან.

ყოველლ ცვლადს უნდა გააჩნდეს უნიკალური სახელი. ეს სახელი ყოველთვის უნდა იწყებოდეს დოლარის სიმბოლოთი და უნდა შეიცავდეს მხოლოდ ლათინური ალფავიტის ასოებს, ციფრებს ან ხაზგასმის ნიშნებს. ამასთან დოლარის ნიშნის მომდევნო სიმბოლო აუცილებლად ასო ან ხაზგასმის ნიშანი უნდა იყოს. აგრეთვე ცვლადის სახელი არ უნდა ემთხვეოდეს **PHP**-ს გასაღებ სიტყვებს. ცვლადების სახელების სიგრძე შეზღუდული არ არის, მაგრამ უკეთესია ისინი რაც შეიძლება მოკლე და რაც შეიძლება გასაგები იყვნენ.

ქვემოთ მოყვანილია რამოდენიმე სწორად დაწერილი ცვლადის სახელი:

$var
$extended_result2
$_temp

ეს კი არასწორი სახელების მაგალითებია:

$2result
$extended output
$გაფართოებული გამოტანა

ამ სახელებიდან პირველში დოლარის ნიშნის შემდეგ მოდის რიცხვი, მეორე-შეიცავს გამოტოვებას, ხოლო მესამე აკრეფილია ქართული ასოებით.
გასაღები სიტყვებისაგან განსხვავებით, ცვლადების სახელები **PHP**-ში მგრძნობიარენი არიან რეგისტრის მიმართ. ასე მაგალითად, **$var** და **$Var**-ეს სხვადასხვა ცვლადებია.
ყოველივე აღნიშნულის შემდეგ, მოდით კიდევ დავუბრუნდეთ ჩვენს სცენარს. განვიხილოთ იგი ნაწილ-ნაწილ.

$a=2;
$b=3;

აქ ჩვენ რიცხვით მნიშვნელობებს ვანიჭებთ ორ ცვლადს: **$a** და **$b**. ეს კეთდება განსაკუთრებული **მარტივი მინიჭების ოპერატორის =:** საშუალებით. მისგან მარცხნივ იწერება ცვლადის სახელი, ხოლო მარჯვნივ-მნიშვნელობა, რომელიც მას უნდა მიენიჭოს.
ვინაიდან **$a** და **$b** ჯერ არ არსებობენ, **PHP** დამმუშავებელი შექმნის, ან როგორც პროგრამისტები ამბობენ **გამოაცხადებს** მათ. ცვლადის გამოცხადება ხდება მისთვის მნიშვნელობის პირველვე მინიჭებისას.

$c=$a+$b;

აქ ჩვენ ვაცხადებთ მესამე ცვლადს-**$c**- და ვანიჭებთ მას გამოთვლის შედეგს 2+3.

Echo $c;

ჩვენი სცენარის ამ უკანასკნელ გამოსახულებაში ოპერატორ echo-ს ამოაქვს ცვლადის $c მნიშვნელობა (ეს იქნება 2 და 3 ჯამი-5) და გამოაქვს იგი ეკრანზე.

ისღა დარჩა სათქმელი, რომ PHP-ს სერვერული გვერდის რომელიმე სცენარში გამოცხადებული ცვლადი ხელმისაწვდომია ამ გვერდზე განთავსებულ ყველა სცენარში. ასე მაგალითად, თუ ჩვენ ჩავწერთ ორ სცენარს ვებ-გვერდის კოდში (თვით HTML კოდი გამოტოვებულია) :

```
<?php $a="Test ! ! !"; ?>
. . .
<?php echo $a; ?.
```

მაშინ ორივე მათგანი შესრულდება სწორად.

მონაცემთა ტიპები

ჩვენს მიერ მე-6 თავში შესწავლილი ცხრილების ველებისაგან განსხვავებით, **PHP**-ს ცვლადები შეიძლება შეიცავდნენ სხვადასხვა ტიპის მონაცემებს. შეიძლება მაგალითად, ცვლადის გამოცხადებისას მისთვის რიცხვითი მნიშვნელობის მინიჭება, ხოლო შემდეგ მასვე მიენიჭოს სიმბოლური ან ლოგიკური მნიშვნელობა და დამმუშავებელი ყველაფერს შეასრულებს. თუმცა რათქმა უნდა ჯობია, რომ ასეთი რამ არ დავუშვათ-ეს პროგრამირების ცუდი სტილია.
მოდით უფრო კარგად გავეცნოთ **PHP**-ს მიერ მხარდაჭერილ მონაცემთა ტიპებს. მთი რაოდენობა არც თუ ისე დიდია.

ლოგიკური ტიპი

ლოგიკური ტიპი წარმოდგენილია მხოლოდ ორი მნიშვნელობით: "ჭეშმარიტი" (**true**) და "მცდარი" (**false**). ორივე ეს მნიშვნელობა იწერება **PHP** ენის გასაღები სიტყვების **true** და **false** სახით.

```
$flag=tru
```

მთელრიცხვიანი ტიპი

მთელი რიცხვის ტიპი, როგორც მისი სახელწოდებიდანაც ჩანს, წარმოადგენს მთელ რიცხვებს -2 147 483 648-დან 2 147 483 647-დე. მონაცემთა ბაზების ცხრილებში ველების ტიპებისაგან განსხვავებით ის მხოლოდ ერთია.

$counter=10;
$delta=-193;

გაჩუმებით მთელი რიცხვები მოიცემიან ათობითი თვლის სისტემაში. თუ საჭიროა რიცხვის მოცემა რვაობით ან თექვსმეტობით თვლის სისტემაში, საკმარისია მას წინ წავუმძღვაროთ შესაბამისად **0** ან **0x**.

$octal_number=0123;
$hex_number=0x4F;

მცოცავ მძიმიანი ტიპი

ტიპი მცოცავი მძიმით წარმოადგენს წილად რიცხვებს- $1,79769313486232*10^{308}$-დან $-2,2250738585072*10^{-308}$-მდე და $2,22507385855072*10^{-308}$ -დან $1,79769313486232*10^{308}$-მდე დაახლოებით, მძიმიდან 14 ნიშნამდე სიზუსტით.

$square=10.56;

შევნიშნოთ, რომ მძიმის ნიშნის ნაცვლად რიცხვის **PHP** კოდში წარმოსადგენად გამოიყენება წერტილი.
რიცხვების ნორმალიზებული სახით წარმოსადგენად (<მანტისა>*10<რიგი>) შეიძლება გამოყენებული იქნას შემდეგი სახის სპეციალური სინტაქსისი <მანტისა>E<რიგი>.

$distance=2.648E+12;

ასეთი სახით შეგვიძლია წარმოვადგინოთ რიცხვი $2,648*10^{12}$. მიაქციეთ ყურადღება იმას, რომ ნიშანი + რიგის მაჩვენებელში აუცილებელია.

$millimeter=1E-3;

ასე წარმოიდგინება რიცხვი $1*10^{-3}$.

ტექსტური ტიპი

ტექსტური ტიპი წარმოადგენს პრაქტიკულად შეუზღუდავი ზომის ტექსტურ სტრიქონებს (ყოველ შემთხვევაში PHP-ს დოკუმენტაციაში ასეა აღნიშნული). საჭიროა ტექსტური მნიშვნელობების ორმაგ ბრჭყალებში ჩასმა. სტრიქონების შიგნით შეიძლება გამოყენებულ იქნას ნებისმიერი სიმბოლოები, რომელთა ეკრანზე გამოყვანა შესაძლებელია.

```
$output="სალამი!";
```

PHP აგრეთვე წარმოგვიდგენს რამოდენიმე ეგრეთ წოდებულ სპეციალურ სიმბოლოს, რომელთა გამოყენებაც შესაძლებელა სტრიქონებში. ეს სიმბოლოები ან რაიმე განსაკუთრებულ მოქმედებებს ასრულებენ ან შეუძლებელია მათი გამოყენება სტრიქონებში ჩვეულებრივი წესით. ყველა ისინი ჩამოთვლილია ცხრილში 7.1.

ცხრილი 7.1. PHP-ს სპეციალური სიმბოლოები

სპეციალური სიმბოლო	აღწერა
\n	სტრიქონის გადაყვანა
\r	ეტლის დაბრუნება
\t	ჰორიზონტალური ტაბულაცია
\\	შებრუნებული სლეში
\$	დოლარის ნიშანი
\"	ორმაგი ბრჭყალი
\<კოდი>	სიმბოლო მოცემული რვაობითი კოდით
\x<კოდი>	სიმბოლო მოცემული თექვსმეტობითი კოდით

აქ საჭიროა ზოგიერთი განმარტებების გაკეთება.
შებრუნებული სლეშის, დოლარის და ორმაგი ბრჭყალების სიმბოლოების გამოყენება სტრიქონებში დაუშვებელია, ამიტომ მათ ნაცვლად გამოყენებული უნდა იყოს მათი შესაბამისი სპეციალური სიმბოლოები. მაგალითად, არასწორი გამოსახულებაა:

$output=”სასტუმრო”თბილისი”;

ვინაიდან სტრიქონის შიგნით ორმაგი ბრჭყალი დაუშვებელია. მათ ნაცვლად უნდა გამოვიყენოთ სიმბოლოების მიმდევრიბა \”, აი ასე:

$output=”სასტუმრო\”თბილისი”\”;

ანუ ტექსტურ მნიშვნელობებში ყოველ დაუშვებელ სიმბოლოს წინ უნდა უსწრებდეს სიმბოლო ”შებრუნებული სლეში”\”.
ეტლის დაბრუნებისა და სტრიქონის გადაყვანის ერთმანეთის მომდევნო ნიშნები-\r\n-შესაძლებელს ხდიან ოპერატორ **echo**-ს საშუალებით, ახალი სტრიქონიდან მონაცემების შეტანას. მაგალითად სცენარი:

echo “დავიწყოთ”;
echo “გამოტანა\r\n”;
echo “ახალი აბზაციდან!!!”;

ეკრანზე გამოიტანს ამას:

დავიწყოთ გამოტანა
ახალი აბზაციდან

ყურადღება მივაქციოთ იმას, რომ ამ სცენარის პირველმა და მეორე გამოსახულებებმა განახორციელეს მონაცემების ერთ სტრიქონად გამოტანა.

შენიშვნა: *ხანდახან ტექსტურ მნიშვნელობებს ათავსებენ ერთეულოვან ბრჭყალებში. მაგრამ ასეთი სტრიქონებისათვის მიუწვდომელი ხდება PHP-ს ზოგიერთი შესაძლებლობა.*

NULL

ტიპი **NULL** ნიშნავს, რომ ცვლადი არ შეიცავს არავითარ მნიშვნელობას. ეს შეიძლება მოხდეს მაშინ, როდესაც ჩვენ შევეცდებით ისეთი ცვლადის მნიშვნელობის მიღებას, რომელიც ჯერ გამოცხადებული არ არის. ასევე ჩვენ შეგვიძლია პირდაპირ

მივანიჭოთ ცვლადს მნიშვნელობა **NULL**, გასაღები სიტყვის **NULL** გამოყენებით:

$null_value=NULL;

ოპერატორები

ოპერატორები, როგორც ჩენთვის უკვე ცნობილია, ასრულებენ უშუალო მოქმედებებს არგუმენტებზე და შესაძლოა შედეგებიც დაგვიბრუნონ.
PHP ენას საკმაოდ ბევრი ოპერატორი გააჩნია, რომელთა განხილვასაც ჩვენ ახლა შევუდგებით.

არითმეტიკული ოპერატორები

როგორც ჩვენთვის ცნობილია, არითმეტიკული ოპერატორები გამოიყენებიან არგუმენტებზე არითმეტიკული მოქმედებების შესასრულებლად. ცხრილში 7.2 ჩამოთვლილია **PHP**-ს მიერ მხარდაჭერილი ყველა არითმეტიკული ოპერატორი.

ცხრილი 7.2. PHP-ს არითმეტიკული ოპერატორები

ოპერატორი	აღწერა
-	რიცხვის ნიშნის შეცვლა
+	შეკრება
-	გამოკლება
*	გამრავლება
/	გაყოფა
%	ნაშთი გაყოფიდან
++	ინკრემენტი(ერთით მომატება)
--	დეკრემენტი (ერთით შემცირება)

გავაკეთოთ ზოგიერთი განმარტება!
რიცხვის ნიშნის შეცვლის ოპერატორი იწერება არგუმენტის წინ და ცვლის მის ნიშანს:

$r=-$r;

ეს გამოსახულება ამოიღებს **$r** ცვლადის მნიშვნელობას, შეცვლის მის ნიშანს და ისევ მოათავსებს ცვლადში **$r**.

ინკრემენტის ოპერატორი შეიძლება იდგეს, როგორც არგუმენტის წინ, ასევე მის შემდეგაც. მოდით განვიხილოთ ამ ოპერატორის შემცველი ორი აგამოსახულბეა.

$t=++$r;

ეს გამოსახულება ჯერ ახდენს **$r** ცვლადის ინკრემენტირებას (გაზრდას), ხოლო შემდეგ შეაქვს იგი **$t** ცვლადში.

$t=$r++ ;

ამ გამოსახულებას კი, პირიქით ჯერ შეაქვს **$r** ცვლადის მნიშვნელობა $t ცვლადში, ხოლო შემდეგ ახდენს **$r** ცვლადის მნიშვნელობის ინკრემენტირებას. ზუსტად ასევე მუშაობს დეკრემენტის ოპერატორიც.

ყურადღება: *ინკრემენტისა და დეკრემენტის ოპერატორების გამოყენება რეკომენდირებულია, თუ საჭიროა რაიმე ცვლადის მნიშვნელობის უბრალოდ გაზრდა ან შემცირება ერთი ერთეულით. ამ ოპერატორების შესრულება უფრო სწრაფად ხდება, ვიდრე შეკრებისა და გამოკლების ოპერატორებისა.*

სტრიქონების გაერთიანების (კონკატენაციის) ოპერატორი

სტრიქონების გაერთიანების (კონკატენაციის) ოპერატორი (.- წერტილი), ორ სტრიქონს აერთიანებს ერთ სტრიქონად. მაგალითად სცენარი:

$s1=”DreamweaverMX”;
$s2=”PHP”;
$s3=”MySQL”;
$output=s1 .” “. s2 . “ “. s3 ;

სტრიქონს **DreamweaverMX PHP MySQL**-ს მოათავსებს ცვლადში $output

მინიჭების ოპერატორები

ჩვენთვის უკვე ნაცნობია მარტივი მინიჭების ერთადერთი ოპერატორი =. მისი საშუალებით ცვლადს ენიჭება ახალი მნიშვნელობა:

$a=2 ;

მარტივი მინიჭების ოპერატორის გარდა, **PHP**-ში მხარდაჭერილია რამოდენიმე **რთული მინიჭების ოპერატორიც.** ასეთი ოპერატორები მინიჭების ოპერაციის, სხვა ოპერაციებთან ერთდროულად შესრულების საშუალებას იძლევიან:

$a = $a + $b;
$a += $b

ეს ორი გამოსახულება შედეგის მიხედვით ერთმანეთის ექვივალენტურია. უბრალოდ მეორეში გამოყენებულია რთული მინიჭების ოპერატორი +=, რომელიც მინიჭებას ახორციელებს მიმატებასთან ერთად.

PHP-ს მიერ მხარდაჭერილი რთული მინიჭების ყველა ოპერატორი და მათი ექვივალენტები მოყვანილია ცხრილში 7.3.

ცხრილი 7.3. **PHP**-ს რთული მინიჭების ოპერატორები

ოპერატორი	ექვივალენტური მნიშვნელობა
$a += $b;	$a = $a + $b
$a -= $b;	$a = $a - $b
$a *= $b;	$a = $a * $b
$a /= $b;	$a = $a / $b
$a %= $b;	$a = $a % $b

ყურადღება: *რთული მინიჭების ოპერატორების შესრულება უფრო სწრაფად ხდება, ვიდრე არითმეტიკული ოპერატორისა და მინიჭების ოპერატორის ერთობლივობის.*

ყველაზე გასაკვირი რამ იმაშია, რომ მინიჭების ოპერატორი ასევა აბრუნებს შედეგს! იგი იმ მნიშვნელობის ტოლია, რომელსაც მიიღებს ცვლადი, რომელიც ოპერატორიდან მარცხენა მხარეს დგას. მაგალითად:

$b = $a =3

ჯერ ცვლადი **$a** მიიღებს მნიშვნელობას 3, ხოლო შემდეგ ცვლადი **Sb** მიიღებს გამოთვლის შედეგის ტოლ მნიშვნელობას **$a** = 3, რომელიც ასევე 3-ის ტოლია.

შედარების ოპერატორები

შედარების ოპერატორები ადარებენ ერთმანეთს ორ არგუმენტს რაიმე პირობის შესაბამისად და აბრუნებენ ლოგიკურ მნიშვნელობას. თუ შედარების პირობა სრულდება, ბრუნდება მნიშვნელობა "ჭეშმარიტი" (**true**), თუ არ სრულდება-"მცდარი" (**false**). ქვემოთ მაყვანილია გამოსახულებების მაგალითები, რომლებშიც გამოყენებულია შედარების ოპერატორები:

$a1 = (2 < 3) ;
Sa2 = (-4 > 0) ;
$a3 = ($r < $t) ;

ამ გამოსახულებების გამოთვლის შედეგად ცვლადი **$a1** იღებს მნიშვნელობას **true** (2 ნაკლებია 3), ცვლადი **$a2**-მნიშვნელობას **false** (-4 არ შეიძლება იყოს ნულზე მეტი), ხოლო **$a3** ცვლადის მნიშვნელობა დამოკიდებული იქნება **$r** და **$t** ცვლადების მნიშვნელობებზე. ასეთ გამოსახულებებს, რომლებიც აბრუნებენ ლოგიკურ სიდიდეებს, **ლოგიკურ** გამოსახულებებს უწოდებენ.
PHP ენის მიერ მხარდაჭერილი ყველ შედარების ოპერატორი მოცემულია ცხრილში 7.4.

ცხრილი 7.4. PHP შედარების ოპერატორები

ოპერატორი	აღწერა
<	ნაკლებობა
>	მეტობა

==	ტოლობა
<=	ნაკლებობა ან ტოლობა
>=	მეტობა ან ტოლობა
!= ან <>	უტოლობა
===	მკაცრი ტოლობა
!==	მკაცრი უტოლობა

ორი უკანასკნელი ოპერატორი-"მკაცრი ტოლობა" და "მკაცრი უტოლობა" განვიხილოთ უფრო დეტალურად. ეს ეგრეთ წოდებული **მკაცრი შედარების** ოპერატორებია. ჩვეულებრივი ოპერატორები "ტოლობა" და "უტოლობა", თუ ხვდებიან სხვადასხვა ტიპის მნიშვნელობების შემცველ არგუმენტებს, ცდილობენ მათ გარდაქმნას ერთი და იმავე ტიპის ცვლადად (ტიპების გარდაქმნებს ჩვენ კიდევ დავუბრუნდებით ამავე თავში). მკაცრი ტოლობის და მკაცრი უტოლობის ოპერატორები ასეთ გარდაქმნებს არ ახორციელებენ, ხოლო არგუმენტების ტიპების შეუთავსებლობის შემთხვევაში ყოველთვის აბრუნებენ მნიშვნელობას **false**.

ლოგიკური ოპერატორები

ლოგიკური ოპერატორები მოქმედებებს ასრულებენ ლოგიკურ მნიშვნელობებზე. ყველა მათგანი მოცემულია ცხრილში 7.5. ხოლო ცხრილებში 7.6 და 7.7 ნაჩვენებია ამ ოპერატორების შესრულების შედეგები.

ცხრილი 7.5. PHP-ს ლოგიკური ოპერატორები

ოპერატორი	აღწერა
!	არა (ლოგიკური ინვერსია)
&& ან and	და (ლოგიკური გამრავლება)
|| ან or	ან (ლოგიკური შეკრება)

ცხრილი 7.6. "AND" (და) და "OR" (ან) ოპერატორების შესრულების შედეგები.

არგუმენტი 1	არგუმენტი 2	&& (და)	|| (ან)
true	true	true	true

true	false	false	true
false	true	false	true
false	false	false	false

ცხრილი 7.7. ოპერატორ ”NOT” (არა) შესრულების შედეგები

არგუმენტი	! (არა)
tru	false
false	tru

ლოგიკური ოპერატორების გამოყენების ძირითადი სფერო-ეს არის შედარების ოპერატორების შემცველი ლოგიკური გამოსახულებები. ასეთი გამოსახულების მაგალითს წარმოადგენს:

$flag = ! ($status ==0) ;

აქ ცვლადი **$flag** მიიღებს მნიშვნელობას **true**, თუ ცვლადის **$status** მნიშვნელობა ”არ” უდრის ნულს (ყურადღება მივაქციოთ ლოგიკური ინვერსიის ოპერატორს, რომელიც **$status == 0** პირობის წინ დგას).

$flag = ($a > 3) || ($b + $c ! = 10) ;

აქ კი ცვლადი **$flag** მიიღებს მნიშვნელობას **true**, თუ ცვლადის $a მნიშვნელობა მეტია 3-ზე ”ან” **$b** და **$c** ცვლადების ჯამი არ უდრის ათს.

როგორ ანგარიშობს PHP ლოგიკური ოპერატორების შემცველ გამოსახულებებს.

ლოგიკური ოპერატორების შემცველი გამოსახულებების გამოთვლისას **PHP** ცდილობს მაქსიმალურად გაიმარტივოს საქმე. იგი შემდეგი წესით მოქმედებს: თუ რაღაც მომენტში ცხადი გახდა, თუ როგორი იქნება ლოგიკური გამოსახულების გამოთვლის შედეგი, მისი გამოთვლა წყდება.

მაგალითისათვის, მოდით ავიღოთ ადრე განხილული გამოსახულება

$flag = ($a > 3) || ($b + $c ! = 10) ;

და შევხედოთ, თუ როგორ მოიქცევა **PHP** მისი გამოთვლისას. (ზოგადად ამ გამოსახულებამ უნდა დააბრუნოს მნიშვნელობა true, თუ ცვლადის $a მნიშვნელობა მეტია სამზე "ან" $b და $c ცვლადების მნიშვნელობათა ჯამი არ უდრის ათს).

დავუშვათ, რომ $a ცვლადის მნიშვნელობა მეტია სამზე. მაშინ (**$a** > 3) მნიშვნელობა იქნება **true**. ხოლო, როგორც ჩვენ ცხრილიდან 7.6. გვახსოვს, თუ ოპერატორ ||-ის (ლოგიკური "ან") ერთერთი არგუმენტი **true**-ს ტოლია, მაშინ ამ ოპერატორის შედეგიც **true** -ს ტოლი იქნება. გამოდის, რომ თუ (**$a** > 3) მნიშვნელობა უდრის true -ს, მაშინ მნიშვნელობა არა აქვს თუ რა მნიშვნელობა ექნება (**$b** + **$c !** = 10 -ს- მთელი ამ გამოსახულების გამოთვლის შედეგი იქნება true.
PHP-იც ასევე ითვლის. ამიტომ, როგორც კი გაარკვევს, რომ (**$a** > 3)-ის გამოთვლა იძლევა **true** შედეგს, იგი წყვეტს ამ გამოსახულების გამოთვლას და ცვლადში **$flag** ათავსებს შედეგს- **true**.
მოდით ახლა სხვა გამოსახულება განვიხილოთ:

$d = ($t == 0) && ($v < 100) ;

დავუშვათ, რომ (**$t == 0**) გამოსახულების გამოთვლის შედეგი არის **false**. ლოგიკური "**AND-&&**" ოპერატორი დააბრუნებს **true**-ს, თუ ორივე მისი არგუმენტი ტოლი იქნება **true**-სი. მოცემულ შემთხვევაში კი ცხადია, რომ მთელი გამოსახულების მნიშვნელობა **false**-ის ტოლი იქნება-ამ ოპერატორის ერთ-ერთი არგუმენტი ნამდვილად არ უდრის **true**-ს. ამიტომაც **PHP** გამოსახულების გამოთვლას დროზე ადრე შეწყვეტს.
ეს თავისებურება მნიშვნელოვნად აჩქარებს **&&** და || ოპერატორების შემცველი ლოგიკური გამოსახულებებისაგან შემდგარი სცენარების დამუშავებას. გამოცდილმა პროგრამისტებმა იციან ამის შესახებ და ხშირად იყენებენ მას.

მონაცემთა ტიპების თავსებადობა და გარდაქმნა.

ახლა ჩვენ უნდა განვიხილოთ ორი მნიშვნელოვანი საკითხი: **მანაცემთა ტიპების თავსებადობა** და ერთი ტიპის მეორე ტიპად **გარდაქმნა**.

რას მივიღებთ, თუ შევკრიბავთ ორ რიცხვით მნიშვნელობას? სწორია-კიდევ ერთ რიცხვით მნიშვნელობას. თუ შევკრიბავთ რიცხვს და სტრიქონს? ძნელი სათქმელია. აქ **PHP** დამმუშავებელი აწყდება მონაცემთა ტიპების შეუთავსებლობის პრობლემას და ერთი ტიპის მეორე ტიპად გარდაქმნის გზით, ცდილობს გახადოს ეს ტიპები თავსებადები. იგი ცდილობს გადააქციოს სტრიქონი რიცხვად და ამის შემდეგ შეასრულოს შეკრების ოპერაციას.
სტრიქონის რიცხვად გადაქცევა ხორციელდება შემდეგი წესების მიხედვით:
-თუ სტრიქონის დასაწყისი შეიცავს ციფრებს, რომელთა გარდაქმნა შესაძლებელია მთელ რიცხვად, სტრიქონი გარდაიქმნება მთელ რიცხვად;
-თუ სტრიქონის დასაწყისი შეიცავს ციფრებს, რომელთა გარდაქმნა შესაძლებელია მცოცავ მძიმიან რიცხვად, სტრიქონი გარდაიქმნება მცოცავ მძიმიან რიცხვად;
-თუ სტრიქონის დასაწყისი საერთოდ არ შეიცავს ციფრებს, მაშინ სტრიქონი გარდაიქმნება ნულად.
განვიხილოთ შემდეგი სცენარი:

$a = 11 ;
$b = “ 10 გოჭი” ;
$c = $a + $b ;

$b-ცვლადში შემავალი სტრიქონი გარდაიქმნება რიცხვად 10.
გამოდის, რომ ცვლადმა **$c** უნდა მიიღოს მნიშვნელობა 21 (11+10). ამასთან შევნიშნოთ, რომ **$b** ცვლადში შემავალი საწყისი სტრიქონი ისევე სტრიქონად დარჩება.
სტრიქონის და რიცხვის გაერთიანების მცდელობისას უკანასკნელი გარდაიქმნება სტრიქონად:

$s = “PHP ;
$n = 4
$result = $s . $n ;

აქ ცვლადი **$result** მიიღებს მნიშვნელობას **PHP4**.
ლოგიკური სიდიდეები გარადაიქმნებიან ან რიცხვითი, ან ტექსტური ტიპის სიდიდეებად კონკრეტული შემთხვევის თავისებურების გათვალისწინებით. მნიშვნელობა **true**

გრდაიქმნება რიცხვი 1-ად ან სტრიქონი ”1”-თად, ხოლო მნიშვნელობა **false** - 0-ად ან ”0”-ად.
სტრიქონების ან რიცხვების ლოგიკურ სიდიდეებად გარდაქმნის წესები უფრო რთულია. ასე მაგალითად, 0-ის, ” ” სტრიქონის (ცარიელი სტრიქონი, რომელიც არც ერთ სიმბოლოს არ შეიცავს) და ”0”-ის რიცხვითი მნიშვნელობები გარდაიქმნებიან **false** მნიშვნელობებად. ყველაფერი დანარჩენი (არანულოვანი რიცხვები და არაცარიელი სტრიქონები) გარდაიქმნებიან **true** - მნიშვნელობებად.
PHP ყოველთვის ცდილობს, არაკორექტულად ჩაწერილი გამოსახულებებიც კი სწორად შეასრულოს. ხანდახან ეს მას გამოსდის, მაგრამ უფრო ხშირად ყველაფერი მუშაობს არა ისე, როგორც ეს დაგეგმილი იყო თავდაპირველად და ბოლოს და ბოლოს, სცენარის შესრულება წყდება სულ სხვა, აბსოლუტურად სწორ გამოსახულებაში შეცდომის აღმოჩენის გამო. ამიტომ ჯობია არ დავუშვათ მსგავსი კაზუსები და გამოვიყენოთ მხოლოდ თავსებადი ტიპების მქონე მონაცემები.
თუ მაინცა და მაინც საჭიროა ერთსა და იმავე ოპერატორში სხვადასხვა ტიპის მქონე არგუმენტების გაერთიანება, ჩვენ მოგვიწევს **ტიპების დაყვანის** ოპერაციის შესრულება. ამისათვის საკმარისი იქნება იმ არგუმენტის წინ, რომლის ტიპის გარდაქმნაც გვსურს, სასურველი ტიპის აღმნიშვნელი, ფრჩხილებში ჩასმული გასაღები სიტყვის ჩაწერა:

-(**bool**)-ლოგიკურ ტიპად გარდაქმნისათვის;
-(**int**)-მთელრიცხვა ტიპად გარდაქმნისათვის;
-(**float**)-მცოცავმძიმიან ტიპად გარადაქმნისათვის;
-(**string**)-ტექსტურ ტიპად გარდაქმნისათვის.

ქვემოთ მოყვანილია სტრიქონისა და რიცხვის გამაერთიანებელი შესწორებული სცენარის მაგალითი-მასში გამოყენებულია ტიპების დაყვანის ოპერაცია:

```
$s = “PHP;
$n = 4;
$result = $s . (string) $n ;
```

ლოგიკურ გამოსახულებებშიც, რომლებშიც შედარების ოპერატორებია გამოყენებული, **PHP** ასევე ახორციელებს ტიპების გარდაქმნას. გამონაკლისს შეადგენენ მხოლოდ ზემოთ აღწერილი მკაცრი შედარების ოპერატორები.

ოპერატორების პრიორიტეტები

ბოლო საკითხი, რომელსაც ჩვენ აქ განვიხილავთ-ეს არის ოპერატორების პრიორიტეტი. ჩვენ უკვე ვიცით, რომ პრიორიტეტი გავლენას ახდენს გამოსახულებაში ოპერატორების შესრულების თანმიმდევრობაზე. ახლა კი დადგა ამაზე უფრო დაწვრილებით საუბრის დრო.

ვთქვათ ჩვენ ჩავწერეთ ერთი გამოსახულებისაგან შემდგარი ასეთი სცენარი:

```
$a = $b + $c -$10 ;
```

აქ ჯერ $b ცვლადის მნიშვნელობას მიემატება $c ცვლადის მნიშვნელობა, ხოლო შემდეგ მიღებულ ჯამს გამოაკლდება 10. ამ გამოსახულების ოპერატორებს გააჩნიათ ერთნაირი პრიორიტეტი და ამიტომ მათი შესრულება ხდება მკაცრად მარცხნიდან მარჯვნივ.
ახლა კი განვიხილოთ მეორე სცენარი:

```
$a = $b + $c * 10 ;
```

აქ ჯერ შესრულდება $c ცვლადის მნიშვნელობის გამრავლება 10-ზე, ხოლო შემდეგ მიღებულ ნამრავლს მიემატება $b ცვლადის მნიშვნელობა. საქმე იმაშია, რომ გამრავლების ოპერატორს (*) უფრო მაღალი პრიორიტეტი გააჩნია, ვიდრე შეკრების ოპერატორს, ამიტომ **PHP**-ს დამმუშავებელი პირველად მას შეასრულებს.
მინიჭების ოპერატორებს-როგორც მარტივებს, ასევე რთულებს-გააჩნიათ ძალიან დაბალი პრიორიტეტი (უფრო დაბალი გააჩნიათ ოპერატორებს **AND** და **OR**. ამიტომ ყოველთვის ჯერ თვით გამოსახულების გამოთვლა ხდება, ხოლო შემდეგ მიენიჭება მისი მნიშვნელობა ცვლადს.
ასე, რომ ყველა ოპერატორის შესრულების ძირითადი პრინციპი ასეთია: ჯერ სრულდებიან უფრო მაღალი პრიორიტეტის მქონე ოპერატორები, ხოლო შემდეგ-უფრო დაბალი პრიორიტეტის მქონე

ოპერატორები. ერთნაირი პრიორიტეტის მქონე ოპერატორები სრულდებიან გამოსახულებაში მათი თანმიმდევრობის შესაბამისად-მარცხნიდან მარჯვნივ.

ცხრილში 7.8, ჩამოთვლილია ჩვენს მიერ შესწავლილი ყველა ოპერატორი მათი პრიორიტეტების კლების მიხედვით.

ცხრილი 7.8. PHP ოპერატორების პრიორიტეტები
(კლების მიხედვით)

ოპერატორები	აღწერა
++ -- - !	ინკრემენტი, დეკრემენტი, ნიშნის შეცვლა, ლოგიკური არა (NOT)
* / %	გამრავლება, გაყოფა, ნაშთის გამოთვლა
+ - ,	შეკრება, გამოკლება და სტრიქონების გაერთიანება
< > <= >= = ! =	შედარება
&&	ლოგიკური (კი)
||	ლოგიკური (ან)
= <>	მინიჭება, მარტივი და რთული
and	ლოგიკური (და)
or	ლოგიკური (ან)

მაგრამ, როგორ მოვიქცეთ, თუ ჩვენ გვინდა ოპერატორების შესრულების ჩვეულებრივი თანმიმდევრობის დარღვევა? ქვემოთ ნაჩვენები სახით ჩაწერის შემთხვევაში, ფრჩხილებში ჩასმულ ოპერატორებს უფრო მაღალი პრიორიტეტი გააჩნიათ, ვიდრე ფრჩხილებს გარეთ მყოფ ოპერატორებს და ამიტომ ყოველთვის პირველად მათი შესრულება ხდება.

$a = ($b + $c) * 10 ;

ამ სცენარში ჯერ შესრულდება **$b** და **$c** ცვლადების მნიშვნელობების შეკრება, ხოლო შემდეგ მიღებული ჯამი გამრავლდება 10-ზე.

ფრჩხილებში მოთავსებული ოპერატორები ასევე ექვემდებარებიან პრიორიტეტულობას. ამიტომ, ხშირად გამოიყენება მრავალჯერადად ჩალაგებული ფრჩხილები;

```
$a = ( ($b + $c) * 10 - $d )/ 2 + 9 ;
```

ამ სცენარში ოპერატორების შესრულება მოხდება შემდეგი თანმიმდევრობით:

1. **$b** და **$c** ცვლადების მნიშვნელობების შეკრება. (ეს ოპერატორი ჩასმულია ორმაგ ფრჩხილებში, ამიტომ უმაღლესი პრიორიტეტი გააჩნია).
2.მიღებული ჯამი მრავლდება 10-ზე. (სრულდება ერთჯერად ფრჩხილებში ჩასმული ოპერატორები).
3. მიღებული ნამრავლიდან **$d** ცვლადის მნიშვნელობის გამოკლება.
4. სხვაობის გაყოფა 2-ზე. (ეს ოპერატორები მდებარეობენ ფრჩხილებს გარეთ, ამიტომ მათი შესრულება ხდება ყველაზე ბოლოს).
5. განაყოფისათვის 9-ის მიმატება.

მოკლედ, ყველაფერი საკმაოდ მარტივია. ამიტომ, ჩვენ ჯერჯერობით შევეშვათ ოპერატორებს და გადავიდეთ უფრო რთულ გამოსახულებებზე, რომელთა გარეშეც უფრო რთულ სცენარებში ვერაფერს გავხდებით. ამასთან ერთად, განვიხილოთ კიდევ რამოდენიმე ოპერატორი, რომელთა საშუალებითაც ხდება ასეთი რთული გამოსახულებების შედგენა.

PHP-ს რთული გამოსახულებები

რთულმა გამოსახულებებმა თავიანთი დასახელება მიიღეს იქედან გამომდინარე, რომ ისინი რამოდენიმე მარტივი გამოსახულებებისაგან შედგებიან. რთული გამოსახულებები იყოფიან რამოდენიმე სახეობად და ძირითადად გამოიყენებიან სცენარების კოდების შესრულების სამართავად. ჩვენ ახლა მათ განვიხილავთ.

ბლოკები

რთული გამოსახულებებიდან ყველაზე მარტივია-**ბლოკური გამოსახულება** ან უბრალოდ **ბლოკი**. ის წარმოადგენს ერთ მთლიანობაში გაერთიანებულ, რამოდენიმე უფრო მარტივ გამოსახულებას. ბლოკის შესაქმნელად საკმარისია მისი შემადგენელი მარტივი გამოსახულებების მოთავსება ფიგურულ ფრჩხილებში.

```
{
   $a = 2 ;
   $b = 3 ;
   $c = $a + $b;
   echo $c ;
}
```

შევნიშნოთ, რომ დამხურავი ფიგურული ფრჩხილის შემდეგ წერტილ-მძიმის დასმა საჭირო არ არის.
ჩვეულებრივ ბლოკების დამოუკიდებლად გამოყენება არ ხდება. უფრო ხშირად ისინი შედიან სხვა რთული გამოსახულებების შემადგენლობაში.

პირობითი გამოსახულებები

პირობითი გამოსახულება მასში შემავალი ორი ბლოკიდან ერთ-ერთის შესრულების საშუალებას იძლევა, იმისდა მიხედვით სრულდება თუ არა რაიმე **პირობა**. ასეთი პირობა შეიძლება იყოს ლოგიკური ცვლადის მნიშვნელობა ან ლოგიკური გამოსახულების გამოთვლის შედეგი.
პირობითი გამოსახულება იქმნება განსაკუთრებული ოპერატორების **if** და **else** საშუალებით და შემდეგი ფორმატი გააჩნია:

```
if (<პირობა>)
   { <ბლოკი, რომელიც უნდა შესრულდეს, თუ  პირობა იქნება ჭეშმარიტი> }
else { <ბლოკი, რომელიც უნდა შესრულდეს, თუ პირობა იქნება მცდარი> }
```

პირობა-ეს სწორედ ის ლოგიკური გამოსახულებაა, რომლის მიხედვითაც **PHP** იღებს გადაწყვეტილებას, თუ რომელი ბლოკი უნდა შეასრულოს. თუ პირობა იღებს მნიშვნელობას **true** (”ჭეშმარიტი”), მაშინ სრულდება პირველი ბლოკი. თუ კი პირობას გააჩნია მნიშვნელობა **false** (”მცდარი”), მაშინ სრულდება მეორე ბლოკი. შევნიშნოთ, რომ პირობით გამოსახულებაში გამოყენებული უნდა იყოს მხოლოდ ბლოკები-ეს აუცილებელია!
ქვემოთ მოცემულია პირობითი გამოსახულების მაგალითი:

```
If ($x ==1)
  { $a = “ერთიანი” ;
    $b = 1 ;}
else { $a = “არაერთიანი” ;
    $b = 22222 ; }
```

აქ ჩვენ **$x** ცვლადის მნიშვნელობას ვადარებთ ერთიანს და შედარების შედეგებიდან გამომდინარე, **$a** და **$b** ცვლადებს ვანიჭებთ სხვადასხვა მნიშვნელობებს.
პირობა შეიძლება იყოს უფრო რთულიც:

```
If ( ($x == 1) && ($v > 10)
 { $f = 3; }
else { $f = 33 ; }
```

აქ ჩვენ გამოვიყენეთ რთული პირობა, რომელიც აბრუნებს მნიშვნელობას **true** იმ შემთხვევაში, თუ **$x** ცვლადის მნიშვნელობა ტოლია ერთის და **$v** ცვლადის მნიშვნელობა მეტია ათზე. უნდა შევნიშნოთ აგრეთვე, რომ ამ შემთხვევაშიც კი ჩვენ გამოვიყენეთ ბლოკები, მიუხედავად იმისა, რომ თითოეული მათგანი შედგება მხოლოდ ერთი მარტივი გამოსახულებისაგან.
არსებობს აგრეთვე პირობითი გამოსახულების მეორე ”გადაგვარებული” ნაირსახეობა, რომელიც მხოლოდ ერთ გამოსახულებას შეიცავს და სრულდება პირობის დაკმაყოფილების შემთხვევაში, ხოლო, თუ პირობა არ იქნება დაკმაყოფილებული ხდება მისი გამოიტოვება. ის ასე ჩაიწერება:

```
if (<პირობა>)
```

<გამოსახულება, რომელიც უნდა შესრულდეს, როდესაც პირობა ჭეშმარიტია>

ამ შემთხვევაში ბლოკების გამოყენება სავალდებულო არ არის. მოდით შევხედოთ:

```
if ($flag) {
   $status = “ოპერაცია შესრულებულია.” ;
   $error_code = 0 ;
  }
```

ამ გამოსახულებაში ჩვენ გამოვიყენეთ ბლოკი, ვინაიდან გამოსახულების ჭეშმარიტების შემთხვევაში (ეს არის $flag ცვლადის მნიშვნელობა) ჩვენ დაგვჭირდება ორი მარტივი გამოსახულების შესრულება.

```
if ($x < 0)
   $x = 0 ;
```

აქ კი ჩვენ ბლოკები არ გამოგვიყენებია, ვინაიდან პირობის ჭეშმარიტების შემთხვევაში ჩვენ დაგვჭირდება მხოლოდ ერთი მარტივი გამოსახულების შესრულება.

თუმცა, უკანასკნელი პირობითი გამოსახულების ჩაწერა ჩვენ სხვანაირადაც შეგვიძლია, თუ გამოვიყენებთ განსაკუთრებულ პირობით ოპერატორს-?:

<პირობა> ? <გამოსახულება, რომელიც უნდა შესრულდეს, როდესაც პირობა ჭეშმარიტი იქნება>:
<გამოსახულება, რომელიც უნდა შესრულდეს, როდესაც პირობა მცდარი იქნება>;

აქ ყველაფერი ისევეა, როგორც ”სრულფასოვანი” პირობითი გამოსახულების შემთხვევაში. თუ პირობა ჭეშმარიტია, სრულდება პირველი გამოსახულება, თუ მცდარია-მეორე. ამასთან ამ გამოსახულებებიდან ერთ-ერთის გამოთვლის შედეგი (პირველის ან მეორესი, პირობის ჭეშმარიტებისდა მიხედვით) ოპერატორის მიერ

დაბრუნებული იქნება და შესაძლებელი იქნება მისი დამახსოვრება ცვლადში ან გამოყენება სხვა გამოსახულებაში.
მოდით გადავწეროთ ჩვენი უკანასკნელი პირობითი გამოსახულება შემდეგი პირობითი ოპერატორის საშუალებით:

$x = ($x < 0) ? 0 : x;

როგორც ვხედავთ საკმაოდ კომპაქტური გამოვიდა. მაგრამ საქმე იმაშია, რომ ოპერატორის ? საშუალებით შესაძლებელია მხოლოდ ძალიან მარტივი პირობითი გამოსახულებების ჩაწერა, რომლებშიც ბლოკები არ იქნება გამოყენებული.

შერჩევის გამოსახულებები

შერჩევის გამოსახულება-ეს ფაქტიურად რამოდენიმე პირობითი ოპერატორების ერთიანობაა. მისი ფორმატი ასეთია:

switch [<ცვლადი ან გამოსახულება] {
case <მნიშვნელობა 1>:
  <გამოსახულებების ანაკრები 1>
      [**break** ;]
[**case** <მნიშვნელობა 2> :
<გამოსახულებების ანაკრები 2>
      [**break** ;]
      . . .
< სხვა **case** სექციები>
. . .]
[**default** :
< გამოსახულებების ანაკრები, რომელიც უნდა შესრულდეს დანარჩენი მნიშვნელობებისათვის>]
  }

შერჩევის გამოსახულებებში გამოიყენება ოპერატორები
Switch, case, default da break .
შერჩევის გამოსახულებაში ცვლადის მნიშვნელობა ან გამოსახულების გამოთვლის შედეგი თანმიმდევრულად შედარდება ***მნიშვნელობა1***-თან, ***მნიშვნელობა2*** -თან და ა.შ. და თუ ასეთი შედარება წარმატებულად დასრულდება, შესრულდება შესაბამის ***გამოსახულებათა ანაკრები*** (პირველი, მეორე და ა.შ.). თუ კი არც

ერთი შედარება არ დასრულდა წარმატებით, შესრულდება გამოსახულებათა ანაკრები, რომელიც **default** სექციაში მდებარეობს (თუ რათქმა უნდა ასეთი არსებობს).
ქვემოთ წარმოდგენილია სცენარის მაგალითი, რომელშიც გამოყენებულია შერჩევის გამოსახულება:

```
switch ($a)  {
  case 1:
$out = “ერთიანი” ;
break;
case 2 :
$out = “ორიანი”
break ;
case 3:
$out = “სამიანი” ;
break ;
default :
$out = “სხვა რიცხვი” ;
  }
```

ყურადღება მივაქციოთ იმას, რომ case სექციაში გამოყენებულია არა ბლოკები, არამედ მარტივი გამოსახულებების ანაკრებები-ეს მნიშვნელოვანია! გამოსახულებები უბრალოდ იწერებიან ერთი მეორეს მიყოლებით, მაგრამ არ თავსდებიან ბლოკებში.
ზოგადად, შერჩევის გამოსახულებები ძალიან ”გამჭვირვალენი” არიან და არ მოითხოვენ დამატებით განმარტებებს. მოდით, უმჯობესია სხვა რამეზე ვისაუბროთ, კერძოდ-**break** ოპერატორის შესახებ. რას აკეთებს იგი და შეიძლება თუ არა მის გარეშე არსებობა? როგორც კი წააწყდება ოპერატორ **break**-ს, **PHP** წყვეტს იმ გამოსახულებათა ანაკრების შესრულებას, რომელიც ამ ოპერატორს შეიცავს და იწყებს შერჩევის ოპერატორის მომდევნო კოდის შესრულებას. თუ გამოსახულებათა რომელიმე ანაკრები არ შეიცავს ოპერატორ **break**-ს, მაშინ **PHP** მისი დასრულებისთანავე იწყებს გამოსახულებათა ანაკრების შესრულებას მომდევნო **case** სექციიდან და ა.შ., მანამ სანამ არ შეხვდება **break** ოპერატორს ან სანამ არ დასრულდება შერჩევის გამოსახულება.

ციკლები PHP-ში

PHP-ში, ისევე, როგორც პროგრამირების სხვა ნებისნიერ ენაში მნიშვნელოვან როლს თამაშობენ ციკლები. იმ შემთხვევაშიც კი, როდესაც შესაძლოა არავითარი წარმოდგენა არ გვქონდეს მათ შესახებ, ძნელი მისახვედრი არ არის, რომ **ციკლი**-ეს არის რაიმე განმეორებადი მოქმედებების ერთობლივობა. სინამდვილეშიც სწორედ ასე არის. იქმნება რაიმე ცვლადი, ხოლო შემდეგ იგი იცვლება მანამ, სანამ არ მიაღწევს რაიმე წინასწარ განსაზღვრულ მნიშვნელობას. პრაქტიკაში ეს ძალზე მოხერხებულია. განსაკუთრებულად მარტივი ხდება მასივების შევსება და წაკითხვა, რაიმე ფუნქციების მნიშვნელობების გამოთვლა და ა.შ., რაც ხშირად გამოიყენება სხვადასხვა საიტებზე, მაგალითად სტუმართა წიგნებში და ფორუმებში.ციკლების რამოდენიმე სახეობა არსებობს **while**, **do..while** და **for**.

while (ციკლი წინაპირობით)

ციკლი **while**-ციკლების უმარტივესი სახეობაა. ის ისევე მოქმედებს, როგორც მისი ანალოგიური ოპერატორი პროგრამირების ენა **C**-ში. ოპეატორ **while**-ის ძირითადი ფრმაა:

```
while (პირობა) {

...

}
```

while ოპერტორის შინაარსი მარტივია. ის PHP-ს მიუთითებს ჩალაგებული ოპერატორების შესრულებაზე მანამ, სანამ სრულდება რაიმე პირობა. გამოსახულების მნიშვნელობის შემოწმება ხდება ყოველი ახალი ციკლის დაწყებისას. ასე რომ, ციკლის შიგნით გამოსახულების მნიშვნელობების ცვლილების მიუხედავად იგი არ შეწყდება მანამ, სანამ არ დაიწყება მომდევნო ახალი ციკლი. იმ შემთხვევებში, როდესაც პირობა თავიდანვე არ სრულდება, ციკლი შესაძლოა არც ერთხელ არ შესრულდეს.

თუ ციკლში მხოლოდ ერთი ოპერატორია, მაშინ შესაძლებელია ფიგურული ფრჩხილების გამოტოვება

while (პირობა) ერთი ოპერატორი.

ქვემოთ მოცემულ მაგალითში ხდება 1-დან 10-მდე რიცხვების გამოტანა:

```
$i = 1;

while ($i <= 10) {

echo $i;
$i++;

}
```

do… while (ციკლი პასტპირობით)

ციკლი **do… while** ძალიან წააგავს **while**-ს, იმ გამონაკლისის გარდა, რომ ლოგიკური გამოსახულების მნიშვნელობის შემოწმება ხდება არა იტერაციის დაწყებამდე, არამედ მისი დასრულებისას.

ძირითადი განსხვავება მდგომარეობს იმაში, რომ **do… while** ერთხელ მაინც გარანტირებულად სრულდება, რაც **while**-ის შემთხვევაში აუცილებელი არ არის.

do… while ციკლებისათვის სინტაქსისის მხოლოდ ერთი სახე არსებობს:

```
$i = 0;

do {

echo $i;

} while ($i > 0);
```

ეს ციკლი სრულდება მხოლოდ ერთხელ, ვინაიდან მომდევნო ნაბიჯებისათვის პირობა აღარ სრულდება ($i არ არის მეტი 0-ზე) და ამიტომ, მთელი ციკლიც აქ მთავრდება.
გარდა აღნიშნულისა, ციკლებში აგრეთვე იყენებენ ჩვენთვის უკვე ნაცნობ ოპერატორის-break. მისი საშუალებით შესაძლებელა ნებისმიერი ციკლის, მათ შორის როგორც **while** და **do..while**, ასევე **for**-ის, ნებისმიერ დროს შეწყვეტა.

მაგალითად:

```
$i = 1;

while ($i <= 10) {

echo $i;
$i++;

if($i > 5) break;

}
```

მოცემულ შემთხვევაში ციკლი შეწყდება, როგორც კი ცვლადი $i გახდება 6 ტოლი.

ციკლი for

for ციკლები-ყველაზე მძლავრი ციკლებია **PHP**-ში. ისინიც **C**-ში მათი ანალოგების მსგავსად მუშაობენ. **for** - ციკლის სინტაქსისი ასეთია:

```
for (გამოსახულება; პირობა; ჩალაგებული ოპერატორები)
{

...

}
```

პირველი გამოსახულება უპირობოდ გამოითვლება (სრულდება) ციკლის დასაწყისშივე.
ყოველი ციკლის დაწყების წინ მოწმდება პირობა. თუ იგი სრულდება, მაშინ ციკლი გრძელდება და ხდება ჩალაგებული ოპერატორების შესრულება, თუ არადა ციკლიმთავრდება. ყოველი ციკლის ბოლოს ხდება ჩალაგებული ოპერატორების გამოთვლა (შესრულება).
თითოეული ამ გამოსახულებათაგან შესაძლოა აღმოჩნდეს ცარიელი. თუ პირობა არ არსებობს, მაშინ ციკლი შეიძლება გაგრძელდეს უსასრულოდ (გაჩუმებით, **PHP** მას მიიჩნევს შესრულებულად (**true**-ს ტოლად), ისევე, როგორც პროგრამირების ენა **C**).ეს არცთუ ისე გამოუსადეგარი რამ არის, როგორც ერთი შხედვით შეიძლება მოგვეჩვენოს, ვინაიდან ხშირად ციკლის დასრულებისათვის შეიძლება დაგვჭირდეს **for** ციკლში ლოგიკური გამოსახულების გამოყენების ნაცვლად, ოპერატორ **break**-ის, რაიმე ლოგიკურ პირობასთან ერთად გამოყენება.

განვიხილოთ შემდეგი მაგალითები. ყველა მათგანს გამოაქვს რიცხვები 1-დან 10-მდე:

// **მაგალითი 1**

```
for ($i = 1; $i <= 10; $i++) {

echo $i;

}
```

// **მაგალითი 2**

```
for ($i = 1;;$i++) {

if ($i > 10) break;
echo $i;

}
```

```
// მაგალითი 3

$i = 1;

for (;;) {

if ($i > 10) break;
echo $i;
$i++;
}

// მაგალითი 4

for ($i = 1; $i <= 10; echo $i, $i++);
```

რათქმა უნდა, პირველი და მეოთხე ვარიანტები შესაძლოა უკეთესებად მოგვეჩვენოს, მაგრამ როგორც აღმოჩნდა, თურმე ხშირ შემთხვევებში **for** ციკლში ცარიელი გამოსახულებების გამოყენება უფრო სასარგებლოა.
მაშ ასე, ციკლების შესახებ ზოგადი ცნობების მიღების შემდეგ განვიხილოთ კიდევ რამოდენიმე მაგალითი:

```
$a = 0 ;
Si = 1;
do {
   $a = $a + Si + 2 ;
   $i = ++ Si ;
} while ($ < 20) ;
```

დავაკვირდეთ მას ყურადღებით. ჩვენ ფაქტიურად შევქმენით ციკლი მთვლელით.
საინტერესო თავისებურება: ციკლი პოსტპირობით ერთხელ მაინც სრულდება, თუნდაც ეს პირობა თავიდანვე მცდარი იყოს. ზოგიერთ შემთხვევაში შესაძლებელია ამ თვისების გამოყენება.
ციკლი წინაპირობით განსხვავდება ციკლისაგან პოსტპირობით იმით, რომ პირობა მოწმდება ციკლის სხეულის შესრულებამდე. ასე რომ, თუ პირობა წინასწარვე მცდარია, სხეული არც ერთხელ არ შესრულდება.

წინაპირობიანი ციკლის შესაქმნელად გამოიყენება ჩვენთვის უკვე ნაცნობი ოპერატორი **while**. ამიტომ ასეთ ციკლებს ასევე ეძახიან "**while ციკლებს**".

While (პირობა)
    <ციკლის სხეული>

წინაპირობიანი ციკლის მაგალითია:

```
while ($a < 100)  {
$a = $a * $i + 2 ;
$i = ++$i ;
}
```

ახლა კი მოდით დავაკვირდეთ ასეთ ციკლს:

```
while (true) {
    $i = ++$i ;
}
```

ამ ციკლის პირობა ყოველთვის ჭეშმარიტია-გამოიყენება მნიშვნელობა **true**. ეს ნიშნავს, რომ ასეთი ციკლის შესრულება უსასრულოდ გაგრძელდება (***უსასრულო ციკლი***). ასეთ ციკლებს ხანდახან იყენებენ პროგრამირებაში, მაგრამ ამასთან ერთად ითვალისწინებენ პირობას, რომლის მიხედვითაც შესაძლებელი იქნება უსასრულო ციკლის შეწყვეტა. ციკლის შესრულების შეწყვეტის და ამისათვის განკუთვნილი ორი ოპერატორის შესახებ საჭიროა განსაკუთრებული საუბარი.

შენიშვნა: *სინამდვილეში, უსასრულო ციკლის შესრულება არ ხდება მუდმივად. ჯერ-ერთი, პროგრამისტს თვითონ შეუძლია შეწყვეტოს იგი, მისი შემცველი პროგრამის დახურვით. მეორე-თვით პროგრამაში შეიძლება მოხდეს შეცდომა და ამის გამო დაიხურება ოპერაციული სისტემის ან PHP-ს დამმუშავებლის მიერ. მესამ-როგორც უკვე ნათქვამი იყო, პროგრამისტს შეუძლია გაითვალისწინოს ციკლის შეწყვეტის პირობა.*

ციკლის შეწყვეტა

როგორც უკვე გავარკვიეთ, უსასრულო ციკლი (ხანდახან-ჩვეულებრივი "სასრულიც") ბოლოს და ბოლოს უნდა შეწყდეს. სპეციალურად ამისათვის **PHP**-ში გათვალისწინებულია ორი ოპერატორი: **break** და **continue**.

ოპერატორი break წყვეტს ციკლის შესრულებას, რის შემდეგაც იწყება ციკლის მომდევნო კოდის შესრულება:

```
while ($a < 100) {
$a = $a * $i + 2:
  If ($a > 50)  {
  break ;
}
Si = ++Si ;
}
```

ამ მაგალითში ციკლის შესრულება შეწყდება, თუ **$a** ცვლადის მნიშვნელობა გადააჭარბებს 50.

ოპერატორი **break** ჩვენთვის უკვე ნაცნობია. ის ჩვენს მიერ გამოყენებული იყო შერჩევის გამოსახულებაში, სადაც ზუსტად ასევე იქცეოდა.

ოპერატორი **continue** ახორციელებს ციკლის ხელახლა გაშვებას. ის შეუსრულებელს ტოვებს ციკლის სხეულში შემავალ ყველა მის მომდევნო გამოსახულებას და შესრულებაზე უშვებს ციკლის მომდევნო ნაბიჯს: პირობის შესრულება, მთვლელის მნიშვნელობის გაზრდა, ციკლის სხეულის შესრულება და ა.შ.

```
while ($a < 100) {
Si = ++ Si ;
If ((Si > 9) && (Si < 21)) {
Continue ;
}
Sa = $a * Si + 2 ;
}
```

აქ ჩვენ გამოვტოვებთ სხეულის ბოლო გამოსახულებას, თუ **$i**-ის მნიშვნელობა მოთავსებულია 10-დან 20-მდე დიაპაზონში და ციკლის შესრულებას ვიწყებთ თავიდან.

ყოველივე ამით ჩვენ დავასრულეთ საუბარი რთული გამოსახულებების შესახებ.

ფუნქციები

ფუნქცია-ეს **PHP** კოდის განსაკუთრებული სახით ჩაწერილი და გაფორმებული ფრაგმენტია, რომლის გამოძახებაც შესაძლებელია მოცემულ სერვერულ გვერდში შემავალი, ნებისმიერი სცენარის, ნებისმიერი ადგილიდან. როგორც წესი, ფუნქციის სახით ფორმდება კოდი, რომელიც ასრულებს სცენარებში ხშირად გამოყენებად, ერთი და იმავე ტიპის ამოცანებს. ასეთ შემთხვევებში, იმის ნაცვლად, რომ ასეთი კოდი რამდენჯერმე ჩავწეროთ სხვადასხვა სცენარში, მას ვაფორმებთ ფუნქციის სახით, ხოლო კოდის შესაბამის ადგილებში უბრალოდ ვსვამთ გამოსახულებას, რომელიც მას გამოიძახებს.
საკუთრივ კოდს, რომლისთვისაც ფუნქცია იქმნება, უწოდებენ **ფუნქციის სხეულს** და აფორმებენ ბლოკის სახით. ყოველ ფუნქციას უნდა გააჩნდეს უნიკალური სახელი, რომლის საშუალებითაც მას მიმართავენ. ფუნქციას, ისევე, როგორც ოპერატორს შეუძლია მიიღოს ერთი ან რამოდენიმე არგუმენტი და დააბრუნოს შედეგი, რომლის გამოყენებაც შესაძლებელი იქნება გამოსახულებებში.

ფუნქციების შექმნა

მანამ სანამ ფუნქციას სადმე გამოვიყენებდეთ, უნდა მოხდეს მისი გამოცხადება. ფუნქციის გამოცხადება ხორციელდება გასაღები სიტყვის **function** საშუალებით:

function <ფუნქციის დასახელება> ([მძიმეებით გამოყოფილი ფორმალური არგუმენტების ჩამონათვალი])

<ფუნქციის სხეული>

როგორც უკვე ავღნიშნეთ, ***ფუნქციის სახელი*** უნდა იყოს უნიკალური. ფუნქციების სახელებისათვის იგივე წესები მოქმედებენ, რაც ცვლადების სახელებისათვის: მხოლოდ ლათინური ასოები, ციფრები და ხაზგასმის ნიშნები, ამასთან პირველი უნდა იყოს ან ასო ან ხაზგასმის ნიშანი. მაგრამ-ყურადღება!-დოლარის ნიშანი ფუნქციის სახელის წინ საჭირო არ არის (დოლარის ნიშანი-**PHP** ენაში ცვლადის აღმნიშვნელია). გარდა ამისა, ფუნქციების

სახელები არ არიან დამოკიდებული სიმბოლოების რეგისტრზე, რომელში მათი აკრეფა ხდება. ასე, რომ **Func, func** და **FUNC**-ეს ერთი და იგივე ფუნქციაა.
ფორმალური არგუმენტების სია წარმოადგენს ცვლადების ანაკრებს, რომლებშიც ფუნქციის გამოძახებისას მისი არგუმენტების მნიშვნელობები მოთავსდებიან. ამ ცვლადებისთვის ჩვენ ნებისმიერი სახელების მოფიქრება შეგვიძლია-სულ ერთია, მათი გამოყენება მაინც მხოლოდ **ფუნქციის სხეულის** შიგნით მოხდება. მათ ასეც ეძახიან-**ფორმალური არგუმენტები.**
ფუნქციის **ფორმალური არგუმენტების სია** თავსდება მრგვალ ფრჩხილებში, ფუნქციის სახელის შემდეგ, ხოლო თვით არგუმენტები ერთმანეთისაგან მძიმეებით გამოიყოფიან. იმ შემთხვევაშიც, თუ ფუნქცია არ მოითხოვს არგუმენტებს, ფრჩხილების ჩვენება მაინც საჭიროა.

ფუნქციის სხეულის ფარგლებში, მის მიერ მიღებულ არგუმენტებზე (თუ ისინი არსებობენ) და სხვა მონაცემებზეც, სრულდება გარკვეული მოქმედებები და შესაძლოა შედეგების გამომუშავებაც. იმისათვის, რომ შედეგი ფუნქციიდან იმ გამოსახულებას დაუბრუნდეს, რომლიდანაც ფუნქციის გამოძახება მოხდა, გამოიყენება ოპერატორი **return**:

Return <ცვლადი ან გამოსახულება>;

აქ ***ცვლადი*** უნდა შეიცავდეს დაბრუნებულ მნიშვნელობას, ხოლო ***გამოსახულებამ*** უნდა გამოითვალოს იგი.
ქვემოთ ნაჩვენებია უმარტივესი ფუნქციის გამოცხადების მაგალითი:

```
function divide ($a, $b) {
    $c = $a / $b ;
     return $c ; }
```

მოცემული ფუნქცია იღებს ორ არგუმენტს-$a და $b-რის შემდეგაც $a-ს ყოფს $b-ზე და შედეგის სახით აბრუნებს განაყოფს. ამასთან შუალედური შედეგის დასამახსოვრებლად ის იყენებს საკუთარ (ეგრეთ წოდებულ-***ლოკალურ***) ცვლადს - $c.
ცხადია, შესაძლებელია ცვლადის არ გამოყენებაც, თუ ფუნქციას divide ჩავწერთ ერთ სტრიქონად:

function divide ($a, $b) { return $a / $b }

ასე გაცილებით მოკლე ჩანაწერი გამოდის.
ფუნქციის ზოგიერთ ფორმალურ არგუმენტს შეიძლება მიენიჭოს გაჩუმებითი მნიშვნელობა **არასავალდებულო არგუმენტის** შექნის გზით. გაჩუმებითი მნიშვნელობა საჭირო არგუმენტს, პირდაპირ ფრჩხილების შიგნით, ოპერატორი = საშუალებით მიენიჭება:

function divide ($a, $b =2) {
$c = $a / $b ;
return $c;
}

ამ ფუნქციის გამოძახებისას, შესაძლებელია მეორე არგუმენტის მნიშვნელობის გამოტოვება-მაშინ იგი 2-ის ტოლი გახდება. ამ შემთხვევაში მხედველობაში უნდა ვიქონიოთ, რომ გაჩუმებითი მნიშვნელობების მქონე არგუმენტები უნდა მდებარეობდნენ ფორმალური არგუმენტების სიის ბოლოში.

ფუნქციების გამოძახება

მას მერე, რაც ფუნქციას გამოვაცხადებთ, ჩვენ შეგვიძლია მისი *გამოძახება* მიმდინარე აქტიური სერვერული გვერდის ფარგლებში არსებული ნებისმიერი სცენარიდან. ფუნქციის გამოძახება ხდება ასე:

<ფუნქციის დასახელება>([<მძიმეებით გამოყოფილი ფაქტიური არგუმენტების სია>])

აქ მიეთითება საჭირო ფუნქციის *სახელი* და მრგვალ ფრჩხილებში ჩამოითვლებიან *ფაქტიური არგუმენტები*, რომლებზედაც უნდა განხორციელდეს შესაბამისი მოქმედებები. ფუნქცია დააბრუნებს შედეგს, რომელიც შემდგომში შეიძლება მიენიჭოს ცვლადს ან გამოყენებულ იქნას გამოსახულების გამოსათვლელად.

ყურადღება: *ფუნქციის გამოძახებისას მასში სწორად **ფაქტიური** არგუმენტები უნდა ჩაისვას და არა **ფორმალური**, რომლებიც გამოყენებული იყო ფუნქციის გამოცხადებისას.*

ქვემოთ მოყვანილია ჩვენს მიერ ადრე გამოცხადებული **divide** ფუნქციის გამოძახების მაგალითი

echo divide (3,2) ;

აქ ჩვენ ფუნქციის გამოძახების გამოსახულებაში ჩავსვით ფაქტიური გამოსახულებები 3 და 2.

$s =4 * divide ($x ,$r) + $v ;

აქ კი ჩვენ ვახორციელებთ ფუნქციის გამოძახებას, რომელშიც ფაქტიური არგუმენტების სახით გამოყენებულია ცვლადები. ხოლო ფუნქციის მიერ დაბრუნებული შედეგი გამოიყენება გამოსახულების გამოსათვლელად.
ისეთი ფუნქციის გამოძახება, რომელიც არ აბრუნებს შედეგს კიდევ უფრო მარტივია:

somefunction ($d , $f , 5 , 0) ;

თუმცა, ასეთივე გზით შესაძლებელია შედეგის დამბრუნებელი ნებისმიერი ფუნქციის გამოძახება. ამ შემთხვევაში აღნიშნული შედეგი იგნორირებული იქნება.
ფუნქციის გამოძახებისას შესაძლებელია იმ არგუმენტების გამოტოვება, რომელთათვისაც გაჩუმებითი მნიშვნელობები იყო განსაზღვრული. მაგალითად ასე:

echo divide (3) ;

ამ შემთხვევაში ფუნქცია **divide**-ის მეორე ფორმალური პარამეტრი (**$b**) მიიღებს მნიშვნელობას 2 (იხილეთ ამ ფუნქციის გამოცხადება).
შესაძლებელია ერთი ფუნქციის მეორისაგან გამოძახება:

```
function samplefunc1 ($a , $b) {
. . .
}
function samplefunc2 ($c) {
. . .
```

```
$k = $v + samplefunc1 ($x , 2) ;
```

საჭიროა მხოლოდ ყოველთვის გვახსოვდეს, რომ ფუნქცია უნდა იყოს გამოცხადებული მანამ, სანამ მისი გამოყენება მოხდება.

ცვლადების გამოყენება ფუნქციის სხეულის შიგნით

უნდა ითქვას, რომ ფუნქციებში შუალედური შედეგების შესანახად განსაზღვრულ და გამოყენებულ ცვლადებზე, ღირს ცალკე საუბარი. ასეთ ცვლადებს **ლოკალურ ცვლადებს** უწოდებენ.
თუ ჩვენ გამოვაცხადებთ რაიმე ცვლადს ფუნქციის გარეთ, ჩვენ შეგვეძლება მისადმი, იმავე სერვერულ გვერდზე არსებული ნებისმიერი სცენარიდან მიმართვა. (ყურადღება! ცვლადისადმი მიმართვა ჩვენ შეგვიძლია მხოლოდ მას შემდეგ, რაც მას გამოვაცხადებთ). პროგრამისტები ამბობენ, რომ ასეთი ცვლადი "ხილულია" გვერდის შიგნიდან და მას **გვერდის დონის ცვლადს** უწოდებენ.
რაც შეეხება ფუნქციის სხეულის ფარგლებში გამოცხადებულ ცვლადებს, ისინი ლოკალურებად ითვლებიან და "ხილულები" არიან მისი სხეულის გარეთ. ანუ ასეთი სცენარი არ იმუშავებს:

```
function func2 () {
$c = 2 ;
return $a * $b + $c ;
}
echo $c ;
```

თუ გავაკეთებთ ცნობილი ანდაზის პერეფრაზს: "საკუთარი ცვლადი ურო ახლოა **"ფუნქციის სხეულთან"**
ახლა კი-ყურადღება! თუ ჩვენ შევეცდებით ფუნქციის გარეთ გამოცხადებული ცვლადისადმი მიმართვას მისი სხეულის შიგნით, ჩვენ აუცილებლად განვიცდით წარუმატებლობას. **PHP**-ს ეგონება, რომ ჩვენ ვცდილობთ ფუნქციის ჯერ გამოუცხადებელი ლოკალური ცვლადისადმი მიმართვას. ხოლო გამოუცხადებელ ცვლადებს ყოველთვის **NULL**-ის ტოლი მნიშვნელობა გააჩნიათ.
მაგრამ როგორ მოვიქცეთ, თუ ჩვენ გვინდა, რომ ფუნქციის სხეულის გარეთ გამოცხადებული ცვლადი "გამოჩნდეს" მის შიგნით? ამისათვის ეს ცვლადი უნდა გამოვაცხადოთ, როგორც გლობალური,

ანუ ყველა მხრიდან ხელმისაწვდომი. მსგავსი გამოცხადების სინტაქსისი ასეთია:

global <მძიმეებით გამოყოფილი ცვლადების სახელების სია>

ქვემოთ მოცემულია სცენარის მაგალითი, რომელშიც გამოყენებულია გლობალური ცვლადები:

```
$a = 3 ;
$b = 4 ;
function func1 () {
global $a , $b ;
return $a *$b ;
}
Echo func () ;
```

ამ სცენარის მუშაობის შედეგად ეკრანზე გამოისახება მნიშვნელობა 12 (3 და 4 ნამრავლი).
მას შემდეგ, რაც ფუნქციის სხეულის შესრულება დასრულდება, მასში გამოცხადებული ყველა ლოკალური ცვლადი ნადგურდება. ხანდახან საჭიროა, რომ განადგურების ნაცვლად, მოხდეს მათი მნიშვნელობების შენარჩუნება, რისთვისაც საკმარისია მათი გამოცხადების გამოსახულების დასაწყისში გასაღები სიტყვის **static** გამოყენება.

```
function count () {
    static $i = 0 ;
    return ++Si ;
}
```

ამ ფუნქციის პირველი გამოძახებისას იქმნება ლოკალური ცვლადი **$i** და ის იღებს მნიშვნელობას 0. შემდგომში, ფუნქციის სხეულში, ეს მნიშვნელობა განიცდის ინკრემენტაციას (ზრდას) და ბრუნდება შედეგის სახით. როდესაც ფუნქციის სხეულის შესრულება დასრულდება, ცვლადი **Si** თავის მნიშვნელობასთან ერთად შეინახება მეხსიერებაში და გამოყენებული იქნება ამ ფუნქციის მომდევნო გამოძახებების დროს.

ამგვარად, ფუნქცია **count** ყოველი გამოძახებისას მოახდენს **$i** ცვლადის მნიშვნელობის ინკრემენტაციას და შედეგის დაბრუნებას.
ასეთ ცვლადებს, რომლებიც თავიანთ მნიშვნელობებს ინახავენ ფუნქციის გამოძახებებს შორის პერიოდის განმავლობაში, **სტატიკურ ცვლადებს** უწოდებენ.

PHP-ს ჩაშენებული ფუნქციები

ჩვენ უკვე გავიგეთ, თუ როგორ უნდა შევქმნათ ჩვენი საკუთარი ფუნქციები. მაგრამ **PHP** ენა გვთავაზობს აგრეთვე მასში უკვე არსებული (**ჩაშენებული**) ფუნქციების უზარმაზარ ანაკრებს, რომელთა გამოყენება ჩვენ ასევე შეგვიძლია ჩვენს სცენარებში.
ყველა ჩაშენებული ფუნქციების სრული აღწერა წარმოდგენილია **PHP**-ს ინტერაქტიულ დოკუმენტაციაში. აქ ჩვენ მათგან მხოლოდ რამოდენიმეს განვიხილავთ.
შედეგის არ დამბრუნებელი ფუნქცია **unset** საშუალებას გვაძლევს მეხსიერებიდან ამოვშალოთ არასაჭირო ცვლადი. მისი გამოძახების ფორმატი ასეთია:

unset (<მძიმეებით გამოყოფილი, წასაშლელი ცვლადების სია>) ;

მაგალითად:

Unset ($b , $b , $c) ;

ფუნქცია **gettype** შედეგის სახით აბრუნებს სტრიქონს, რომელიც აღწერს ამ ფუნქციისათვის გადაცემული არგუმენტის მონაცემების ტიპს:

gettype (<ცვლადი ან გამოსახულება>) ;

ამ ფუნქციას შეუძლია შემდეგი სტრიქონული მნიშვნელობებიდან ერთ-ერთის დაბრუნება:

-**boolean**- ლოგიკური ტიპი;
-**integer**- მთელი რიცხვის ტიპი;
-**double**- ტიპი მცოცავი მძიმით;
-**string**-სტრიქონული ტიპი;
-**NULL**

შემდეგში, როდესაც ჩვენ დავიწყებთ რეალური სცენარების წერას, ჩვენ კიდევ რამოდენიმე ჩაშენებულ **PHP** ფუნქციას შევისწავლით. ამჯერად კი ამით დავასრულოთ ფუნქციებთან დაკავშირებული საკითხების შესწავლა.

მასივები

ჩვენ უკვე საკმაოდ ბევრი ვიცით ცვლადების და მათთან მუშაობის შესახებ. მაგრამ ჩვენი ცოდნა ჯერ კიდევ არ არის სრული. ასე მაგალითად, ჩვენ ჯერ არაფელი არ ვიცით მასივების-მონაცემების შენახვის განსაკუთრებული ხერხის შესახებ, რომელიც ხელმისაწვდომია **PHP**-ში. მოდით გავარკვიოთ, თუ რა არის ეს.

მასივების შექმნა და მათთან მუშაობა

მასივი-ეს გადანომრილი ცვლადების ერთობლივობაა, რომელიც ერთ მთლიანობას წარმოადგენს. მასივში შემავალ ცვლადებს, მის **ელემენტებს** უწოდებენ. მასივის საჭირო ელემენტისადმი წვდომა ხორციელდება მისი რიგითი ნომრის მიხედვით, რომელსაც **ინდექსი** ეწოდება. ხოლო მასივის ელემენტების საერთო რაოდენობას მის **ზომას** უწოდებენ.

მასივები იდეალური საშუალებაა იმ შემთხვევებში, როდესაც საჭიროა ერთსა და იმავე სტრუქტურაში შენახულ იქნას ერთი და იმავე ტიპის მონაცემთა ერთობლივობა. ასე მაგალითად, მასივში შეიძლება შევინახოთ კვირის ყველა შვიდი დღის დასახელებები, ხოლო შემდეგ მივიღოთ საჭირო დასახელება მისი ნომრის მიხედვით, რომელიც ამ მასივის ინდექსი იქნება.

მასივის შესაქმნელად, საკმარისია ნებისმიერ ცვლადს უბრალოდ მივანიჭოთ მისი ერთმანეთისაგან მძიმეებით გამოყოფილი და კვადრატულ ფრჩხილებში მოთავსებული ელემენტების სია:

$days = [“ორშაბათი” , “სამშაბათ” , “ოთხშაბათი” , “ხუთშაბათი” , “პარასკევი” , “შაბათი” , “კვირა”] ;

ჩვენს მიერ შექმნილი მასივის **$days** ლემენტები მიიღებენ ინდექსებს 0-დან 6-მდე. ხოლო **$days** მასივის ზომა ტოლი იქნება 7.

ყურადღება: *მასივის ელემენტების ნუმერაცია იწყება ნულიდან და არა ერთიდან.*

ამის შემდეგ, მასივის საჭირო ელემენტთან წვდომისათვის, საჭიროა მასივის სახელის შემდეგ, ამ ელემენტის ინდექსის მითითება კვადრატულ ფრჩხილებში.

echo $days [2] ;

ეს გამოსახულება ეკრანზე გამოიტანს $days მასივის მესამე ელემენტის მნიშვნელობას, ანუ სტრიქონს **ოთხშაბათი**.
არსებობს მასივის შექმნის მეორე ხერხიც-განსაკუთრებული ფუნქციის **array** გამოყენება.

$days = array [“ორშაბათი” , “სამშაბათ” , “ოთხშაბათი” , “ხუთშაბათი” , “პარასკევი” , “შაბათი” , “კვირა”] ;

ცხადია, რომ ამ შემთხვევაში მომავალი მასივის ელემენტები უბრალოდ მიეთითებიან ფუნქციის არგუმენტების სახით.
თუ საჭირო იქნება, ჩვენ ადვილად შევძლებთ ადრე შექმნილ მასივში კიდევ ერთი ელემენტის დამატებას, მისთის უბრალოდ საჭირო მნიშვნელობის მინიჭებით. მაგალითად, ასე:

$arr = [1 , 2 , 3 , 4] ;
$arr [4] = 9

აქ ჩვენ ჯერ შევქმენით მასივი $arr ოთხი ელემენტისაგან, ხოლო შემდეგ დავუმატეთ მას კიდევ ერთი, რიგით მეხუთე ელემენტი, რომლის მნიშვნელობა 9 ტოლია.
შესაძლებელია გაკეთდეს ასეც:

$arr [7] = 23 ;

ეს გამოსახულება დაამატებს მასივში $arr კიდევ ერთ, მეექვსე ელემენტს მნიშვნელობით 23. მაგრამ მისი ინდექსი 7 ტოლი იქნება. ძალიან საინტერესო შესაძლებლობაა.
თუ კი ჩვენ შევქმნით მასივის ახალ ელემენტს შემდეგი სახის გამოსახულების საშუალებით

$arr [] =888 ;

ანუ, არ მივუთითებთ ფრჩხილებში შესაქმნელი ელემენტის ინდექსს, **PHP** თვითონ მიანიჭებს მას ინდექსს, რომელიც ტოლი იქნება უკანასკნელი ელემენტის ინდექსს მიმატებული ერთიანი (ჩვენს შემთხვევაში-7+1=8).
უნდა აღინიშნოს, რომ მასივის შესაქმნელად ფუნქცია **array**-ის გამოყენებისას, ჩვენ შეგვიძლია პირდაპირ მასში მივუთითოთ ამ მასივის ელემენტების ინდექსები:

$arr = [1 ,2 ,3 ,4 ,9 ,7 => 23 , 888] ;

ეს გამოსახულება შექმნის შვიდი ელემენტისაგან შემდგარ $arr მასივს. პირველ ხუთს ექნება ინდექსები 0-დან 5-მდე შესაბამისად. მეექვსე ელემენტი მიიღებს ინდექსს-7 (7 => 23 სახის ჩანაწერში, მარცხნივ იწერება მასივის შესაქმნელი ელემენტის ინდექსი, ხოლო მარჯვნივ-ამ ელემენტის მნიშვნელობა). ბოლო მეშვიდე ელემენტი მიიღებს რიგით მომდევნო "თავისუფალ" ინდექსს-8.
PHP-ს კიდევ ერთი შესანიშნავი შესაძლებლობა-ეს არის მასივის ელემენტებისათვის სტრიქონული ინდექსების მინიჭება:

$digits = array ("ერთი" => 1, "ორი" => 2, "სამი" => 3) ;
Echo $digits ["ორი"] +2

ამ სცენარის პირველი გამოსახულება შექმნის შესაბამისად **ერთი**, **ორი** და **სამი** ინდექსების მქონე, სამი ელემენტისაგან შემდგარ მასივს. ხოლო მეორე გამოსახულება ამოიღებს **ორი** ინდექსის მქონე ელემენტის მნიშვნელობას, მიუმატებს მას 2 და მიღებულ ჯამს გამოიტანს ეკრანზე.

შენიშვნა: *დაპროგრამების მრავალ ენაში მასივებს, რომელთა ელემენტებს სტრიქონული ინდექსები გააჩნიათ, უწოდებენ* ***ასოციატურ მასივებს*** *ან* ***ხეშებს****. **PHP**-ში კი ასეთ მასივებს არანაირი განსაკუთრებული დასახელება არ გააჩნიათ-მათ უბრალოდ "მასივებს" უწოდებენ.*

ჩვენ შეგვიძლია მასივის ნებისმიერ ელემენტს მივანიჭოთ მეორე მასივი (ან როგორც გამოცდილი პროგრამისტები ამბობენ, შევქმნათ **ჩალაგებული** მასივი).

$arr[16] = [1=> "n1", "n2", 10 => "n10"] ;

ამის შემდეგ შესაძლებელი იქნება ჩალაგებული მასივის ნებისმიერი ელემენტისადმი წვდომა, თუ მასივის სახელის შემდეგ მიმდევრობით ორივე ინდექსის მივუთითებთ, ამასთან ყოველი ინდექსი უნდა იყოს მოთავსებული კვადრატულ ფრჩხილებში:

$str = $arr [6] [2] ;

ცვლადი $str მნიშვნელობის სახით მიიღებს სტრიქონს, რომელიც ჩალაგებული მასივის მეორე ელემენტშია მოთავსებული-**n2**.
მასივის გამოუყენებელი ელემენტის ან ერთდროულად მთელი მასივის წასაშლელად, შეგვიძლია უკვე ჩვენთვის ნაცნობი ფუნქციის **unset** გამოყენება:

unset ($arr [6]) ;
unset ($digits) ;

ადრე განხილული სცენარის პირველი გამოსახულება წაშლის **$arr** მასივის ელემენტს ინდექსით 6 (ეს ელემენტი შეიცავს ჩალაგებულ მასივს). მეორე გამოსახულება კი შლის **$digits** მასივს მთლიანად.
მხოლოდ ის დაგვრჩა სათქმელი, რომ ფუნქცია **gettype** მასივისათვის აბრუნებს სტრიქონს **array**.

დათვალიერების ციკლი

ადრე ჩვენ შევისწავლეთ **PHP**-ში გათვალისწინებული ციკლების სამი ნაირსახეობა. მაგრამ მას კიდევ ერთი ციკლი გააჩნია, რომელიც სპეციალურად მასივებთან სამუშაოდ არის განკუთვნილი. ეს ეგრეთ წოდებული **დათვალიერების ციკლია**, რომელიც მასივის ყველა ელემენტებზე გარკვეული მოქმედებების შესრულების შესაძლებლობას იძლევა.
დათვალიერების ციკლი იქმნება ოპერატორ **foreach** საშუალებით, ამიტომ მას ხშირად "**foreach ციკლს**"-აც უწოდებენ:

foreach (<მასივის დასახელება> as <ცვლადი-ინდექსი>=> <ცვლადი-მნიშვნელობა>)
<ციკლის სხეული>

აქ, ფრჩხილებში ჯერ იმ მასივის სახელი მიეთითება, რომლის ელემენტებზეც უნდა განხორციელდეს ციკლის სხეულში მოცემული მოქმედებები. შემდგომ, გასაღები სიტყვის as, მერე მიეთითება იმ ცვლადის სახელი, რომელშიც უნდა შეტანილი იქნას მასივის მორიგი ელემენტის ინდექსი, ხოლო ნიშნის => შემდეგ-ცვლადის სახელი, რომელსაც მიენიჭება ამ ელემენტის მნიშვნელობა.
ქვემოთ მოცემულია დათვალიერების ციკლის მაგალითი:

```
foreach ($days as $index => $day) {
echo “$days [“ . $index. “]=” . $day . “\r\n” ;
}
```

ეს პატარა სცენარი სტრიქონ-სტრიქონ (მიაქციეთ ყურადღება სპეციალურ სიმბოლოს **\r\n**, რომელიც განსაზღვრავს ახალ სტრიქონზე გადასვლას) გამოიტანს ეკრანზე **$days** მასივის ყველა ელემენტების მნიშვნელობებს, მათ ინდექსებთან ერთად.

არსებობს დათვალიერების ციკლის შემოკლებული ვარიანტიც, რომელიც ასე იწერება:

foreach (<მასივის დასახელება> as <ცვლადი-მნიშვნელობა>
<ციკლის სხეული>

იგი გამოიყენება, თუ მასივის ელემენტების ინდექსების მნიშვნელობები საჭირო არ არის.

```
foreach ($days as $day) {
    echo $day .  “\r\n “ ;
}
```

კონსტანტები

PHP გვთავაზობს კიდევ ერთ შესანიშნავ შესაძლებლობას, რომლის შესახებაც აუცილებლად უნდა ითქვას. ეს არის **კონსტანტები**-

რიცხვითი, სტრიქონული ან ლოგიკური მნიშვნელობები, რომელთაც მინიჭებული აქვთ გარკვეული სახელები.
კონსტანტების შექმნა ხდება ჩაშენებული ფუნქციის **define** საშუალებით, რომელსაც ასეთი გამოძახების ფორმატი გააჩნია:

define (<კონსტანტის მნიშვნელობა>, <კონსტანტის დასახელება>) ;

ამ ფუნქციის პირველი არგუმენტი უნდა იყოს კონსტანტის მნიშვნელობა სტრიქონულ, რიცხვით ან ლოგიკურ ფორმატში. მეორე არგუმენტი-ეს არის კონსტანტის სახელი აუცილებლად სტრიქონულ ფორმატში. კონსტანტების სახელებს არ გააჩნიათ წინ დოლარის ნიშანი და ისინი მგრძნობიარენი არიან სიმბოლოების რეგისტრისადმი, რომლითაც ხდება მათი აკრეფა. ფუნქცია **difine** არ აბრუნებს მნიშვნელობას.

მოდით მაგალითისათვის შევქმნათ რაიმე კონსტანტა:

define (“PHP” , “PLATFORM”) ;

ეს გამოსახულება შექმნის კონსტანტას **PLATFORM**, რომელსაც ექნება **PHP**-ს სტრიქონული მნიშვნელობა. მიაქციეთ ყურადღება იმას, რომ კონსტანტის სახელი აკრეფილია დიდი ასოებით-ეს თავისებური დე-ფაქტო სტანდარტია **PHP**-ში.
კონსტანტის შექმნის შემდეგ, ჩვენ შეგვიძლია მისი გამოყენება სადმე ჩვენ სცენარებში:

echo “ ჩვენ ვმუშაობთ“ . PLATFORM-თან **;**

ამ შემთხვევაშიც კონსტანტის სახელს წინ არ გააჩნია დოლარის ნიშანი.
ცვლადებისაგან კონსტანტები იმით განსხვავდებიან, რომ მათი მნიშვნელობა ყოველთვის მუდმივია და შეცვლა არ შეიძლება. ასე მაგალითად, თუ ჩვენ ჩავწერთ:

PLATFORM = “ASP” ;

მაშინ მივიღებთ შეტყობინებას შეცდომის შესახებ.

PHP აგრეთვე გვთავაზობს ჩაშენებული კონსტანტების მცირე ნაკრებს, რომელიც თვითონ მასშია განსაზღვრული. მაგალითად, კონსტანტა **_FILE_** სტრიქონული სახით შეიცავს იმ **PHP** ფაილამდე ბილიკს და მის დასახელებას, რომელშიც მოთავსებულია მოცემულ მომენტში შესრულებადი სცენარი.
კონსტანტების საშუალებით შესაძლებელია **PHP** სცენარებში გამოსაყენებელი, მუდმივი მნიშვნელობების რაიმე ანაკრების შექმნა და შენახვა ერთ რომელიმე ადგილას. თუ კი შემდგომში საჭიროება მოითხოვს ერთ-ერთი მნიშვნელობის შეცვლას, ამის გაკეთება შესაძლებელი იქნება მხოლოდ იმ გამოსახულებაში, რომლითაც შეიქმნა შესაბამისი კონსტანტა და არ დაგვჭირდება ყველა იმ სცენარების ჩასწორება, სადაც ანაკრებში შემავალი კონსტანტები იქნა გამოყენებული.

კომენტარები

ძალიან ხშირად სცენარების დაწერისას, ჩნდება სცენარის კოდში რაიმე შენიშვნების (დამხმარე ინფორმაციის) მოთავსების აუცილებლობა. ამისათვის გამოიყენებიან **PHP** ენის განსაკუთრებული გამოსახულებები, რომელთაც **კომენტარებს** უწოდებენ. **PHP**-ს დამმუშავებლის მიერ ხდება კომენტარების გამოტოვება, ამიტომ მათში შესაძლებელია ნებისმიერი რამის ჩაწერა. კომენტარების **ActionScript** კოდში ჩასასმელად გათვალისწინებულია სამი ოპერატორი: **// , # , და /*. . .* / '.** პირველი და მეორე, ნებისმიერი გამოსახულების ბოლოში, ერთსტრიქონიანი კომენტარის ჩასმის საშუალებას იძლევიან:

```
// ეს არის ერთსტრიქონიანი კომენტარი
$a = $b + $c #  ესეც ერთსტრიქონიანი კომენტარია
```

შევნიშნოთ, რომ კომენტარი იწერება გამოსახულების დასასრულის აღმნიშვნელი წერტილ-მძიმის ნიშნის შემდეგ.
ოპერატორი **/*. . .* /** სცენარის კოდში ნებისმიერი ზომის კომენტარის ჩასმის საშუალებას იძლევა:

```
/* ამ გამოსახულებაში ჩვენ ვკრიბავთ ორი ცვლადის შიგთავსს და
შედეგს ვათავსებთ მესამეში
*/
$a = $b + $c
```

ბუნებრივია, რომ მარტივი კოდის კომენტირება არც ღირს-ისედაც ნათელია, რომ ვახორციელებთ ორი ცვლადის აჯამვას და მიღებულ შედეგს ვათავსებთ მესამე ცვლადში. რაც შეეხება უფრო რთულ კოდს, აქ კომენტარი შესაძლოა მეტად სასარგებლო იყოს. სხვა შემთხვევაში, შეიძლება დროთა განმავლობაში ჩვენ თვითონ დაგვავიწყდეს თუ რას აკეთებს და როგორ მუშაობს იგი.

რა იქნება ამის მერე?

ძალიან ნუ შევყვებით **PHP**-ს შესაძლებლობების შესწავლას. ის ძირითადი შესაძლებლობები, რასაც ჩვენ გავეცანით, საკმარისი უნდა იყოს დასაწყისისათვის. თუ კიდევ რაიმე დაგვჭირდება **PHP**-სთან დაკავშირებით, ჩვენ მას შევისწავლით გზადაგზა, საქმის მსვლელობისას.
ახლა კი - წავიწიოთ წინ! კმარა თეორია, პრაქტიკული საქმიანობის დრო დადგა. შემდეგ თავში დავიწყებთ ჩვენი პირველი, აქტიური სერვერული ვებ-გვერდების შექმნას **PHP** ენაზე. ამაში დაგვეხმარება ჩვენს მიერ რამდენადმე დავიწყებული **DreamweaverMX**-ი.

თავი 8

უმარტივესი სერვერული ვებ-გვერდები.

მონაცემების გამოტანა

თეორიული კურსი და ვებ-დიზაინთან პირდაპირი კავშირის არ მქონე სხვადასხვა მოსამზადებელი სამუშაოები დავასრულეთ. ჩვენ უკვე გვაქვს მონაცემთა ბაზა, მოკლედ გავიარეთ **PHP**-ს კურსიც და მზად ვართ შემდგომი სამუშაოებისათვის.
მოდით, შევუდგეთ ჩვენი ახალი ვებ-საიტის შექმნას ყველაზე თანამედროვე ინტერნეტ-ტექნოლოგიების გამოყენებით. შევქმნათ საქაღალდე ამ საიტის ლოკალური ასლის ფაილებისათვის და დავარქვათ მას **Site2**, მასში გადმოვაკოპიროთ ჩვენი საიტის მთავარი გვერდი **default.htm**-ადრინდელი დათქმის შესაბამისად, იმის შესახებ, რომ მცირედი ცვლილებებით მას გამოვიყენებდით ახალ საიტშიც. ყოველივე ამის შემდეგ გავუშვათ **DreamweaverMX**-ი.

როდესაც **DreamweaverMX**-ი ჩაიტვირთება, გავხსნათ ჩვენი ახალი საიტის **default.htm** გვერდი და შევასწოროთ ჰიპერმინიშნებების-**ფაილები** და **სტატიები** ინტერნეტ-მისამართები. ისინი უნდა მიუთითებდნენ ერთსა და იმავე ვებ-გვერდზე-**Categories.php**, რომელიც შესაბამისად ფაილებისა და სტატიების კატეგორიების სიას შეიცავს. ჩვენ პირველ რიგში სწორედ ამ გვერდზე მუშაობით დავკავდებით.
მაგრამ ჯერ, როგორც წესი დავიწყოთ მცირედი მოსამზადებელი სამუშაოებით.

მომზადება სერვერული გვერდების შესაქმნელად

მოდით მოვაწყოთ მცირედი რევიზია-შევამოწმოთ, გვაქვს თუ არა ყველაფერი რაც საჭიროა **DreamweaverMX**-ის გარემოში აქტიური სერვერული ვებ-გვერდების შესაქმნელად. საჭიროა არცთუ ისე ბევრი რამ და მათგან თითქმის ყველაფერი უკვე გვაქვს.

1.**მთლიანად გამართული და მომუშავე ვებ-სერვერი.** უკვე დავაყენეთ ჩვენს კომპიუტერზე ვებ-სერვერი **Apache** (მისი დაყენებისა და გამართვის პროცესი აღწერილია ***დანართ 1***-ში).
2.**მთლიანად გამართული და მომუშავე მონაცემთა სერვერი.** ჩვენ უკვე გვაქვს კომპიუტერზე **MySQL** სერვერი, რომელთანაც უკვე ვიმუშავეთ კიდევაც, როდესაც ვსწავლობდით **მე-5 თავს**-ვქმნიდით ჩვენს მონაცემთა ბაზას (მისი დაყენების და გამართვის პროცესი აღწერილია **დანართ 2**-ში).
3. **მთლიანად გამართული, მომუშავე და ვებ-სერვერთან ინტეგრირებული PHP დამმუშავებელი.** ისიც ჩვენ უკვე გვაქვს-ჩვენ დავაყენეთ იგი, როდესაც ვსწავლობდით მე-**7 თავს.** (მისი დაყენებისა და გამართვის პროცესი აღწერილია დანართ 3-ში).
სიის პირველი სამი პუნქტი ჩვენ უკვე გვაქვს. ხოლო, რაც შეეხება მეოთხეს?...

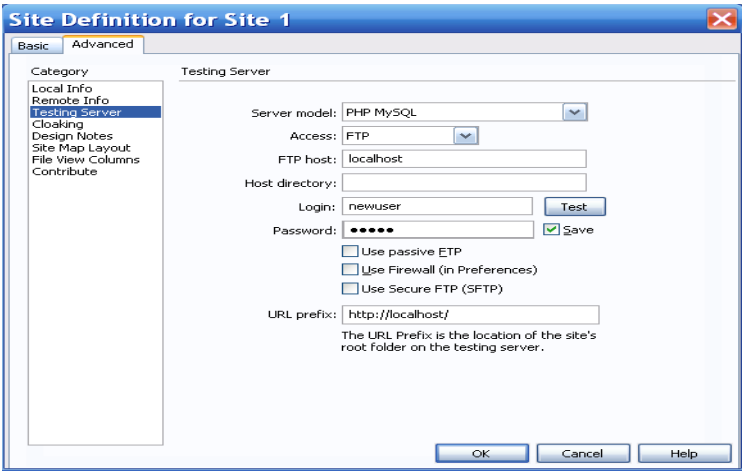
თუ რანაირად უნდა დავარეგისტრიროთ ვებ-საიტი **DreamweaverMX**-ში ჩვენ ვიცით-უკვე შევასრულეთ ეს ოპერაცია მე-4 თავში, როდესაც გამოვაქვეყნეთ მხოლოდ სტატიკური ვებ-გვერდების შემცველი ჩვენი პირველი საიტი. მაგრამ საქმე იმაშია, რომ აქტიური სერვერული გვერდების შემცველი საიტების დარეგისტრირება ხდება **DreamweaverMX**-ში ზოგიერთი

დამატებითი მონაცემების შეტანის გზით, განსაკუთრებული წესის მიხედვით. თუ როგორ ამას ჩვენ ახლავე გავიგებთ.
მაშ ასე, მენიუ **Site**-ში ვირჩევთ პუნქტს **Manage Sites** და ეკრანზე გამოსულ დიალოგურ ფანჯარაში **Manage Sites** ვაჭერთ ღილაკს **New** (იხ.ნახ. 4.1.). ამის შემდეგ ეკრანზე გამოჩნდება პატარა ზომის მენიუ, რომელშიც ვირჩევთ პუნქტს **Site**. ეკრანზე კიდევ გამოჩნდება მეორე დიალოგური ფანჯარა-**Site Definition** (იხ. ნახ. 4.2). ამ ფანჯარაში ჩავრთოთ **Local Info** კატეგორია და შევიტანოთ ცნობები ჩვენი ახალი საიტის ლოკალური ასლის შესახებ. ისინი იგივე იქნება, რაც ძველ საიტს ჰქონდა, მხოლოდ იმ განსხვავებით, რომ საქაღალდის და შესაბამისად საიტის სახელები იქნება არა **Site1**, არამედ Site2. შემდეგ, თანმიმდევრობით გადავერთოთ კატეგორიებში **Remote Info** და **Site Layout** და შევიტანოთ მათში მონაცემები; ისინი იგივე იქნება, რაც ძველი საიტისათვის, ყოველგვარი ცვლილებების გარეშე. ყოველივე ეს ჩვენთვის უკვე ნაცნობია.
ახლა კი, ჩვენ მზად ვართ იმისთვის, რომ შევიტანოთ **DreamweaverMX**-ში, სერვერული გვერდების შექმნის, ჩვენს მიერ არჩეული ტექნოლოგიის სწორად მხარდაჭერისათვის, საჭირო დამატებითი მონაცემები. ამისათვის ვირჩევთ Site **Definition** დიალოგური ფანჯრის **Testing Server** კატეგორიას (ნახ.8.1.).
ჩამოსაშლელი სიის **Server model** საშუალებით, განისაზღვრება ჩვენს მიერ გამოსაყენებელი, სერვერული გვერდების შექმნის, თვით ტექნოლოგია. რადგანაც ჩვენ ვაპირებთ "**PHP**"-ს **My SQL**-თან კავშირის გამოყენებას, ჩვენ უნდა ავირჩიოთ პუნქტი **PHP MySQL**. პრინციპში, შეიძლება დავასრულოთ ამით და დავაჭიროთ ღილაკს **OK**. მაგრამ თუ ასე მოვიქცევით, ვერ გამოვიყენებთ **DreamweaverMX**-ის ერთ-ერთ ყველაზე ღირსშესანიშნავ შესაძლებლობას-სერვერული გვერდების პირდაპირ მისივე გარემოში ("ცოცხალი" დათვალიერება) გახსნის შესაძლებლობას. მითუმეტეს, რომ "ცოცხალი" დათვალიერების გამოსაყენებლად ჩვენ სულ ცოტა რამის გაკეთება დაგვჭირდება.
როგორ მუშაობს "ცოცხალი" დათვალიერების რეჟიმი? ძალიან მარტივად. დავუშვათ, რომ ჩვენ გვინდა რაიმე სერვერული გვერდის მუშაობის შემოწმება. ჩამოვთვალოთ ამისათვის საჭირო ყველა მოქმედება:
1.ამ ვებ-გვერდის ყველა ფაილის (თვით ვებ-გვერდის, გამოსახულებების, სტილების ცხრილების და ა.შ.) კოპირება ვებ-სერვერის ძირეულ ან ერთ-ერთ ვირტუალურ საქაღალდეში.

აღნიშნული მოქმედების წინ, შესაძლებელია მოგვიწიოს ამ გვერდის შენახვა, თუ იგი ჩვენ ახლახან შევასწორეთ **DreamweaverMX**-ში და ჯერ კიდევ არ შეგვინახავს.

2.ვებ-დამთვალიერებლის გახსნა.

3. ვებ-დამთვალიერებლის მისამართების სტრიქონში, ვებ-სერვერისა და ჩვენთვის საჭირო ვებ-გვერდის ფაილის სახელების შეტანა.



ნახ.8.1. დიალოგური ფანჯარა Site Definition (კატეგორია Testing Server)

ამის გაკეთება რათქმა უნდა რთული არ არის. მაგრამ მოდით წარმოვიდგინოთ, რომ ამ მოქმედებების შესრულება ჩვენ ყოველწუთიერად დაგვჭირდება, რომ დავინახოთ ვებ-გვერდის ყოველი შესწორების შედეგი. **DreamweaverMX**-ს კი ჩვენს ნაცვლად, თვითონ შეუძლია ამ სამი მოქმედების შესრულება და პირდაპირ თავის საკუთარ გარემოში გახსნის სერვერულ გვერდს. ჩვენ ვებ-დამთვალიერებლის გაშვებაც კი აღარ დაგვჭირდება!
არ შეიძლება ასეთი შესაძლებლობის არ გამოყენება. ასე, რომ მოდით მივცეთ **DreamweaverMX**-ს ყველა საჭირო მონაცემები, რათა მან შეძლოს "ცოცხალი" დათვალიერების რეჟიმის გაშვება.

ჯერ **Access**-ის ჩამოსაშლელ სიაში ავირჩიოთ დასატესტირებელი სერვერული გვერდების, ვებ-სერვერის საქაღალდეში (ამ საქაღალდეს დავარქვათ ტესტური, რათა სხვებში არ აგვერიოს) გადმოკოპირების ხერხი. რადგანაც ვებ-სერვერი დაყენებულია ჩვენივე კომპიუტერზე, ავირჩიოთ პუნქტი **Local/Network**-კოპირება ლოკალურ საქაღალდეში ან ქსელში.

ამის შემდეგ შესამოწმებელია, ჩასვა თუ არა **DreamweaverMX**-მა შესატან ველში **Testing server folder**, ბილიკი ჩვენი საიტის ძირეულ საქაღალდემდე, რომელიც ჩვენ განვსაზღვრეთ კატეგორიაში **Remote Info**. სწორედ, ჩვენი ვებ-სერვერის ძირეულ საქაღალდეს ვიყენებთ ჩვენ, ტესტური საქაღალდის სახით. ასევე უნდა დავაკვირდეთ, რომ **DreamweaverMX**-მა შესატან ველში **URL prefix** ჩასვას ჩვენი ვებ-სერვერის ინტერნეტ-მისამართი.

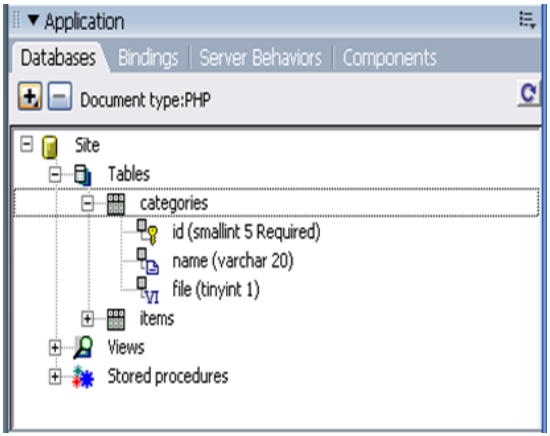
საჭირო პარამეტრების განსაზღვრის შემდეგ, დავაჭიროთ **Site Definition** დიალოგური ფანჯრის **OK** ღილაკს, ხოლო შემდეგ **Manage Sites** დიალოგური ფანჯრის **Done** ღილაკს. თუ ჩვენ გახსნილი გვექნება პანელი **Files**, მასში გამოტანილი იქნება ჩვენი ახალი საიტის **Site2** ყველა ფაილი (მოცემულ მომენტში იქ მხოლოდ მთავარი გვერდი-**default.htm**-ია გამოტანილი). **DreamweaverMX**-ი კი გაიგებს, რომ ჩვენ სერვერულ გვერდებთან ვაპირებთ მუშაობას.

მონაცემთა ბაზების რეგისტრაცია DreamveawerMX-ში.

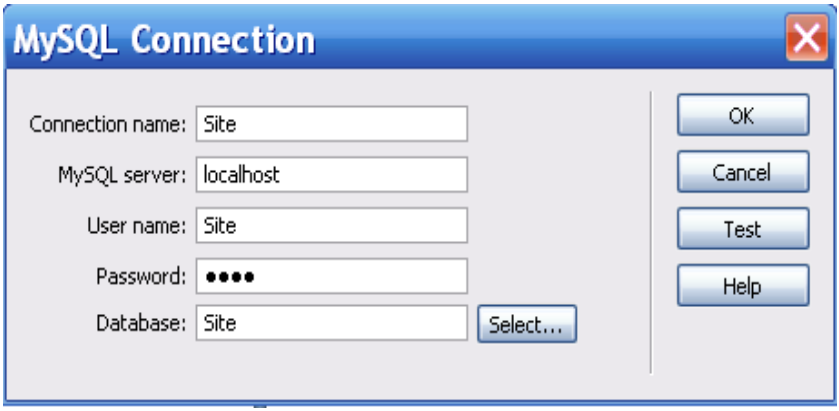
ჩვენი შემდგომი ნაბიჯია-**DreamveawerMX**-ში **მონაცემთა ბაზის რეგისტრაცია.** ეს იმისთვის არის საჭირო, რომ **DreamveawerMX**-ი გაერკვეს მონაცემთა ბაზის სტრუქტურაში, რომელთანაც მუშაობას ვაპირებთ.

მონაცემთა ბაზის რეგისტრაციამდე, საჭიროა თვით მონაცემთა სერვერის **MySQL** და **Web**-სერვერის **Apache** გაშვება. საქმე იმაშია, რომ მონაცემთა ბაზის სტრუქტურის გასარკვევად **DreamveawerMX**-ი Web-სერვერის სატესტო საქაღალდეში ათავსებს **PHP**-ს განსაკუთრებულ სერვერულ გვერდებს, ხოლო ჩვენ უკვე ვიცით, რომ მათი შესრულებისათვის საჭიროა **Web**-სერვერი. მონაცემთა სერვერის შესახებ ხომ, ლაპარაკიც ზედმეტია-ის ცალსახად საჭიროა! ბოლო ნაბიჯი მონაცემთა ბაზების რეგისტრაციის წინ-ეს არის პირველი სერვერული გვერდის შექმნა. ასეთია **DreamveawerMX**-ის მოთხოვნა და ამას ვერაფერს ვერ ვუზამთ. ასე რომ, მენიუ **File**-ში ავირჩიოთ პუნქტი **New, New Document** დიალოგური ფანჯრის სიაში **Category**, ავირჩიოთ პუნქტი **Dynamic Page**, ხოლო მარჯვენა სიაში-

პუნქტი **PHP**. ამის შემდეგ დავაჭიროთ ღილაკს **Create** და აქტიური სერვერული გვერდი შეიქმნება. შევინახოთ იგი დახურვის გარეში საიტის **Site2**, ძირეულ საქაღალდეში, **Categories.php** სახელით (შემდეგ მას გამოვიყენებთ კატეგორიების სიის გამოსატანად).
მონაცემთა ბაზის **DreamveawerMX**-ში დასარეგისტრირებლად, ჩვენ დაგვჭირდება **Databases** პანელი (ნახ.8.2.). ეკრანზე მის გამოსატანად, ჩავრთოთ პუნქტი-ჩამრთველი **Databases**, მენიუ **Window**-ში, ან დავაჭიროთ კლავიშების კომბინაციას **<Ctrl>+<Shift>+<F10>.**
ამ პანელის უდიდესი ნაწილი დაკავებული ექნება უკვე დარეგისტრირებული მონაცემთა ბაზების სიას. მაგრამ, ვინაიდან ჩვენ ჯერ არც ერთი მონაცემთა ბაზა არ დაგვირეგისტრირებია, **Databases** პანელზე გაჩნდება სია, რომელშიც ჩამოთვლილი იქნება ყველა ნაბიჯი, რომლებიც ჩვენ უნდა შევასრულოთ, **DreamveawerMX**-ში მონაცემთა ბაზების წარმატებული რეგისტრაციისათვის. ჩვენ ეს ნაბიჯები უკვე შესრულებული გვაქვს, ასე რომ, გადავიდეთ პირდაპირ რეგისტრაციის პროცესზე.



ნახ.8.2. პანელი Databases



ნახ.8.3. დიალოგური ფანჯარა MySQL Connection

დავაჭიროთ სიის ზემოთ მდებარე "პლიუს" ნიშნიან ღილაკს და ეკრანზე გამოსულ მცირე ზომის მენიუში ავირჩიოთ ერთადერთი პუნქტი **MySQL Connection**. ეკრანზე გაჩნდება დიალოგური ფანჯარა **MySQL Connection**, რომელიც ნახ.8.3.-ზეა ნაჩვენები.

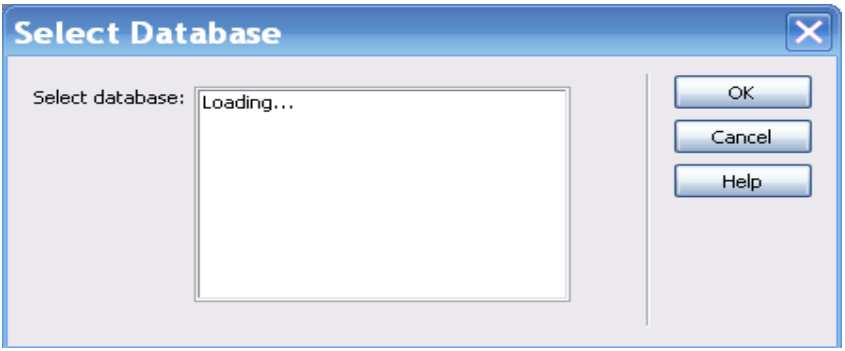
შესატან ველში **Connection name** შევიტანოთ ჩვენს მონაცემთა ბაზასთან კავშირის სახელი; **DreamveawerMX**-ი ამ სახელით შეინახავს მონაცემთა ანაკრებს, რომელიც მასთან დასაკავშირებლად არის საჭირო. რათა არ ვიმტვრიოთ თავი, შევიტანოთ აქ იგივე სახელი, რაც თავის დროზე მონაცემთა ბაზას მივანიჭეთ-**Site**.

შესატანი ველი **MySQL server** გამოიყენება მონაცემთა სერვერის ინტერნეტ-მისამართის განსაზღვრისათვის. რადგანაც სერვერი ჩვენსავე კომპიუტერზეა დაყენებული, შევიტანოთ იქ **localhost**.

შესატან ველებში **User name** და **Password** მონაცემთა ბაზასთან დასაკავშირებლად შეიტანება შესაბამისად მომხმარებლის სახელი და პაროლი. მოდით გავიხსენოთ, თუ რა სახელი და პაროლი მივანიჭეთ ჩვენ ჩვენს მომხმარებელს-ესენია **site** და **site** შესაბამისად.

დაგვრჩა მხოლოდ შესატან ველში **Databases** თვით მონაცემთა ბაზის სახელის-site შეტანა. შესაძლებელია აგრეთვე ველის მარჯვენა მხარეს მდებარე **Select** ღილაკზე დაწკპუნება. ეკრანზე გამოჩნდება მცირე ზომის დიალოგური ფანჯარა **Select Database** (ნახ.8.4.), რომლის **Select Database** სიაში მხოლოდ ერთადერთი მონაცემთა ბაზა

იქნება მოთავსებული, რომელშიც მომხმარებელ **site**-ს აქვს შეღწევის უფლება. ავირჩიოთ იგი და დავაჭიროთ ღილაკს OK.



ნახ.8.4. დიალოგური ფანჯარა Select Databases

იმისათვის, რომ შევაწოწმოთ სწორად შევიტანეთ თუ არა ყველა მონაცემი, დავაჭიროთ **MySQL Connection** დიალოგური ფანჯრის ღილაკს **Test**. საპასუხოდ **DreamveawerMX**-ი შეეცდება მონაცემთა ბაზასთან დაკავშირებას და გამოიყვანს შესაბამისი გაფრთხილების ფანჯარას. თუ დაკავშირება ნორმალურად განხორციელდა, დავაჭიროთ ღილაკს **OK**. თუ კი დაკავშირების მცდელობა წარუმატებელი აღმოჩნდა, უნდა მოხდეს შეტანილი მონაცემების შესწორება და მცდელობის განმეორება. აგრეთვე უნდა შემოწმდეს, გაშვებულია ტუ არა მონაცემთა და ვებ სერვერები.

მორჩა, ჩვენს მიერ დარეგისტრირებული მონაცემთა ბაზა არსებობს **Databases** პანელის სიაში იერარქიული სიის "ძირის" სახით (იხ.ნახ.8.2.), რომლის სახელიც ემთხვევა თვით მონაცემთა ბაზის სახელს. ეს "ძირი" განიტოტება სამ "ხედ": **Tables, Views და Stored procedures**. ჩვენ, მოცემულ მომენტში, უფრო გვაინტერესებს "ხე" **Tables**, რომელიც შეიცავს მონაცემთა ბაზაში-**site**, შექმნილი ყველა ცხრილის სახელებს. ყოველი ცხრილი თავის მხრივ, აგრეთვე წარმოადგენს "ხეს", რომლის გაშლის შედეგად ჩვენ დავინახავთ ამ ცხრილში არსებული ველების სიას.

მონაცემთა ბაზების რეგისტრაციის პარამეტრების შესაცვლელად, საჭიროა სასურველი "ფუძის" მონიშვნა და კონტექსტურ მენიუში, რომელიც თაგუნას მარჯვენა ღილაკის დაჭერის შედეგად გამოვა, **Edit Connection** პუნქტის არჩევა. ეკრანზე გამოჩნდება ჩვენთვის უკვე

ნაცნობი დიალოგური ფანჯარა **MySQL Connection**, რომელშიც მონაცემთა ბაზების შესახებ ნებისმიერი ინფორმაციის შეცვლა შეგვეძლება, გარდა კავშირის სახელწოდებისა.

Databases პანელის სიიდანგამოუყენებელი მონაცემტა ბაზის წასაშლელად, საჭიროა შესაბამისი "ფუძის" მონიშვნა და "მინუს" ნიშნიანი ღილაკის დაჭერა. ეკრანზე გამოჩნდება გამაფრთხილებელი ფანჯარა, რომელშიც უნდა დავაჭიროთ ღილაკს **Yes.**

ახლა კი მოდით შევხედოთ საიტის ლოკალური ასლის ფაილების სიას, რომელიც გამოისახება **Files** პანელზე (თუ ეს პანელი არ არის გამოტანილი ეკრანზე, უნდა ჩავრთოთ პუნქტი-ჩამრთველი **Files**, მენიუ **Window**-ში ან დავაჭიროთ კლავიშას <**F**>). დავინახავთ, რომ ძირეულ საქაღალდეში **DreamveawerMX**-ს შექმნილი აქვს საქაღალდე **Connections**, ხოლო მის შიგნით ფაილი-**Site.php**. ამ ფაილის სახელწოდება ემთხვევა მონაცემთა ბაზასთან კავშირის სახელწოდებას, რაც იმას ნიშნავს, რომ ეს ფაილი შეიცავს რაღაცა **PHP**-კოდს, რომელიც უშუალოდ ამყარებს აღნიშნულ კავშირს.

მოდით დავაკვირდეთ ამ კავშირს. **DreamveawerMX**-ის გარემოში გავხსნათ ფაილი **Site.php** და გადავერთოთ **HTML**-კოდის დათვალიერების რეჟიმში. დოკუმენტის ფანჯარაში ჩვენ დავინახავთ შემდეგს:

```
<?php
# FileName="Connection_php_mysql.htm"
# Type="MySQL"
# HTTP="true"
$hostname_Site = "localhost" ;
$database_Site = "site" ;
$username_Site = "site" ;
$password_Site = "site" ;
$Site = mysql_pconnect ($hostname_Site,$username_Site,$password_Site)
or
trigger_error(mysql_error( ),E_USER_ERROR ;
?>
```

თუ გამოვრიცხავთ ტეგს <?php. . .?> და **DreamveawerMX**-ის დამხმარე მონაცემების შემცველ კომენტარებს, მაშინ დარჩება მხოლოდ ხუთი გამოსახულება, რომლებიც ჩვენ უნდა განვიხილოთ. ეს გამოსახულებები გამოყოფილია მუქი შრიფტით.

პირველი ოთხი გამოსახულება განსაზღვრავს ცვლადების მნიშვნელობებს, რომლებიც შესაბამისად შეიცავენ მონაცემთა სერვერის ინტერნეტ-მისამართს, მონაცემტა ბაზის სახელს, მომხმარებლის სახელს და მის პაროლს. ყოველივე ეს ჩვენ განვსაზღვრეთ დიალოგურ ფანჯარაში **MySQL Connection** (იხ.ნახ.8.3).

მეხუთე გამოსახულებაზე კი განსაკუთრებულად უნდა შევჩერდეთ. მასში **PHP**-ს ორი ჩაშენებული ფუნქცია არის გამოყენებული, რომელთაც ჩვენ, ჯერ არ გავცნობივართ.

პირველი ფუნქცია-**mysql_pconnect**-მონაცემთა სერვერთან მუდმივ კავშირს ამყარებს. (მუდმივი კავშირების შესახებ იხ. თავი1-ში). არგუმენტების სახით იგი იღებს მონაცემთა სერვერის ინტერნეტ-მისამართს, მომხმარებლის სახელს და პაროლს. (შევნიშნოთ, რომ მონაცემთა ბაზის სახელი აქ ჯერ არ ფიგურირებს). ამ ფუნქციის შედეგი იქნება განსაკუთრებული არანულოვანი რიცხვი (**იდენტიფიკატორი**), რომელიც მიენიჭება ცვლადს **$Site** და ცალსახად განსაზღვრავს განხორციელებულ კავშირს (ამ შემთხვევაშიც, ცვლადის სახელი ემთხვევა დარეგისტრირებული კავშირის სახელს).

მეორე ფუნქციას-**trigger_error**-ეკრანზე გამოყავს (რათქმა უნდა, არა უშუალოდ ეკრანზე, არამედ ვებ-გვერდზე) შეტყობინება შეცდომის შესახებ. ამ ფუნქციის პირველი არგუმენტი უნდა იყოს თვით შეტყობინება შეცდომის შესახებ სტრიქონული სახით (რომელსაც ჩვენს შემთხვევაში, აბრუნებს რიგით მესამე ფუნქცია-**mysql_error**), ხოლო მეორე-შეცდომის რიცხვითი კოდი (თითქმის ყოველთვის გამოიყენება კონსტანტის მნიშვნელობა **E_USER_ERROR**).

ახლა კი მოდით გავარკვიოთ, თუ როგორ გამოითვლება ეს მეხუთე გამოსახულება ადრე განხილული სცენარიდან.

დავუშვათ, რომ **PHP**-მ შეძლო **MySQL**-თან კავშირის დამყარება, და ფუნქცია **mysql_pconnect**-მა დააბრუნა კორექტული იდენტიფიკატორი და მიანიჭა იგი ცვლადს **$Site**. მაშინ გამოთვლის შედეგი

$Site=mysql_pconnect($hostname_Site,$username_Site, $password_Site)

ტოლი იქნება თვით ამ იდენტიფიკატორის.

შემდგომ, მეხუთე გამოსახულებაში **PHP** შეხვდება ლოგიკურ **(OR)** ოპერატორს და ჩათვლის რომ, ეს ლოგიკური გამოსახულებაა. ეს

იმას ნიშნავს, რომ დაბრუნებული იდენტიფიკატორი უნდა გარდაიქმნას ლოგიკურ სიდიდედ. ასეთი სიდიდე იქნება **true**, ვინაიდან ყველა არანულოვანი რიცხვები გარდაიქმნებიან **true**-დ. ხოლო, რადგანაც ოპერატორი OR აბრუნებს **true**-ს იმ შემთხვევაში, თუ ერთ-ერთი მისი არგუმენტი **true**-ს ტოლია, მაშინ გამოსახულების შემდგომი გამოთვლა აღარ იქნება საჭირო და ფუნქცია **trigger_error** არ შესრულდება.

ახლა, მოდით განვიხილოთ ყველაზე უარესი ვარიანტი- **PHP**-მ ვერ შეძლო **MySQL**-თან კავშირის დამყარება, და ფუნქციამ **mysql_pconnect** დააბრუნა 0 (ის ყოველთვის ასეთ შემთხვევებში აბრუნებს 0-ს). 0-ი გარდაიქმნება **false**-ში. **PHP** განაგრძობს გამოსახულების გამოთვლას, ვინაიდან **OR** ოპერატორის შედეგი ასეთ შემთხვევაში გაურკვეველია და შეასრულებს ფუნქციას **trigger_error**. საიტის დამთვალიერებელი მიიღებს შეტყობინებას შეცდომის შესახებ.

აი, ასეთი სცენარი შექმნა ჩვენთვის **DreamweaverMX**-მა. უნდა გვახსოვდეს, რომ არ უნდა წაიშალოს ფაილი **Site.php**, რომელშიც იგი ინახება, წინააღმდეგ შემთხვევაში ჩვენ მოგვიწევს **DreamweaverMX**-ში მონაცემთა ბაზის ხელახლა დარეგისტრირება.

უმარტივესი სერვერული გვერდების შექმნა

ახლა კი შეიძლება შევუდგეთ **DreamweaverMX**-ში სერვერული გვერდების შექმნას. ყველი მოსამზადებელი ოპერაციები ჩვენ უკვე ჩავატარეთ.

პირველი გვერდი, რომელსაც ჩვენ შევქმნით, იქნება კატეგორიების სიის შემცველი გვერდი-**Categories.php**, რომელიც ჩვენ პრინციპში უკვე შექმნილი გვაქვს. მართალია ის ჯერ არ შეიცავს რაიმე შიგთავსს გარდა **DreamweaverMX**-ში შექმნილი **HTML**-ის აუცილებელი სტრუქტურისა. ჩვენ იგი უნდა შევავსოთ მონაცემებით.

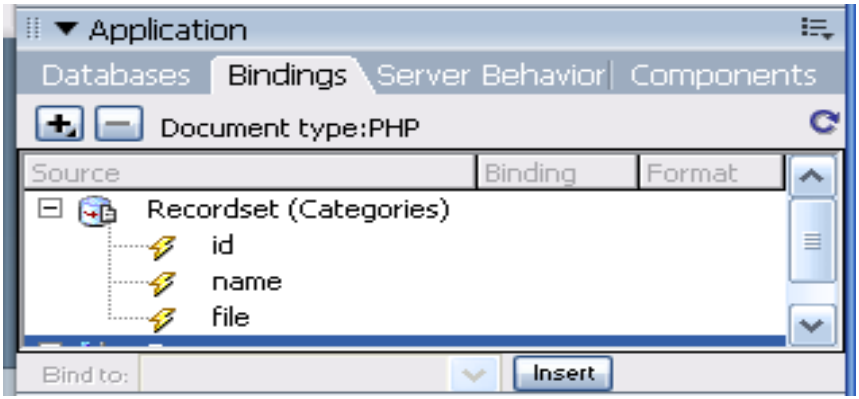
მაშ ასე, **DreamweaverMX**-ში გავხსნათ გვერდი **Categories.php**, თუ რათქმა უნდა ის, ჯერ კიდევ გახსნილი არ არის. მივანიჭოთ მას სახელი, შევიტანოთ რაიმე განმარტებითი ტექსტი და შევქმნათ მასში, ერთი სტრიქონისა და ერთი სვეტისაგან შემდგარი ცხრილი. შემდგომში, სწორად ამ ცხრილში მოვათავსებთ კატეგორიების სიას. ხოლო, კატეგორიების დასახელებები გარდაიქმნებიან ჰიპერმინიშნებებად, რომლებიც მიანიშნებენ ფაილებისა და სტატიების შემცველ გვერდებზე.

ჩანაწერების ანაკრების შექმნა.

ცხადია, რომ კატეგორიების სიას, ამოვიღებთ ჩვენი მონაცემთა ბაზის ცხრილიდან **categories**. ხოლო მისი ამოღებისათვის დავწერთ **SQL** მოთხოვნას. უფრო სწორად **DreamweaverMX**-ს ვაიძულებთ ამის გაკეთებას ჩვენს მაგივრად-მას შეუძლია ასეთი რამის კეთება. მონაცემთა სერვერი შეასრულებს ამ მოთხოვნას და დაგვიბრუნებს პასუხს, რომელშიც იქნება ჩვენთვის საჭირო ჩანაწერები-ეგრეთ წოდებულ ჩანაწერების ანაკრებს (ინგლისურად-**Recordset**).

რადგანაც, ჩვენ ვაპირებთ **DreamweaverMX**-ის გამოყენებას, მოდით შევატყობინოთ მას , თუ რომელი ჩანაწერების ანაკრები გვჭირდება. ამისათვის გამოვიყენოთ პანელი **Bindings,** რომელიც ნახ.8.5-ზეა ნაჩვენები.

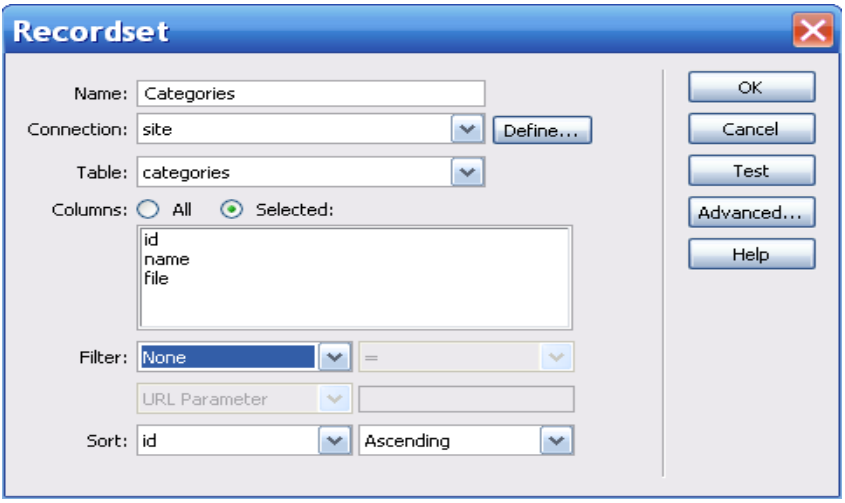
მისი ეკრანზე გამოსაძახებლად ჩავრთოთ პუნქტი-ჩამრთველი **Bindings** მენიუ **Window**-ში ან დავაჭიროთ კლავიშების კომბინაციას <Ctrl>+<F10>.



ნახ.8.5. პანელი Bindings

ეს პანელი, ძირითადად მოცემული მომენტისათვის უკვე შექმნილი ჩანაწერების ანაკრებების სიისაგან შედგება. თავდაპირველად იგი შეიცავს მხოლოდ დამხმარე ტექსტს, რომელშიც დამწყები ვებ-პროგრამისტებისათვის, ჩამოთვლილია ის ნაბიჯები, რომლებიც მათ უნდა გაიარონ, სანამ ისინი შეძლებენ საკუთარი, პირველი ჩანაწერების ანაკრების შექმნას. ჩვენ ხომ, ის უკვე გავიარეთ. ამიტომ, ეგრევე შევუდგეთ საქმეს.

დავაჭიროთ ამ პანელზე განთავსებულ "პლიუს"-ნიშნიან ღილაკს, და ეკრანზე გამოსულ მენიუში ავირჩიოთ პუნქტი **Recordset (Query)**. ეკრანზე გაჩნდება დიალოგური ფანჯარა **Recordset** (ნახ.8.6.)



ნახ.8.6. დიალოგური ფანჯარა Recordset

შესატან ველში **Name** იწერება ჩანაწერების შესაქმნელი ანაკრების სახელი. მივანიჭოთ მას სახელი **Categories**-ასე უფრო გასაგები იქნება, თუ რისთვის არის იგი საჭირო.

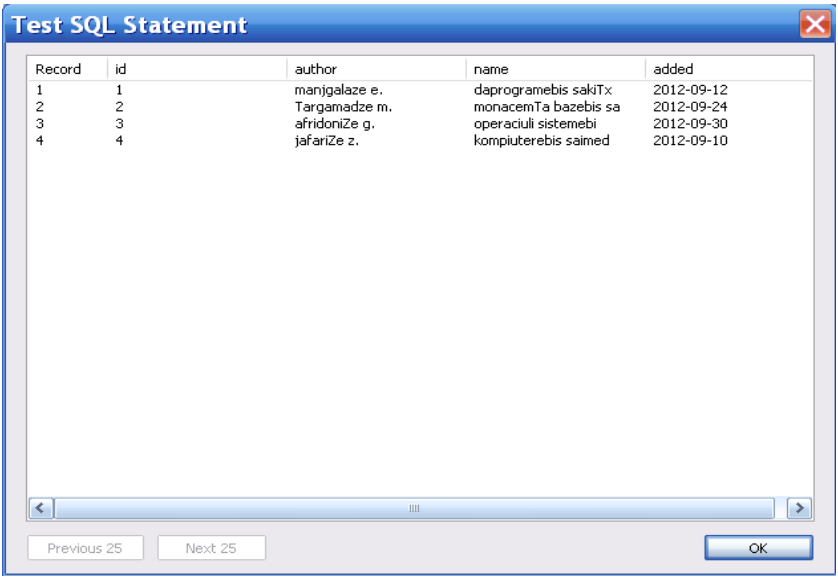
ჩამოსაშლელ სიაში **Connection** ავირჩიოთ ჩვენს მიერ უკვე დარეგისტრირებული მონაცემთა ბაზა-**Site**.

მონაცემთა ბაზიდან, ჩამოსაშლელი სიის **Table** საშუალებით, ავირჩიოთ ჩვენთვის საჭირო ცხრილი. ეს არის ცხრილი- **categories**.

გადამრთველების ჯგუფი **Columns** საშუალებას გვაძლევს ავირჩიოთ ცხრილის ველები, რომლებიც ჩანაწერების ანაკრებში იქნებიან ჩართული. ჩვენ შეგვიძლია ჩავრთოთ გადამრთველი **All** (თუმცა იგი ისედაც ჩართულია გაჩუმებით), რათა ჩანაწერების ანაკრებში შეტანილ იქნას ყველა ველი. მაგრამ მოდით ვიფიქროთ: ჩვენ მხოლოდ **id** და **name** ველები გვჭირდება, ხოლო ველი-**file** ამ შემთხვევაში ფაქტიურად არ არის საჭირო. ამიტომ ჩავრთოთ გადამრთველი **Selectid**, დავაჭიროთ კლავიშას **<Ctrl>** და მისი, დაჭერილ მდგომარეობაში შენარჩუნებით, ავირჩიოთ ჩვენთვის სასურველი ველები ქვემოთ განლაგებული სიიდან.

ჩამოსაშლელი სიიდან **Sort** ავირჩიოთ ველი, რომლის მიხედვითაც მოხდება ანაკრებში შემავალი ჩანაწერების დახარისხება (სორტირება), ხოლო მარჯვენა მხარეს განლაგებული ჩამოსაშლელი სიიდან-დახარისხების მიმართულება (პუნქტი **Ascending**-ზრდადობის მიხედვით, ხოლო **Descending**-კლებადობის მიხედვით). მოდით ავირჩიოთ ველი **id** და სორტირების მიმართულება ზრდადობის მიხედვით, რათა ჩანაწერები ანაკრებში განლაგდნენ იმავე თანმიმდევრობით, რა თანმიმდევრობითაც ჩვენ შევიტანეთ ისინი. (რათქმა უნდა, სორტირება შესაძლებელია **name** ველის მიხედვითაც, მაგრამ ამ შემთხვევაში არც თუ ისე კორექტულ თანმიმდევრობას მივიღებთ-კატეგორია **სხვადასხვა-Прочее**, რომლის ადგილიც ბოლოში უნდა იყოს, მოექცევა შუაში).

იმისათვის, რომ შევამოწმოთ სწორად არის თუ არა შეტანილი ყველა მონაცემი, დავაჭიროთ ღილაკს **Test**. ამის შემდეგ ეკრანზე უნდა გამოჩნდეს დიალოგური ფანჯარა **Test SQL Statement**, რომელიც ნაჩვენებია ნახ.8.7. აქ შეიძლება ჩანაწერების მიღებული ანაკრების დათვალიერება; ფანჯრის დასახურად, საკმარისია **OK** ღილაკზე დაჭერეა. რუსულ ენასთან მიმართებაში **Test SQL Statement** ფანჯარას გარკვეული პრობლემები ექმნება, რაც იმაში გამოიხატება, რომ **name** ველში ჩანაწერები არ იკითხება.



ნახ.8.7. დიალოგური ფანჯარა Test SQL Statement

თუ კი ჩანაწერების ანაკრების პარამეტრები არასწორად იქნება შეტანილი, **DreamweaverMX**-ი ეკრანზე გამოიტანს შესაბამისი გაფრთხილების ფანჯარას. ჩვენ მოგვიწევს ხელახლა შემოწმება, ყველაფერი სწორად გავაკეთეთ თუ არა.

როგორც კი მოვრჩებით ჩანაწერების პარამეტრების განსაზღვრას, ჩვენ შეგვიძლია დიალოგური ფანჯრის **Recordset** დახურვა, **OK** ღილაკზე დაჭერით. ამის შემდეგ **Bindings** პანელის სიაში გაჩნდება მცირე ზომის "ხე", რომლის "ფესვი"(ფუძე) იქნება ჩანაწერების ანაკრები, ხოლო "შტოები"-შესაბამისი ველები.

თუ ჩვენ დაგვჭირდება ჩანაწერების ანაკრების პარამეტრების შესწორება, ორჯერ უნდა დავაწკაპუნოთ სიის შესაბამის "ფესვს". ეკრანზე გამოჩნდება დიალოგური ფანჯარა **Recordset**, რომელშიც ჩვენ შეგვეძლება საჭირო პარამეტრების შეცვლა. გამოუყენებელი ჩანაწერების ანაკრების წასაშლელად საჭიროა შესაბამისი "ფესვის" მონიშვნა და "მინუს" ნიშნიანი ღილაკის დაჭერა.

ახლა მოდით შევინახოთ ჩვენი ვებ-გვერდი **Categories.php**. ამისათვის მენიუ **File**-ში ავირჩიოთ პუნქტი **Save** ან დავაჭიროთ კლავიშების კომბინაციას **<Ctrl>+<S>.** ამით ჩვენ მოვრჩით ჩანაწერების ანაკრების შექმნას.

ამასობაში, **DreamweaverMX**-მა ჩვენს მაგივრად შექმნა **PHP** სცენარი, რომელიც ამყარებს კავშირს მონაცემთა ბაზასთან, ასრულებს **SQL** მოთხოვნას და მონაცემთა სერვერისაგან იღებს ჩანაწერების ანაკრებს. რაც შეეხება **SQL** მოთხოვნის შესაქმნელად საჭირო მონაცემებს, ჩვენ ისინი შევიტანეთ იმავე დიალოგურ ფანჯარაში **Recordset**.

საკუთრივ სერვერული გვერდის შექმნა.

რადგანაც მოვრჩით ჩანაწერების ანაკრების შექმნას, მოდით გადავიდეთ თვით სერვერული ვებ-გვერდის შექმნაზე, რომელიც ეკრანზე კატეგორიების სიას გამოიტანს.

მაშ ასე, ჩვენ უკვე გახსნილი გვაქვს ვებ-გვერდი **Categories.php**. შევამოწმოთ, გახსნილია თუ არა **Bindings** პანელიც-ჩვენ ახლა იგი დაგვჭირდება. დოკუმენტის ფანჯარა, რომელშიც გვერდია გახსნილი, და პანელი, ერთმანეთის მიმართ განვათავსოთ ისეთნაირად, რომ ისინი არ ეფარებოდნენ ერთმანეთს და თუ

ეფარებიან, მაშინ ისე მაინც, რომ ჩვენს მიერ ვებ-გვერდზე შექმნილი ცხრილი გამოჩნდეს. ამის შემდეგ შეიძლება შევუდგეთ მუშაობას.

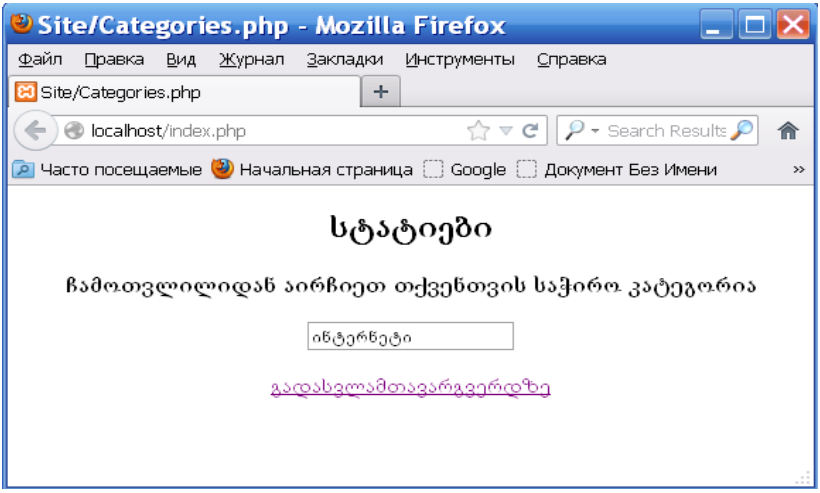
Bindings პანელის იერარქიულ სიაში ავირჩიოთ "შტო", რომელიც შეესაბამება **name** ველს-სწორედ ამ ველის შიგთავსი იქნება გამოტანილი ჩვენს ვებ-გვერდზე. ამის შემდეგ, უნდა გადავათრიოთ ეს ველი ვებ-გვერდზე და "დავაგდოთ" ცხრილის ერთად-ერთ უჯრედში. მოცემული მოქმედების შედეგი ნაჩვენებია ნახ.8.8.

ჩვენ ამწუთას შევქმენით **დინამიური ტექსტი**-ასე ეწოდება ვებ-გვერდზე რაიმე მონაცემების გამომტან **PHP** სცენარს **DreamweaverMX**-ის ტერმინოლოგიით. ჩვენს შემთხვევაში დინამიურ ტექსტს გამოაქვს ჩანაწერების ანაკრების-**categories, name**-ველის შიგთავსი. ახლა სწორედ ის დროა, როდესაც უნდა დავინახოთ, თუ როგორ მუშაობს ეს ყოველივე რეალურად. გამოვიყენოთ **DreamweaverMX**-ის "ცოცხალი" დათვალიერების რეჟიმი, რისთვისაც უნდა ჩავრთოთ დოკუმენტის ინსტრუმენტების პანელზე განთავსებული ღილაკი-გადამრთველი **Live Data View** (). საპასუხოდ, **DreamweaverMX**-ი ჩვენს გვერდს **Categories.php**, მოათავსებს ვებ-სერვერის სატესტო საქაღალდეში და გაუშვებს მას შესრულებაზე. ჩვენ დავინახავთ იმას, რაც ნაჩვენებია ნახ.8.9.-ზე.

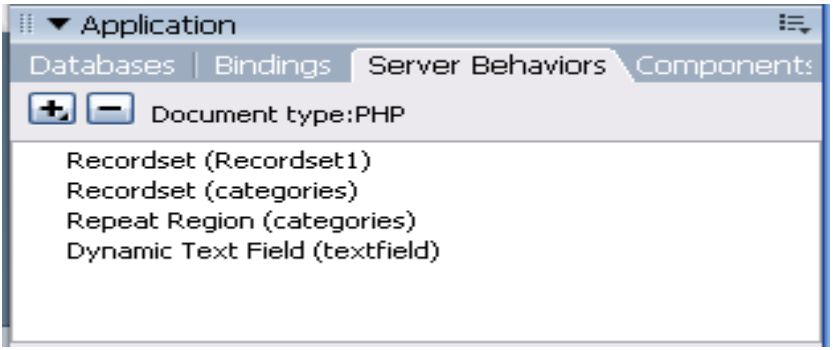
შესანიშნავია! მუშაობს! მაგრამ, რატომღაც მხოლოდ ერთი ჩანაწერი გამოიტანა. ჩვენ კი გვინდა ყველა.

იმისათვის, რომ გვერდმა გამოიტანოს ყველა ჩანაწერი, ჩვენ მასზე უნდა შევქმნათ **განმეორებადი არე**-გვერდის განსაკუთრებული ელემენტი, რომელიც გამოიტანება იმდენჯერ, რამდენი ჩანაწერიც არის ანაკრებში **categories**. ამ არეში მოთავსებული იქნება ცხრილის ერთადერთ სტრიქონი (ტეგი **<TR>** თავისი შიგთავსით); ეს ნიშნავს, რომ ანაკრების ყოველი ჩანაწერის **name** ველის შიგთავსი, გამოტანილი იქნება ცხრილის ცალკე სტრიქონში.

ნახ.8.8.name ველის შიგთავსის გამომტანი დინამიური ტექსტი.



ნახ.8.9. სერვერული გვერდის Categories.php შესრულების შედეგი

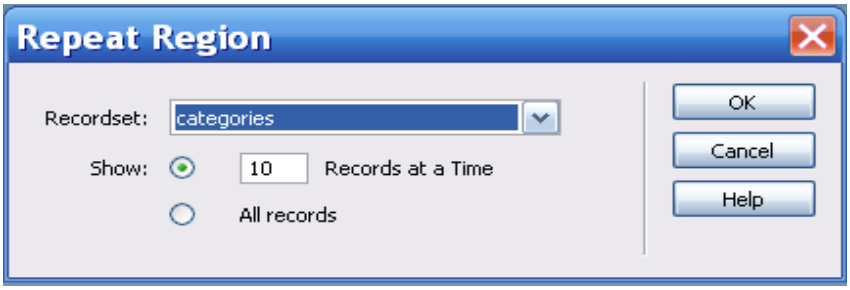


ნახ.8.10. პანელი Server Behaviors

მოვათავსოთ ტექსტური კურსორი ცხრილის რომელიმე უჯრაში და დავაწკაპუნოთ **<tr>** ღილაკზე, რომელიც განთავსებულია დოკუმენტის ფანჯრის, სტატუსის სტრიქონის, ტეგების სექციაში. ამგვარად ჩვენ მოვნიშნავთ ცხრილის სტრიქონს. ამის შემდეგ, მენიუ **Window**-ში ჩავრთოთ პუნქტი-გადამრთველი **Server Behaviors** ან დავაჭიროთ კლავიშების კომბინაციას **<Ctrl>+<F9>**. ეკრანზე გამოჩნდება პანელი **Server Behaviors** (ნახ. 8.10).

თითქმის მთლიანად, ეს პანელი დაკავებული იქნება **DreamweaverMX**-ის მიერ, მოცემულ სერვერულ ვებ-გვერდზე შექმნილი სერვერული ქცევების სიით. **სერვერულ ქცევებს (server behaviors)** უწოდებენ **PHP** სცენარებს, რომლებსაც **DreamweaverMX**-ი ქმნის სერვერული ვებ-გვერდის კოდში. მოცემულ მომენტში **Categories.php** გვერდზე შექმნილია ჩვენთვის უკვე ნაცნობი ორი სერვერული ქცევა: **Recordset** (მონაცემთა ანაკრების შექმნა) და **Dynamic Trxt** (დინამიური ტექსტი). სერვერული ქცევის სახელწოდების შემდეგ, ფრჩხილებში მითითებულია ჩანაწერების ანაკრების სახელი, რომელსაც იგი ეკუთვნის.

განმეორებადი არეს შესაქმნელად, ჩვენ დაგვჭირდება ცხრილში კიდევ ერთი სერვერული ქცევის დამატება. ამისათვის დავაჭიროთ "პლიუს" ნიშნიან ღილაკს და ეკრანზე გამოსულ მენიუში, ავირჩიოთ პუნქტი **Repeat Region**. ეკრანზე გამოჩნდება დიალოგური ფანჯარა Repeat Region, რომელიც ნაჩვენებია ნახ.8.11-ზე.

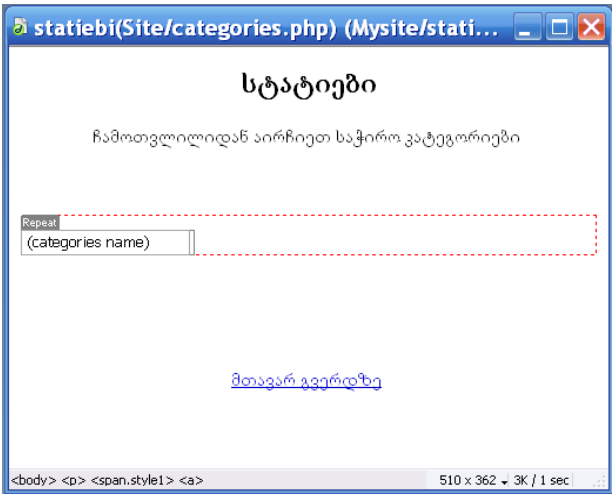


ნახ.8.11. დიალოგური ფანჯარა Repeat Region

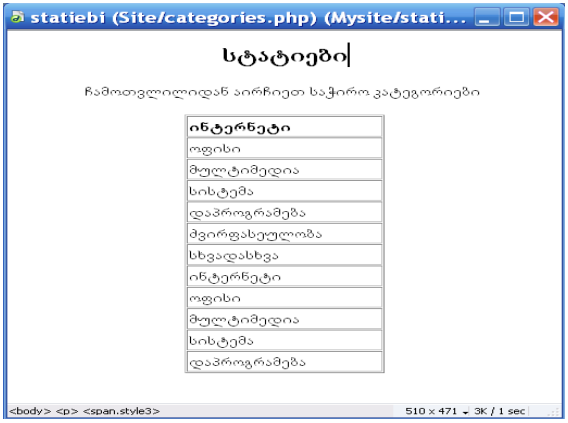
ჩამოსაშლელ სიაში **Recordset** ავირჩიოთ ჩანაწერების სასურველი ანაკრები-ჩვენ შემთხვევაში, ეს რათქმა უნდა იქნება **Categories.** შესაქმნელი განმეორებადი არეს შიგთავსი გამოტანილი იქნება იმდენჯერ, რამდენი ჩანაწერიც არის ამ ანაკრებში.
თუ ჩართულია **Show** ჯგუფის ზედა გადამრთველი, მაშინ ჩანაწერების ანაკრების შიგთავსის ვებ-გვერდზე გამოტანა მოხდება ნაწილ-ნაწილ, თითქოს გვერდებად. ეკრანზე ერთდროულად გამოსატანი ჩანაწერების რაოდენობა (ასეთი "გვერდის" ზომა) განისაზღვრება შესატან ველში **Records at Time**. ჩანაწერების "გვერდებად" გამოტანა გამართლებულია იმ შემთხვევაში, როდესაც ჩანაწერების რაოდენობა ძალიან დიდია და ყველა მათგანის ერთ ვებ-გვერდზე გამოსატანად, მომხმარებელს დიდ ხანს მოუწევდა ლოდინი. მაგრამ ჩვენს შემთხვევაში ამის საშიშროება არ არსებობს და თავისუფლად შეგვიძლია **All records** გადამრთველის ჩართვა, რომელიც უზრუნველყოფს ანაკრების ყველა ჩანაწერის ერთ გვერდზე გამოტანას.

შენიშვნა: *თუ გამოიყენება ანაკრებში შემავალი ჩანაწერების "გვერდებად" გამოტანა, მაშინ საჭიროა ზრუნვა ჰიპერმინიშნებებზე, რომლებიც უზრუნველყოფენ გადასვლებს "გვერდი"-დან , "გვერდ"-ზე. ამისათვის Dreamweaver-ს გააჩნია სერვერული ქცევები Move To First Page (გადასვლა პირველ გვერდზე), Move To Previous Page (გადასვლა წინა გვერდზე), Move To Next Page (გადასვლა შემდეგ გვერდზე), Move To Last Page (გადასვლა ბოლო გვერდზე). მათ შესაქმნელად, პანელზე Server Behaviors უნდა დავაჭიროთ ღილაკს "პლიუსი" ნიშნით და ეკრანზე გამოსული მენიუს, Recordset Paging ქვემენიუში ავირჩიოთ შესაბამისი პუნქტი.*

ყველა საჭირო მონაცემების განსაზღვრის შემდეგ შეგვიძლია დავაჭიროთ ღილაკს **OK**. ჩვენს მიერ შექმნილი განმეორებადი არე ნაჩვენები იქნება დოკუმენტის ფანჯარაში. მას ნახ.8.12.-ზე ნაჩვენები სახე ექნება. ხოლო **Server Behaviors** პანელის სიაში გაჩნდება ახალი პუნქტი-**Repeat Region** (განმეორებადი არე).



ნახ.8.12. გვერდი Categories.php განმეორებადი არით.



ნახ.8.13. განმეორებადი არის შემცველი სერვერული გვერდის-Categories.php შესრულების შედეგი

თუკი ჩვენ ჩავრთავთ "ცოცხალ" დათვალიერებას, მაშინ დავინახავთ, რომ ჩვენი სერვერული გვერდი მუშაობს ისე, როგორც საჭიროა და გმოყავს ყველა ჩანაწერი (ნახ.8.13.). (თუმცა, ის მთლად ისე არ მუშაობს, როგორც უნდა იმუშაოს, მაგრამ ამის შესახებ ჩვენ მოგვიანებით ვისაუბრებთ).

ამგვარად, ჩვენ შევქმენით ჩვენი პირველი აქტიური სერვერული ვებ-გვერდი! ამაში ჩვენ დახმარება გაგვიწია **DreamweaverMX**-მა.

სულ ცოტა რამ დაგვრჩა სათქმელი. იმისათვის, რომ შევცვალოთ რაიმე სერვერული ქცევის პარამეტრები, საჭიროა **Server Behaviors** პანელის სიის შესაბამის პუნქტზე დაწკაპუნება. ეკრანზე გამოჩნდება დიალოგური ფანჯარა, რომელშიც ჩვენ შეგვეძლება აღნიშნული პარამეტრების შეცვლა. ასე მაგალითად, თუ ჩვენ ორჯერ დავაწკაპუნებთ პუნქტზე **Recordset**, გამოვა დიალოგური ფანჯარა **Recordset** (იხ.ნახ.8.6.), ხოლო თუკი იგივეს გავაკეთებთ **Repeat Region** პუნქტთან დაკავშირებით, ჩვენ დავინახავთ დიალოგურ ფანჯარას **Repeat Region** (იხ.ნახ.8.11.). არასაჭირო სერვერული ქცევის წასაშლელად საკმარისია მისი შესაბამისი პუნქტის მონიშვნა და "მინუს" ნიშნიანი ღილაკის დაჭერა.

ახლა, მოდით შევინახოთ გვერდი **Categories.php** და **DreamweaverMX**-ში გადავერთოთ **HTML**-კოდის დათვალიერების რეჟიმში. დროა შევხედოთ, თუ რა **PHP**-სცენარი შექმნა **DreamweaverMX**-მა ჩვენს ზურგს უკან.

ჩანაწერების გამოსატანად განკუთვნილი PHP სცენარების გარჩევა

DreamveaverMX-ის მიერ შექმნილი **PHP** კოდის გარჩევას ჩვენ ნაწილ-ნაწილ შევასრულებთ. ასე უფრო მარტივად გავიგებთ, თუ რას აკეთებს იგი და რომელ სერვერულ ქცევას განეკუთვნება.

PHP-ს პირველი სცენარი ძალიან მოკლეა და ერთად ერთი გამოსახულებისაგან შედგება:

```
<?php require_once ('Connections/Site.php') ;?>
```

აქ ჩვენ უცებ წავაწყდით ჩვენთვის უცნობ **PHP** ოპერატორს-**require_once**. ეს ოპერატორი იღებს ერთ არგუმენტს (საყურადღებოა, რომ იგი მიეთითება ბრჩხილებში, როგორც ფუნქციის არგუმენტი)-

სტრიქონს, რომელიც შეიცავს ბილიკს **PHP** ფაილამდე. ეს სტრიქონი, როგორც ჩვენ ვხედავთ, ჩასმულია ერთეულოვან ბრჭყალებში, რაც პრინციპში დასაშვებია.

require_once ოპერატორისათვის გადაცემული სახელების მატარებელ ფაილებში შენახული ყველა **PHP** სცენარები სრულდებიან. მაგრამ ისინი შესრულდებიან მხოლოდ ერთხელ; თუკი მეორე სცენარში ასევე შეგვხვდება ოპერატორი **require_once** იმავე არგუმენტით, მაშინ მოხდება მისი იგნორირება.

რაც შეეხება ფაილს **Site.php**, რომელიც ინახება **Connections** საქაღალდეში, იგი შეიცავს ჩვენს მიერ განსაზღვრული სახელისა და პაროლის მქონე მონაცემთა სერვერთან მუდმივი კავშირის დამამყარებელ სცენარს. ჩვენ უკვე განვიხილეთ ეს სცენარი, როდესაც ვსაუბრობდით **DreamweaverMX**-ში მონაცემთა ბაზების რეგისტრაციის შესახებ.

პირველ სცენართან დაკავშირებით სათქმელი მხოლოდ ისღა დაგვრჩა, რომ მისი ჩასმა ხდება ყოველი სერვერული **PHP** გვერდის დასაწყისში. მიუხედავად იმისა, რომ ის არ შეესაბამება არც ერთ სერვერულ ქცევას, იგი აუცილებელია მონაცემთა ბაზებთან მომუშავე სერვერული ქცევებისათვის.

შემდეგი სცენარი შეესაბამება სერვერულ ქცევას **Recordset**. მას ასეთი სახე აქვს:

```
<?php
mysql_select_db($database_Site,$Site) ;
$query_Categories = “SELECT id, name FROM categories ORDER BY id ASC” ;
$Categories = mysql_query ($query_Categories,$Site) or die(mysql_error ( ) ) ;
$row_Categories = mysql_fetch_assoc ($Categories);
$totalRows_Categories = mysql_num_rows ($Categories) ;
?>
```

პირველი გამოსახულება იყენებს ჩაშენებულ ფუნქციას **mysql_select_db**, რომელიც ახორციელებს მიერთებას მანაცემთა ბაზასთან. არგუმენტების სახით იგი იღებს მონაცემთა ბაზის სახელს სტრიქონულ ფორმატში (ფაილში **Site.php** გამოცხადებული ცვლადი **$database_Site)** და მონაცემთა სერვერთან დაკავშირების

იდენტიფიკატორს (ასევე, ფაილში **Site.php** გამოცხადებული ცვლადი **$Site**).
შემდეგი გამოსახულება კიდევ უფრო მარტივია-იგი ცვლადს **$query_Categories** ანიჭებს **SQL**-მოთხოვნის კოდს. ეს მოთხოვნა თვით **DreamweaverMX**-ის მიერ არის აგებული, Recordset დიალოგურ ფანჯარაში ჩვენს მიერ შეტანილი მონაცემების საფუძველზე (იხ. ნახ.8.6).
მესამე გამოსახულება იძახებს ჩაშენებულ ფუნქციას **mysql_query**, რომელიც ასრულებს SQL მოთხოვნას და ცვლადში **$Categories** აბრუნებს ჩანაწერების ანაკრების იდენტიფიკატორს. (ყურადღება მივაქციოთ იმას, რომ ცვლადის სახელი ემთხვევა დიალოგურ ფანჯარაში Recordset ჩვენს მიერ შეტანილი ჩანაწერების ანაკრების სახელს). არგუმენტების სახით ეს ფუნქცია იღებს **SQL** მოთხოვნის კოდს (ცვლადი **$query_Categories**) და მონაცემთა ბაზასთან მიერთების იდენტიფიკატორს.
თუ ფუნქცია **mysql_query** ვერ შეძლებს მოთხოვნის შესრულებას, იგი დააბრუნებს 0-მნიშვნელობას და ამ შემთხვევაში შესრულდება ფუნქცია die. ეს ფუნქცია წყვეტს ყველა **PHP** სცენარების შესრულებას მოცემულ გვერდზე და გამოაქვს შეტყობინება შეცდომის შესახებ, რომელიც მას არგუმენტის სახით გადაეცემა. ხოლო თვით ამ შეტყობინებას შეცდომის შესახებ სტრიქონული სახით დააბრუნებს ფუნქცია **mysql_error**.

მეოთხე გამოსახულება ასევე შეიცავს ჩვენთვის უცნობ ჩაშენებულ ფუნქციას-**mysql_fetch_assoc**. იგი არგუმენტის სახით იღებს ჩანაწერების ანაკრების იდენტიფიკატორს (ცვლადი **$Categories**), ამოალაგებს მონაცემებს ამ ანაკრების პირველი ჩანაწერიდან და დასამუშავებლად მოამზადებს მომდევნო-მეორე ჩანაწერს.
შედეგის სახით ფუნქცია **mysql_fetch_assoc** აბრუნებს მასივს (და ათავსებს მას ცვლადში **$row_Categories**). ამ მასივის ელემენტებია-ანაკრების დამუშავებული ჩანაწერების ველების მნიშვნელობები; მათ გააჩნიათ ინდექსები, რომლებიც წარმოადგენენ შესაბამისი ველების სახელებს სტრიქონული სახით. ანუ, ჩვენს შემთხვევაში მასივს **$row_Categories** ასეთი ელემენტები ექნება:

-$row_Categories[“name”] -name ველის მნიშვნელობა;
-$row_Categories[“id”] -id ველის მნიშვნელობა.

სწორედ ამ მასივიდან იქნება შემდგომში ამოლაგებული **Categories** ანაკრების ყველა ჩანაწერის მნიშვნელობები.
ბოლო გამოსახულებაც ძალიან მარტივია. ჩაშენებული ფუნქცია **mysql_num_rows**, არგუმენტის სახით იღებს ჩანაწერების ანაკრების იდენტიფიკატორს და აბრუნებს ამ ანაკრებში არსებული ჩანაწერების სრულ რაოდენობას (და ათავსებს მას ცვლადში **$totalRows_Categories**).

შენიშვნა! *პრინციპში მოცემული გამოსახულება აქ არც არის საჭირო. Categories.php გვერდზე არსებულ არცერთ სცენარში, ჩანაწერების რაოდენობა არ არის გამოყენებული. Dreamweaver-მა ეს გამოსახულება ჩასვა PHP კოდში იმ შემთხვევისათვის, თუ ჩვენ შევქმნიდით სერვერულ ქცევებს, რომლებშიც რაიმე მიზნებიდან გამომდინარე გამოყენებული იქნებოდა ჩანაწერების რაოდენობა (მაგალითად, სერვერულ ქცევებს, რომლებიც ახორციელებენ "გადაფურცვლას"). თუკი ჩვენ არ ვაპირებთ ასეთი სერვერული ქცევების შექმნას, მოცემული სცენარის უკანასკნელი გამოსახულება შეიძლება წაიშალოს.*

ყურადღება მივაქციოთ იმას, რომ ადრე განხილულ ორივე **PHP** სცენარი თავსდება ნებისმიერი **HTML**-კოდის წინ.
ვებ-გვერდის სათაურის სექციის განმსაზღვრელ **HTML**-კოდს ჩვენ გამოვტოვებთ, რადგანაც იქ არაფელია ჩვენთვის საინტერესო. პირდაპირ მივმართოთ ჩვენი ცხრილის შემქმნელ **HTML**-კოდს, რომელიც აგრეთვე მოიცავს ჩანაწერების ანაკრებიდან-**Categories**, ამ ცხრილში მონაცემების გამომტან **PHP** სცენარებსაც. ეს კოდი ასეთია (ყველა **PHP** სცენარი გამოყოფილია მუქი ფერით):

```
<TABLE WIDTH="400" BORDER="1" CELLSPACING="2" CELLPADDING="1">
<?php do { ?>
<TR>
<TD><?php echo $row_Categories['name'] ; ?><TD>
</TR>
<?php} while {$row_Categories = mysql_fetch_assoc ($Categories) ) ; ?>
</TABLE>
```

აქ ჩვენ ვხედავთ ერთი სტრიქონიდან და ერთი სვეტიდან შემდგარ ცხრილს. სტრიქონის შემქმნელი **HTML**-კოდი (ტეგი **<TR>** შიგთავსთან ერთად), წარმოადგენს პოსტპირობიანი ციკლის სხეულს (პოსტპირობიანი ციკლების შესახებ იხ. ***თავი7)***. ეს ნიშნავს იმას, რომ ცხრილის სტრიქონი განმეორებული იქნება საბოლოო ვებ-გვერდის (საიტის დამთვალიერებლისთვის გამოსატანი ვებ-გვერდის) **HTML**-კოდში იმდენჯერ, რამდენჯერაც შესრულდება ამ ციკლის სხეული.

ციკლის პირობის სახით გამოიყენება ასეთი, ჩვენთვის უკვე ნაცნობი გამოსახულება:

$row_Categories = mysql_fetch_assoc {$Categories));

ჩვენ უკვე ვიცით, რომ ის ახორციელებს მონაცემების ამოლაგებას მონაცემთა ანაკრების მიმდინარე ჩანაწერიდან და ამზადებს დასამუშავებლად მომდევნო ჩანაწერს. თუ ეს გამოსახულება მიმდევრობით მრავალჯერ შესრულდება (ის სწორედ ამგვარად შესრულდება, ვინაიდან ციკლის პირობას წარმოადგენს, რომელიც სხეულის ყოველი შესრულებისას გამოითვლება), ის ბოლოს და ბოლოს ჩანაწერთა ანაკრების-**Categories**-ის, ყველა ჩანაწერიდან ამოალაგებს მონაცემებს.
როდესაც მიღწეული იქნება ჩანაწერთა ანაკრების საბოლოო ჩანაწერი (ანუ ყველა ჩანაწერი უკვე დამუშავებული იქნება), ეს გამოსახულება დააბრუნებს მნიშვნელობას false-ანუ შეტყობინებას იმის შესახებ, რომ მეტი აღარ შეუძლია მონაცემების ამოლაგება. პირობა მაშინათვე გახდება მცდარი, შეწყდება მისი შესრულება და ციკლი დასრულდება.
მსგავსი ციკლი შეესაბამება სერვერულ ქცევას **Repeat Region** (განმეორებადი არე). თუმცა ამის მიხვედრა ისედაც არ არის ძნელი. ჩანაწერების ანაკრების-**Categories**, საკუთრივ **name** ველის გამოტანას ახორციელებს სცენარი

<?php echo $row_Categories [“name”] : ?>

აქ ყველაფერი ძალიან მარტივად არის. მასივიდან **$row_Categories** ამოიღება ელემენტი ინდექსით **name** (კატეგორიების შემცველი ველის სახელი სტრიქონული სახით) და ჩვენთვის კარგად ნაცნობი

ოპერატორის **echo**-ს საშუალებით გამოიტანება იგი ვებ-გვერდზე. ძნელი მისახვედრი არ არის, რომ ეს სცენარი შეესაბამება სერვერულ ქცევას **Dynamic Text** (დინამიური ტექსტი).
დაგვრჩვა მხოლოდ უკანასკნელი, ძალიან მარტივი სცენარის განხილვა, რომელიც **DreamweaverMX**-მა **HTML**-კოდის მთლად ბოლოში მოათავსა. აგერ ისიც:

<?php
mysql_free_result ($Categories) ;
?>

აქ გამოყენებულია **PHP**-ს ჩვენთვის უცნობი, კიდევ ერთი ჩაშენებული ფუნქცია-**mysql_free_result**. იგი მეხსიერებიდან შლის ჩანაწერების ანაკრებს, რომლის იდენტიფიკატორიც მისთვის გადაცემულია ერთად ერთი არგუმენტის სახით.

ყურადღება! *ჩანაწერების ანაკრები იკავებს სერვერული კომპიუტერის მეხსიერებას. რაც მეტ ჩანაწერს შეიცავს იგი, მით მეტი მეხსიერებაა საჭირო მისი შენახვისათვის. ამიტომ, ყოველთვის, ჩანაწერების ანაკრებთან მუშაობის დასრულებისთანავე აუცილებელია მისი წაშლა.*

ჩვენს მიერ განხილული **PHP**-ს სცენარები იყენებენ მრავალ, ჩვენთვის უცნობ ჩაშენებულ ფუნქციებს და ხერხებს. მაგრამ, სინამდვილეში ისინი არანაირ სირთულეს არ წარმოადგენენ. მათი დაწერის პრინციპების გაცნობის და გარკვეული პრაქტიკის მიღების შემდეგ, ჩვენ შეგვეძლება მათი დაწერა ხელითაც, **DreamweaverMX**-ის დახმარების გარეშე.
ამჯერად კი, მოდით ვნახოთ, ჩვენი ერთი შეხედვით დასრულებული ვებ-გვერდი **Categories.php** და დავფიქრდეთ იმაზე, თუ რას ვერ აკეთებს იგი ისე, როგორც საჭიროა.

მონაცემების ურთიერთგადაცემა სერვერულ ვებ-გვერდებს შორის

სწორია! მას გამოაქვს ყველა კატეგორიების სია-როგორც სტატიების, ასევე ფაილების-მიუხედავად იმისა, **default.htm** გვერდზე, თუ რომელ ჰიპერმინიშნებაზე გვაქვს დაწკაპუნებული. ეს კი ჩვენ სულაც არ გვჭირდება!

ჩვენ გვინდა როგორმე განვასხვავოთ ფაილების კატეგორია სტატიების კატეგორიისაგან, ხოლო ამისათვის ჩვენ უნდა ვიცოდეთ, თუ რომელ ჰიპერმინიშნებაზე დააწკაპუნა ჩვენი საიტის დამთვალიერებელმა. მოკლედ, ჩვენ სასწრაფოდ უნდა მოვიფიქროთ მონაცემთა გადაცემის ხერხი ერთი სერვერული ვებ-გვერდიდან მეორეზე. ახლა სწორედ ამით დავკავდებით.

მონაცემების გადაცემის ხერხი-GET.

საბედნიეროდ ჩვენ არაფრის გამოგონება არ დაგვჭირდება. ყველაფერი უკვე გამოგონილია ჩვენამდე. მოცემულ შემთხვევაში, ეს გახლავთ, ქსელში ერთი ვებ-გვერდიდან მეორეზე მონაცემების გადაცემის ***განსაკუთრებული მეთოდი,*** რომელსაც **GET-მეთოდი** ეწოდება. ჩვენ ახლა სწორედ მას განვიხილავთ.
თავი 1-დან ჩვენ გვახსოვს, რომ ვებ-დამთვალიერებელი, იმისათვის, რომ, მიიღოს ვებ-სერვერიდან მისთვის საჭირო ფაილი, ამ სერვერს უგზავნის კლიენტურ მოთხოვნას, რომელიც შეიცავს აღნიშნული ფაილის ინტერნეტ-მისამართს. ასე რომ, **GET**-მეთოდის შემთხვევაში, მონაცემები გადაიცემიან როგორც ამ მისამართის ნაწილი.
GET-მეთოდით გადასაცემი მონაცემები წარმოდგენილი უნდა იყვნენ

<არგუმენტის დასახელება> = <არგუმენტის მნიშვნელობა>

წყვილების ანაკრების სახით . (ჩანს ანალოგია ცვლადებთან). წყვილების ეს ანაკრები თავსდება ინტერნეტ-მისამართის მთლად ბოლოში და მისგან კითხვის ნიშნით გამოიყოფა. თვით წყვილები კი ერთმანეთისაგან გამოყოფილები უნდა იყვნენ კომერციული "და" ნიშნით-(&).

ასეთი ინტერნეტ-მისამართის მიღებისთანავე, ვებ-სერვერი გამოყოფს იმ ფაილის სახელს, რომელშიც შენახულია სერვერული ვებ-გვერდი, მასთან ერთად გადაცემული მონაცემებისაგან და აღნიშნულ მონაცემებს გადასცემს **PHP**-ს დამმუშავებელს. სერვერულ ვებ-გვერდზე არსებულ სცენარებს, განსაკუთრებული, ჩაშენებული (თვით **PHP**-ში გამოცხადებული) **$_GET**-მასივისათვის მიმართვის გზით, შეუძლიათ მოითხოვონ გადაცემული მონაცემები. ეს მასივი შეიცავს ელემენტებს, რმლებიც წარმოადგენენ **GET**-მეთოდით გადაცემული არგუმენტების მნიშვნელობებს, რომელთაც გააჩნიათ ინდექსები-ანუ ამ არგუმენტების სახელები სტრიქონული სახით.

ყოველივე ზემოთ აღნიშნული შეიძლება ძალიან რთულად მოგვეჩვენოს, ამიტომ მოდით განვიხილოთ **GET**-მეთოდის გამოყენება ჩვენს მაგალითზე. იმისათვის, რომ გავმიჯნოთ **ფაილების** კატეგორია **სტატიების** კატეგორისაგან, შემოვიღოთ არგუმენტი **file**. მნიშვნელობა 1-ით ავღნიშნოთ **ფაილების** კატეგორია, ხოლო მნიშვნელობა 0-ით **სტატიების** კატეგორია. მაშინ ჰიპერმინიშნებას, რომელიც მიუთითებს ფაილების კატეგორიების შემცველ გვერდზე, ექნება ასეთი სახე:

Categories.php?file=1,

ხოლო ჰიპერმინიშნებას, რომელიც მიუთითებს სტატიების სიაზე-
Categories.php?file=0

ცხადია, რომ კატეგორიების სიის გამოსატანად ჩვენ ვიყენებთ ერთსა და იმავე სერვერულ გვერდს-**Categories.php**. უბრალოდ ამ გვერდის სცენარი, ჩვენს მიერ გადაცემული **file** არგუმენტის მნიშვნელობის მიხედვით, **site**-მონაცემთა ბაზის **categories**-ცხრილიდან ამოალაგებს ჩანაწერების სხვადასხვა ანაკრებებს.
file არგუმენტისათვის მისამართად, ჩვენმა სცენარებმა უნდა გამოიყენონ შემდეგი სახის გამოსახულება:

$is_file = $_GET [“file”] ;

რომლის შესრულების შემდეგ ცვლადში **$is_file** მოთავსებული იქნება გადაცემული არგუმენტის-**file**-ის მნიშვნელობა.
რათქმა უნდა, მსგავსი რამის ხელით წერა ჩვენ არ მოგვიწევს-ყველაფერს ჩვენს ნაცვლად **DreamweaverMX**-ი გააკეთებს. ის შეცვლის **Categories.php** გვერდზე უკვე შექმნილ სცენარებს ისეთნაირად, რომ მათ შეძლონ **GET**-მეთოდით გადასაცემი მონაცემების დამუშავება. მაგრამ, ცხადია ჩვენც მოგვიწევს გარკვეული ქმედებების შესრულება.

ერთმანეთისათვის მონაცემების გადამცემი ვებ-გვერდების შექმნა.

ჯერ გავხსნათ **DreamweaverMX**-ში ჩვენი საიტის მთავარი გვერდი **default.htm** და შევასწოროთ ჰიპერმინიშნებები, რომლებიც მიანიშნებენ ფაილებისა და სტატიების კატალოგების სიებზე.

ფაილების კატეგორიების სიაზე მიმნიშნებელ ჰიპერმინიშნებას უნდა გააჩნდეს შემდეგი ინტერნეტ-მისამართი:

Categories.php?file=1,

ხოლო სტატიების კატეგორიების სიაზე მიმნიშნებელ ჰიპერმინიშნებას-ასეთი ინტერნეტ მისამართი:

Categories.php?file= 0,

შევინახოთ შესწორებული გვერდი **default.htm** და დავხუროთ იგი.
ახლა კი დადგა კატეგორიების სიის გვერდში **Categories.php**, ცვლილებების შეტანის დრო. გავხსნათ იგი, თუ ის ჯერ კიდევ არ არის გახსნილი. შემდეგ პანელში **Server Behaviors** მოვძებნოთ ჩანაწერების ანაკრების-Categories, შესაბამისი პუნქტი **Recordset** და ორჯერ დავაწკაპუნოთ მასზე. ეკრანზე გაჩნდება ჩვენთვის უკვე ნაცნობი დიალოგური ფანჯარა Recordset (იხ.ნახ.8.6).

ყურადღება! *თუკი ფანჯარას Recordset, გააჩნია ნახ.8.6.-ზე ნაჩვენებისაგან განსხვავებული სახე, ეს იმას ნიშნავს, რომ იგი გადართულია გაფართოებულ რეჟიმზე. ჩვეულებრივ რეჟიმზე დასაბრუნებლად, საჭიროა ღილაკზე Simple დაჭერა.*

მაშ ასე, ჩვენ უნდა შევარჩიოთ ჩანაწერები, **default.htm** გვერდიდან **GET** მეთოდით გადაცემული, **file** არგუმენტის მნიშვნელობაზე დაყრდნობით. ამისათვის ჩვენ დაგვჭირდება ბაზიდან მონაცემთა ამოკრების **SQL** მოთხოვნაში, ჩანაწერების ფილტრაციის კრიტერიუმის ჩამატება. უფრო ზუსტად თუ ვიტყვით, ჩვენ უნდა ვაცნობოთ **DreamweaverMX**-ს აღნიშნულის შესახებ.
ჩვენ გვჭირდება ჩანაწერების ფილტრაცია **file** ველის მიხედვით. ამიტომ ჩამოსაშლელ სიაში **Filter**, რომელიც განსაზღვრავს ველს, რომლის მიხედვითაც ხორციელდება ფილტრაცია, ავირჩიოთ პუნქტი **file.** მარჯვენა მხარეს განლაგებული ჩამოსაშლელი სია განსაზღვრავს შედარების ოპერატორს; ავირჩიოთ პუნქტი =.
Filter სიის ქვემოთ მდებარეობს მეორე ჩამოსაშლელი სია, რომლიდანაც აირჩევა იმ მნიშვნელობის სახე, რომელთანაც მოხდება ჩვენს მიერ არჩეული ველის **file**-ის მნიშვნელობის შედარება. ვინაიდან, ჩვენ გვჭირდება ამ ველის მნიშვნელობის შედარება **GET**

მეთოდით გადაცემულ არგუმენტთან, ამ სიიდან ავირჩიოთ პუნქტი **URL Parameter**. ამის შემდეგ გვრჩება მხოლოდ მარჯვენა მხარეს განლაგებულ შესატან ველში, **file** არგუმენტის სახელის შეტანა და ამით მორჩება ყველაფერი.

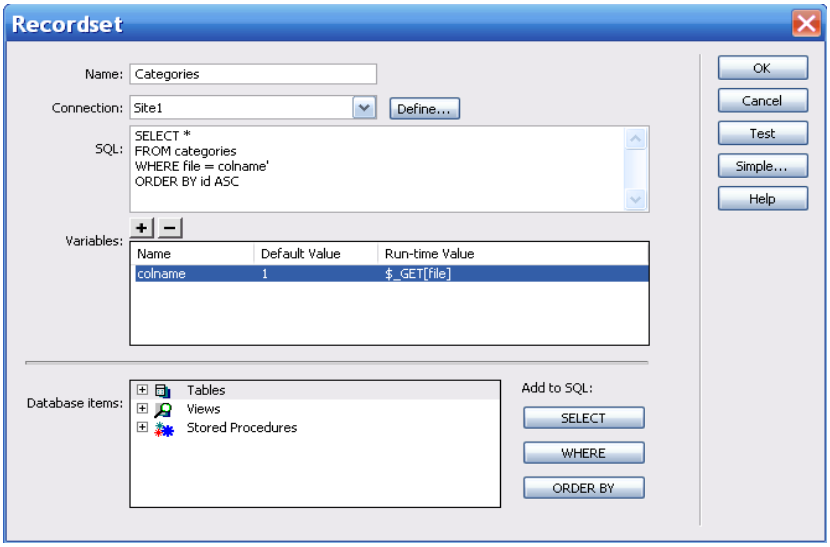
მე-7 თავიდან ჩვენთვის ცნობილია, რომ **PHP**-ში ლოგიკურ მნიშვნელობას **true** შეესაბამება ნებისმიერი არანულოვანი მნიშვნელობა, ხოლო მნიშვნელობას **faulse-0**. შეიძლება ითქვას, რომ **MySQL**-იც ექვემდებარება იგივე წესებს. (უფრო ზუსტად, ის ინახავს ლოგიკურ მნიშვნელობებს **true** და **faulse**-1 და 0 რიცხვების სახით, შესაბამისად).

ჩვენს მიერ შეტანილი მონაცემები ნაჩვენებია ნახ.8.14. შევამოწმოთ, სწორად შევიტანეთ ისინი თუ არა და შევასწოროთ შეცდომები თუ კი ასეთები არსებობენ.

ნახ.8.14. Recordset დიალოგური ფანჯრის ფრაგმენტი მართვის ელემენტებით, რომლებიც განსაზღვრავენ ჩანაწერების ფილტრაციის კრიტერიუმებს

პრინციპში, აწი უკვე შეიძლება **OK** ღილაკზე დაჭერა. მაგრამ, მოდით კიდავ ცოტა წავიმუშაოთ ჩანაწერების ანაკრებთან **Categories.** მოდით გავაკეთოთ ისე, რომ თუ არგუმენტი **file** საერთოდ არ იყო გადაცემული, გვერდმა **Categories.php**-იმ გაჩუმებით, სტატიების კატეგორიების სია გამოიტანოს.

დავაჭიროთ ღილაკს **Advanced**. დიალოგური ფანჯარა **Recordset** შეიცვლის სახეს-იხ.ნახ.8.15. ეს ეგრეთ წოდებული გაფართოებული რეჟიმია, რომელშიც ჩვენ შეგვიძლია **Recordset** სერვერული ქცევის დამატებითი პარამეტრების განსაზღვრა. კერძოდ, სწორედ აქ არის შესაძლებელი **file** არგუმენტისათვის გაჩუმებითი მნიშვნელობის მინიჭება.



ნახ.8.15. დიალოგური ფანჯარა Recordset (გაფართოებული რეჟიმი)

ჩამოსაშლელი სიები **Name** და **Connection** ჩვენთვის უკვე ნაცნობია. **SQL**-ის რედაქტირების არეში გამოისახება **DreamweaverMX**-ის საშუალებით, ჩვენს მიერ შეტანილი მონაცემების საფუძველზე ფორმირებული **SQL**-მოთხოვნის კოდი. აგერ ისიც:

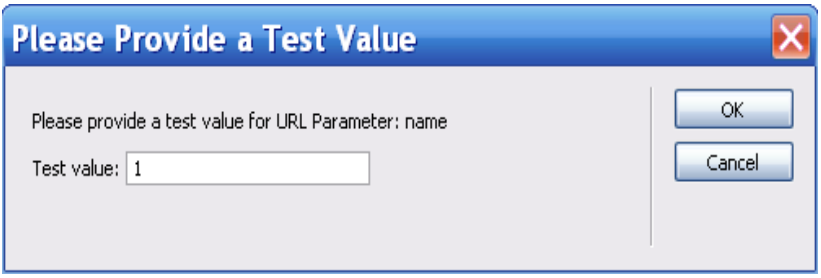
SELECT id, name FROM categories WHERE file = colname ORDER BY id ASC

აქ ყველაფერი ნათელია: **categories** ცხრილიდან ამოიკრიბება იმ ჩანაწერების id და **name** ველები, რომელთა **file** ველის მნიშვნელობები ტოლია **colname**-ის. ამოკრეფლი ჩანაწერების დახარისხება ხდება **id** ველის მზარდი მნიშვნელობების მიხედვით.
მაგრამ რას წარმოადგენს **colname**? ეს არის SQL-მოთხოვნის ცვლადი, რომელსაც ამ მოთხოვნის შესრულებისას **DreamweaverMX**-ი ანიჭებს საჭირო მნიშვნელობას (ჩვენს შემთხვევაში-**file** არგუმენტის მნიშვნელობა). **DreamweaverMX**-მა ეს ცვლადი შექმნა თავისთავისათვის სამუშაოს გასამარტივებლად.
ჩვენს მიერ ან **DreamweaverMX**-ის მიერ შექმნილი **SQL**-მოთხოვნის ყველა ცვლადი გამოისახება სიაში **Variables**. ეს სია შედგება სამი

სვეტისაგან: **Name** (ცვლადის სახელი), **Default Value** (გაჩუმებითი მნიშვნელობა) და **Run-time Value** (რეალური მნიშვნელობა, რომელიც ჩაისმება შესრულების დროს). სიის ზედა ნაწილში მდებარეობს ორი ღილაკი "**Plus-Плюс**" და "**Minus-Минус**" ნიშნაკებით. პირველი ქმნის ახალ ცვლადს, მეორე კი ამოშლის მონიშნულ ცვლადს სიიდან.

ჩვენ გვჭირდება **colname** ცვლადისათვის ახალი გაჩუმებითი მნიშვნელობის განსაზღვრა. ამისათვის, თაგუნათი დავაწკაპუნოთ მნიშვნელობაზე, რომელიც მოცემულია **Default Value** სვეტში (იქ მოცემულია 1-თვით **DreamweaverMX**-ის მიერ ჩასმული გაჩუმებითი მნიშვნელობა), შევიტანოთ იქ 0 და დავაჭიროთ კლავიშას **<Enter>**. მორჩა!

ახლა კი მოდით, გადავერთოთ **Recordset** დიალოგური ფანჯრის ჩვეულებრივ რეჟიმში, რისთვისაც საკმარისია **Simple** ღილაკზე დაჭერა. და მოდით შევამოწმოთ მონაცემთა ჩვენს მიერ შექმნილი ანაკრები მოქმედებაში. დავაჭიროთ ღილაკს **Test**, ეკრანზე გამოსული დიალოგური ფანჯრის **Please Provide** a **Test Value** (ნახ.8.16), ერთად ერთ შესატან ველში შევიტანოთ 0 ან 1, დავაჭიროთ ამ ფანჯრის **OK** ღილაკს და შევხედოთ შედეგს. ეს შედეგი გამოტანილი იქნება დიალოგურ ფანჯარაში **Test SQL Statement** (იხ. ნახ.8.7).



ნახ. 8.16. დიალოგური ფანჯარა Please Provide a Test Value

OK ღილაკზე მიმდევრობითი დაჭერით, დავხუროთ ყველა დიალოგური ფანჯარა. "ცოცხალი" დათვალიერების რეჟიმის საშუალებით, შევამოწმოთ მოქმედებაში მზა გვერდი **Categories.php** რადგანაც **file** არგუმენტი არ ყოფილა გადაცემული ამ გვერდისათვის, ის გამოიტანს სტატიების კატეგორიების სიას.

ახლა კი შეიძლება მზა სერვერული გვერდების შემოწმება "საბრძოლო" პირობებში. ამისათვის შევინახოთ **Categories.php**

გვერდი და **Site2** საიტის მთელი შიგთავსი გადავწეროთ ვებ-სერვერის ძირეულ საქაღალდეში (თუ როგორ უნდა გაკეთდეს ეს აღწერილი იყო მე-4 თავში). ამის შემდეგ გავუშვათ ვებ-დამთვალიერებელი და მისამართების სტრიქონში ავკრიბოთ **http://localhost:8080/**. როდესაც გაიხსნება ჩვენი საიტის მთავარი გვერდი, მიმდევრობით დავაჭიროთ ჰიპერმინიშნებებს **ფაილები** და **სტატიები** და შევხედოთ თუ რას მივიღებთ.
თუკი ჩვენს მიერ ყველაფერი სწორად იქნება გაკეთებული, ჩვენმა პირველმა სერვერულმა ვებ-გვერდმა **Categories.php**-იმ სწორად უნდა დაამუშაოს **file** არგუმენტის სხვადასხვა მნიშვნელობები და გამოიტანოს ფაილებისა და სტატიების კატეგორიების სიები. მართალია, მისი სათაური იგივე დარჩება-**სტატიები**, მაგრამ ამას ჩვენ შემდგომში შევასწორებთ.

მიღებული მონაცემების დამუშავების უზრუნველმყოფი PHP-კოდის გარჩევა

ჯერ მოდით ვნახოთ, რა ცვლილებები შეიტანა **DreamweaverMX**-მა მის მიერ ადრე შექმნილ **PHP** კოდში.
საკუთრივ ცვლილებები, განიცადა **Recordset** ქცევის შესაბამისმა ერთად ერთმა სცენარმა. (ეს სწორიც არის, ჩვენ ხომ მხოლოდ ჩანაწერების ანაკრების პარამეტრებს ვცვლიდით). ახლა ის ასე გამოიყურება (შეცვლილი ფრაგმენტები გამოყოფილია მუქი შრიფტით):

```
<?php
$colname_Categories = “0” ;
If (isset($_GET[‘file’])) {
$colname_Categories = {get_magic_quotes_gpc( ) ) ? $_GET [‘file’] : addslashes ($_GET [‘file’] ;
}
mysql_select_db($database_Site, $Site) ;
$query_Categories = sprintf(“SELECT id, name FROM categories WHERE file = %s ORDER BY id ABC”, $colname_Categories) ;
$Caategories = mysql_query($query_Categories, $Site) or die(mysql_error ( )) ;
$row_Categories = mysql_fetch_assoc($Categories) ;
$totalRows_Categories = mysql_num_rows($Categories);
?>
```

პირველივე გამოსახულება აცხადებს ცვლადს **$colname_Categories** და ანიჭებს მას სტრიქონულ მნიშვნელობას ”0”-ს. ეს ცვლადი შეესაბამება **SQL**-მოთხოვნის ცვლადს **colname**, ხოლო ”0”-მისი გაჩუმებითი მნიშვნელობაა.

შემდეგ მოდის პირობითი გამოსახულება, რომელიც ამოიღებს file არგუმენტის მნიშვნელობას და მიანიჭებს მას ცვლადს **$colname_Categories**. განვიხილოთ იგი უფრო დაწვრილებით.

ჩაშენებული ფუნქცია **isset** აბრუნებს **true** მნიშვნელობას, თუ მისთვის პარამეტრის სახით გადაცემული ცვლადი (ან მასივის ელემენტი, როგორც ჩვენს შემთხვევაში) გამოცხადებულია, ხოლო წინააღმდეგ შემთხვევაში მნიშვნელობას **false**. ეს ფუნქცია გამოიყენება შემოწმების პირობაში, არსებობს თუ არა **$_GET** მასივის ელემენტი **file** ინდექსით, ან სხვა სიტყვებით რომ ვთქვათ, გადაეცა თუ არა მოცემულ გვერდს არგუმენტი **file.**
თუ არგუმენტი **file** გადაცემული იყო, მაშინ მისი მნიშვნელობა უნდა მიენიჭოს ცვლადს **$colname_Categories** (თუკი არა, მაშინ ამ ცვლადს ისევე გაჩუმებითი მნიშვნელობა ექნება). ამას ასრულებს გამოსახულება.

$colname_Categories = (get_magic_quotes_gpc ( )) ? $_GET [‘file’] : addslashes ($_GET[‘file’]) ;

აქ თავს იჩენს პრობლემა, რომლის გასწორებასაც ცდილობს **DreamweaverMX**-ი. საქმე იმაშია, რომ **PHP**-ს სტრიქონულ მნიშვნელობებში დაუშვებელია ზოგიერთი სიმბოლოების გამოყენება (მაგალითად, ორმაგი ბრჭყალების). თუკი **file** არგუმენტს ექნება მნიშვნელობა, რომელიც შეიცავს დაუშვებელ სიმბოლოებს, წარმოიქმნება შეცდომა. იმისათვის, რომ სტრიქონულ გამოსახულებებში შესაძლებელი იყოს დაუშვებელი სიმბოლოების გამოყენება, მათ წინ უნდა დაიწეროს ”შებრუნებული სლეშის” ნიშანი (\). ამას აკეთებს ფუნქცია **addslashes.**
მაგრამ არსებობს კიდევ სხვა პრობლემა. **PHP**-ს დამმუშავებელი შეიძლება გაწყობლი იყოს ისეთნაირად, რომ **GET** მეთოდით გადაცემული მონაცემების მიღებისას, ყველა დაუშვებელ სიმბოლოებს ავტომატურად წარუმძღვაროს წინ ”შებრუნებული

სლეშის" ნიშანი (ეგრეთ წოდებული "**magic_quotes_gpc** პარამეტრი"). გამოდის, რომ საჭიროა იმის შემოწმება, ჩართულია თუ გამორთული ეს პარამეტრი; თუკი ის ჩართულია, მაშინ არ არის საჭირო **addslashes** ფუნქციის გამოძახება. ამაზე "პასუხისმგებელია" ფუნქცია, რომელიც აბრუნებს ლოგიკურ მნიშვნელობას **true**-ს, თუ პარამეტრი **magic_quotes_gpc** ჩართულია და **false**-ს-წინააღმდეგ შემთხვევაში.

სწორედ ამიტომ არის **file** არგუმენტის მნიშვნელობის ამომღები და მისი **$colname_Categories** ცვლადისათვის მიმნიჭებელი გამოსახულება ასეთი რთული. **DriamweaverMX**-ი უბრალოდ ცდილობს გვერდი აუაროს "წყალქვეშა ქვებს", რაც მას საკმაოდ კარგადაც გამოსდის.
ბოლო გამოსახულება, რომელიც ჩვენ უნდა განვიხილოთ, ასრულებს **SQL**-მოთხოვნის კოდის მინიჭებას **$query_Categories** ცვლადისათვის. მოდით კიდევ ერთხელ შევხედოთ მას:

$query_Categories = sprintf("SELECT id , name FROM categories WHERE file = %s ORDER BY id ASC" , $colname_Categories) ;

აქ გამოიყენება მორიგი, ჩვენთვის უცნობი ჩაშენებული ფუნქცია **sprintf**. ამ ფუნქციის მოვალეობაა მისთვის პირველი არგუმენტით გადაცემული სტრიქონული ცვლადის აღება, მასში ეგრეთ წოდებული **შაბლონების** მოძებნა და მათი შეცვლა მომდევნო არგუმენტების მნიშვნელობებით: პირველი შაბლონი-მეორე არგუმენტი, მეორე შაბლონი-მესამე არგუმენტი და ა.შ.. შაბლონებს გააჩნიათ (%)-ის ნიშნის სახე, თანმდევი მნიშვნელობების მონაცემთა ტიპების აღმნიშვნელი ასოებით, რომლებიც შაბლონის ნაცვლად უნდა იქნან ჩასმული.
შევხედოთ ამ ფუნქციის პირველ არგუმენტს-**SQL** მოთხოვნის კოდს. მასში მხოლოდ ერთი შაბლონია გამოყენებული-%s, რომელიც აღნიშნავს სტრიქონულ მნიშვნელობას. ეს მნიშვნელობა აღებული იქნება **sprint** ფუნქციის მეორე არგუმენტიდან-ცვლადიდან **$colname_Categories**-და ჩაისმება შაბლონის ნაცვლად.
ამ სცენარის დანარჩენი **PHP** კოდი ჩვენთვის ნაცნობია. ასე, რომ აღარ შევუდგებით მის განხილვას.

უფრო რთული სერვერული გვერდები

მაშ ასე. ძირითადი ცნებები ჩვენ უკვე შევისწავლეთ. დადგა დრო, როდესაც ხელი უნდა მოვკიდოთ რაიმე უფრო რთულ საქმეს.
წინ კიდევ ბევრი საქმე გველის. კატეგორიების სიის შემცველ ვებ-გვერდს ყოველთვის გამოაქვს ერთი და იგივე სათაური-**სტატიები**, მიუხედავად იმისა, სტატიების კატეგორია გამოიტანს თუ არა ფაილებს. იგივე ეხება სათაურსაც (<TITLE> ტეგის შიგთავსი). თუმცა, რა სათაურებზე და სახელწოდებებზეა საუბარი-ჩვენ ხომ ჯერ არ შეგვიქმნია მომხმარებლის მიერ არჩეულ კატეგორიას მიკუთვნებული **სტატიებისა** და **ფაილების** სიის ვებ-გვერდი. შევუდგეთ მუშაობას!

PHP სცენარების ხელით დაწერა

დავიწყოთ იმით, რომ ვაიძულოთ გვერდი **Categories.php** გამოიტანოს მისი შიგთავსის შესაბამისი სათაური, იმისდა მიხედვით, თუ რას შეიცავს იგი, **სტატიების** თუ **ფაილების** კატეგორიებს.
ხელით დავწეროთ **PHP** სცენარი, რომელიც ამას განახორციელებს-**DreamweaverMX**-ი ამ შემთხვევაში ვერაფერში ვერ დაგვეხმარება.
მაშ ასე, გავხსნათ გვერდი Categories.php, თუ ის ჯერ კიდევ არ არის გახსნილი და გადავიდეთ **HTML**-კოდის წარმოჩენის რეჟიმში. ჩვენი პირველი, ხელით დაწერილი სცენარი, ჩავსვათ უშუალოდ **Rekordset** სერვერული ქცევის, შესაბამისი სცენარის შემდეგ (სწორედ იქ იყო გამოცხადებული ცვლადი **$colname_Categories**, რომელსაც ჩვენ ვიყენებთ) და **HTML**-კოდის წინ.
ჩვენს სცენარს ასეთი სახე ექნება:

```
<?php
switch ($colname_Categories) {
   case “0”:
     $scat = “სტატიები” ;
     Break ;
   case “1” :
      $cat = “ფაილები” ;
      break ;
 }
?>
```

ამაში რთული არაფერია. ჩვენ უბრალოდ ვიყენებთ შერჩევის გამოსახულებას **$colname_Categories** ცვლადის მნიშვნელობის შესამოწმებლად და **$scat** ცვლადისათვის შესაბამისი მნიშვნელობის მისანიჭებლად. **$scat** ცვლადის მნიშვნელობა კი გამოიტანება საბოლოო ვებ-გვერდზე.
მაშ ასე, ჩავსვათ ადრე განხილული სცენარი, უშუალოდ **Recordset** სერვერული ქცევის შესაბამისი სცენარის შემდეგ. ამის შემდეგ მოვძებნოთ ვებ-გვერდის სახელის შემცველი ტეგი-**<TITLE>.** მას ასეთი სახე ექნება:

<TITLE>სტატიები</TITLE>

ოდნავ შევასწოროთ იგი, რათა ჩვენს ვებ-გვერდს ჰქონდეს მოცემული შემთხვევის შესაბამისი სახელწოდება :

<TITLE><?php echo $scat ; ?></TITLE>

ახლა, მოსაძებნი გვრჩება ტეგი **<H1>**, რომელმაც ვებ-გვერდზე უნდა გამოიტანოს პირველი დონის სათაური:

<H1>სტატიები</H1>

და ისიც შევასწოროთ:

<H1><?php echo $scat ; ?></H1>

თუ ჩვენ ამის შემდეგ გადავერთვებით ვებ-გვერდის წარმოჩენის რეჟიმში, დავინახავთ, რომ **DreamweaverMX**-მა სათაურის ”სტატიები” ნაცვლად გამოიტანა ასეთი ნიშანი-  .
ის აღნიშნავს **HTML**-კოდში ჩასმულ **PHP**-სცენარს.
ყურადღება მივაქციოთ იმ გარემოებას, რომ არც ერთი სცენარი, რომელიც მდებარეობს **HTML** კოდის წინ და მის შემდეგ, დოკუმენტის ფანჯარაში არანაირად არ აისახება. მათ დასანახად უნდა გადავერთოთ **HTML** კოდის წარმოჩენის რეჟიმში.
ახლა შევინახოთ ჩვენი გვერდი, საიტის ყველა ფაილი გადავწეროთ ვებ-სერვერის ძირეულ საქაღალდეში, გავუშვათ ვებ-

დამთვალიერებელი, მისამართების სტრიქონში ავკრიბოთ **http://localhost:8080** და დავაწკაპუნოთ ერთ-ერთ ჰიპერმინიშნებაზე-ფაილები ან სტატიები. გილოცავთ-ჩვენი სცენარი მუშაობს!
აქ ჩნდება კითხვა: რატომ არ უნდა ჩავსვათ **$scat** ცვლადისათვის მნიშვნელობის მიმნიჭებელი **PHP**-კოდი, პირდაპირ **Recordset** სერვერული ქცევის შესაბამის სცენარში? რათქმა უნდა შეიძლება ამის გაკეთება, მაგრამ ამ შემთხვევაში ჩვენ დავკარგავთ ამ სერვერულ ქცევასთან **Server Behaviors** პანელის საშუალებით მუშაობის შესაძლებლობას, რადგანაც **DreamweaverMX**-ი აღარ მიიჩნევს ამ სცენარს "თავისად". ამ შემთხვევაში ჩვენ **PHP** კოდის ხელით ჩასწორება მოგვიწევს.
ახლა ჩვენ გვჭირდება ნიადაგის მომზადება მოცემულ კატეგორიაში შემავალი სტატიებისა და ფაილების სიის გამომტანი, სერვერული ვებ-გვერდის შესაქმნელად(მას, როგორც ჩვენ უკვე მე-5 თავში გადავწყვიტეთ, ეწოდება **Items.php**).
ამ მიზნით, ჩვენ დაგვჭირდება კატეგორიის ყოველი დასახელების ჰიპერმინიშნებად გარდაქმნა და მისი საშულებით **Items.php** გვერდისათვის რაიმე არგუმენტის გადაცემა.

ეს არგუმენტი იქნება **Categories**-ანაკრების, შესაბამისი ჩანაწერის **id** ველის მნიშვნელობის ტოლი. (აი, ეს ველიც დაგვჭირდა!). დავარქვათ მას **catid**.
სრულიად გასაგებია, თუ რა სახე უნდა ჰქონდეს ასეთ ჰიპერმინიშნების ინტერნეტ-მისამართს:

Items.php?catid =<id ველის მნიშვნელობა>

ჰიპერმინიშნებების შექმნაც ჩვენ უკვე ვიცით-ამის შესახებ ვისაუბრეთ ***მე-3 თავში.*** მაშ ასე, გადავერთოთ ვებ-გვერდის წარმოჩენის რეჟიმში, თაგუნას დაწკაპუნებით მოვნიშნოთ კატეგორიის დასახელების გამომტანი დინამიური ტექსტი და თვისებების რედაქტორის ჩამოსაშლელ სიაში **Link** შევიტანოთ ასეთი რამ:

Items.php?catid=

ამის შემდეგ გადავერთოთ **HTML**-კოდის წარმოჩენის რეჟიმში და ეკრანზე გამოვიტანოთ პანელი **Bindings**-ამის გაკეთება ყველაზე მარტივია **<Ctrl>+<F10>** კლავიშების კომბინაციაზე დაჭერით.
პანელში **Bindings,** ვხსნით ჩანაწერების ანაკრების-Categories, შესაბამის "ხეს", თაგუნას დაწკაპუნებით ვირჩევთ "შტოს"-id, გადავათრევთ მას ჩვენს მიერ ახლახანს შექმნილი ჰიპერმინიშნების **HREF** ატრიბუტის მნიშვნელობაზე და ვსვამთ უშუალოდ ტოლობის ნიშნის შემდეგ. აღნიშნულის მერე ამ ჰიპერმინიშნების **HTML**-კოდმა უნდა მიიღოს ასეთი სახე:

<A HREF = "Items.php? catid=<?php echo $row_Categories['id'] ;?>"><?php echo $row_Categories['name'] ; ?></A>

მნიშვნელობების ერთდროული გამოტანა რამდენიმე ველიდან

მაშ ასე, ჰიპერმინიშნება უკვე მზად გვაქვს. დაგვრჩა მხოლოდ თვით **Items.php** ვებ-გვერდის შექმნა. ამისათვის, მენიუ **File**-ში ვირჩევთ პუნქტს **New, New Document** დიალოგური ფანჯრის Category სიიდან ვირჩევთ პუნქტს **Dynamic Page**, ხოლო მარჯვენა სიაში პუნქტს-**PHP**. ამის შემდეგ ვაჭერთ ღილაკს Create-და ახალი სერვერული ვებ-გვერდი მზად არის. შევიტანოთ მასში რაიმე განმარტებითი ტექსტი და დახურვის გარეშე შევინახოთ **Site2** საიტის ძირეულ საქაღალდეში **Items.php** სახელით.
ახლა შევქმნათ ჩანაწერების ანაკრები. ამისათვის, ეკრანზე გამოვიტანოთ პანელი **Bindings**, დავაწკაპუნოთ **"plus(плюс)"** ნიშნაკიან ღილაკზე და გამოსულ მენიუში ავირჩიოთ პუნქტი **Recordset(Query).** დიალოგურ ფანჯარაში **Recordset** განვსაზღვროთ ასეთი პარამეტრები:

-მონაცემთა ანაკრების სახელი (შესატანი ველი **Name**)-**Items**;
-კავშირის სახელი (ჩამოსაშლელი სია **Connection**)-**Site**;
-ცხრილის სახელი (ჩამოსაშლელი სია **Table**)-**items**;
-ცხრილის არჩეული ველები (გადამრთველი **Selected**, ჯგუფში **Columns**);
-**author, name, added და href** (სია **Columns**) ველების შემცველი ჩანაწერების ანაკრები ;
-ფილტრაცია **catid** ველის მიხედვით (ჩამოსაშლელი სია **Filter**);
-**catid** ველის მნიშვნელობა ტოლი უნდა იყოს (ჩამოსაშლელი სიის პუნქტი=, რომელიც **Filter** სიის მარჯვნივ მდებარეობს);

-**GET** მეთოდით გადაცემული არგუმენტის მნიშვნელობა (**Filter** სიის ქვემოთ მდებარე ჩამოსაშლელი სიის პუნქტ **URL Parameter)**;
-რომლის სახელიც არის-**catid** (**Filter** სიის მარჯვენა ქვედა ნაწილში მდებარე შესატანი ველი);
-დახარისხება **added** ველის მიხედვით (ჩამოსაშლელი სია **Sort**);
-ამ ველის მნიშვნელობების კლებადობის მიხედვით (**Sort** სიის მარჯვნივ მდებარე ჩამოსაშლელი სიის პუნქტი **Descending**).

ახლა დავაჭიროთ ღილაკს **OK** და შევხედოთ **DreamweaverMX**-ის მიერ შექმნილ ჩანაწერების ანაკრებს-იგი გამოსახული იქნება **Binding** პანელის სიაში "ხეს" სახით.
ამის შემდეგ ვებ-გვერდზე **Items.php** შევქმნათ ცხრილი. მას ექნება ორი სტრიქონი (ზედა-დასახელების სტრიქონი), ოთხი სვეტი და სიგანე 100 %.
ამ ცხრილის სვეტებში წარმოდგენილი იქნება შემდეგი მონაცემები:

-პირველი-სტატიის ავტორის და ფაილის შემმუშავებლის სახელი;
-მეორე-სტატიის ან ფაილის დასახელება;
-მესამე-განახლების თარიღი;
-მეოთხე-გრაფიკული ჰიპერმინიშნება სტატიაზე ან ფაილზე.

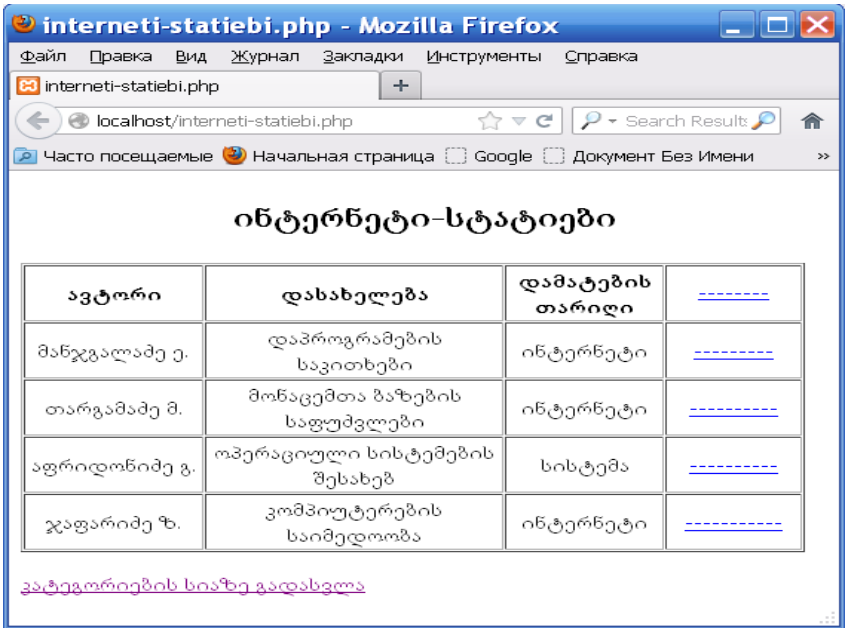
ამ მიზნით ჩვენ გამოვიყენებთ **Arrow.gif** ფაილში შენახულ გრაფიკულ გამოსახულებას.
ცხრილის პირველ სტრიქონში შეტანით, მივანიჭოთ ამ სვეტებს სახელები. შემდეგ, შევინახოთ ვებ-გვერდი და გავაგრძელოთ მუშაობა.
ჩვენთვის უკვე ცნობილია, თუ როგორ უნდა შევქმნათ ჩანაწერების ანაკრების რაიმე ველიდან, მონაცემების ამომკრეფი დინამიური ტექსტი. ამისათვის საკმარისია ვებ-გვერდის წინასწარ განსაზღვრულ ადგილზე გადავათრიოთ **Binding** პანელის იერარქიული სიის შესაბამისი "შტო". გადავათრიოთ "შტო" **author**, ცხრილის მეორე სტრიქონის პირველ უჯრედში, "შტო" **name**-მეორეში, ხოლო "შტო" added-მესამეში. "შტო"-ს **href** ჯერ-ჯერობით შევეშვათ.
აღნიშნულის შემდეგ ცხრილის მეორე სტრიქონის მეოთხე უჯრედში მოვათავსოთ გრაფიკული გამოსახულება **Arrow.gif**, მოვნიშნოთ იგი თაგუნაზე დაწკაპუნებით და გადავაქციოთ ჰიპერმინიშნებად. თვისებების რედაქტორის ჩამოსაშლელ სიაში **Link**, შევიტანოთ ნებისმიერი ინტერნეტ-მისამართი, რაც თავში მოგვივა-სულ ერთია

შემდგომში ჩვენ მას მაინც წავშლით. გადავერთოთ **HTML**-კოდის წარმოჩენის რეჟიმში, მოვძებნოთ ახლახანს შექმნილი ჰიპერმინიშნების აღმწერი კოდი და წავშალოთ <A> ტეგის **HREF** ატრიბუტის მთელი შიგთავსი. თვით ეს ატრიბუტი და თვით ტეგი დავტოვოთ. **Bindings** პანელიდან, ბრჭყალებში, რომლებშიც მოთავსებულია **HREF** ატრიბუტის მნიშვნელობა, გადავათრიოთ "შტო" href და შევხედოთ მიღებულ შედეგს:

<A HREF="<?php echo $row_Items['href'];?><IMG SRC="Arrow.gif"></A>

(**IMG** ტეგი შესაძლებელია შეიცავდეს სხვა ატრიბუტებსაც, მაგრამ ისინი აქ არ არიან წარმოდგენილი).
ზემოთ აღნიშნულიდან გამომდინარე ცხადია, რომ უნდა შეიქმნას განმეორებადი არე, რათა ჩვენმა ვებ-გვერდმა შეძლოს **Items** ანაკრების ყველა ჩანაწერის გამოტანა. ტექსტური კურსორი მოვათავსოთ ცხრილის მეორე სტრიქონის ნებისმიერ უჯრედში და მთლიანად ამ სტრიქონის მოსანიშნად, სტატუსის სტრიქონის ტეგების სექციაში დავაწკაპუნოთ ღილაკზე **<tr>. Server Behaviors** პანელის გამოსატანად დავაჭიროთ კლავიშების კომბინაციას **<Ctrl>+<F10>**, შემდეგ დავაჭიროთ ამ პანელის **"plus(плюс)"** ნიშნაკიან ღილაკს და ეკრანზე გამოსულ მენიუში ავირჩიოთ პუნქტი **Repeat Region**.
ეკრანზე გამოსულ დიალოგურ ფანჯარაში **Repeat Region**, განვსაზღვროთ განმეორებადი არის შემდეგი პარამეტრები:
-მონაცემთა ანაკრების სახელი (ჩამოსაშლელი სია **Recordset)-Items;**
-ანაკრების ყველა ჩანაწერის გამოტანა (**Show** ჯგუფში შემავალი გადამრთველი **All records**).

ყოველივე ამის შემდეგ დავაჭიროთ ღილაკს OK და შევინახოთ გვერდი.
ახლა მოდით ვნახოთ, თუ რა გამოგვივიდა. საიტის ყველა ფაილი გადავწეროთ ვებ-სერვერის ძირეულ საქაღალდეში, გავუშვათ ვებ-დამთვალიერებელი და მისამართების სტრიქონში ჩავწეროთ **http://localhost:8080**. როდესაც ვებ-დამთვალიერებელი გახსნის ჩვენი საიტის მთავარ გვერდს, მივყვეთ ნებისმიერ ჰიპერმინიშნებას სტატიების და ფაილების სიის ნებისმიერ გვერდამდე. თუ ჩვენ ყველაფერი სწორად გვაქვს გაკეთებული, ასეთი რამე უნდა დავინახოთ-იხ.ნახ.8.17.



ნახ.8.17. სტატიებისა და ფაილების სიის გვერდი Items.php.

მართალია, განსაკუთრებული სიხარულის საფუძველი ჯერ-ჯერობით არა გვაქვს. ჯერ-ერთი, თარიღის მნიშვნელობები გამოიტანება **PHP**-ს შიდა ფორმატით-"წელი-თვე-რიცხვი", ხოლო ჩვენთვის უფრო მისაღებია "რიცხვი-თვე-წელი". მეორე-ვებ-გვერდის სათაურიც და მისი დასახელებაც ისევ "მოწყვეტილი არიან რეალობას." მესამე-კარგი იქნებოდა თუ ქვედა ნაწილში მითითებული იქნებოდა ჰიპერმინიშნება, რომელიც მიანიშნებდა სტატიებისა და ფაილების კატეგორიების შემცველ გვერდებზე, შესაბამისად.
დავიწყოთ ჩვენი გვერდის სათაურით და მისი დასახელებით.

რამოდენიმე მონაცემთა ანაკრებიანი ვებ-გვერდები

იმისათვის, რომ შეგვეძლოს ვებ-გვერდის ტექსტში იმ კატეგორიის დასახელბის ჩასმა, რომლის სტატიებიც (ან ფაილებიც) მიმდინარე მომენტში მასზეა წარმოდგენილი, ჩვენ უნდა შეგვეძლოს საიდანმე მისი ამოღება. ჩვენს შემთხვევაში, ამოღება შესაძლებელია მხოლოდ

site-მონაცემთა ბაზის, **categories** ცხრილიდან. ეს იმას ნიშნავს, რომ ვებ-გვერდზე **Items.php**, ჩვენ დაგვჭირდება კიდევ ერთი ჩანაწერთა ანაკრების შექმნა, რომელშიც მხოლოდ ერთი ჩანაწერი იქნება მოთავსებული-ის ჩანაწერი, რომლის **id** ველის მნიშვნელობა ტოლი იქნება **catid** ატრიბუტისათვს **GET** მეთოდით გადაცემული მნიშვნელობისა.
დავაჭიროთ კლავიშების კომბინაციას **<Ctrl>+<F10>**, რათა ეკრანზე გამოვიტანოთ პანელი **Bindings,** დავაწკაპუნოთ **"plus(плюс}"** ნიშნაკიან ღილაკზე და ეკრანზე გამოსულ მენიუში ავირჩიოთ პუნქტი **Recordset (Query)**. დიალოგურ ფანჯარაში Recordset განვსაზღვროთ შემდეგი პარამეტრები:

-მონაცემთა ანაკრების სახელწოდება (შესატანი ველი **Name**)-**Category**;
-კავშირის სახელწოდება (ჩამოსაშლელი სია **Connection**)-**Site**;
-ცხრილის სახელწოდება (ჩამოსაშლელი სია **Table**)-**categories**;
-ცხრილის არჩეული ველები (**Columns** ჯგუფში, გადამრთველი-**Selected**);
-ჩანაწერების ანაკრები შეიცავს ველებს **nam**e და **file** (სია **Columns)**;
-ფილტრაცია **id** ველის მიხედვით (ჩამოსაშლელი სია **Filter**);
-**id** ველის მნიშვნელობა უნდა უდრიდეს (ჩამოსაშლელი სიის პუნქტი=, რომელიც მდებარეობს **Filter** სიის მარჯვენა მხარეს);
-**GET** მეთოდით გადაცემული არგუმენტის მნიშვნელობას (ჩამოსაშლელი სიის პუნქტი **URL Parameter**, რომელიც **Filter** სიის ქვემოთ მდებარეობს);
-რომლის სახელი არის-**catid** (შესატანი ველი **Filter** სიის მარჯვენა ქვედა მხარეს);
-დახარისხების გარეშე (ჩამოსაშლელი სია **Sort**-ის პუნქტი **None**).
ჩანაწერების ანაკრების შესაქმნელად დავაჭიროთ ღილაკს **OK**. **Bindings** პანელის იერარქიულ სიაში გაჩნდება ახალი "ხე".
ამის შემდგომ ყველაფერი ძალზე ადვილია. გადავერთვებით **HTML**-კოდის წარმოსახვის რეჟიმში და ვეძებთ ტეგს **<TITLE>**, რომელიც ვებ-გვერდის სახელს განსაზღვრავს. აგერ ისიც:

<TITLE>ინტერნეტი-სტატიები</TITLE>

მთლიანად ვშლით მის შიგთავსს, მოვნიშნავთ "შტო"-ს name **Bindinds** პანელის **Recordset (Category)** სიის "ხეზე", გადავათრევთ მას და

”ვაგდებთ” პირდაპირ გამხსნელ და დამხურავ **<TITLE>** ტეგებს შორის. ასეთი რამე უნდა მივიღოთ:

```
<TITLE><?php echo $row_Category['name'] ; ?></TITLE>
```

იგივეს ვაკეთებთ **<H1>** ტეგთან მიმართებაშიც, რომელიც განსაზღვრავს ვებ-გვერდის დასახელებას.
ახლა მოდით გავაკეთოთ ისე, რომ კატეგორიის დასახელების შემდეგ, ბრჭყალებში გამოისახოს სიტყვები **სტატიები** და **ფაილები**. ამისათვის ვებ-გვერდის კოდში ჩავსვათ ჩვენთვის უკვე ნაცნობი ასეთი სცენარი:

```
<?php
switch ($row_Category ['file']) {
    case “0” :
      $scat = “სტატიები” ;
      break ;
    case “1” :
      $scat = “ფაილები” ;
      break ;
}
?>
```

და მოვათავსოთ იგი უშუალოდ Recordset სერვერული ქცევის შესაბამისი სცენარის შემდეგ. ეს სცენარი ამოიღებს მნიშვნელობას ჩანაწერთა ანაკრების Categories, **file** ველიდან და იმისდა მიხედვით თუ როგორია ეს მნიშვნელობა, ცვლადს $scat მიანიჭებს სტრიქონის მნიშვნელობას **სტატიები** ან **ფაილები**.
დაგვრჩა მხოლოდ ჩვენს მიერ **<TITLE>** და **<H1>** ტეგებში ჩასმული სცენარის დასრულება. იგი ასე უნდა გამოიყურებოდეს:

```
<TITLE><?php echo $row_Category ['name'] . “(“ . $scat . “)” ;?></TITLE>
```

(ეს არის სცენარი **<TITLE>** ტეგისათვის; სცენარს **<H1>** ტეგისათვის ასეთივე სახე ექნება).
უკანასკნელი, რასაც ჩვენ გავაკეთებთ-ეს არის ვებ-გვერდის ქვემოთ მდებარე, კატეგორიების სიის შემცველ გვერზე მიმნიშნებელი ჰიპერმინიშნების შესწორება. გავაკეთოთ ისე, რომ მასზე

დაწკაპუნებისას მომხმარებელი ფაილების სიიდან ხვდებოდეს ფაილების კატეგორიების სიაში, ხოლო სტატიების სიიდან-სტატიების კატეგორიების სიაში. ამისათვის მას უნდა განუსაზღვროთ ასეთი ინტერნეტ-მისამართი:

Categories.php?file=<1-ფაილებისათვის, 0-სტატიებისათვის>

ხოლო, **file** არგუმენტისათვის მნიშვნელობის აღება ჩვენ შეგვიძლია Category ჩანაწერთა ანაკრების **file** ველიდან.
მაშ ასე, მოვნიშნოთ სტრიქონი **სიების კატეგორია** და თვისებების რედაქტორის ჩამოსაშლელ სიაში Link ავკრიბოთ:

Categories.php?file=

შემდეგ გადავერთოთ **HTML**-კოდის წარმოჩენის რეჟიმში, მოვძებნოთ ახლახანს შექმნილი ჰიპერმინიშნების კოდი, და **<A>** ტეგის **HREF** ატრიბუტის მნიშვნელობაში, უშუალოდ ტოლობის შემდეგ, ხელით ჩავამატოთ ის, რაც ქვემოთ მუქი შრიფტით არის გამოყოფილი:

Categories.php? file=<?php echo $row_Category['file'] ; ?>

ბოლოს და ბოლოს ჩვენ უკვე საკმარისი გამოცდილება შევიძინეთ **PHP** დაპროგრამებაში, იმისათვის, რათა შეგვეძლოს კოდში ზოგიერთი წვრილმანის ხელით ჩამატება! რათქმა უნდა Category "ხე"-ს **file** "შტო"-ს გადათრევა შეიძლება იქ **Bindings** პანელიდან, მაგრამ ეს უკვე აღარ არის ჩვენთვის საინტერესო.
ესეც ასე! უკვე შეიძლება ჩვენი საიტის გამოქვეყნება ლოკალურ ვებ-სერვერზე და მისი გამოცდა ვებ-დამთვალიერებელში. შეეცადეთ გააკეთოთ ეს დახმარების გარეშე-დროა მიეჩვიოთ დამოკიდებელ მუშაობას.

თარიღის მნიშვნელობის სწორად გამოტანა

ახლა უკვე შეგვიძლია მივხედოთ თარიღის მნიშვნელობებს. უზრუნველვყოთ მათი გამოტანა არა ისე, როგორც ეს სურს **PHP**-ს, არამედ ისე, როგორც ჩვენთვის იქნება უფრო მოსახერხებელი-ფორმატში "რიცხვი, თვე, წელი".

PHP იყენებს ჩაშენბულ ფუნქციას **date**, რომელიც იღებს ორ არგუმენტს. პირველი არგუმენტი-ეს არის იმ ფორმატის აღმწერი სტრიქონული გამოსახულება, რომლითაც უნდა მოხდეს თარიღის გამოტანა, ხოლო მეორე-ეს თვით რიცხვით ფორმატში გამოსატანი თარიღის მნიშვნელობაა. ეს ფუნქცია აბრუნებს თარიღის დაფორმატებული მნიშვნელობის შემცველ სტრიქონს.
მოდით შევხედოთ თარიღის მნიშვნელობის გამომტანი დინამიური ტექსტის შესაბამის სცენარს:

<?php echo $row_Items ['added'] ; ?>

და დავამატოთ მასში ფუნქცია date:

<?php echo date ("d.m.Y' , strtotime ($row_Items ['added'])) ; ?>

აქ ჩვენ, აღნიშნულ ფუნქციას, პირველ არგუმენტად გადავეცით სტრიქონი "**d.m.Y**", რომელიც აღწერს ჩვენთვის სასურველ ფორმატს "რიცხვი, თვე, წელი)". ეს სტრიქონი შეიცავს შაბლონებს, რომელთა ნაცვლად **PHP** ჩასვამს რიცხვის მნიშვნელობებს და თვისა და წლის შესაბამის ნომრებს. ასეთი შაბლონები სულ სამია:
-"d"-რიცხვის გამოტანა ორი ციფრის საშუალებით. თუ რიცხვი შედგება ერთი ციფრისაგან, ის გამოტანილი იქნება ნულთან ერთად. ასე მაგალითად, რიცხვი 2-სათვის გამოსასვლელზე ჩვენ მივიღებთ 02-ს, ხლო რიცხვი 21-სათვის-21-ს.
-"**m**"-ისთვის შესაბამისი ნომრის გამოტანა, ასევე ორი ციფრის საშუალებით. თუ თვის შესაბამისი რიცხვი ერთი ციფრისაგან შედგება, იგი გამოტანილი იქნება ნულთან ერთად.

-"**Y**"-წლის გამოტანა ოთხი ციფრის საშუალებით.
და კიდევ ერთი. რადგანაც ფუნქცია date, მეორე არგუმენტს იღებს რიცხვითი ფორმატით, ხოლო მასივში **$row_Items** ეს არგუმენტი ინახება სტრიქონულ ფორმატში, ჩვენ მოგვიწევს კიდევ ერთი ჩაშენებული ფუნქციის **strtotime** გამოძახება, რომელიც თარიღის სტრიქონულ მნიშვნელობას გარდაქმნის რიცხვით მნიშვნელობად.

ჩავსვათ დასრულებული სცენარი **HTML**-კოდში ძველის ადგილას. შევინახოთ ვებ-გვერდი და შევამოწმოთ იგი ვებ-

დამთვალიერებელში. დავინახავთ, რომ თარიღის მნიშვნელობა სწორად იქნება გამოტანილი.

ვებ-გვერდის ელემენტების პირობითი გამოტანა.

ჩვენი **Items.php** ვებ-გვერდი ყველანაირად კარგია. ერთად-ერთი გამონაკლისის გარადა-თუ ანაკრებში არ იქნება არც ერთი ჩანაწერი (მომხმარებლის მიერ არჩეულ კატეგორიაში არ არის არცერთი ფაილი და არცერთი სტატია), ცხრილი მანც იქნება გამოტანილი, თანაც ძალიან ლამაზად. მოდით რაიმე მოვუხერხოთ ამას.
ერთ-ერთი შესაძლო გადაწყვეტა ასეთია-ვამოწმებთ, არის თუ არა ანაკრებში ჩანაწერები:

If ($row_Items = mysql_fetch_assoc ($Items)) {

და გამოგვაქვს სიის შემცველი ცხრილი. თუ კი ჩანაწერების ანაკრები ცარიელია:

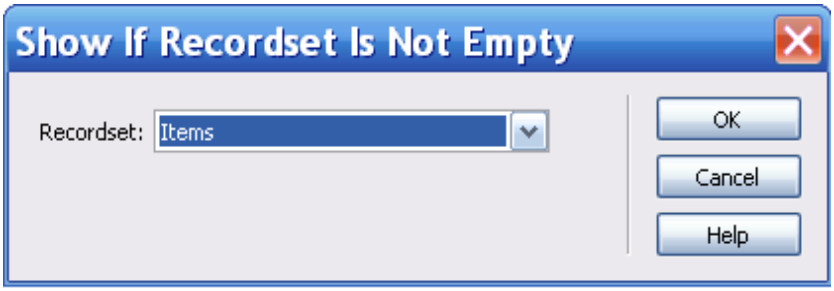
} else {

გამოგვაქვს შემდეგი სახის გაფრთხილება:

უკაცრავად, ეს კატეგორია ადმინისტრატორს არა აქვს შევსებული სტატიებით (ფაილებით).

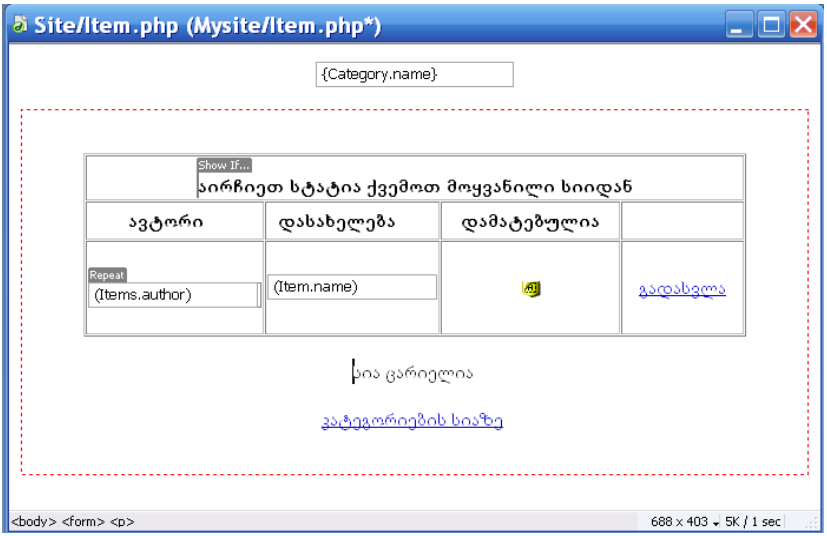
ჩვენ სწორედ ასე გავაკეთებდით, ჩვენი ვებ-გვერდის კოდს ხელით რომ ვწერდეთ. მაგრამ, ვინაიდან ჩვენ ვიყენებთ შესანიშნავ პაკეტს **DreamveaverMX**-ს, მოდით ვისარგებლოთ მისი მომსახურებით. მითუმეტეს რომ, მას შეუძლია ჩვენთვის დახმარების გაწევა, რაიმე პირობის შესრულებისას გამოსატანი ვებ-გვერდების ***არასავალდებულო არეების*** შექმნის შესაძლებლობის უზრუნველყოფით.
მაშ ასე, მოდით ჩავსვათ პირდაპირ ცხრილის ქვეშ, კატეგორიების სიის მიმთითებელი ჰიპერმინიშნების ზევით, მოკლე აბზაცი **Recordset Is Empty-Список пуст-** ***სია ცარიელია.*** ამის შემდეგ მოვნიშნოთ სტრიქონი **choose the article from the list given below** (აირჩიეთ სტატია ქვემოთ მოცემული სიიდან)-***აირჩიეთ სტატია ქვემოთ წარმოდგენილი სიიდან*** და ცხრილი. ყველაფერი ამისაგან

შევქმნათ არასავალდებულო არე, რომელიც იმ შემთხვევაში გამოიტანება, როდესაც ჩანაწერების ანაკრები არ იქნება ცარიელი.
უკვე ჩვეული მოძრაობით დავაჭიროთ კლავიშების კომბინაციას **<Ctrl>+<F10>**, რათა გამოვიტანოთ ეკრანზე პანელი **Server Behaviors**. ამ პანელში დავაჭიროთ "**plus(плюс)**" ნიშნაკიან ღილაკს, ეკრანზე გამოსულ მენიუში ავირჩიოთ პუნქტი **Show Region**, ხოლო ამის შემდეგ გამოსულ ქვემენიუში-პუნქტი **Show If Recordset Is Empty**. ეკრანზე გაჩნდება დიალოგური ფანჯარა **Show If Recordset Is Not Empty** (ნახ. 8.18).



ნახ.8.18. დიალოგური ფანჯარა Show If Recordset Is Empty

ერთად-ერთი, რაც ჩვენ უნდა გავაკეთოთ ამ ფანჯარაში-ეს არის ის, რომ ჩამოსაშლელ სიაში **Recordset**, ავირჩიოთ ჩვენი ჩანაწერების ანაკრები **Items**. ამის შემდეგ შეგვიძლია დავაჭიროთ ღილაკს **OK**. ჩვენს მიერ შექმნილი არასავალდებულო არე ნაჩვენებია ნახ. 8.9-ზე.



ნახ.8.19. ვებ-გვერდი Items.php არასავალდებულო არით

ახლა მოვნიშნოთ აბზაცი **Recordset Is Empty-Список пуст-** ***სია ცარიელია.*** ჩვენ მას გარდავქმნით არასავალდებულო არედ, რომელიც გამოიტანება იმ შემთხვევაში, თუ ჩანაწერების ანაკრები პირიქით, არ არის ცარიელი.
დავაჭიროთ **Server Behaviors** პანელის "**plus(плюс)**" ნიშნაკიან ღილაკს, ეკრანზე გამოსულ მენიუში ავირჩიოთ პუნქტი **Show Region**, ხოლო შემდეგ გამოსულ ქვემენიუში-პუნქტი **Show If Recordset Is Empty**. ეკრანზე გაჩნდება დიალოგური ფანჯარა **Show If Recordset Is Empty**, რომელიც ძალიან წააგავს ჩვენთვის უკვე ნაცნობ ფანჯარას **Show If Recordset Is Not Empty**. ამ ფანჯრის ჩამოსაშლელ სიაში **Recordset**, ავირჩიოთ ისევ იგივე ჩანაწერების ანაკრები **Items** და დავაჭიროთ ღილაკს **OK**.

თუ ამის შემდეგ ჩვენ გამოვაქვეყნებთ ჩვენს საიტს ლოკალურ ვებ-სერვერზე და შევამოწმებთ მას ვებ-დამთვალიერებელში, იმ ვებ-გვერდებზე, რომლებზეც გამოსახული იქნება "ცარიელი" კატეგორიების შიგთავსი, გამოტანილი იქნება ტექსტი **Recordset Is Empty-Список пуст-სია ცარიელია.**

თუ კი ამის შემდეგ, ჩვენ გადავერთვებით ჩვენი ვებ-გვერდის **HTML**-კოდის წარმოჩენის რეჟიმში და მოვძებნით **Show If Recordset Is Empty** სერვერული ქცევის შესაბამის სცენარს:

```
<?php if ($totalRows_Items == 0) { //Show if recordset empti ?>
<P>სია ცარიელია.</P>
<? Php } // Show if recordset emty ?>
```

გაკვირვებულები დავრჩებით-**DreamweaverMX**-ი თითქოს მიხვდა ჩვენს ჩანაფიქრს. მის მიერ შექმნილი სცენარი, ამოწმებს **Recordset** სერვერული ქცევის შესაბამის სცენარში გამოცხადებული ცვლადის **$totalRows_Items** მნიშვნელობის, ნულთან ტოლობას:

```
$totalRows_Items = mysql_num_rows ($Items) ;
```

როდესაც ჩვენ ამ სცენარს ვიხილავდით, იმასაც კი ვფიქრობდით, რომ ეს გამოსახულება შეიძლებოდა ზედმეტიც ყოფილიყო-მაშინ ხომ ცვლადი **$totalRows Items** არსად არ იყო გამოყენებული. მაგრამ აღმოჩნდა, რომ ის მაინც დაგვჭირდა!
რაც შეეხება სერვერული ქცევის **Show If Recordset Is Not Empty_Показать если набор не пустой-ჩვენე თუ სია ცარიელი არ არის** შესაბამის სცენარს, იგი ამოწმებს პირობას, მეტია თუ არა ნულზე ცვლადის მნიშვნელობა:

```
<?php if ($totalRows_Items > 0) { // Show if recordset not empty ?>
<P>აირჩიეთ სტატია ქვემოთ მოტანილი სიიდან</P>
. . .
<?php } // Show if recordset not empty ?.
```

ამით ჩვენ დავასრულეთ არასავალდებულო არეებზე საუბარი.

ჩანაწერების ანაკრებების შესახებ ცნობების გამოტანა.

უკანასკნელი, რასაც ჩვენ გავაკეთებთ **Items.php** გვერდთან დაკავშირებით-ეს არის ის, რომ ვაიძულებთ მას გვაჩვენოს არჩეულ კატეგორიაში შემავალი სტატიებისა და ფაილების რაოდენობა, ანუ ანაკრებში ჩანაწერების რაოდენობა. (ზოგადად ეს არც არის საჭირო მაგრამ, მაინც იყოს).

ანაკრებში ჩანაწერების რაოდენობა გამოისახება ცალკე სტრიქონში, რომელიც სტატიების (ფაილების) სიის შემცველი ცხრილის ქვემოთ მდებარეობს. ცხადია, რომ ეს სტრიქონი უნდა მოხვდეს ჩვენს მიერ ადრე შექმნილ არასავალდებულო არეში, რომელიც ეკრანზე იმ შემთხვევაში გამოიტანება, თუ ანაკრები შეიცავს ჩანაწერებს. ამიტომ ჩვენ როგორმე უნდა "ჩავაკვეხოთ" ეს სტრიქონი არასავალდებულო არეში. ამის გაკეტება ყველაზე მარტივია **HTML**-კოდის წარმოჩენის რეჟიმში.

მაშ ასე, გადავერთოთ **HTML**-კოდის წარმოჩენის რეჟიმში და მოვძებნოთ ის ადგილი, სადაც მთავრდება პირველი არასავალდებულო არე, რომელიც ეკრანზე გამოიტანება იმ შემთხვევაში, თუ ანაკრები შეიცავს ჩანაწერებს. ქვემოთ მოცემულია ამ ადგილის აღმწერი **HTML**-კოდი:

```
<?php } while ($row_Items = mysql_fetch_assoc ($Items) } ; ?>
</TABLE>
<?php } // Show if recordset not empty ?>
<?php if ($totalRows_Items == 0) { // Show if recordset empty ?>
```

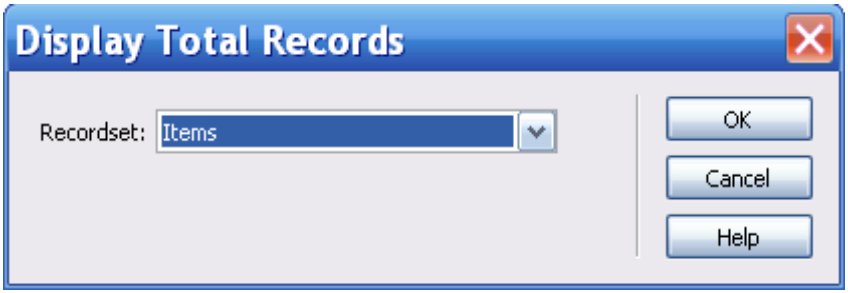
(მართალია, აქ **PHP** უფრო მეტია, ვიდრე **HTML**). უშუალოდ **</TABLE>**ტეგის შემდეგ ჩავსვათ ტეგი **<P>**, რომელიც შეიცავს ტექსტს ***სულ ჩანაწერები:*** (+აუცილებელი გამოტოვება!). ამის შემდეგ კოდი მიიღებს შემდეგ სახეს (ჩასმული ფრაგმენტი გამოყოფილია მუქი ფერით):

```
<?php } while ($row_Items = mysql_fetch_assoc ($Items) } ; ?>
</TABLE>
<P>სულ ჩანაწერები:</P>
<?php } // Show if recordset not empty ?>
<?php if ($totalRows_Items == 0) { // Show if recordset empty ?>
```

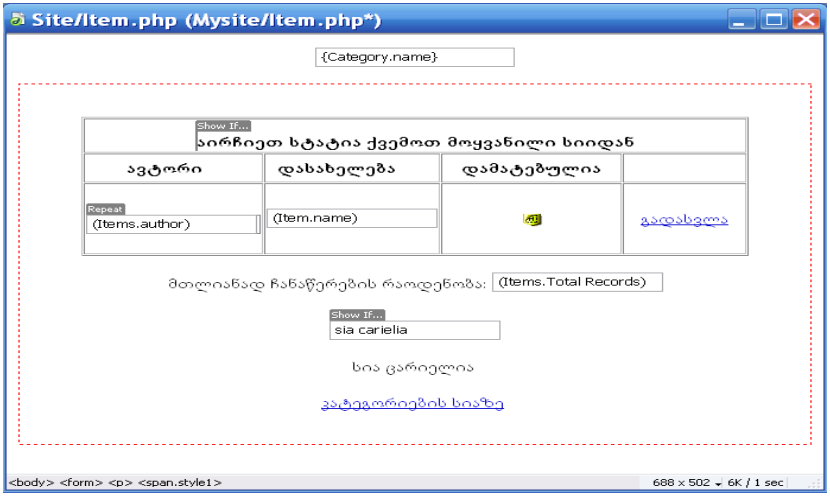
ახლა ჩვენ უკვე გვაქვს იმავე არასავალდებულო არეში შემავალი ტექსტური აბზაცი, რომელ არეშიც მოთავსებულია სიების შემცველი ცხრილი. გვრჩება მხოლოდ ანაკრებში ჩანაწერების რაოდენობის გამომტანი PHP სცენარის ჩამატება მასში. ამისათვის ჩვენ ისევ **DreamveaverMX**-ის მომსახურებით ვისარგებლებთ.

გადავერთოთ ისევ ვებ-გვერდის წარმოჩენის რეჟიმში და ტექსტური კურსორი დავსვათ ახლახანს შექმნილი აბზაცის ბოლოს, უშუალოდ

“გამოტოვების” შემდეგ. ეკრანზე გამოვიტანოთ პანელი **Server Behaviors**, დავაჭიროთ მასზე **”plus (плюс)”** ნიშნაკიან ღილაკს, ეკრანზე გამოსულ მენიუში ავირჩიოთ პუნქტი **Display Record Count**, ხოლო შემდეგ გამოსულ ქვემენიუში-პუნქტი **Display Total Records**. ეკრანზე გამოჩნდება მცირე ზომის დიალოგური ფანჯარა **Display Total Records** (ნახ. 8.20.).



ნახ.8.20. დიალოგური ფანჯარა Display Total Records.



ნახ.8.21. არასავალდებულო არის და ანაკრებში ჩანაწერების რაოდენობის შესაბამის სტრიქონის შემცველი ვებ-გვერდი Items.php

ჩამოსაშლელ სიაში **Recordset**, ვირჩევთ ჩანაწერების ანაკრებს **Items**, რომლის ჩანაწერების რაოდენობის ჩვენებაც გვინდა ვებ-გვერდზე,

და ვაჭერთ ღილაკზე **OK**. ამის შემდეგ ჩვენი სერვერული ვებ-გვერდის **Items.php** სახე იქნება ისეთი, როგრიც ნაჩვენებია ინახ.8.21-ზე.
დაგვრჩა მხოლოდ გამზადებული ვებ-გვერდის შემოწმება მოქმედებაში.
თუ ჩვენ შევხედავთ ახლახანს შექმნილი **Display Total Records** სერვერული ქცევის შესაბამის **PHP** სცენარს, დავინახავთ, რომ იგი ძალზე მარტივია:

<P>სულ ჩანაწერები:<?php echo $total_Items ?></P>

აქ ეკრანზე გამოიტანება **Recordset** სერვერული ქცევის შესაბამის სცენარში გამოცხადებული, **$totalRows Items** ცვლადის შიგთავსი. ეს ცვლადი ჩვენ ისევ გამოგვადგა!

რა იქნება ამის შემდეგ?

ჩვენ ახლახან შევქმენით მარტივი ვებ-საიტი, რომელშიც აქტიური სერვერული ვებ-გვერდებია გამოყენებული. წინა "სტატიკური" ვებ-საიტისაგან განსხვავებით, ამ შემთხვევაში სულ სამი ვებ-გვერდი დაგვჭირდა, ესენია-**default.html**, **Categories.php** და **Items.php**.
შეიძლება ითქვას, რომ ჩვენ უკვე ვიცით **DreamweaverMX**-ის გარემოში მარტივი სერვერული ვებ-გვერდების შექმნა. ახლა დადგა რთული ვებ-გვერდებისა და რთული სცენარების შექმნის დრო, რომლებიც უზრუნველყოფენ მონაცემთა ბაზაში **site**, მონაცემების შეტანისა და ჩასწორების შესაძლებლობების რეალიზაციას. ჩვენ ხომ მოგვიწევს სიებში ახალი სტატიებისა და ფაილების დამატება, შეცდომების გასწორება და ზოგიერთი მათგანის, რომლებმაც უკვე შეწყვიტეს ქსელში თავიანთი არსებობა-წაშლა. წინააღმდეგ შემთხვევაში ჩვენი საიტი ძალიან სწრაფად დაკარგავს თავის აქტუალურობას და შესაბამისად მომხმარებლებსაც.
შემდეგი თავი მიეძღვნება მონაცემების დამატების, შეცვლის და წაშლისთვის განკუთვნილი სერვერული ვებ-გვერდების შექმნას.
ამიტომ, ჯერ-ჯერობით არ ვხურავთ **DreamweaverMX**-ს, არ ვაჩერებთ არც **MySQL**-ს და არც Apache-ს. ჩვენი უმარტივესი საიტი ჯერ ხომ ბოლომდე არ არის დასრულებული.

თავი 9

მონაცემების შეტანისა და ჩასწორების რეალიზაცია.

მაშ ასე, ბაზიდან ვებ-გვერდზე მონაცემების გამოტანის საკითხებში ჩვენ უკვე გავერკვიეთ მე-8 თავში. დადგა დრო, რომ გავერკვეთ, თუ როგორ შეიძლება **HTML**-ისა და **PHP**-ს საშუალებით ბაზაში ახალი მონაცემების შეტანისა და არსებული მონაცემების ჩასწორების რეალიზაცია. ასევე დადგა დრო, როდესაც ჩვენ ბოლოსდაბოლოს უნდა შევქმნათ **site** ბაზაში მონაცემების შემტანი გვერდი, რათა ამ მიზნით აღარ გამოვიყენოთ მონაცემების პროგრამა-კლიენტით.
მიუხედავად იმისა, რომ მე-8 თავის ბოლოს ვებ-გვერდებს, რომლებიც ახორციელებენ მონაცემების შეტანასა და ჩასწორებას, ჩვენ ვუწოდეთ რთული ვებ-გვერდები, სინამდვილეში არავითარი განსაკუთრებული სირთულე მათში არ არის. უბრალოდ საჭიროა ზოგიერთ პრინციპებში გარკვევა, რომელთა საფუძველზეც ხდება მათი აგება და **DreamweaverMX**-ის იმ სერვერული ქცევების შესწავლა, რომელთა გამოყენებითაც ხდება მომხმარებლის მიერ მონაცემების უშუალოდ გამოტანა.
თავიდან, როგორც წესი, გავიაროთ მცირე თეორიული კურსი. სწორედ მისი საშუალებით შევისწავლით ჩვენ მომხმარებლისაგან მონაცემების მიღების და მათი სერვერული პროგრამისათვის გადაცემის ძირითად პრინციპებს. ასე, რომ კომპიუტერი ჩვენ ჯერ-ჯერობით არ დაგვჭირდება.

როგორ ხდება მონაცემების შეტანისა და გადაცემის რეალიზაცია.

ზოგადად, მომხმარებლისაგან მონაცემების მიმღები და მათი დამმუშავებელი (ჩვენს შემთხვევაში-მათი მონაცემთა ბაზაში შემტანი) ვებ-გვერდების შესაქმნელად სამი პრობლემაა გადასაწყვეტი:
-სახელდობრ, ამ მონაცემების მიღება;
-მათი კოდირება ვებ-დამთვალიერებლის მხარეზე;
-მათი გადაცემა სერვერული პროგრამისათვის კოდირებული სახით.
მიღებული მონაცემების დეკოდირება ხორციელდება თვით **PHP**-დამმუშავებლის მიერ, ჩვენი მხრიდან ყოველგვარი ჩარევის გარეშე, ასე რომ, ჩვენ ამაში მონაწილეობის მიღება არ დაგვჭირდება.

შენიშვნა! *ვინაიდან PHP ტექნოლოგიის დანიშნულება სწორედ სერვერული პროგრამების შექმნაში მდგომარეობს, მას გააჩნია ჩაშენებული საშუალებები მიღებული მონაცემების დეკოდირებისათვის. თუკი სერვერული პროგრამა იწერება სხვა ტექნოლოგიების გამოყენებით (მაგალითად C++-ში ან Pascal-ში), მაშინ მათი დეკოდირებით დაკავება მოუწევს თვით პროგრამისტს.*

მოდით ვნახოთ, თუ როგორ ხდება ამ პრობლემების გადაწყვეტა **HTML**-ისა და **PHP**-ს საშუალებით.

მონაცემების შეტანა. ფორმები

როგორ ხდება მონაცემების შეტანა **Windows**-ის ჩვეულებრივ გამოყენებით პროგრამებში? ძალიან მარტივად-ფანჯრების და მართვის ელემენტების საშუალებით: შესატანი ველები, სიები, ალმები, გადამრთველები და ღილაკები. ეკრანზე ჩნდება მართვის ელემენტების შემცველი ფანჯარა, რომლებშიც ჩვენ უნდა შევიტანოთ რაიმე მონაცემები. მათი შეტანის შემდე ვაჭერთ ღილაკს **OK** და პროგრამა იწყებს შეტანილი მონაცემების დამუშავებას. ყველაფერი ეს ჩვენ უკვე ვიხილეთ, იმავე **DreamweaverMX**-ის შემთხვევაშიც.
როგორღა იქნება ვებ-გვერდების შემთხვევაში? ზუსტად იგივენაირად! ვებ-დამთვალიერებელს გამოაქვს ისეთივე მართვის ელემენტების ანაკრების შემცველი ვებ-გვერდი, როგორიც გამოყენებულია **Windows**-ის გამოყენებით პროგრამებში. და ჩვენ შეგვყავს მათში მონაცემები.
Windows-პროგრამების მართვის ელემენტების მსგავსად, რომლებიც აუცილებლად შესაბამის ფანჯრებში უნდა მდებარეობდნენ, ვებ-გვერდების მართვის ელემენტებიც ფორმებში უნდა იყვნენ განთავსებული. **ფორმა**-ეს ვებ-გვერდის განსაკუთრებული ელემენტია, მართვის ელემენტების თავისებური კონტეინერი, რომლიდანაც მონაცემები გადაეცემიან ვებ-სერვერს. ნახ.9.1.-ზე ნაჩვენებია ასეთი ფორმის მაგალითი. ეს ფორმა, რომელიც შეიცავს მხოლოდ ორ შესატან ველს, გამოიყენება მომხმარებლის სახელისა და პაროლის განსაზღვრისათვის, დაცულ საიტში შესაღწევად.

ნახ.9.1. ფორმის მაგალითი.

ყურადღება! *მართვის ყველა ელემენტები, რომლებიც გამოიყენებიან ვებ-სერვერისათვის გადასაცემი მონაცემების შესატანად, აუცილებლად ფორმაში უნდა იყვნენ განთავსებული.*

თვით მონაცემების გაგზავნის პროცესი იწყება მას შემდეგ, რაც მომხმარებელი დააჭერს განსაკუთრებულ ღილაკს. ამ ღილაკს ეწოდება **გაგზავნა-Отправить-Send** (ინგლისურენოვან პროგრამებში-**Submit)** და იგი აუცილებლად უნდა არსებობდეს ფორმაზე.
ასევე შესაძლებელია რომ ფორმაზე წარმოდგენილი იყოს ღილაკი **ჩამოგდება-Reset-Сброс** , რომელიც ანულებს მომხმარებლის მიერ შეტანილ მონაცემებს. როგორც წესი ორივე ეს ღილაკი მდებარეობს ფორმის უკიდურეს ქვედა ნაწილში.
ფორმა იქმნება განსაკუთრებული წყვილი ტეგის **<FORM>** საშუალებით, რომლის შიგნითაც თავსდებიან მართვის ელემენტების შესაბამისი ტეგები:

<FORM ACTION = “givemedata.php>
. . .
</FORM.

ამ ტეგის აუცილებელი ატრიბუტი **ACTION** განსაზღვრავს იმ სერვერული პროგრამის სახელს, რომელმაც უნდა მიიღოს შეტანილი მონაცემები.
მართვის ელემენტების უმეტესობა იქმნება ერთეულოვანი ტეგის **<INPUT>**-ის საშუალებით. ამ ტეგს გააჩნია აუცილებელი ატრიბუტები **TYPE** და **NAME**. პირველი ატრიბუტი განსაზღვრავს მართვის ელემენტის **ტიპს-შესატანი ველი, ალამი** თუ **ღილაკი**, ხოლო მეორე-მის უნიკალურ სახელს.
მოდით განვიხილოთ ისეთი მცირე ზომის ფორმის, როგორიც ნაჩვენებია ნახ. 9.1.-ზე შემქმნელი HTML-კოდი:

```
<FORM ACTION = “givemedata.php”>
   <P>დასახელება: <INPUT TYPE = “text”  NAME=”name”></P>
   <P>პაროლი: <INPUT TYPE = “text” NAME=”password”></P>
   <P><INPUT TYPE = “submit” NAME = “submit”></P>
</FORM>
```

ამ კოდის პირველი სტრიქონი ჩვენთვის უკვე ნაცნობია-იგი ქმნის ფორმას, რომელიც შეტანილ მონაცემებს გაუგზავნის სერვერულ გვერდს givemedata.php. ეს ყველაფერი ჩვენ უკვე ვიცით.
მეორე და მესამე სტრიქონები ქმნიან შესატან ველებს, შესაბამისად მომხმარებლის სახელისა და მისი პაროლის განსასაზღვრავად. ჩანს, რომ მათ შემქმნელ <INPUT> ტეგებს გააჩნიათ ატრიბუტები <TYPE>, text მნიშვნელობებით, ანუ წარმოადგენენ შესატან ველებს.
მეოთხე სტრიქონი ქმნის ღილაკს Submit (Отправить). აქ <INPUT> ტეგის ატრიბუტს-TYPE, უნდა ჰქონდეს მნიშვნელობა submit.
მართვის ელემენტების სახელების განსასაზღვრავად შესაძლებელია გამოყენებულ იქნას აგრეთვე ტეგი **ID**:

```
<INPUT TYPE = “text”  ID=”name”>
```

მაგრამ, ვინაიდან იგი არ არის მხარდაჭერილი ყველა ვებ-დამთვალიერებლის მიერ, პრაქტიკაში **<INPUT>** ტეგში, როგორც წესი, იყენებენ ორივე ატრიბუტს-**NAME**-საც და **ID**-საც.
სხვადასხვა მართვის ელემენტების შესაქმნელად გამოსაყენებელ ყველა **HTML** ტეგებს, ჩვენ შემდგომში, ამავე თავში განვიხილავთ. ახლა კი მოდით გავარკვიოთ, თუ რა გზით ხდება მეორე პრობლემის გადაწყვეტა.

მონაცემების კოდირება.

მანამ, სანამ მონაცემები გაეგზავნება ვებ-სერვერს და მისი გავლით, სერვერულ პროგრამას, ისინი განსაკუთრებული წესით უნდა იყვნენ კოდირებული. მონაცემების კოდირება ხორციელდება ფორმის საშუალებით, ანუ უფრო ზუსტად, ვებ-დამთვალიერებლის მიერ, **<FORM>** ტეგის განსაკუთრებული ატრიბუტების საფუძველზე. მოდით, გავარკვიოთ, სახელდობრ, თუ როგორ ხდება მონაცემების კოდირება და რომელი ატრიბუტები არიან მასზე ”პასუხისმგებელნი”.

მაშ ასე, შეიძლება ითქვას, რომ მომხმარებლის მიერ შეტანილი მონაცემების კოდირება ხდება ფორმის საშუალებით. ფორმას მართვის ელემენტებიდან ამოაქვს შეტანილი მონაცემები და **<INPUT>** ტეგის, **NAME** და **ID** ატრიბუტებით განსაზღვრული სახელები, და აერთიანებს მათ

<მართვის ელემენტის დასახელება> = <მნიშვნელობა>

სახის წყვილებად. მაგალითად, ჩვენს ფორმაში შეტანილი მონაცემების საფუძველზე, ვებ-დამთვალიერებელი შექმნის ასეთ ანაკრებს:

Name = admin
Password = superuser
Submit = 1

ასე, რომ სერვერულ პროგრამას შეეძლება გაიგოს, თუ რომელი მონაცემები, რომელ მართვის ილემენტებში იყო შეტანილი.
რაც შეეხება ღილაკს **გაგზავნა-Submit-Отправить**. მისი დაჭერის შემთხვევაში, ის ყოველთვის იღებს ერთის ტოლ მნიშვნელობას. ეს მნიშვნელობა პრაქტიკაში არასდროს არ გამოიყენება-სერვერული პროგრამისათვის ისედაც ცხადია, რომ დაჭერილი იყო ღილაკი **გაგზავნა-Submit-Отправить**. წინააღმდეგ შემთხვევაში ეს პროგრამა არ მიიღებდა მონაცემებს ფორმისაგან.
აღნიშნულის შემდეგ ფორმა ახორციელებს სიმბოლოების გარდაქნას. ასე მაგალითად, გამოტოვების ნიშანი იცვლება +სიმბოლოთი. ყველა სიმბოლო, რომლებიც არ წარმოადგენენ ციფრებს, ლათინურ ასოებს და ხაზგასმის ნიშნებს, გარდაიქმნებიან შემდეგი სახის მიმდევრობებად-**%NN**, სადაც **NN** წარმოადგენს სიმბოლოს თექვსმეტობით კოდს.
ამის შემდეგ, უკვე ფორმა იწყებს მონაცემების კოდირებას წინასწარ განსაზღვრული წესების ერთობლივობის -კოდირების მეთოდის შესაბამისად. კოდირების მეთოდი მოიცემა **<FORM>** ტეგის განსაკუთრებული ატრიბუტის **ENCTYPE**-ის საშუალებით.

<FORM ACTION = “givemedata.php”
ENCTYPE = “application/x-www-form-urlencoded”.
. . .

</FORM>

გაჩუმებით (თუ ატრიბუტი ENCTYPE არ არის მითითებული) მონაცემების კოდირება სრულდება

application/x-www-form-urlencoded

მეთოდის საშუალებით. ეს მეთოდი გამოიყენება უმეტეს შემთხვევებში. ვებ-დამთვალიერებლების უმრავლესობის მიერ მხარდაჭერილია ასევე კოდირების მეთოდები **multipart/form-data** და **text/plain**, მაგრამ მათი გამოყენება შედარებით იშვიათად ხდება.

შენიშვნა! *კოდირების მეთოდი* multipart/form-data *გამოიყენება იმ შემთხვევაში, თუ სერვერული პროგრამისათვის ფაილების გაგზავნაა საჭირო; იგი ახორციელებს ასეთი შემთხვევის შესაბამისი ორობითი მონაცემების გარდაქმნას. ხოლო მეთოდი* text/plain, *მონაცემებს წარმოადგენს ჩვეულებრივი ტექსტის სახით, რაც შეიძლება სასარგებლო იყოს, თუ ფორმის მონაცემების გაგზავნა მოხდება ელექტრონული ფოსტის საშუალებით (ხანდახან მონაცემების გადაცემის ასეთ ხერხსაც იყენებენ).*

უნდა აღინიშნოს, რომ მონაცემების კოდირება ზემოთ აღნიშნული მეთოდებიდან ერთ-ერთის საშუალებით გამოიყენება არა ყოველთვის. ჩვენ ამის შესახებ უფრო დაწვრილებით ვისაუბრებთ მაშინ, როდესაც განვიხილავთ მე-3 პრობლემის გადაწყვეტასთან დაკავშირებულ საკითხს.

მონაცემების გადაცემა

ქსელში მონაცემების გადაცემისას ფორმას (ცხადია არა თვით ფორმას, არამედ მის დამმუშავებელ ვებ-დამთვალიერებელს) შეუძლია გამოიყენოს მონაცემთა გადაცემის ორი მეთოდიდან ერთ-ერთი. ერთი მათგანი ჩვენთვის უკვე ნაცნობი **GET** მეთოდია, ხოლო მეორე ჯერჯერობით უცნობი **POST** მეთოდი. განვიხილოთ ისინი რიგრიგობით.

GET მეთოდი ჩვენ შევისწავლეთ მე-8 თავში, როდესაც ვიხილავდით ერთი ვებ-გვერდიდან მეორეში მონაცემების გადაცემის ხერხს. ამ მეთოდით გადასაცემი მონაცემები წარმოიდგინებიან

<მართვის ელემენტის დასახელება>=<მნიშვნელობა>

წყვილების ანაკრების სახით, რომელიც ისმება ინტერნეტ-მისამართის ბოლოს და მისგან კითხვის ნიშნით გამოიყოფა. თვით წყვილები კი ერთმანეთისაგან გამოიყოფიან "კომერციული და" (**&**) ნიშნით.
მოდით ავიღოთ ვებ-დამთვალიერებლის მიერ, ჩვენს ფორმაში შეტანილი მონაცემების საფუძველზე ფორმირებული მონაცემების ანაკრები:

name = admin
password = superuser
submit = 1

და გავუგზავნოთ იგი სერვერულ პროგრამას **GET** მეთოდით:

givemedata.php?name = admin&password = superuser&submit = 1

ამ შემთხვევაში მონაცემების არანაირი კოდირება არ გამოიყენება. ეს ნიშნავს იმას, რომ **GET** მეთოდით მონაცემების გადაცემის შემთხვევაში, შესაძლებელია **<FORM>** ტეგის **ENCTYPE** ატრიბუტის გამოტოვება-ვინაიდან იგი მაინც არ იქნება დამუშავებული.
მონაცემების წარმოდგენის სიმარტივე და თვალსაჩინოება-**GET** მეთოდის ერთად-ერთი უპირატესობაა. აღნიშნულის გამო, მნიშვნელოვნად მარტივდება სერვერული პროგრამების გამართვა: რადგანაც ვებ-სერვერისათვის გადასაცემი მისამართი, ყოველთვის მის, მისამართების სტრიქონში იქნება გამოსახული, ჩვენ ყოველთვის შეგვეძლება დავინახოთ, თუ რა იყო კონკრეტულად გადაცემული. (გასაგებია, რომ კონფიგურაციასთან დაკავშირებული მონაცემების გადაცემა ამ მეთოდით არ შეიძლება-მათ ყველა დაინახავს).
GET მეთოდს ორი ნაკლოვანება გააჩნია და ორივე ძალიან სერიოზული. პირველი: მისი საშუალებით არ შეიძლება მონაცემების დიდი მოცულობების გადაცემა. ეს გამოწვეულია ინტერნეტ-მისამართების სიგრძეზე, შესაბამისი სტანდარტებით დაწესებული შესღუდვებით (არაუმეტეს 256 სიმბოლო). თუ ამას გამოვაკლებთ თვით სერვერული პროგრამის ინტერნეტ-მისამართის სიგრძეს-მივიღებთ ჩვენი მონაცემებისათვის მაქსიმალურად დასაშვებ ზომას.
GET მეთოდის მეორე ნაკლოვანება-მისი უპირატესობის მეორე

მხარეა. მის მიერ გადაცემული მონაცემები ღიაა ყველას დასანახად და მარტივად არის შესაძლებელი მათი წაკითხვა ვებ-დამთვალიერებლის მისამართების სტრიქონში.

GET მეთოდი გამოიყენება, იმ შემთხვევაში, თუ სერვერული პროგრამისათვის გადასაცემი მონაცემები წინასწარი შეფასებით მცირე მოცულობისაა და საიდუმლოებას არ წარმოადგენენ. ძირითადში ეს ვებ-საიტის შიდა მოხმარების მონაცემებია: კატეგორიების ნომრები, სხვადასხვა კოდები, გასაღებები და სხვა. თუ ჩვენ გვჭირდება მოცულობითი ან კონფიდენციალური მონაცემების გადაცემა, უნდა გამოვიყენოთ გადაცემის მეორე მეთოდი-**POST**.

POST მეთოდი, სერვერულ პროგრამას, მონაცემებს გადასცემს, უკვე არა ინტერნეტ-მისამართის ნაწილის სახით, არამედ ეგრეთ წოდებული კლიენტური მოთხოვნის დამატებითი მონაცემების სახით. ვინაიდან ასეთი დამატებითი მონაცემების ზომა არ არის შეზღუდული, ჩვენ შეგვიძლია გადავცეთ ნებისმიერი მონაცემები განუსაზღვრელი რაოდენობით. **POST** მეთოდით ვებ-სერვერისათვის ფაილის გადაცემაც კი არის შესაძლებელი.

მონაცემების **POST** მეთოდით გადაცემის შემთხვევაში, ხორციელდება მონაცემების კოდირება და ყოველთვის მუშავდება **<FORM>** ტეგის **ENCTYPE** ატრიბუტი, თუ რათქმა უნდა ის წარმოდგენილია **HTML** კოდში. (თუ კი ატრიბუტი **ENCTYPE** გამოტოვებულია, მაშინ გამოიყენება მონაცემების გაჩუმებითი კოდირების მეთოდი:

application/x-www-form-urlencoded.

მეთოდ **POST**-ს ორი ღირსება გააჩნია: გადასაცემი მონაცემების მოცულობაზე შეზღუდვის არ არსებობა და მათი ”უხილავობა”. ნაკლოვანებები: მონაცემების დეკოდირების სირთულე და გამართვის სიძნელე. მაგრამ, ვინაიდან მონაცემების დეკოდირებას ჩვენს ნაცვლად **PHP** დამმუშავებელი ახორციელებას, ხოლო გამართვის სირთულეების, გამოცდილ პროგრამისტებს არ ეშინიათ, შესაძლებელია ამ ნაკლოვანებების უგულვებელყოფა.

POST მეთოდის გამოყენებით, ხდება საიტის მონაცემთა ბაზაში შესატანი მოცულობითი ან საიდუმლო მონაცემების (სახელები და პაროლები) გადაცემა. ამიტომ ჩვენს ვებ-გვერდებზე, რომლებიც

ახორციელებენ მონაცემების შეტანას და შესწორებას, ჩვენ სწორედ ამ **POST** მეთოდს გამოვიყენებთ.

შენიშვნა! *ამბობენ, რომ კომიტეტი W3C აპირებს დროთა განმავლობაში აიძულოს ვებ-დიზაინერები და ვებ-პროგრამისტები საერთოდ უარი თქვან GET მეთოდზე და გამოიყენონ მხოლოდ POST მეთოდი. ჯერ-ჯერობით GET მეთოდი გამოცხადებულია არა რეკომენდირებულად ახლად შესაქმნელ საიტებში. თუმცა ყველა ბოლო დროს გამოშვებული ვებ-დამთვალიერებული, ახორციელეს ამ მეთოდის მხარდაჭერას, ამიტომ ის ჯერ კიდევ საქმეშია.*

მაშ ასე, თეორია დამთავრდა და როგორც ყოველთვის, დადგა პრაქტიკის დრო. დროა ვიბრძოლოთ **DreamweaverMX**-ის დასაცავად.

ადმინისტრაციული და მომხმარებლების ვებ-გვერდები

მოდით, ჯერ დროებით შევეშვათ მონაცემების შეტანას და ჩასწორებას და სხვა რამეზე ვიფიქროთ. რაზე? უსაფრთხოებაზე და შეღწევის უფლებების განაწილებაზე.

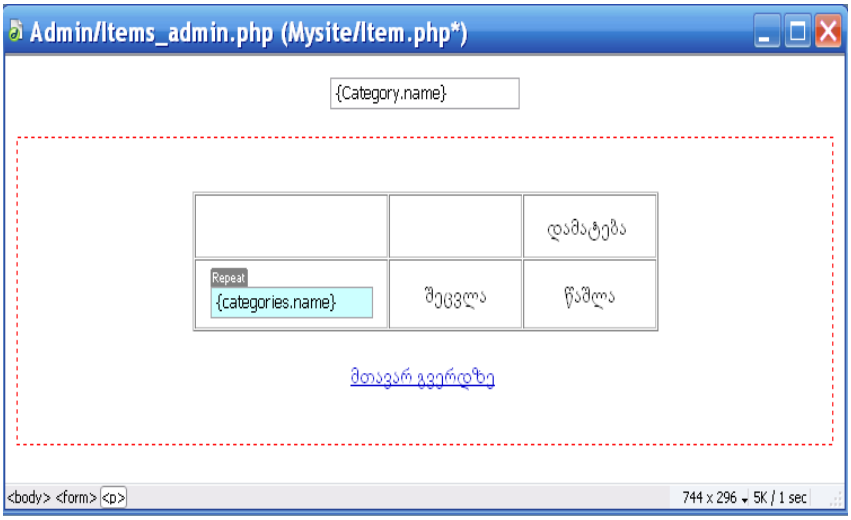
ჩვენი მონაცემთა ბაზის უსაფრთხოება ჩვენ უზრუნველვყავით ჯერ კიდევ მე-6 თავში, მაშინ როდესაც ის შევქმენით. მომხმარებელს **site**, რომლის სახელითაც ჩვენი სერვერული გვერდები უერთდებიან მონაცემთა ბაზას, გააჩნია **categories** და **items** ცხრილებში, მონაცემების წაკითხვის და ჩაწერის მიზნით, შეღწევის უფლება, მაგრამ ცხრილების შექმნის უფლება მას არ გააჩნია. ეს თითქოს საკმარისი უნდა იყოს ჩვენი ბაზის დასაცავად. ბოლოს და ბოლოს, ჩვენ კიდევ გვყავს მომხმარებელი root, რომელსაც ყველაფრის უფლება აქვს (გააჩნია სერვერის ადმინისტრატორის უფლებები).
მაგრამ საქმე იმაშია, რომ ჩვენი საიტის მომხმარებელთა უმეტესობას არ უნდა ჰქონდეს მონაცემთა ბაზაში ჩანაწერის განხორციელების უფლება. საიტის ჩვეულებრივ დამთვალიერებლებს, რომლებიც უბრალოდ ათვალიერებენ სტატიებისა და ფაილების კატეგორიების შემცველ გვერდებს, არ უნდა გააჩნდეთ მათში რაიმეს ჩაწერის შესაძლებლობა. ჩანაწერის განხორციელების უფლებას უნდა ფლობდეს მხოლოდ საიტის ადმინისტრატორი, ანუ ჩვენ, ვინაიდან სწორედ ჩვენ მოგვიწევს სიების შევსება და მათში არსებული შეცდომების შესწორება (საიტის ადმინისტრირება).

მაშ, როგორ გავმიჯნოთ ჩვეულებრივი "უუფლებო" მომხმარებლების (როგორც პროფესიონალი პროგრამისტები ამბობენ, **სტუმრების**) და საიტის ადმინისტრატორების უფლებამოსილებები? ძალიან მარტივად. უნდა შევქმნათ სხვადასხვა დანიშნულების ვებ-გვერდების ორი კომპლექტი, ერთი-ეგრეთ წოდებული **ადამინისტრატიული ვებ-გვერდები** და მეორე-ეგრეთ წოდებული **მომხმარებელთა ვებ-გვერდები.**

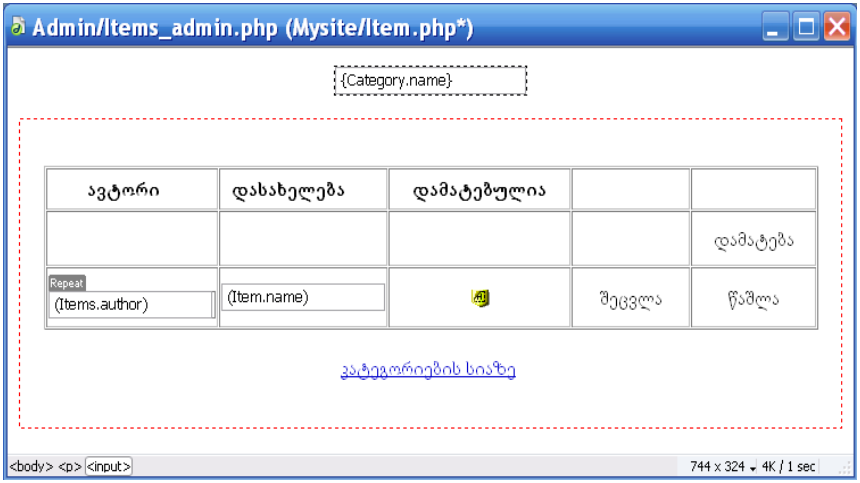
მომხმარებელთა ვებ-გვერდებმა უნდა უზრუნველყონ სტუმრების მიერ, მხოლოდ მონაცემების, მაქსიმალურად კომფორტულად დათვალიერება. ასეთი გვერდები ჩვენ უკვე შევქმენით მე-8 თავში- **Categories.php** და **Items.php**. რაც შეეხება ადმინისტრატიულ ვებ-გვერდებს, მათ უნდა უზრუნველყონ მონაცეების დამატებისა და ჩასწორების შესაძლებლობები, ისეთივე მაქსიმალური კომფორტით. სწორედ ამ გვერდების შექმნა იქნება ჩვენი უახლოესი ამოცანა.

მოდით, ჩვენი საიტის ძირეულ საქაღალდეში შევქმნათ საქაღალდე **Admin** (თუ როგორ უნდა გავაკეთოთ ეს, აღწერილია მე-4 თავში). ამ საქაღალდეში მოვათავსოთ ყველა ადმინისტრატიული გვერდები, რათა გავმიჯნოთ ისინი მომხმარებლების ვებ-გვერდებისაგან.

კატეგოეიების სიასთან სამუშაოდ, **Admin** საქაღალდეში შევქმნათ ვებ-გვერდი **Categories_admin.php**. ეს გვერდი, ჩვენს მიერ უკვე შექმნილი გვერდის **Categories.php**-ს ანალოგიური იქნება, იმ განსხვავებით, რომ მასში შემავალ ცხრილს ექნება სამი სვეტი. პირველი სვეტი-ისევე, როგორც **Categories.php** გვერდზე, იქნება კატეგორიის დასახელება, ხოლო დანარჩენ ორში მოთავსებული იქნება ჰიპერმინიშნებები **შეცვლა- Change-Изменить** და **წაშლა-delete-Удалить**, რომლებიც შესაბამისად მოცემული ჩანაწერის ჩასწორებისა და მისი წაშლის ვებ-გვერდებზე მიუთითებენ. ასევე ჩვენ დაგვჭირდება ამ ცხრილში კიდევ ერთი, განმეორებად არეში არ შემავალი სტრიქონის ჩამატება; ამ სტრიქონის ბოლო უჯრედში, მოთავსებული იქნება ჰიპერმინიშნება **დამატება-add-Добавить**, რომელიც მიანიშნებს ახალი ჩანაწერის დასამატებელ გვერდზე. **Categories_admin.php** გვერდის მიახლოებითი სახე ნაჩვენებია ნახ. 9.2.



ნახ. 9.2. Categories_admin.php ვებ-გვერდის სახე Dreamweaver-ის დოკუმენტების ფანჯარაში.



ნახ.9.3. Items_admin.php ვებ-გვერდის სახე Dreamweaver-ის დოკუმენტების ფანჯარაში

რაც შეეხება არჩეული კატეგორიის სტატიებთან (ფაილებთან) მუშაობას, ჩვენ ამ მიზნით იმავე **Admin** საქაღალდეში შევქმნით ვებ-გვერდს **Items_admin.php**. ეს გვერდი არაფრით არ იქნება

განსხვავებული ჩვენს მიერ უკვე შექმნილი **Items.php** გვერდისაგან, თუ არ ჩავთვლით რამოდენიმე გამონაკლისს. პირველი-ჩანაწერების ანაკრებში **Items**, ვამატებთ კიდევ ერთ ველს-id. მეორე-ვშლით სვეტს **გახსნა-Open-Открыть**, ხოლო მის ნაცვლად ვამატებთ სვეტებს **შეცვლა-Change -Изменить** და **წაშლა-Delete-Удалить**, ისევე, როგორც ეს გავაკეთეთ **Categories_admin.php** ვებ-გვერდის შემთხვევაში. მესამე-ვამატებთ ცხრილში კიდევ ერთ, განმეორებად არეში არ შემავალ სტრიქონს და მის ბოლო უჯრედში ვათავსებთ ჰიპერმინიშნება **დამატება-Add-Добавить**. მეოთხე-ვშლით ორივე არასავალდებულო არეს; ამასთან მეორეს, რომელიც ეკრანზე გამოიტანება ჩანაწერების ცარიელი ანაკრების შემთხვევაში-ვშლით შიგთავსთან ერთად. მეხუთე-შეგვიძლია წავშალოთ ანაკრებში ჩანაწერების რაოდენობის შესახებ ინფორმაციის შემცველი სტრიქონიც (თუმცა შეგვილია არც წავშალოთ). ყოველივე აღნიშნულის შედეგად, უნდა მივიღოთ ის, რაც ნაჩვენებია ნახ. 9.3-ზე.

ყურადღება მივაქციოთ იმ გარემოებას, რომ **Items_admin.php** ვებ-გვერდი არ შეიცავს სვეტს **ინტერნეტ-მისამართი**. საქმე იმაშია, რომ ინტერნეტ-მისამართი შეიძლება იყოს ძალიან გრძელი, და მაშინ ჩვენი ცხრილი ვეღარ ჩაეტევა ვებ-დამთვალიერებლის ფანჯარაში. ასე, რომ ჯობია, ის არ იყოს გამოტანილი **Items_admin.php** გვერდზე. თუკი ჩვენ დაგვჭირდება მისი დანახვა, ჩვენ ყოველთვის შეგვეძლება გადავიდეთ ჩანაწერების ჩასწორების გვერდზე, სადაც გამოტანილი იქნება მისი ყველა ველის მნიშვნელობები.

ცხადია, რომ ჩვენს მიერ შექმნილი ადმინისტრატიული ვებ-გვერდები უნდა შეიცავდნენ ერთმანეთზე, და არა მომხმარებლებისათვის განკუთვნილ მათ "კოლეგებზე" მიმთითებელ ჰიპერმინიშნებებს. ზოგადად ადმინისტრაციული გვერდები არ უნდა მიანიშნებდნენ მომხმარებლების ვებ-გვერდებზე და პირიქითაც.

კატეგორიებისა და **სტატიების** (ფაილების) სიების ადმინისტრატიული გვერდების შექმნის შემდეგ, შეიძლება გადავიდეთ მათი დამატების, ჩასწორების და წაშლის ვებ-გვერდების შექმნაზე. ახლა ჩვენ, სწორედაც ამით დავკავდებით.

მონაცემების შეტანისა და ჩასწორების მარტივი სერვერული ვებ-გვერდების შექმნა.

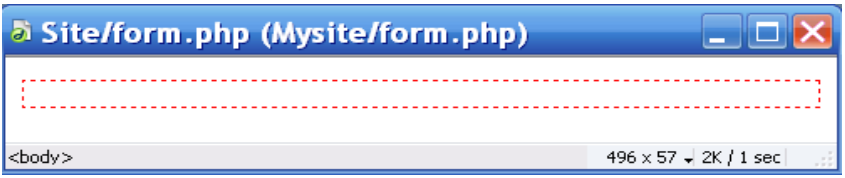
დასაწყისში, ჩვენ შევქმნით **კატეგორიების** დამატების, ჩასწორების და წაშლისათვის განკუთვნილი ადმინისტრატიული ვებ-გვერდების ანაკრებს-ეს უფრო მარტივია. **სტატიების** და **ფაილების** დამატების, ჩასწორების და წაშლის ვებ-გვერდებით ჩვენ უფრო მოგვიანებით დავკავდებით.

მარტივი ვებ-გვერდი ჩანაწერის დამატებისთვის

დავიწყოთ ახალი კატეგორიის დამატებისთვის განკუთვნილი ვებ-გვერდით. მას დავარქვათ **Category_add.php**. მისი საშუალებით ჩვენ შეგვეძლება ახალი კატეგორიისათვის სახელწოდების მინიჭება, აგრეთვე იმის განსაზღვრა, თუ რას მიეკუთვნება იგი **სტატიებს** თუ **ფაილებს**.

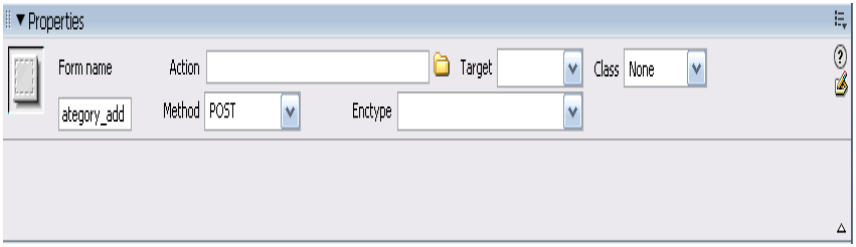
მაშ ასე, შევქმნათ **PHP**-ს ახალი სერვერული ვებ-გვერდი და შევინახოთ იგი **Admin** საქაღალდეში **Category_add.php** სახელწოდებით. მივცეთ მას დასახელება **კატეგორიების დამატება**, შევიტანოთ რაიმე განმარტებითი ტექსტი ასეთივე სათაურით. ამის შემდეგ შევქმნათ ჩვენი პირველი ფორმა.

ფორმის შესაქმნელად, ავირჩიოთ მენიუ **Insert**-ის **Form**-ქვემენიუს, პუნქტი **Form**. ჩვენს მიერ შექმნილ, ჯერ კიდევ ცარიელ ფორმას ექნება ნახ.9.4.-ზე ნაჩვენები სახე.



ნახ.9.4. ცარიელი ფორმა **Category_add.php** ვებ-გვერდზე.

დავსვათ ამ ფორმაზე ტექსტური კურსორი და შევხედოთ თვისებების რედაქტორს. ის უნდა გამოიყურებოდეს ისე, როგორც ნახ 9.5.-ზეა ნაჩვენები.

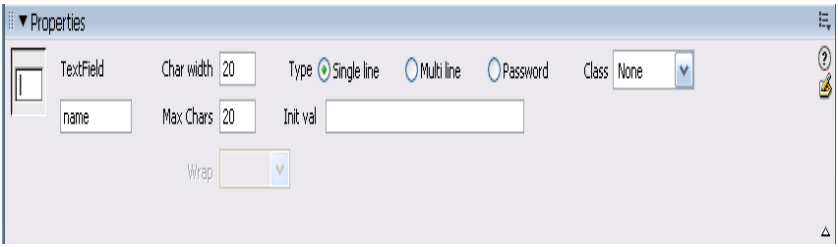


ნახ.9.5. თვისებების რედაქტორის სახე არჩეული ფორმის შემთხვევაში.

აქ ჩვენ გვაინტერესებს მხოლოდ შესატანი ველი **Form Name**, რომელშიც განისაზღვრება ვებ-გვერდის ფარგლებში, ფორმისათვის უნიკალური სახელი. მივანიჭოთ ჩვენს ფორმას "შინაარსის შესაბამისი" სახელი **category_add**-ასე უფრო გაგვიადვილდება საკუთარ ვებ-გვერდებში გარკვევა. შევიტანოთ აღნიშნული სახელი და დავაჭიროთ კლავიშას **<ENTER>**, რათა **DreamweaverMX**-მა შესაბამისი ცვლილებები შეიტანოს **HTML**-კოდში.

შენიშვნა! *ნახ. 9.5-ზე ნაჩვენები თვისებების რედაქტორის დანარჩენი მართვის ელემენტები განსაზღვრავენ <FORM> ტეგის შესაბამისი ატრიბუტების მნიშვნელობებს. ჩვენ მათ არ შევიტანთ-ამას თვითონ Dreamweaver-ი გააკეთებს, როდესაც შექმნის ახალი ჩანაწერის დასამატებელ სერვერული ქცევას.*

ახლა ისევ დავაყენოთ ტექსტური კურსორი ფორმაში, შევიტანოთ ტექსტი **დასახელება,** და დავაჭიროთ "ჰარი" კლავიშას. ტექსტური კურსორის ნიშნის შემდეგ შევქმნათ ახალი კატეგორიის სახელწოდების შეტანისათვის განკუთვნილი ველი.
შესატანი ველი იქმნება **Insert** მენიუს, **Form** ქვემენიუს, **Text Fild** პუნქტის არჩევით. შექმნილი შესატანი ველი მაშინათვე გაჩნდება იმ ადგილას სადაც ტექსტური კურსორი დგას, ანუ უშუალოდ წარწერის "**დასახელება"** შემდეგ: (+ჰარი). დავაწკაპუნოთ მასზე, რათა მოვნიშნოთ იგი და შევხედოთ თვისებათა რედაქტორს (ნახ. 9.6).



ნახ.9.6. თვისებების რედაქტორის სახე მონიშნული შესატანი ველის შემთხვევაში.

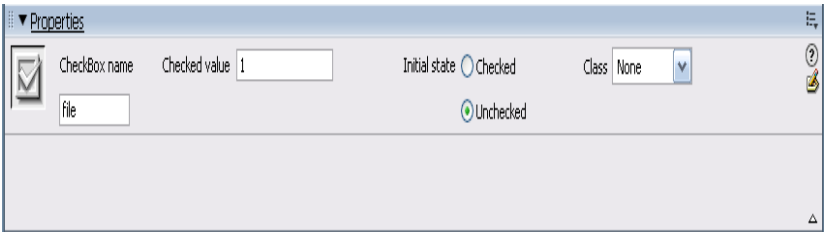
თვისებების რედაქტორში შევიტანოთ ახლად შექმნილი შესატანი ველის შემდეგი პარამეტრები:
-სახელი (შესატანი ველი **Text Fild**)-**name**;
-სიგანე სიმბოლოებში (შესატანი ველი **Char width**)-20(**categories** ცხრილის **name** ველის ზომა არის 20 სიმბოლო);
-სიმბოლოების მაქსიმალური რაოდენობა, რომელთა შეტანაც არის შესაძლებელი ამ შესატან ველში (შესატანი ველი **Max Chars**)-20;
-შესატანი ველის ტიპი (გადამრთველების ანაკრები **Type**)-ჩვეულებრივი ერთსტრიქონიანი შესატანი ველი (**Single line**);
-გაჩუმებითი მნიშვნელობა (შესატანი ველი Init val)-არაფელი.

შენიშვნა! *ერთსტრიქონიანი შესატანი ველების გარადა, HTML საშუალებებით შესაძლებელია რედაქტირების არეების შექმნა (თვისებების რედაქტორში Type ჯგუფის Multi line გადამრთველი) და პაროლების შესატანი განსაკუთრებული ველები (გადამრთველი Password).*

ამ მონაცემების შეტანის შემდეგ, ტექსტური კურსორი დავსვათ უშუალოდ ახლად შექმნილი შესატანი ველის შემდეგ და დავაჭიროთ კლავიშას **<ENTER>**, რათა შევქმნათ მეორე აბზაცი. ამ აბზაცში ავკრიბოთ ტექსტი **ფაილი,** და დავაჭიროთ გამოტოვების ღილაკს. ახლა კი ცოტათი ვიფიქროთ.

Categories ცხრილის **file** ველი ლოგიკური ტიპისაა. ეს ნიშნავს იმას, რომ მას შეუძლია **true** ან **false** ”ჩართულია” ან ”გამორთულია” მნიშვნელობების მიღება. რომელ ელემენტს მივანიჭოთ უპირატესობა ამისათვის?

რათქმა უნდა, შესაძლებელია "**კი**" და "**არა**" პუნქტების შემცველი, ჩვეულებრივი ან ჩამოსაშლელი სიის შექმნა, ან მსგავსი ორი გადამრთველიდან ანაკრების შექმნა. მაგრამ სია და გადამრთველების ანაკრები ვებ-გვერდზე დიდ ადგილს დაიკავებს, ხოლო ჩამოსაშლელი სია არც თუ ისე მოსახერხებელია-ყველა პუნქტის დასანახავად მისი გახსნა იქნება საჭირო. ასე, რომ ჩვენი არჩევანი იქნება ალამი-მისი საშუალებით უფრო თვალსაჩინოდ იქნება შესაძლებელი ლოგიკური მნიშვნელობის განსაზღვრა.
ალმის შექმნა ხდება **Insert** მენიუს, იმავე **Form** ქვემენიუს, **Checkbox** პუნქტის არჩევით. როდესაც ის გამოჩნდება **ფაილი** (+ჰარი) წარწერის შემდეგ, მოსანიშნად დავაწკაპუნოთ მასზე მაუსით და მივმართოთ თვისებათა რედაქტორს (ნახ. 0.7.).

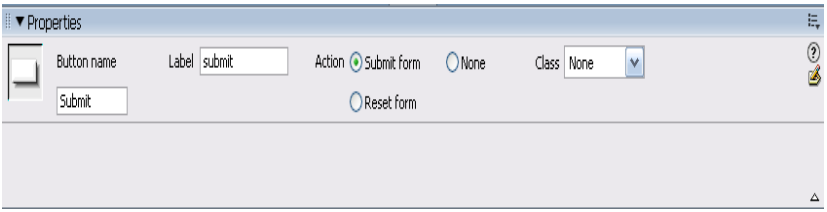


ნახ.9.7. თვისებების რედაქტორის სახე მონიშნული ალმის შემთხვევაში.

თვისებების რედაქტორში ჩვენ დაგვჭირდება ჩვენი ალმისათვის შემდეგი პარამეტრების შეტანა:
-სახელი (შესატანი ველი **CheckBox name**)-**file**;
-ჩართული მდგომარეობის შესაბამისი მნიშვნელობა (შესატანი ველი **Checked value**)-1 (პრინციპში შესაძლებელია ნებისმიერი მნიშვნელობის შეტანა);
-საწყისი მდგომარეობა (გადამრთველების ანაკრები **Initial state**)-გამორთული (გადამრთველი **Unchecked**; გადამრთველი **Checked** განსაზღვრავს ჩართულ მდგომარეობას).
თუ ჩვენი ალამი ჩართული იქნება, ფორმა ასეთ წყვილს შექმნის:

File=1.

თუკი ალამი არ იქნება ჩართული, მაშინ ფორმა საერთოდ უგულვებელყოფს მას (თითქოს ის არც არსებობს) და მის საფუძველზე არანაირ მონაცემებს არ შექმნის.
გამოტანის დამთავრებისთანავე ტექსტური კურსოეი დავაყენოთ უშუალოდ ჩვენი ალმის შემდეგ და დავაჭიროთ კალავიშს **<ENTER>**, რათა შევქმნათ კიდევ ერთი, მესამე აბზაცი. ამ აბზაცში ჩვენ შევქმნით მართვის უკანასკნელ ელემენტს, რომლის გარეშეც არც ერთი ფორმა არ არსებობს-ღილაკს **Submit**.
ღილაკის შესაქმნელად, ავირჩიოთ მენიუ **Insert**-ის, **Form**-ქვემენიუს პუნქტი **Button**. როდესაც ღილაკი გაჩნდება ფორმაზე, გამოვყოთ იგი მაუსის დაწკაპუნებით და დახმარებისათვის მივმართოთ თვისებების რედაქტორს (ნახ.9.8), რათა შევიტანოთ აღნიშნული ღილაკის პარამეტრები.



ნახ. 9.8. თვისებების რედაქტორის სახე მონიშნული ღილაკის შემთხვევაში.

ჩვენი ღილაკის პარამეტრები ასეთი იქნება:
-სახელი (შესატანი ველი **Button name**)-**submit**;
-წარწერა ღილაკზე (შესატანი ველი Label)-დამატება-**add(Добавить);**
-ღილაკის მიერ შესასრულებელი მოქმედება (გადამრთველების ერთობლივობა Action)-მონაცემების გაგზავნა (გადამრთველი **Submit form**).

შენიშვნა! *ფორმაში ჩანაწერის გაუქმების Cancel(Сброс) ღილაკის შესაქმნელად საჭიროა გადამრთველების Action ჯგუფში, Reset form ღილაკის ჩართვა (ნახ.9.8.). ასევე არსებობს ისეთი ღილაკის შექმნის შესაძლებლობა, რომელიც არ შეასრულებს HTML-ით გათვალისწინებულ არანაირ ქმედებას; ამისათვის უნდა ჩავრთოთ გადამრთველი None.*

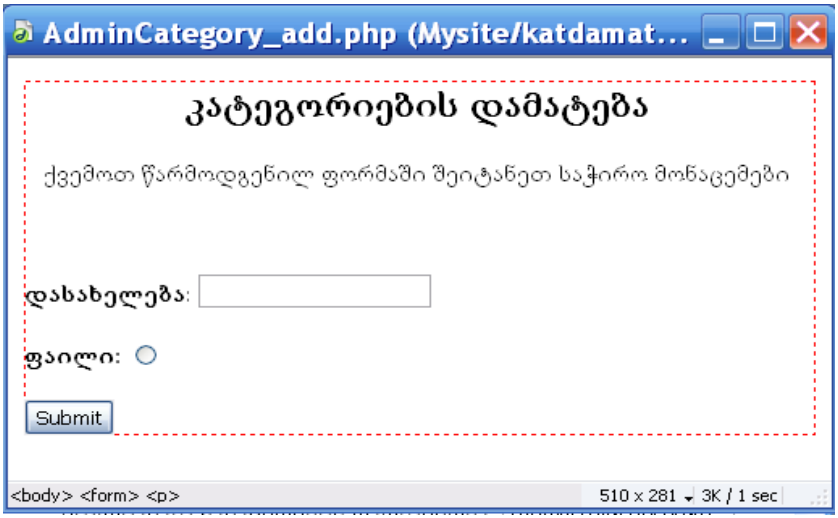
ამით ჩვენი ფორმის შექმნა დასრულებულია-იხ. ნახ.9.9. ახლა უკვე შეიძლება ვებ-გვერდის შენახვა და ჩანაწერის დამატების განმახორციელებელი სერვერული ქცევის შექმნაზე გადასვლა.

სერვერული ქცევის შესაქმნელად ჩვენ დაგვჭირდება შეუცვლელი **Server Behaviors** პანელი. გამოვიტანოთ იგი ეკრანზე, დავაჭიროთ ”**plus(плюс)**” ნიშნაკიან ღილაკს და ეკრანზე გამოსულ მენიუში ავირჩიოთ პუნქტი **Insert Record**. ეკრანზე გაჩნდება დიალოგური ფანჯარა **Insert Record** (ნახ.9.10.).

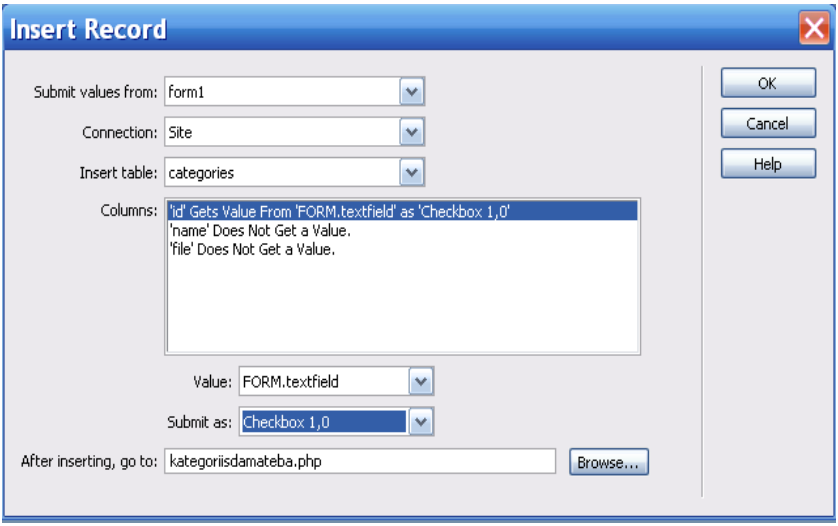
ჩამოსაშლელ სიაში **Submit valus from** ვირჩევთ ფორმის სახელს, რომლიდანაც ახალი ჩანაწერისათვის უნდა მოხდეს მონაცემების არჩევა. ჩვენს შემთხვევაში ეს არის ფორმა **category_add**.

ჩამოსაშლელი სია **Connection** უკვე ნაცნობია ჩვენთვის სხვა ფანჯრებიდან. იგი განსაზღვრავს მონაცემთა ბაზასთან კავშირს; ჩვენს შემთხვევაში ეს იქნება-**Site**.

ცხრილი, რომელშიც უნდა მოხდეს ჩანაწერის დამატება, განისაზღვრება ჩამოსაშლელი სიის **Insert table** საშუალებით. ავირჩიოთ მასში ცხრილი **categories.**



ნახ.9.9. მზა ფორმის შემცველი ვებ-გვერდი Category_add.php

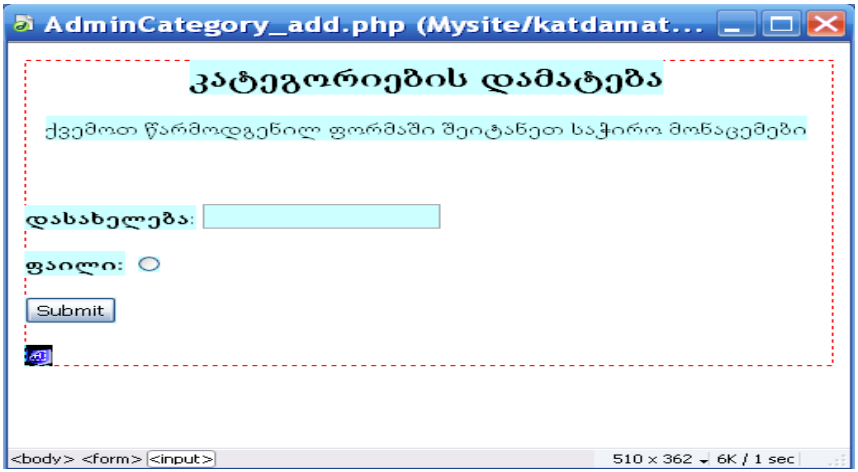


ნახ.9.10. დიალოგური ფანჯარა Insert Record

დიდ სიაში **Columns** ჩამოთვლილია არჩეული ცხრილის ყველა ველი და ფორმაზე მათი შესაბამისი მართვის ელემენტები. ჩვენ შეგვიძლია ნებისმიერი ველის არჩევა და არჩეული ველისათვი მართვის ელემენტის და მონაცემების ტიპის განსაზღვრა, შესაბამისად **Value** და **Submit us** ჩამოსაშლელი სიების საშუალებით. მოდით **Columns** სიიდან, რიგრიგობით ავირჩიოთ categories ცხრილის ველები და მათთვის განვსაზღვროთ პარამეტრები. მაშ ასე, id ველის მნიშვნელობა არ განისაზღვრება ფორმიდან (ის თვით სერვერის მიერ განისაზღვრება, ვინაიდან მთვლელის ტიპისაა), ამიტომ ჩვენ, ჩამოსაშლელ სიაში **Value**, ავირჩევთ ტიპს **None**. **name** ველის მნიშვნელობა აღებული იქნება ფორმის **name** შესატანი ველიდან (სიაში **Value** ავირჩიოთ პუნქტი **FORM.name**), ხოლო მისი ტიპი იქნება ტექსტური (**Submit as** სიის პუნქტი **Text)**. ველი **file** მიიღებს მნიშვნელობას ლოგიკური ტიპის ალმისგან **file** (**Value** სიის პუნქტი **FORM.file**).

მოდით აქ ცოტახნით შევჩერდეთ. **Submit as** სიაში წარმოდგენილია სამი ალამი, რომლებიც განსაზღვრავენ მონაცემების ლოგიკურ ტიპებს: **Checkbox Y,N, Checkbox 1,0** და **Checkbox-1,0**. მათგან რომელი ავირჩიოთ? მოდით ვიფიქროთ. **MySQL** ლოგიკურ ცვლადებს ინახავს ერთი ციფრის "სიგრძის" მქონე ველებში, ხოლო **PHP** ყველა

არანულოვან რიცხვებს აღიქვამს როგორც **true**-ს და ნულს-როგორც **false**-ს. ესეიგი, უნდა ავირჩიოთ **Submit as** პუნქტის **Checkbox 1,0** პუნქტი.
რაც შეეხება შესატან ველს **After inserting**, **go to**, იგი განსაზღვრავს ვებ-გვერდს რომელზეც, გადასვლა მოხდება, თუკი შექმნილი იქნება ახალი ჩანაწერი. ამ ველში შევიტანოთ ჩვენი კატეგორიების სიის-**Categories_admin.php** ადმინისტრატიული ვებ-გვერდის სახელი. ასევე შეიძლება დავაწკაპუნოთ მარჯვენა მხარეს მდებარე ღილაკზე Browse და **DreamweaverMX**-ის გასახსნელ დიალოგურ ფანჯარაში ავირჩიოთ საჭირო ფაილი.
ყველა საჭირო მონაცემების შეტანის შემდეგ, შეიძლება ღილაკზე **OK** დაჭერა. ნახ.9.11. ნაჩვენებია გამზადებული ფორმა, რომელზეც მიბმულია **Insert record** სერვერული ქცევა.



ნახ.9.11. **Category_add.php** ვებ გვერდი გამზადებული ფორმით, რომელზეც მიბმულია სერვერული ქცევა **Insert record**.

ფორმის ქვედა ნაწილში კარგად ჩანს ერთი უცნაური ნიშანი  . ეს ეგრეთ წოდებული ფარული ველია, რომელიც ფორმაში ისეთი მონაცემების შესანახად გამოიყენება, რომელთა ჩვენება მომხმარებლებისათვის არ არის საჭირო. **DreamweaverMX**-მა თვითონ შექმნა ეს ველი საკუთარი საჭიროებებისათვის.
დაგვრჩა მხოლოდ **Categories_admin.php** ვებ-გვერდის გახსნა და ცხრილში მოთავსებული ტექსტის-**დამატება**, გარდაქმნა,

Category_add.php ვებ-გვერდზე მიმნიშნებელ ჰიპერმინიშნებად. ამის შემდეგ შეიძლება ჩვენს მიერ შექმნილი ვებ-გვერდების შემოწმება მოქმედებაში.
ვინაიდან ჩვენ, **default.htm** ვებ-გვერდზე, ჯერ კიდევ არ შეგვიქმნია ადმინისტრატიულ ვებ-გვერდებზე მიმთითებელი ჰიპერმინიშნებები, კატეგორიების სიაში მოსახვედრად დაგვჭირდება, ვებ-დამთვალიერებლის მისამართების სტრიქონში, **სტილებისათვის**

http://localhost:8080/Admin/Categories_admin.php?file=0,

ხოლო ფაილებისათვგის

http://localhost:8080/Admin/Categories_admin.php?file=1

მისამართის აკრეფა. შეიძლება ვსინჯოთ რამოდენიმე ახალი კატეგორიის ჩამატება-სულ ერთია ჩვენ მათ მერე მაინც წავშლით, როგორც კი შევქმნით ვებ-გვერდს **Category_delete.php**-ს.
ჩვენს ვებ-გვერდს ჩანაწერის დასამატებლად მხოლოდ ორად-ორი ნაკლოვანება გააჩნია. პირველი-ჩანაწერის დამატებისას ყოველთვის ხდება დაბრუნება სტატიების კატეგორიების სიაში. მეორე-არ არსებობს ჰიპერმინიშნება კატეგორიების სიაში დასაბრუნებლად. ეს კი ვებ-დიზაინში ცუდ ტონად ითვლება. მაგრამ ჩვენ ორივე ამ ნაკლოვანებას აღმოვფხვრით მოგვიანებით.

ჩანაწერების დასამატებლად გამოსაყენებელი PHP სცენარების გარჩევა.

დადგა **DreamweaverMX**-ის მიერ შექმნილი **PHP** კოდის დათვალიერების დრო. ის საკმაოდ მოცულობითია, ასე, რომ ჩვენ მას განვიხილავთ ნაწილ-ნაწილ. გადავერთოთ **HTML** კოდის წარმოჩენის რეჟიმში და დავიწყოთ.
თავიდან მოდით მოვძებნოთ **HTML** კოდის წინ მდებარე, Insert record სერვერული ქცევის შესაბამის დიდ სცენარში ასეთი **PHP** კოდის ფრაგმენტი:

```
$editFormAction=$_SERVER['PHP_SELF'];
If (isset($_server['QUERY_STRING'])) {
$editFormAction.="?" . htmlentities($_SERVER['QUERING']) ;
```

}

ჯერ ცვლადს **$editFormAction** ენიჭება მოცემულ მომენტში დამუშავებაში მყოფი, ანუ Category_add.php სერვერული ვებ-გვერდის ფაილის სახელი. ეს სახელი ამოიღება ჩაშენებული **$_SERVER** მასივის **PHP_SELF** ინდექსიანი ელემენტიდან.
შემდეგ მოწმდება, იყო თუ არა ამ გვერდისათვის **GET** მეთოდით რაიმე არგუმენტები გადაცემული (ის უნდა მდებარეობდეს იმავე **$_SERVER** ჩაშენებული მასივის **QUERY_STRING** ინდექსიან ელემენტში). თუ კი გადაცემული იყო (მასივის მოცემული ელემენტი არსებობს), იგი ემატებიან ვებ-გვერდის სახელს, რომელიც **$editFormAction** ცვლადში ინახება.
რაც შეეხება ფუნქციას **htmlentities**, რომელიც ახორციელებს **GET** მეთოდით გადასაცემი მონაცემების კოდირებას (კერძოდ, სიმბოლოების გარდაქმნას), მისი გამოძახება ფაქტიურად არც არის საჭირო, ვინაიდან **$_SERVER** მასივის **PHP_SELF** ინდექსიანი ელემენტი შეიცავს ჯერ-ჯერობით არადეკოდირებულ მონაცემებს. თუ რატომ არის ის მაინც წარმოდგენილი **PHP** კოდში-გაუგებარია; შესაძლოა ეს **DreamweaverMX**-ის შეცდომა იყოს.
გამოვტოვოთ დანარჩენი **PHP** და **HTML** კოდები და გადავიდეთ ფორმის შემქმნელ **HTML** კოდზე. ის ასე გამოიყურება:

```
<FORM ACTION = ”<?php echo $editFormAction ; ?> “METOD =POST”
NAME = “category_add” ID = “category_add”>
<P>დასახელება:
<INPUT NAME = “name” TYPE = “text” ID=”name” SIZE=”20”
MAXLENGTH=”20>
</P>
<P>ფაილი:
<INPUT NAME = “file” TYPE = “checkbox” ID = “file” VALUE=”1”
</P>
<P>
<INPUT NAME = “submit” TYPE =“submit” ID=”submit”
VALUE=”დამატება”>
</P>
<INPUT TYPE = “hidden” NAME= “MM_insert” VALUE=”category_add>
</FORM>
```

აქ ჩვენთვის პრინციპში ყველაფერი ნაცნობია. ჩანს, რომ **DreamweaverMX**-მა თვითონ განსაზღვრა მონაცემების გადაცემის მეთოდი **(POST)** და სერვერული გვერდი, რომელიც ამ მონაცემებს მიიღებს **($editFormAction**(ცვლადის მნიშვნელობა). ასევე ჩვენ ვხედავთ უხილავი ველის შემქმნელ კოდს:

<INPUT TYPE="hidden"NAME="MM_insert" VALUE="category_add".

<INPUT> ტეგის **TYPE** ატრიბუტი, უხილავი ველის შემთხვევაში, უნდა შეიცავდეს მნიშვნელობას-**hidden**. ასევე ჩვენ ვხედავთ, რომ სახელის (**NAME** ატრიბუტის მნიშვნელობა) მნიშვნელობად განსაზღვრულია **MM_insert**, ხოლო უხილავ ველში მოთავსებულ მონაცემებად (**VALUE** ატრიბუტის მნიშვნელობა)-**category_add**. ეს ყველაფერი შექმნა **DreamweaverMX**-მა თავსი საჭიროებებისთვის, ხოლო თუ რა საჭიროებებისთვის, ამას ჩვენ მალე გავიგებთ.
დავუბრუნდეთ ტეგს **<FORM>**, რომელიც ქმნის ჩვენს ფორმას და დავაკვირდეთ **ACTION** ატრიბუტის მნიშვნელობას. თუ ცვლადი შეიცავს მოცემული ვებ-გვერდის სახელს, მაშინ გამოდის, რომ მონაცემები გადაეცემა ამავე-**Category_add.php** ვებ-გვერდს და რომ მან უნდა დაამუშაოს აღნიშნული მონაცემები.
ეს **DreamweaverMX**-ის მიდგომაა-მონაცემებს ამუშავებს იგივე ვებ-გვერდი, რომელშიც ხდება მათი შეტანა. სერვერული ქცევა **Insert record** შეიცავს სცენარს, რომელიც ამოწმებს მისთვის **GET** მეთოდით გადაცემულ მონაცემებში **category_add** მნიშვნელობის (უხილავი ველის შიგთავსი) მქონე **MM_insert** არგუმენტის არსებობას. თუკი არგუმენტი **MM_insert** არ არსებობს, მაშინ გამოიტანება მონაცემების შესატანი ფორმა, ხოლო თუ არსებობს, ეს იმას ნიშნავს, რომ მონაცემები უკვე შტანილია და საჭიროა მათი დამუშავება.
DreamweaverMX-ის მიერ შექმნილი ვებ-გვერდი **Category_add.php** ასე მუშაობს:

1. პირველი გახსნისას, ვინაიდან ვებ-გვერდისთვის ჯერ არანაირი მონაცემები არ ყოფილა გადაცემული **POST** მეთოდით, გამოიტანება ფორმა.
2. ფორმაში მონაცემების შეტანის შემდეგ, მომხმარებელი აჭერს ღილაკს **დამატება-Add-Добавить** და მონაცემები **POST** მეთოდით გადაეცემა ამმავე ვებ-გვერდს.

3. გვერდი იხსნება მეორე ჯერ. **PHP** სცენარი ხელახლა ამოწმებს იყო, თუ არა ვებ-გვერდისათვის **POST** მეთოდით გადაცემული მონაცემები, და თუ კი ისინი გადაცემული იყო, ამუშავებს მათ. ამ შემთხვევაში ფორმა კი არ გამოიტანება, არამედ ხდება გადასვლა კატეგორიების სიის **Categories_admin.php** გვერდზე.

ახლა დაწვრილებით, სტრიქონ-სტრიქონ განვიხილოთ მონაცემების დამუშავების სცენარი:

If ((isset($_POST[“MM_insert”]))&&
(S_POST[“MM_insert”] == “category_add”)) {

აქ მოწმდება, იყო თუ არა მოცემული გვერდისათვის **POST** მეთოდით გადაცემული category_add მნიშვნელობის მქონე **MM_insert** არგუმენტი-**DreamweaverMX**-ის მიერ ჩვენს ფორმაზე შექმნილი უხილავი ველის შიგთავსი. თუკი ის გადაცემული იყო, მაშინ მომდევნო ბლოკი მთლიანად სრულდება.

$insertSQL = sprintf(“INSERT INTO categories (name, file) VALUES (%s,%s)”, GetSQLValueString($_POST[‘name’],”text”),
GETSQLValueString(isset($_POST[‘file’]) ? “true” : “ ”,
“defined” , “1” , “0”)) ;

ცვლადი **$insertSQL** იღებს მნიშვნელობას-ჩანაწერის დამატების SQL მოთხოვნის კოდს. მისი შექმნისათვის ისევ გამოიყენება ჩაშენებული ფუნქცია **sprintf**, რომელიც ასრულებს სტრიქონზე წარმოდგენილი შაბლონების შეცვლას რეალური მნიშვნელობებით.
ასევე ამ გამოსახულებაში გამოიყენება ჩვენთვის უცნობი ფუნქცია **GetSQLValueString**. ეს არ არის ჩაშენებული ფუნქცია-ის თვით **DreamweaverMX**-მა გამოაცხადა სერვერული ქცევის **Insert record**-ის შესაბამისი სცენარის დასაწყისშივე.
მისი დანიშნულება მდგომარეობს იმაში, რომ გარდაქმნას მისთვის პირველი არგუმენტით გადაცემული მნიშვნელობა ისე, რომ იგი

აღქმული იქნას MySQL-ის მიერ და დააბრუნოს ეს მნიშვნელობა, როგორც შედეგი. ამასთან ის ხელმძღვანელობს მეორე არგუმენტის მნიშვნელობით, რომლითაც განისაზღვრება გარდასაქმნელი მნიშვნელობის მონაცემების ტიპი. ასე მაგალითად, თუ მეორე არგუმენტის მიერ გადაცემულია სტრიქონია **text**, მაშინ პირველი არგუმენტი უნდა იქნას დამუშავებული, როგორც სტრიქონული ტიპის მნიშვნელობა (ჩასმულია ერთეულოვან ბრჭყალებში).

თუკი ფუნქცია **GetSQLValueeString**-ის მეორე არგუმენტის მიერ გადაცემულია სტრიქონი **defined**, ეს ფუნქცია იქცევა შემდეგნაირად. თუ პირველი არგუმენტი შეიცავს არაცარიელ სტრიქონს, იგი დააბრუნებს მესამე არასავალდებულო არგუმენტის მიერ გადაცემულ მნიშვნელობას. თუ ამ ფუნქციას პირველი არგუმენტით გადაეცემა ცარიელი სტრიქონი, იგი დააბრუნებს არასავალდებულო მეოთხე არგუმენტს.

მოდით შევხედოთ ასეთ გამოსახულებას:

```
GetSQLValueString(isset($_POST["file"]) ? "true" : " " ,
"defined" , "1" , "0")) ;
```

აქ **GetSQLValueString** ფუნქციის პირველ არგუმენტად გამოყენებულია გამოსახულება, რომელიც ამოწმებს გადაცემული იყო, თუ არა ფორმისათვის **file** არგუმენტი (ანუ იყო, თუ არა ჩართული შესაბამისი ალამი). თუ ის იყო გადაცემული, მაშინ **GetSQLValueString** ფუნქციის პირველი არგუმენტი იქნება სტრიქონი true და ეს ფუნქცია დააბრუნებს 1-ს. თუკი არგუმენტი **file** არ ყოფილა გადაცემული ფორმისათვის, მაშინ **GetSQLValueString** ფუნქციის პირველი არგუმენტი გახდება ცარიელი სტრიქონი (" ") და ფუნქცია დააბრუნებს 0-ს. ამ ფუნქციის მიერ დაბრუნებული მნიშვნელობა ჩასმული იქნება ჩანაწერის დამატების **SQL** მოთხოვნაში.

```
mysql_select_db($databaase_Site, $Site) ;
$Result1 = mysql_query($insertSQL,$Site) or die (mysql_error ( )] ;
```

ეს ორი, ჩვენთვის უკვე ნაცნობი გამოსახულებები ირჩევენ მონაცემთა ბაზას **site** და ასრულებენ ადრე შექმნილ და **$insertSQL** ცვლადში მოთავსებულ **SQL** მოთხოვნას. მოთხოვნის შესრულების შედეგი თავსდება **$Result1**ცვლადში და მეტი სხვაგან არსად არ გამოიყენება. (ამ შემთხვევაში გასაგებია თუ რატომ არის საჭირო მისი სადმე შენახვა).

$insertGoTo = "Categories_admin.php" '

ეს გამოსახულება კი **$insertGoTo** ცვლადში ათავსებს იმ ვებ-გვერდის სახელს, რომელზეც გადასვლა მოხდება ჩანაწერის დამატების შემდეგ. ეს არის კატეგორიების სიის ადმინისტრაციული გვერდი.

If (isset($SERVER['QUERY_STRING'])) {

ეს პირობითი გამოსახულება კი **DreamweaverMX**-მა შექმნა, როგორც იტყვიან ყოველი შემთხვევისათვის. იგი ამოწმებს, იყო თუ არა **Category_add.php**-გვერდისათვის გადაცემული რაიმე მონაცემები **GET** მეთოდით. თუ კი გადაცემული იყო, მაშინ შესრულდება ორი ქვემოთ მოყვანილი გამოსახულება.

$insertGoTo = (strpos($insertGoTo, '?')) ? "&" : "?" ;

ეს გამოსახულება ამოწმებს, წარმოდგენილია თუ არა **$insertGoTo** ცვლადის მნიშვნელობაში კითხვითი ნიშნის სიმბოლო **(?)**. თუკი ის წარმოდგენილია, მაშინ ამ ცვლადის მნიშვნელობას დაემატება &, წინააღმდეგ შემთხვევაში-?. ეს საჭიროა იმისათვის, რომ ვებ-გვერდს, რომლის სახელიც მოცემულია **$insertGoTo** ცვლადში, გადაეცეს **Category_add.php** გვერდისათვის **GET** მეთოდით გადაცემული ყველა მონაცემი.
(ჩაშენებული ფუნქცია **strops** აბრუნებს იმ სიმბოლოს პოზიციის ნომერს, რომელიც გადაცემული იყო მეორე არგუმენტის მიერ, პირველი არგუმენტით გადაცემულ სტრიქონში. თუ ეს სიმბოლო სტრიქონში არ იქნება წარმოდგენილი, ბრუნდება 0).

$insertGoTo = $_SERVER['QUERY_STRING'] ; }

ეს გამოსახულება კი, ცვლადის მნიშვნელობას **$insertGoTo**-ს ამატებს **Category_add.php** გვერდისათვის **GET** მეთოდით გადაცემულ კოდირებულ მონაცემებს. ანუ **Category_add.php** გვერდი, ახორციელებს მისთვის გადაცემული მონაცემების გადამისამართებას **category_admin.php** გვერდისათვის.

header (sprintf("Location: %s" , $insertGoTo)) ; }

უკანასკნელი გამოსახულება, ახორციელებს გადასვლას გვერდზე, რომლის სახელიც (ყველა **GET** მეთოდით გადაცემულ არგუმენტებთან ერთად) მდებარეობს ცვლადში **$insertGoTo**. ამისათვის გამოიყენება შემდეგი სახის სინტაქსისი:

header ("Location : <ვებ-გვერდის ფაილის დასახელება") ;

როგორც კი **PHP** დამმუშავებელი წააწყდება ფუნქციას **header**-ს, იგი ვებ-სერვერის გამოყენებით, უგზავნის ვებ-დამთვალიერებელს სხვა გვერდზე გადასვლის ბრძანების შემცველ განსაკუთრებულ სერვერულ პასუხს. როდესაც ვებ-დამთვალიერებელი მიიღებს მას, ის ვებ-სერვერს გაუგზავნის მოთხოვნას ვებ-გვერდზე, რომლის სახელიც მითითებული იყო ბრძანებაში (ჩვენს შემთხვევაში-ეს არის ვებ-გვერდი **Categories_admin.php**).

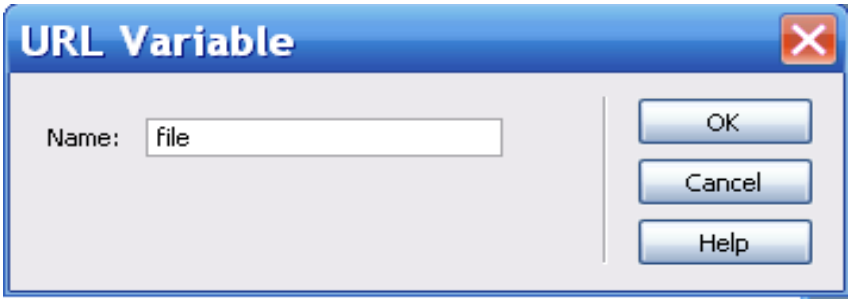
მართვის ელემენტებისათვის გაჩუმებითი მნიშვნელობების განსაზღვრა.

დიახ, ჩვენი ვებ-გვერდი **Category_add.php** მუშაობს, მაგრამ არც თუ ისე გამართულად, როგორც ეს საჭიროა. ჩანაწერის დამატების შემდეგ ის ყოველთვის უბრუნდება სტატიების კატეგორიების სიას, მიუხედავად იმისა რომ არ შეიცავს კატეგოეიების სიაზე დამაბრუნებელ ჰიპერმინიშნებას. გარდა ამისა, არ იქნებოდა ცუდი თუ **file** ალამის თავდაპირველი ჩართვა ან გამორთვა მოხდებოდა იმისდა მიხედვით, თუ რომელი კატეგორიის სიიდან, სტატიების თუ ფაილების, მოხდა ამ გვერდზე გადასვლა.
ამ სიტუაციიდან გამოსავალი ნათელია-**Category_add.php** ვებ-გვერდს **GET** მეთოდით უნდა გადაეცეს არგუმენტი **file**. ამისათვის, გავხსნათ ვებ-გვერდი **Categories_admin.php**, გადავერთოთ **HTML** კოდის წარმოჩენის რეჟიმში და მოვძებნოთ ჰიპერმინიშნების-**Добавить-Add-**

დამატება, შემქმნელი კოდი. მივანიჭოთ ამ ჰიპერმინიშნებას ასეთი ინტერნეტ-მისამართი (**<A>** ტეგის, **HREF** ატრიბუტის მნიშვნელობა):

Category_add.php?file=<?php echo $colname_Categories ; ?>

(ცვლადი **$colname_Categories** ცხადდება სცენარში, რომელიც შეესაბამება სერვერულ ქცევას **Recordset** და შეიცავს **Categories_admin.php** ვებ-გვერდისათვის **GET** მეთოდით გადაცემულ არგუმენტს-**file**.
შევინახოთ შესწორებული **Categories_admin.php** ვებ-გვერდი და დავხუროთ იგი. გადავერთოთ **Category_add.php** ვებ-გვერდზე.
თავიდან ჩვენ დაგვჭირდება **PHP** კოდის დამატება, რომელიც **file** არგუმენტიდან ამოიღებს მნიშვნელობას და მოათავსებს მას რომელიმე ცვლადში. მისი დაწერა ხელითაც შეიძლება, მაგრამ რადგანაც ჩვენ **DreamweaverMX**-ში ვმუშაობთ, მოდით, მან გააკეთოს ეს ჩვენს მაგივრად.
დავაჭიროთ კლავიშვების კომბინაციას <Ctrl>+<F10>, რათა ეკრანზე გამოვიტანოთ პანელი **Bindings**. ამ პანელზე დავაჭიროთ "**plus(плюс)**" ნიშნაკიან ღილაკს და ეკრანზე გამოსულ მენიუში ავირჩიოთ პუნქტი **URL Variable**. ეკრანზე გამოჩნდება მცირე ზომის დიალოგური ფანჯარა **URL Variable**, რომელიც ნაჩვენებია ნახ.9.12.-ზე.



ნახ.9.12. დიალოგური ფანჯარა URL Variable

ამ ფანჯრის ერთად-ერთ შესატან ველში **Name**, შევიტანოთ ჩვენი არგუმენტის სახელი-**file**. დავაჭიროთ ღილაკს OK. ამის შემდეგ

Bindings პანელის სიაში გაჩნდება ახალი "**ხე**" **URL** "ფესვით" და ერთად-ერთი "შტო"-თი - **file**.

ჩვენ ახლახანს განვახორციელეთ ჩვენი **file** არგუმენტის რეგისტრაცია **DreamweaverMX**-ში. ფაქტიურად ჩვენ ვაცნობეთ **DreamweaverMX**-ს, რომ ვაპირებთ ამ არგუმენტის გამოყენებას ჩვენს სცენარებში.

არგუმენტის დარეგისტრირების შემდეგ, მაუსის დაწკაპუნებით მოვნიშნოთ ალამი **file**. თვისებათა რედაქტორში (იხ. ნახ. 9.7.) დავაჭიროთ ღილაკს **Dynamic**. ეკრანზე გაჩნდება დიალოგური ფანჯარა **Dynamic CheckBox** (ნახ. 9.13.).

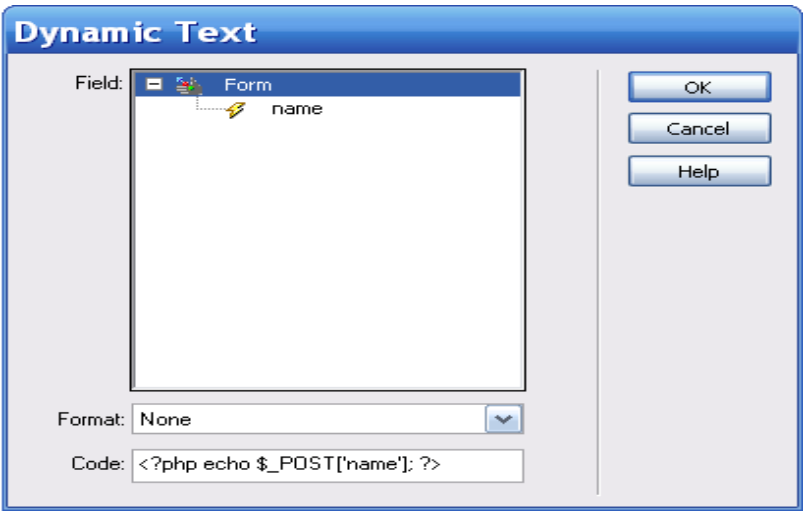
ნახ. 9.13. დიალოგური ფანჯარა Dynamic CheckBox.

ჩამოსაშლელ სიაში **CheckBox,** ავირჩიოთ ალამი, რომლისთვისაც გაჩუმებითი მნიშვნელობის მინიჭება გვსურს. გაჩუმებით იქ არჩეული იქნება ალამი-**file**, ასე რომ, თვითონ ჩვენ ამის გაკეთება არ მოგვიწევს.

შესატან ველში **Check if** მოიცემა პარამეტრი, რომლის საფუძველზეც განისაზღვრება ალმის გაჩუმებითი მნიშვნელობა. ჩვენს შემთხვევაში-ეს არის **GET** მეთოდით გადაცემული არგუმენტი **file**. ჩვენ შეგვიძლია პირდაპირ ამ ველში **PHP** კოდის ასეთი ფრაგმენტის შეტანა:

<?php $_GET ['file'] ?>

გარდა ამისა შეიძლება კიდევ "ელვა' ნიშნაკიანი ღილაკის დაჭერა, რომელიც **Check if** შესატანი ველის მარჯვნივ მდებარეობს. ამის შემდგ ეკრანზე გამოჩნდება დიალოგური ფანჯარა **Dynamic Data** (ნახ. 9.14.).



ნახ. 9.14. დიალოგური ფანჯარა Dynamic Data

ამ ფანჯრის იერარქიულ სიაში **Fild**, ავირჩიოთ თვით ის პარამეტრი, რომლის საფუძველზეც განისაზღვრება გაჩუმებითი მნიშვნელობა. ამ სიაში ჩვენ დაგვჭირდება **URL** "ფუძიანი" "**bg**"-ს გაშლა და ჩვენი **file** არგუმენტის შესაბამისი **file** "შტოს" არჩევა. შესატან ველში Code გაჩნდება **PHP** კოდს ფრაგმენტი, რომელიც ამოიღებს აღნიშნული არგუმენტის მნიშვნელობას.

რაც შეეხება ჩამოსაშლელ სიას **Format**, ის განსაზღვრავს **Fild** სიაში არჩეული პარამეტრის მნიშვნელობის დამატებით ფორმატირებას. ჩვეულებრივ, არავითარი დამატებითი ფორმატირება არ არის საჭირო, ამიტომ თავისუფლად შეგვიძლია ამ სიაში **None** პუნქტის არჩევა.

საჭირო პარამეტრის არჩევის შემდეგ, დავაჭიროთ **Dynamic Data** დიალოგური ფანჯრის ღილაკს **OK** და ისევ მოვხვდებით **Dynamic CheckBox** ფანჯარაში.
უკანასკნელი, რაც ჩვენ დაგვრჩა-ეს არის შესატან ველში **Equal to** იმ მნიშვნელობის შეტანა, რომელთანაც შედარდება **Check if** შესატან ველში მოცემული პარამეტრის მნიშვნელობა. თუ ისინი ერთმანეთის ტოლები აღმოჩნდებიან, ალამი ჩაირთვება, წინააღმდეგ შემთხვევაში-გამოირთვება. ჩვენ იქ უნდა შევიტანოთ მნიშვნელობა **1-true** მნიშვნელობის სტრიქონის სახით წარმოდგენილი რიცხვითი ექვივალენტი, რადგანაც **$_GET** მასივის ელემენტების მნიშვნელობები სწორედ სტრიქონული სახით არიან წარმოდგენილნი.

ახლა უკვე შეგვიძლია **OK** ღილაკზე დაჭერა, რათა დავხუროთ ფანჯარა **Dynamic CheckBox** და ვაიძულოთ **DreamweaverMX**-ი შექმნას საჭირო სცენარი. მორჩა, ალმისთვის-**file,** გაჩუმებითი მნიშვნელობა განსაზღვრულია.
მოდით, აქვე შევქმნათ კატეგორიების სიის ვებ-გვერდზე (ცხადია, ადმინისტრატიულ ვებ-გვერდზე) მიმთითებელი ჰიპერმინიშნება. ტექსტური კურსორი დავაყენოთ ფორმის მარჯვენა მხარეს და დავაჭიროთ კლავიშას **<ENTER>**, რათა შევქმნათ ცარიელი აბზაცი. ამ ცარიელ აბზაცში ავკროფოთ ტექსტი **კატეგორიების სიაზე** და გადავაქციოთ იგი შემდეგი ინტერნეტ-მისამართის მქონე ჰიპერმინიშნებად:

Categories_admi.php? file=<?php echo $_GET[“file”] ; ?>

ახლა უკვე შეიძლება გამზადებული ვებ-გვერდის შენახვა და მისი შემოწმება მოქმედებაში. როდესაც დავტკბებით მისი ხილვით, მოდით გადავერთოთ **HTML** კოდის წარმოჩენის რეჟიმში და შევხედოთ, თუ როგორი **PHP** კოდი შექმნა **DreamweaverMX**-მა, რათა **file** ალმისათვის, მიენიჭებინა გაჩუმებითი მნიშვნელობა. კოდი ძალიან მცირე ზომისაა (ის გამოყოფილია მუქი შრიფტით):

<INPUT <?php if (! (strcmp($_GET['file'], "1"))) {echo "checked" ;} ?>
NAME = "file" TYPE = "checkbox" ID = "file" VALUE="1">

ჩაშენებული ფუნქცია **strcmp** ახორციელებს, მისთვის არგუმენტების სახით გადაცემული სტრიქონების, საკმაოდ რთულ შედარებებს. ამ ფუნქციის სრული აღწერა მოცემულია **PHP**-ს სახელმძღვანელოში; ჩვენ კი მხოლოდ იმის ცოდნა გვჭირდება, რომ იგი აბრუნებს **0**-ს თუ სტრიქონები ერთმანეთის ტოლია, ხოლო წინააღმდეგ შემთხვევაში, ნებისმიერ არანულოვან მნიშვნელობას. თუკი გამოვიყენებთ ლოგიკური უარყოფის ოპერატორს "**არა**"(**NOT**), სტრიქონების ტოლობის შემთხვევაში მივიღებთ მნიშვნელობას **true**, ხოლო წინააღმდეგ შემთხვევაში მნიშვნელობას -**false**.
თუ სტრიქონები ტოლები არიან (პირობითმა გამოსახულებამ დააბრუნა მნიშვნელობა **true**), მაშინ ალმის შემქმნელ ტეგში **<INPUT>**, ჩაისმება ატრიბუტი **CHECKED**, რომელიც ჩართავს ალამს. ეს არის ეგრეთ წოდებული **ატრიბუტი მნიშვნელობის გარეშე**; მას არ გააჩნია მნიშვნელობა, ტეგში მხოლოდ მისი არსებობაც საკმარისია.
მორჩა, **DreamweaverMX**-ს სხვა არავითარი **PHP** კოდი არ დაუმატებია ვებ-გვერდზე (ჩვენს მიერ, ჰიპერმინიშნების ინტერნეტ მისამართში, ხელით ჩამატებული კოდი არ ითვლება). მაშ ასე, ამით დავასრულოთ მუშაობა **Category_add.php** ვებ-გვერდზე, და გადავიდეთ არჩეული კატეგორიის მონაცემებში, ცვლილებების განმახორციელებელი ვებ-გვერდის **Category_edit.php** შექმნაზე.

ჩანაწერების ჩასწორების განმახორციელებელი მარტივი ვებ-გვერდები.

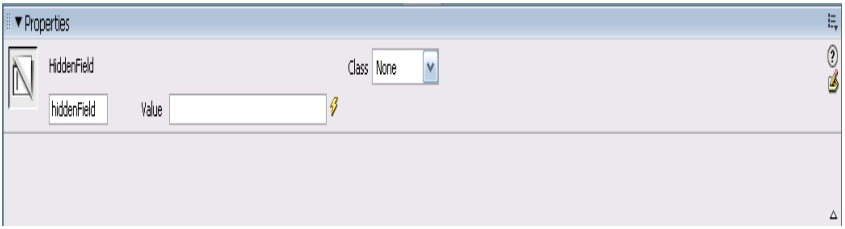
შევქმნათ ახალი სერვერული **PHP** ვებ-გვერდი და **Category_edit.php** სახელით შევინახოთ იგი **Admin** საქაღალდეში. მივცეთ მას დასახელება "**კატეგორიის შეცვლა**", სათაურადაც დავტოვოთ იგივე და დაურთოთ რაიმე განმარტებითი ტექსტი. შევქმნათ ზუსტად ისეთივე ფორმა, როგორიც გვქონდა **Category_add.php** ვებ-გვერდზე,

მხოლოდ იმ განსხვავებით, რომ მისი სახელი იქნება **category_edit**. ღილაკზე **submit** გავაკეთოთ განსხვავებული წარწერა-**Изменить-შეცვლა**.
ახლა კი, შევჩერდეთ და დავფიქრდეთ. ჩანაწერების ჩასწორებისათვის ჩვენ დაგვჭირდება ორი ამოცანის შესრულება:

-ცხრილიდან-**categories**, საჭირო ჩანაწერის ამოღება და ამ ჩანაწერების ველების მნიშვნელობების შეტანა **category_edit** ფორმის მართვის ელემენტებში;
-ღილაკზე-**Изменить-Change-შეცვლა,** დაჭერის შემდეგ, **category_edit** ფორმის მართვის ელემენტებიდან, შესწორებული მნიშვნელობების ამოღება და მათი ჩაწერა **categories** ცხრილის ამავე ჩანაწერის შესაბამის ველებში.

მაგრამ, საქმე იმაშია, რომ **DreamweaverMX**-ში გათვალისწინებული, ჩანაწერის შეცვლის განმახორციელებელი, სერვერული ქცევა (წინსწრებით ავღნიშნოთ, რომ მას **Update Record** ეწოდება), ასრულებს ამ სამუშაოს მხოლოდ მეორე ნაწილს-ცხრილში გადააქვს შესწორებული მნიშვნელობები. ფორმის მართვის ელემენტების შევსება ჩასწორებისათვის არჩეული ჩანაწერის ძველი მნიშვნელობებით, ჩვენ თვითონ მოგვიწევს. ამისათვის კი, საჭიროა ჩასწორებისათვის განკუთვნილი ჩანაწერის შემცველი ანაკრების შექმნა.
საჭირო ჩანაწერის მოძებნა პრობლემას არ წარმოადგენს. საკმარისია ვებ-გვერდისათვის, **GET** მეთოდით, **categories** ცხრილის **id**-ველის მნიშვნელობის შემცველი id არგუმენტის გადაცემა. ეს ველი შეიცავს მთელი ცხრილის ფარგლებში უნიკალურ მნიშვნელობებს. ასე, რომ ჩვენთვის საჭირო ჩანაწერს ჩვენ ყოველგვარი სირთულეების გარეშე მოვძებნით.
გარდა ამისა, ჩვენ მოგვიწევს ამოღებული ჩანაწერის **id** ველის მნიშვნელობების სადმე შენახვა. შემდგომში ის გამოყენებული იქნება სერვერული ქცევის **Update Record**-ის მიერ ჩასასწორებელი ჩანაწერის

მოსაძებნად. ამიტომ, მოდით ჩავსვათ ჩვენს ფორმაში **category_edit**, უხილავი ველი.
ჩავსვათ ტექსტური კურსორი უშუალოდ **file** ალმის შემდეგ-სწორედ აქ იქნება მოთავსებული უხილავი ველი, რომელშიც შენახული იქნება არჩეული ჩანაწერის **id** ველის მნიშვნელობა. აღნიშნული უხილავი ველის შესაქმნელად, ავირჩოთ **Insert** მენიუს, **Form**-ქვემენიუს, **Hidden Field** პუნქტი. როდესაც შეიქმნება უხილავი ველი, მოვნიშნოთ იგი მაუსზე დაწკაპუნებით და შევხედოთ თვისებების რედაქტორს (ნახ. 9.15.).



ნახ. 9.15. თვსებების რედაქტორის სახე მონიშნული უხილავი ველის შემთხვევაში.

ერთად-ერთი პარამეტრი, რომლის განსაზღვრაც ჩვენ აქ დაგვჭირდება-ეს არის უხილავის ველის სახელი. იგი შეიტანება შესატან ველში **HiddenField**.

ეხლა კი შეიძლება შევუდგეთ ჩანაწერების ანაკრების შექმნას. ეკრანზე **Bindings** პანელის გამოსატანად დავაჭიროთ კლავიშების კომბინაციას **<Ctrl>+<F10>**, დავაწკაპუნოთ **"плюс"** ნიშნაკიან ღილაკზე da ეკრანზე გამოსულ მენიუში ავირჩიოთ პუნქტი **Recordset(Query).** დიალოგურ ფანჯარაში **Recordset** განვსაზღვროთ შემდეგი პარამეტრები:

-მონაცემთა ანაკრების სახელი (შესატანი ველი **Name**)-**category**;
-კავშირის სახელწოდება (ჩამოსაშლელი სია **Connection**)-**Site**;

-ცხრილის სახელწოდება (ჩამოსაშლელი სია **Table**)-**categories**;
-ცხრილის ყველა ველი (გადამრთველი **All**, **Columns** ჯგუფში);
-ფილტრაცია **id** ველის მიხედვით (ჩამოსაშლელი სია **Filter**);
-**id** ველის მნიშვნელობა ტოლი უნდა იყოს (**Filter** სიის მარჯვნივ მდებარე ჩამოსაშლელი სიის პუნქტი-=):
-**GET** მეთოდით გადაცემული არგუმენტის მნიშვნელობის (**Filter** სიის გვემოთ მდებარე ჩამოსაშლელი სიის პუნქტი **URL Parameter**);
-რომლის სახელია-**id** (**Filter** სიის მარჯვენა ქვედა მხარეს მდებარე შესატანი ველი);
-სორტირების გარეშე (**Sort** ჩამოსაშლელი სიის პუნქტი **None**).

ჩანაწერების ანაკრების შესაქმნელად დავაჭიროთ ღილაკს **OK**. **Bindings** პანელის იერარქიულ სიაში გაჩნდება ჩვენს მიერ ახლახანს შექმნილი ანაკრების შესაბამისი "ხე".
ჩვენი შემდეგი ამოცანა არის-**category_edit** ფორმის მართვის ელემენტებისათვის, გაჩუმებითი მნიშვნელობების განსაზღვრა. ამ მნიშვნელობების აღება ხდება ჩვენს მიერ ახლად შექმნილი ჩანაწერების ანაკრების-**Category**-ის, შესაბამისი ველებიდან.
დავაწკაპუნოთ შესატან ველზე **name**, რათა მოვნიშნოთ იგი, და თვისებების რედაქტორში დავაჭიროთ ელვის ნიშნაკიან პატარა ღილაკს, რომელიც მდებარეობს **Init val** შესატანი ველის მარჯვნივ (იხ. ნახ. 9.6.). ეკრანზე გაჩნდება ჩვენთვის უკვე ნაცნობი დიალოგური ფანჯარა **Dynamic Data** (იხ. ნახ. 9.14.).
ამ ფანჯრის იერარქიულ სიაში **Fild** ჩამოვშალოთ "ხე" **Recordset (Category)** და ავირჩიოთ **Category** ჩანაწერების ანაკრების, **name** ველის შესაბამისი "შტო"-**name**. შესატან ველში **Code**, გაჩნდება **PHP** კოდი, რომელიც ამოიღებს ამ ველის მნიშვნელობას. ამის შემდეგ დავაჭიროთ ღილაკს **OK.**
შემდეგ ავირჩიოთ ალამი **file** და დავაჭიროთ თვისებათა რედაქტორის ღილაკს **OK**. **Dynamic CheckBox** ფანჯარაში დავაჭიროთ ელვის ნიშნაკიან ღილაკს, რომელიც მდებარეობს შესატანი ველის **Check if** მარჯვენა მხარეს. აღნიშნული მოქმედებების შემდეგ, ეკრანზე გამოსული **Dynamic Data**

დიალოგური ფანჯრის იერარქიულ სიაში **Fild**, ავირჩიოთ **Recordset (Category)** "ხის", **file** "შტო" და დავაჭიროთ ღილაკს **OK.** ისევ აღმოვჩნდებით დიალოგურ ფანჯარაში **Dynamic CheckBox**, სადაც **Equal to** შესატან ველში, უნდა შევიტანოთ მნიშვნელობა 1 და ისევ დავაჭიროთ ღილაკს **OK**.
ბოლოს მოვნიშნოთ უხილავი ველი **id** და დავაჭიროთ იმავე ელვის ნიშნაკიან ღილაკს, რომელიც თვისებების რედაქტორში, **Value**-შესატანი ველის მარჯვნივ მდებარეობს (იხ. ნახ. 9.15.). **Recordset** (**Category**) ფანჯრის იერარქიულ სიაში **Field**, გავშალოთ "ხე" **Recordset** (**Category**) და ავირჩიოთ id "შტო", რის შემდეგაც ვაჭერთ ღილაკს **OK**.

ეხლა კი შეიძლება შევუდგეთ სერვერული ქცევის **Update Record** შექმნას. ეკრანზე გამოვიტანოთ **Server Behaviors** პანელი, დავაჯიროთ მასზე განთავსებულ "**плюс**" ნიშნაკიან ღილაკს და ეკრანზე გამოსულ მენიუში ავირჩიოთ პუნქტი **Update Record**. ეკრანზე გამოჩნდება დიალოგური ფანჯარა **Update Record** (ნახ.9.16).

ნახ. 9.16. დიალოგური ფანჯარა Update Record

ჩამოსაშლელ სიაში **Submit values from** ვირჩევთ იმ ფორმის სახელს, რომლიდანაც მოხდება შესწორებული მონაცემების არჩევა ჩანაწერში შესატანად. ჩვენს შემთხვევაში ეს არის ფორმა **category_edit**.

ჩამოსაშლელი სია **Connection**, როგორც წესი, განსაზღვრავს მონაცემთა ბაზასთან კავშირს; ჩვენს შემთხვევაში-Site.
ცხრილი, რომელშიც შესაცვლელი ჩანაწერი მდებარეობს, განისაზღვრება ჩამოსაშლელი სიის **Update table** საშუალებით. მასში ავირჩიოთ ცხრილი categories.
დიდ სიაში **Columns** ჩამოითვლება არჩეული ცხრილის ყველა ველი და მათი შესაბამისი, ფორმის მართვის ელემენტები. ჩვენ შეგვიძლია შესაბამისად **Value** და **Submit as** ჩამოსაშლელი სიების საშუალებით, მასში ნებისმიერი ველის არჩევა და არჩეული ველისთვის მართვის ელემენტის და მონაცემთა ტიპის განსაზღვრა.
მოდით, სიაში **Columns**, რიგრიგობით ავირჩიოთ **categories** ცხრილის ველები და განვსაზღვროთ მათთვის პარამეტრები. **name** ველის მნიშვნელობა აიღება ფორმის შესატანი ველიდან **name** (სიაში **Value**, ავირჩიოთ პუნქტი **FORM.name**), ხოლო მისი ტიპი იქნება ტექსტური (**Submit as** სიის,პუნქტი **Text**). ველი **file** მნიშვნელობას მიიღებს **file** ალმისაგან, ხოლო მისი ტიპი იქნება რიცხვითი ფორმით გამოსახული ლოგიკური მნიშვნელობა (**Submit as** სიის პუნქტი **Checkbox 1,0**).
რაც შეეხება **id** ველს, მისთვის ჩვენ სიაში Value ავირჩევთ პუნქტს **FORM.id**, ხოლო **Submit as** სიაში, პუნქტს-Integer (მონაცემთა მთელნიშნა ტიპი).
გარდა ამისა, სცენარი, რომელსაც **DreamweaverMX**-ი ქმნის, სწორედ ამ ველის მნიშვნელობის მიხედვით უნდა ახორციელებდეს შესასწორებელი ჩანაწერის მოძებნას. მან ეს რომ გააკეთოს, ჩვენ აუცილებლად უნდა გვქონდეს ჩართული ალამი-**Primary key**.

შესატანი ველი **After updating**, **go to** განსაზღვრავს ვებ-გვერდს, რომელზეც გადასვლა მოხდება ჩანაწერის შეცვლის შემდეგ. ამ ველში შევიტანოთ ჩვენი კატეგორიების სიის ადმინისტრატიული ვებ-გვერდის სახელი-**Categories_admin.php.** შესაძლებელია ასევე მარჯვენა მხარეს განლაგებულ Browse ღილაკზე დაწკაპუნება და **DreamweaverMX**-ის ფაილის გახსნის დიალოგურ ფანჯარაში საჭირო ფაილის არჩევა.

ყველა საჭირო მონაცემების შეტანის შემდეგ, შეიძლება OK ღილაკზე დაჭერა. ამის შემდეგ **DreamweaverMX**-ი შექმნის ჩვენთვის საჭირო სერვერულ ქცევას-**Update Record**(ჩანაწერის განახლება).

თითქმის დასრულებული ვებ-გვერდი **Category_edit.php** შევინახოთ და გავხსნათ კატეგორიების სიის ადმინისტრატიული ვებ-გვერდი **Category_admin.php**. კატეგორიების სიის შემცველ ცხრილში მოვძებნოთ სტრიქონი **Change-Изменить-შეცვლა** და გარდავქმნათ იგი შემდეგი ინტერნეტ-მისამართის მქონე ჰიპერმინიშნებად:

Category_edit.php?id=<?php echo $row_Categories[‘id’] ; ?>

(მასივი **$row_Categories**, შეიცავს **Categories_admin.php** ვებ-გვერდის, **Categories**-ჩანაწერების ანაკრების ველების მნიშვნელობებს).

როდესაც ჩვენ დავიწყებთ **Category_edit.php** ვებ-გვერდის ექსპლუატაციას, მაშინათვე წავაწყდებით ისეთ სიტუაციას, როდესაც ჩანაწერის შეცვლის შემდეგ დაბრუნება ხდება მაინც და მაინც სტატიების სიაზე (**GET** მეთოდით **Categories_admin.php** ვებ-გვერდზე გადასაცემი **file** არგუმენტის მნიშვნელობა ტოლია 0-ის, როგორც ეს ჩვენ თვითონვე განვსაზღვრეთ მე-8 თავში). როგორ გამოვასწოროთ ეს სიტუაცია?

გამოსავალი საკმაოდ მარტივია. როდესაც ჩვენ ვქმნიდით **Category_add.php** ვებ-გვერდს, წავაწყდით ანალოგურ სიტუაციას, როდესაც ეს გვერდი, გვერდზე გადასვლისას ”უბრუნებს” მას, მისთვის **GET** მეთოდით გადაცემულ ყველა არგუმენტს. ესეიგი, თუ ჩვენ **id** არგუმენტთან ერთად, **Category_edit.php** ვებ-გვერდს, გადავცემთ არგუმენტს **file**, მაშინ ჩანაწერის შეცვლის შემდეგ ორივე

მათგანი ”დაუბრუნდება” **Categories_admin.php** ვებ-გვერდს. უცნობ **id** არგუმენტს ეს გვერდი უგულვებელყოფს, ხოლო არგუმენტ **file**-ს, გამოიყენებს შესაბამისი კატეგორიების სიის გამოსატანად.
რახან ვთქვით-უნდა გავაკეთოთ კიდევაც. ჰიპერმინიშნებას **შეცვლა** ვანიჭებთ ახალ ინტერნეტ-მისამართს:

Category_edit.php? id = <?php echo $row_Categories [‘id’] ; ?>&
File = <?php echo $colname_Categories ; ?>

(ცვლადი **$colname_Categories** შეიცავს **Categories_admin.php** გვერდისათვის **GET** მეთოდით გადაცემული **file** არგუმენტის მნიშვნელობას).
დაგვრჩა მხოლოდ **Categories_edit.php** ვებ-გვერდზე დაბრუნება და მასში, **Categories_admin.php**-კატეგორიების სიის ვებ-გვერდზე მიმთითებელი ჰიპერმინიშნების შექმნა, რათა შეგვეძლოს მასზე დაბრუნება, თუ რაიმე მიზეზების გამო, გადავიფიქრებთ ჩანაწერის ჩასწორებას. ტექსტური კურსორი დავაყენოთ ფორმის მარჯვენა მხარეს და ცარიელი აბზაცის შესაქმნელად, დავაჭიროთ კლავიშას **<ENTER>**. ამ აბზაცში ავკრიფოთ ტექსტი **კატეგორიების სიაზე** და გარდავქმნათ იგი ასეთი ინტერნეტ-მისამართის მქონე ჰიპერმინიშნებად:

Categories_admin.php?file=<?php echo $_GET[“file”] ; ?>

ამით ჩანაწერების ჩასწორების ვებ-გვერდის შექმნა დასრულებულია! შეიძლება მისი შემოწმება მოქმედებაში.

ჩანაწერების ჩასწორებისათვის გამოსაყენებელი PHP სცენარების გარჩევა.
აქ მაინცადამაინც ბევრი არაფერია გასარჩევი. **Update Record** სერვერული ქცევის შესაბამისი სცენარი, პრაქტიკულად არაფრით განსხვავდება იმ სცენარისაგან, რომელიც შეესაბამება ჩვენთვის უკვე ნაცნობ სერვერულ ქცევას **Insert Record**. ის კი ჩვენ უკვე გავარჩიეთ.

ამიტომ მოდით, განვიხილოთ სცენარების მხოლოდ ის ფრაგმენტები, რომლებიც განსხვავებულია მათგან.

ქვემოთ მოცემულია **category_edit** ფორმის შემქმნელი **HTML** კოდი:

```
<FORM METHOD="POST" ACTION="<?php echo $editFormAction; ?>" NAME="category_edit" ID="category_edit">
<P>დასახელება:
<INPUT NAME="name" TYPE="text" ID="name" VALUE="<?php echo $row_Category['name'] ; ?>" SIZE="20" MAXLENGTH="20">
</P>
<P>ფაილი:
<INPUT <?php if (!(strcmp($row_Category['file'],"1"))) {echo "checked" ; } ?> NAME="file" TYPE="checkbox" ID="file" VALUE="1">
<INPUT NAME="id" TYPE="hidden" ID="id" VALUE="<? php echo $row_Category['id']; ? >">
</P>
<P>
<INPUT NAME="submit" TYPE="submit" ID="submit" VALUE="შეცვლა">
</P>
<INPUT TYPE="hidden" NAME="MM_update" VALUE="category_edit">
</FORM>
```

აქ დაემატა კიდევ ერთი უხილავი **id** ველის განმსაზღვრელი **<INPUT>** ტეგი, რომლის **TYPE** ატრიბუტის მნიშვნელობა ტოლია **hidden**-ის. **DreamweaverMX**-ის მიერ შექმნილ დამხმარე უხილავ ველს უკვე MM_update სახელიც გააჩნია და category_edit სტრიქონის (**<INPUT>** ტეგის, **VALUE** ატრიბუტის მნიშვნელობა) მატარებელია თავის თავში. შესაბამისად, ჩანაწერის შემცვლელი სცენარი, ამოწმებს მისთვის **POST** მეთოდის საშუალებით გადაცემულ მონაცემებში ჩაშენებული **$_POST** მასივის **MM_update** ელემენტის არსებობას.

ასევე აქ დამატებულია მცირე ზომის სცენარები, რომლებიც მართვის ელემენტებში ათავსებენ, მათი შესაბამისი ველების მნიშვნელობებს **categories** ცხრილიდან. მათ ეს მნიშვნელობები შეაქვთ **<INPUT>** ტეგების ყველა **VALUE** ატრიბუტებში. ჩვენ არ

განვიხილავთ ამ სცენარებს-ისინი ძალზე მარტივები არიან და ჩვენ მათ უკვე ვიცნობთ.
ქვემოთ ნაჩვენებია ჩანაწერის შეცვლაზე **SQL** მოთხოვნის მაფორმირებელი **PHP** კოდის ფრაგმენტი:

$updateSQL = sprintf ("UPDATE categories SET name=%s, file=%s WHERE id=%s", GetSQLValueString($_POST['name'] , "text") , GetSQLValueString(isset($_POST['file']) ? "true" : " ", "defined" , "1" , "0"), GetSQLValueString ($_POST ['id'], "int')) ;

როგორც ვხედავთ, **name** შესატანი ველის და **file** ალმის მნიშვნელობები შეიტანებიან მათ შესაბამის ველებში, ხოლო უხილავი id ველის მნიშვნელობა გამოიყენება ჩასასწორებელი ჩანაწერის მოსაძებნად.
DreamweaverMX-ის მიერ შექმნილ **PHP** სცენარებში, სხვა რაიმე საინტერესო არ არის. ამიტომ მოდით შევეშვათ მათ და შევუდგეთ ჩანაწერის წაშლის ვებ-გვერდის **Category_delete.php** შექმნას.

ჩანაწერის წაშლის მარტივი ვებ-გვერდი

მესამედაც შევქმნათ ახალი სერვერული **PHP** გვერდი და შევინახოთ ის **Admin** საქაღალდეში **Category_delete.php** სახელით. მივანიჭოთ მას სათაური კატეგორიის წაშლა, შევიტანოთ ასეთივე სახელწოდება და განმარტებითი ტექსტი. ბოლოს ამ ვებ-გვერდზე შევქმნათ ცარიელი ფორმა და დავარქვათ მას სახელად **category_delete**.
ჩანაწერის წასაშლელად ჩვენ ისევ ორი ამოცანის შესრულება მოგვიწევს:

-ცხრილიდან categories საჭირო ჩანაწერის ამოღება და მისი ველების მნიშვნელობების გამოტანა ვებ-გვერდზე;
-ღილაკზე **წაშლა-Delete-Удалить** დაჭერის შემდეგ-არჩეული ჩანაწერის წაშლა ცხრილიდან **categories.**

ეს ამოცანები გაცილებით მარტივია, ვიდრე **Category_edit.php** ვებ-გვერდის შემთხვევაში. მაგრამ მაინც მოგვიწევს თავის შეწუხება.

დავიწყოთ ერთად-ერთი-წაშლისათვის განკუთვნილი ჩანაწერის შემცველი, ჩანაწერების ანაკრებიდან. მას ისეთივე პარამეტრები ექნება, რაც იმ ანაკრებს, რომელიც ჩვენ შევქმენით, როდესაც ვმუშაობდით **Category_edit.php** ვებ-გვერდზე, და ისეთივე სახელი-**Category**.

ჩანაწერების ანაკრების შექმნის შემდეგ, ტექსტური კურსორი განვათავსოთ ფორმაზე **category_delete**, ავკრიფოთ ტექსტი **დასახელება:** და დავსვათ გამოტოვების ნიშანი (ჰარი). უშუალოდ გამოტოვების ნიშნის შემდეგ ჩავსვათ **Category**-ჩანაწერების ანაკრების, **name** ველის მნიშვნელობის ამსახველი დინამიური ტექსტი. შემდეგ ახალი აბზაცის შესაქმნელად დავაჭიროთ კლავიშას **<ENTER>**, ავკრიფოთ ამ აბზაცში ტექსტი **ფაილი:** და დავსვათ გამოტოვების ნიშანი (ჰარი). ამის შემდეგ გადავერთოთ **HTML** კოდის წარმოჩენის რეჟიმში, რათა ხელით შევიტანოთ **file** ველის მნიშვნელობის გამომტანი სცენარი. ის ძალიან მარტივი იქნება:

```
<?php if (!(strcmp($row_Category['file'] , "1"))) { echo "+" ; } ?>
```

როგორც ვხედავთ, თუ ველი **file** შეიცავს **true** მნიშვნელობას გამოიტანება ნიშანი "**plus(плюс)**". თუკი ეს ველი შეიცავს მნიშვნელობას **false** არაფერლი არ გამოიტანება.

გადავერთოთ უკან, ვებ-გვერდის წარმოჩენის რეჟიმში და კურსორი დავაყენოთ მეორე აბზაცის ბოლოს. მენიუ **Insert**-ის, ქვემენიუ **Form**-ის, **Hidden Fild** პუნქტის არჩევით მოვათავსოთ იქ უხილავი ველი. აღნიშნულ უხილავ ველს მივანიჭოთ სახელი **id**.

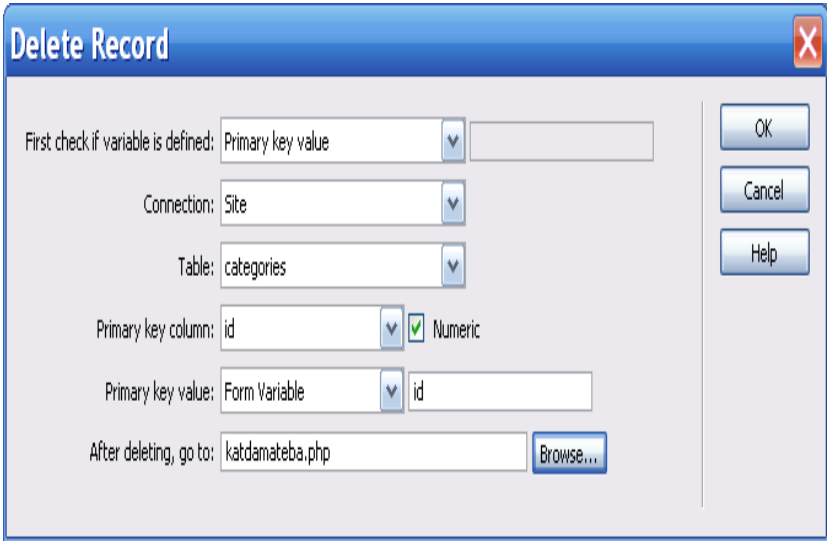
ავირჩიოთ უხილავი ველი **id** და დავაჭიროთ ელვის ნიშნაკიან ღილაკს, რომელიც თვისებების რედაქტორში, **Value** შესატანი ველის მარჯვნივ მდებარეობს (იხ. ნახ. 9.15.).

Recordset (Category) დიალოგური ფანჯრის იერარქიულ სიაში **Field** ჩამოვშალოთ "ხე" **Recordset (Category)** და ავირჩიოთ "შტო" **id**, რის შემდეგაც დავაჭიროთ **OK.**

შემდეგ, დავაყენოთ ტექსტური კურსორი ახლად შექმნილი უხილავი ველის შემდეგ და დავაჭიროთ კლავიშას **<ENTER>,** რათა შევქმნათ კიდევ ერთი აბზაცი. მასზე განვათავსოთ შემდეგი პარამეტრების მქონე ღილაკი:

-სახელი (თვისებების რედაქტორის შესატანი ველი **Button name)-submit**;

-წარწერა ღილაკზე (შესატანი ველი Lable)-**წაშლა-Delete-Удалить**.
-ღილაკის მიერ შესასრულებელი მოქმედება (გადამრთველების ერთობლივობა **Action**)-მონაცემების გაგზავნა (გადამრთველი **Submit form**).
შევინახოთ გვერდი და შევუდგეთ ჩანაწერის წაშლის განმახორციელებელი სერვერული ქცევის შექმნას. მას ეწოდება **Delete Record**.
ეკრანზე გამოვიტანოთ პანელი **Server Behaviors**, დავაჭიროთ მის "**plus(плюс)**" ნიშნაკიან ღილაკს და ეკრანზე გამოსულ მენიუში ავირჩიოთ პუნქტი **Delete Record** (ნახ. 9.17).



ნახ.9.17. დიალოგური ფანჯარა Delate Record

აღნიშნული ფანჯრის ჩამოსაშლელ სიაში **First check if variable is defined** შეიძლება არჩეულ იქნას ნებისმიერი პუნქტი. გაჩუმებით იქ არჩეულია პუნქტი **Primary key value**-დავტოვოთ იგი უცვლელად.
ჩამოსაშლელი სია **Connection**, ისევე ბაზასთან კავშირს განსაზღვრავს; ჩვენს შემთხვევაში-Site.

ცხრილი, რომლიდანაც უნდა მოხდეს ჩანაწერის წაშლა განისაზღვრება ჩამოსაშლელი სიის **Table** საშუალებით. ავირჩიოთ აქ ჩვენი ცხრილი **categoryes**.

ჩამოსაშლელ სიაში **Primary key column**, ხდება ჩვენი ცხრილის იმ ველის არჩევა, რომლის მიხედვითაც უნდ განხორციელდეს წასაშლელი ჩანაწერის მოძებნა. ჩვენს შემთხვევაში-ეს არის ველი **id**, ამიტომ აღნიშნულ სიაში ვირჩევთ შესაბამის პუნქტს. ამის შემდეგ ვრთავთ ალამს **Numeric**, ვინაიდან ამ ველს რიცხვითი ტიპი გააჩნია.

ჩამოსაშლელი სია **Primary key value** განსაზღვრავს, თუ საიდან უნდა იყოს არჩეული **Primary key column** სიით განსაზღვრულ ველში მოსაძებნი მნიშვნელობა. ვინაიდან ჩვენ დაგვჭირდება უხილავ **id** ველში მოთავსებული მნიშვნელობის ძებნის განხორციელება, ავირჩიოთ პუნქტი **Form Parameter-POST** მეთოდით გადაცემული არგუმენტი. შევიტანოთ ამ არგუმენტის სახელი-id, მარჯვენა მხარეს გამოსულ შესატან ველში.

შესატანი ველი **After deleting**, go to განსაზღვრავს ვებ-გვერდს, რომელზედაც მოხდება გადასვლა ჩანაწერის შეცვლის შემდეგ. შევიტანოთ მასში ჩვენი კატეგორიების სიის ადმინისტრატიული ვებ-გვერდის სახელი-**Categories_admin.php**. ასევე შესაძლებელია მარჯვენა მხარეს განლაგებულ Browse ღილაკზე დაწკაპუნება და **DreamweaverMX**-ის ფაილების გახსნის დიალოგურ ფანჯარაში, საჭირო ფაილის არჩევა.

ახლა, დავაჭიროთ ღილაკს **OK** და შეიქმნება სერვერული ქცევა **Delete Record**.

შევინახოთ **Category_delete.php** და გავხსნათ კატეგორიების სიის ადმინისტრატიული ვებ-გვერდი-Categories_admin.php. კატეგორიების სიის შემცველ ცხრილში, მოვძებნოთ სტრიქონი **წაშლა** და გარდავქმნათ იგი შემდეგი ინტერნეტ-მისამართის მქონე ჰიპერმინიშნებად:

Category_delete.php? id=<?php echo $row_Categories ['id'] ; ?> &
file = <?php echo $colname_Categories ; ?>

დაგვრჩა მხოლოდ, **Category_delete.php** ვებ-გვერდზე დაბრუნება და მასზე, კატეგორიების სიის ვებ-გვერდზე-**Categories_admin.php**, მითითებელი ჰიპერმინიშნების შექმნა.

დავაყენოთ ტექსტური კურსორი ფორმის მარჯვენა მხარეს და ცარიელი აბზაცის შესაქმნელად დავაჭიროთ კლავიშს **<ENTER>**.

აღნიშნულ ცარიელ აბზაცში ავკრიფოთ ტექსტი **კატეგორიების სიაზე** და გარდავქმნათ იგი ქვემოთ ნაჩვენები ინტერნეტ-მისამართის მქონე ჰიპერმინიშნებად:

Categories_admin.php?file=<?php echo $_GET[“file”] ; ?>

ამით დასრულდა. სამი ვებ-გვერდი, რომლებიც ახორციელებენ ჩანაწერების დამატებას, ჩასწორებას და წაშლას, ჩვენ უკვე შევქმენით. ახლა დადგა ანალოგიური ვებ-გვერდების შექმნის რიგი, რომლებიც განახორციელებენ თვით სტატიების და ფაილების დამატებას, შეცვლას და წაშლას.

უფრო რთული სერვერული ვებ-გვერდები მონაცემების შეტანისა და ჩასწორებისათვის

სტატიებისა და **ფაილების** დამატების, ჩასწორების და წაშლისათვის განკუთვნილი ვებ-გვერდების შექმნა რამდენადმე რთული იქნება. და არა მხოლოდ იმიტომ, რომ ცხრილი **items** შეიცავს მნიშვნელობების შესატანად განკუთვნილი ველების უფრო მეტ რაოდენობას, არამედ იმიტომაც, რომ ამ ცხრილის ჩანაწერებთან სამუშაოდ განკუთვნილი ვებ-გვერდების შემადგენლობაში წარმოდგენილი იქნება სიები.

დავიწყოთ სტატიებისა და ფაილების დამატებისათვის განკუთვნილი ვებ-გვერდიდან-**Item_add.php**. შევქმნათ ახალი სერვერული **PHP** ვებ-გვერდი და **Item_add.php** სახელწოდებით შევინახოთ იგი **Admin** საქაღალდეში. მივცეთ მას დასახელება **სტატიის(ფაილის**) დამატება, შევიტანოთ ასეთივე სათაური და რაიმე განმარტებითი ტექსტი. შევქმნა აგრეთვე ფორმა, რომლის სახელი იქნება **item_add**.

ჩვენი მონაცემთა ბაზის-**site**, ცხრილი-**Items**, გაცილებით მეტი რაოდენობის ველებს შეიცავს, ვიდრე ცხრილი **categories**. ამიტომ მართვის ელემენტების შექმნაზე ჩვენ გაცილებით მეტი მუშაობა მოგვიწევს. არა იმიტომ, რომ ისინი განსაკუთრებულ სირთულეს წარმოადგენენ, არამედ იმიტომ, რომ უბრალოდ მათი რაოდენობა ძალიან დიდია.

მაშ ასე, ფორმაში ვსვამთ ტექსტურ კურსორს, ვწერთ ტექსტს **ავტორი:**, ვსვამთ გამოტოვების ნიშანს(ჰარი) და ვქმნით შესატან ველს ასეთი პარამეტრებით:

-სახელი (შესატანი ველი-**TextField**)-**author**;
-სიგანე სიმბოლოებში (შესატანი ველი-**Char width**)-30 (**items** ცხრილის, **author** ველის ზომა განისაზღვრება 30 სიმბოლოთი);
-სიმბოლოების მაქსიმალური რაოდენობა, რომელთა შეტანაც არის შესაძლებელი ამ შესატან ველში (შესატანი ველი **Max Chars**)-30;
-შესატანი ველის ტიპი (გადამრთველების ერთობლივობა **Type**)-ჩვეულებრივი ერთსტრიქონიანი შესატანი ველი (გადამრთველი **Single line**);
-გაჩუმებითი მნიშვნელობა (შესატანი ველი **Init val**)-არაფელი.

ტექსტურ კურსორს ვაყენებთ უშუალოდ ახლად შექმნილი შესატანი ველის შემდეგ და ვაჭერთ კლავიშს **<ENTER>**. ახალ აბზაცში ვწერთ **დასახელება:**, ვსვამთ გამოტოვების ნიშანს და ვქმნით მეორე შესატან ველს, შემდეგი პარამეტრებით:

-სახელი-**name**;
-სიგანე სიმბოლოებში-60 (**items** ცხრილის **name** ველის ზომა 60 სიმბოლოა);
-შეტანილი სიმბოლოების მაქსიმალური რაოდენობა-60;
-შესატანი ველის ტიპი-ჩვეულებრივი ერთსტრიქონიანი შესატანი ველი;
-გაჩუმებითი მნიშვნელობა-არაფელი.
შემდეგ აბზაცში ვწერთ **დამატებულია:**, ვსვამთ გამოტოვების ნიშანს და ვქმნით მესამე შესატან ველს. მისი პარამეტრები ასეთია:
-სახელი -**added;**
-სიგანე სიმბოლოებში-14 (იყოს ოდნავ მეტი, ვიდრე საჭიროა);
-შეტანილი სიმბოლოების მაქსიმალური რაოდენობა-10 (რიცხვებისათვის ორი სიმბოლო, თვეებისათვის ორი სიმბოლო, წლებისათვის ოთხი სიმბოლო და ორი წერტილი);
-შესატანი ველის ტიპი, ჩვეულებრივი ერთსტრქონიანი შესატანი ველი.
რაც შეეხება გაჩუმებით მნიშვნელობას, ჩვენ მას ამ ველს მივანიჭებთ-მასში მიმდინარე თარიღი ავტომატურად ჩაიწერება. თვისებების რედაქტორში დავაჭიროთ ელვის ნიშნაკიან ღილაკს და **Dynamic Data** დიალოგური ფანჯრის Code შესატან ველში შევიტანოთ ასეთი **PHP** კოდი:

<?php echo date (“Y-m-d”) ; ?>

ჩაშენებული ფუნქცია date ჩვენთვის ნაცნობია მე-8 თავიდან. ის ახდენს მეორე არგუმენტის მიერ გადაცემული თარიღის მნიშვნელობის ფორმირებას პირველი არგუმენტის მიერ გადაცემული შაბლონების სტრიქონის შესაბამისად და აბრუნებს მას სტრიქონული სახით. თუკი მეორე არგუმენტი არ არის მითითებული, ეს ფუნქცია დააბრუნებს მიმდინარე თარიღის მნიშვნელობას. ჩვენ სწორედ ეს გვჭირდება.
კი მაგრამ, რატომ ავირჩიეთ თარიღის წარმოდგენის ასეთი უცნაური ფორმატი-"წელი-თვე-რიცხვი"? საქმე იმაშია, რომ ეს ერთად-ერთი ფორმატია, რომელიც "გასაგებია" **MySQL**-ისათვის. თუკი ის თარიღს მიიღებს ჩვენთვის ჩვეული სახით "რიცხვი.თვე.წელი" , მაშინ **added** ველში ჩაწერს ისეთ რამეს, რისი დანახვისაც შეგვეშინდება. რას ვიზამთ-ეს თვით **MySQL** პროგრამის შეზღუდვაა და ჩვენ ვალდებული ვართ გავითვალისწინოთ იგი.

შენიშვნა: *ზოგადად, შესაძლებელია ამ პრობლემის გადაწყვეტა; თან ეს გადაწყვეტა არ იქნება რთული. როგორი იქნება? მოძებნეთ იგი დამოუკიდებლად. რჩევა-1: PHP-ს ჩაშენებული ფუნქცია სახელწოდებით substr საშუალებას გვაძლევს სტრიქონიდან "ამოვჭრათ" ნებისმიერი სიგრძის ფრაგმენტი. რჩევა-2: სტრიქონების გაერთიანების ოპერატორი....*

და კიდევ ერთი, მეოთხე აბზაცი. შეგვაქვს ტექსტი **ჰიპერმინიშნება:**, ვსვამთ გამოტოვების ნიშანს, შემდეგ ახალ შესატან ველს. ველის პარამეტრებია:

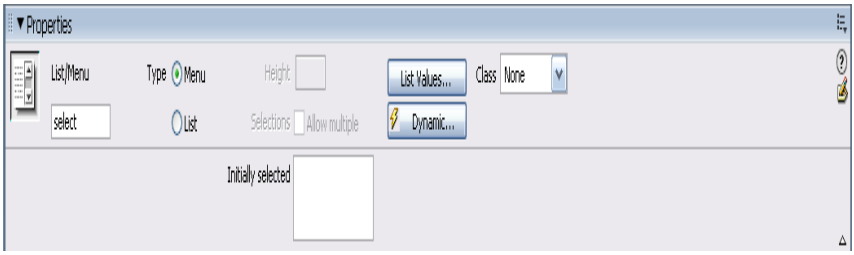
-სახელი-**href;**
-სიგანე სიმბოლოებში-60 (სხვა შემთხვევაში ის არ დაეტევა გვერდზე)
-შეტანილი სიმბოლოების მაქსიმალური მნიშვნელობა-255 (სწორედ ასეთი ზომა გააჩნია **items** ცხრილის ველს **href);**
-შესატანი ველის ტიპი-ჩვეულებრივი ერთსტრიქონიანი შესატანი ველი;
-გაჩუმებითი მნიშვნელობა-არაფელი.

ცოტაც მოვითმინოთ-მხოლოდ ორი მართვის ელემენტიღა დაგვრჩა! ვქმნით კიდევ ერთ აბზაცს, ვწერთ მასში **კატეგორია:**, ვსვამთ გამოტოვების ნიშანს და ვქმნით... რას ვქმნით?
აქ ჩვენ უნდა უზრუნველვყოთ არჩევანის შესაძლებლობა რამოდენიმე პოზიციიდან. ამის გაკეთება შესაძლებელია სამი სხვადასხვა მართვის ელემენტის საშუალებით: ჩვეულებრივი სიის, ჩამოსაშლელი სიის და გადამრთველების ერთობლივობის. თითოეულ ამ მართვის ელემენტს გააჩნია თავისი უპირატესობაც და ნაკლოვანებაც, რომელთაც ჩვენ ახლა განვიხილავთ.

ჩვეულებრივი სია-ასარჩევად ხელმისაწვდომი, საკმაოდ დიდი რაოდენობის პოზიციების, ერთდროულად წარმოჩენის საშუალებას იძლევა, მაგრამ საკმაოდ მოზრდილია. ის ეფექტურია, თუ ასარჩევად ხელმისაწვდომი პოზიციების რაოდენობა ძალიან დიდია-ათობით, თუ არა ასობით. ეს კი ჩვენი შემთხვევა არ არის.

გადამრთველების ანაკრების გამოყენება ყველაზე კარგია იმ შემთხვევაში, როდესაც პოზიციების რაოდენობა არცთუ ისე ბევრია-ათის ფარგლებში-და ყველა მათგანი სახეზე უნდა იყოს. თუ კი პოზიციების რაოდენობა მეტია, მაშინ გადამრთველების ანაკრების ზომა უბრალოდ ძალიან გაიზრდება და ნამდვილად აღარ დაეტევა არც ერთ ვებ-გვერდზე.

ჩამოსაშლელი სია კომპაქტურია და შეიძლება შეიცავდეს ძალიან ბევრ პოზიციას. მისი ერთად-ერთი ნაკლი ალბათ ის არის, რომ ყოველთვის სახეზეა მხოლოდ ერთი, მოცემული მომენტისათვის არჩეული პოზიცია. მაგრამ ეს ნაკლოვანება არ არის პრინციპიალური და ამიტომ მოდით შევჩერდეთ ჩამოსაშლელ სიაზე.
ჩამოსაშლელი სიის შესაქმნელად, საჭიროა **Insert** მენიუს, **Form** ქვემენიუს, **List/Menu** პუნქტის არჩევა. როდესაც სია შეიქმნება, დავაწკაპუნოთ მასზე და მივმართოთ თვისებების რედაქტორს (ნახ. 9.18), რათა განვსაზღვროთ ამ სიის პარამეტრები.



ნახ. 9.18. თვისებების რედაქტორის სახე არჩეული სიის შემთხვევაში

ჯერჯერობით ჩვენ მხოლოდ ორ პარამეტრს განვსაზღვრავთ:

-სახელი (შესატანი ველი **List/Menu**)-**category**;
-ტიპი (გადამრთველების ჯგუფი **Type**)-ჩამოსაშლელი სია (გადამრთველი **Menu**).

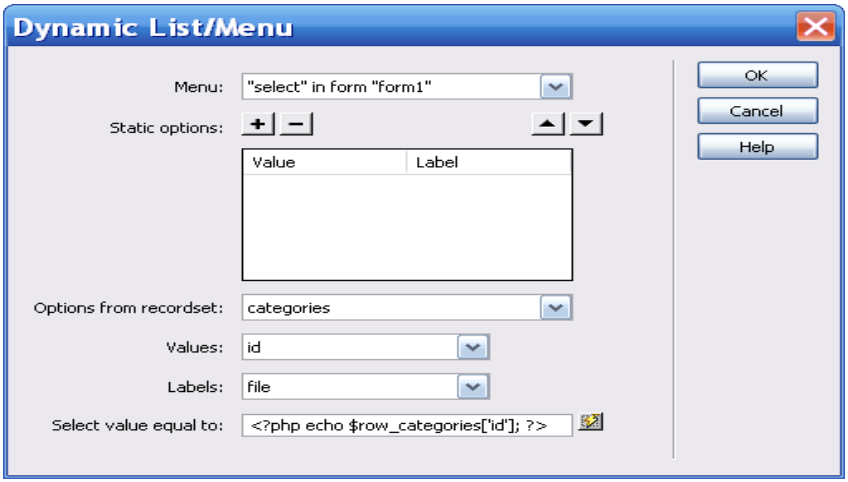
შენიშვნა! *თუ ჩავრთავთ Type ჯგუფის List გადამრთველს, შეიქმნება ჩვეულებრივი სია.*

დაგვრჩა მხოლოდ ტექსტური კურსორის დაყენება უშუალოდ ახლად შექმნილი სიის შემდეგ, **<ENTER>** კლავიშზე დაჭერა და ახალ აბზაცში ღილაკის შექმნა. მისი პარამეტრები ასეთი იქნება:
-სახელი-**submit**;
-წარწერა ღილაკზე-**დამატება**;
-ღილაკის მიერ შესასრულებელი მოქმედება-მონაცემების გაგზავნა.

შევინახოთ ვებ-გვერდი **Item_add.php** და ყურადღება გავამახვილოთ სიაზე **category.**
რას უნდა შეიცავდეს იგი? კატეგორიების სიას? საიდან უნდა აიღოს მან ეს სია? ცხრილიდან **categories**. ესეიგი, ჩვენ უნდა შევქმნათ **სტატიებისა** და **ფაილების** შემცველი ჩანაწერების ანაკრები და ჩვენს მიერ შექმნილი სია შევავსოთ ამ ანაკრებიდან აღებული მნიშვნელობებით.
ჩანაწერების ანაკრების შექმნის პროცესის ხელახლა აღწერა არ ღირს-ჩვენ საკმაო რაოდენობით გვაქვს ისინი შექმნილი. მაშ ასე, ჩვენი ჩანაწერების ანაკრებში წარმოდგენილი იქნება მონაცემთა ბაზა **site**-ში შემავალი ცხრილის **categories**, ველები **id** და **name**. ფილტრაცია განხორციელდება ცხრილის **file** ველის მნიშვნელობის მიხედვით,

ხოლო შესადარებელი მნიშვნელობა აღებული უნდა იქნას **Items_admin.php** ვებ-გვერდისაგან **GET** მეთოდით გადაცემული file არგუმენტიდან. ანაკრების ჩანაწერების დახარისხება განხორციელდება **id** ველის მიხედვით. ანაკრების სახელწოდება იქნება **Categories.**

და კიდევ ერთი რამე. მოდით ისე გავაკეთოთ, რომ, **Item_add.php** ვებ-გვერდის გახსნისთანავე, **category** სიაში, უკვე ჩასმული იყოს ის კატეგორია, რომლის სტატიებიც (ფაილებიც) ჩამოთვლილი იყო **Items_admin.php** ვებ-გვერდზე. ამისათვის ჩვენ მოგვიწევს ვებ-გვერდიდან **GET** მეთოდით კიდევ ერთი არგუმენტის-**catid** გადაცემა, რომელშიც მოთავსებული იქნება **categories** ცხრილის **id** ველის მნიშვნელობა მოცემული კატეგორიისათვის. აღნიშნულის განსახორციელებლად ეკრანზე გამოვიტანოთ პანელი **Bindings**, დავაჭიროთ **"plus(плюс)"** ნიშნაკიან ღილაკს, ეკრანზე გამოსულ მენიუში ავირჩიოთ პუნქტი **URL Variable**, შევიტანოთ **URL Variable** დიალოგური ფანჯრის **Name** შესატან ველში (იხ. ნახ.9.12.), შესაქმნელი არგუმენტის სახელი-**catid** და დავაჭიროთ ღილაკს **OK**. შემდეგ, **item_add** ფორმაში მოვნიშნოთ სია **category** და თვისებების რედაქტორში დავაჭიროთ ღილაკს **Dynamic** (იხ. ნახ.9.18.). ეკრანზე გამოჩნდება დიალოგური ფანჯარა **Dynamic List/Menu** (ნახ.9.19).



ნახ. 9.19. დიალოგური ფანჯარა Dynamic List.Menu

ამ ფანჯრის ჩამოსაშლელ სიაში **Menu**, მოიცემა სია, რომელიც ჩვენ უნდა შევავსოთ პუნქტებით. წესით, იქ უკვე უნდა იყოს არჩეული ჩვენი ერთად-ერთი სია **category**. ასე რომ, მას ხელს აღარ ვახლებთ.
Static option სია, ღილაკების ანაკრებთან ერთად, სიაში ზოგიერთი პუნქტების ხელით შეტანის საშუალებას გვაძლევს. მოცემულ მომენტში ჩვენ ეს არ გვჭირდება.
არის კიდევ ერთი სია-**Option from recordset**. ის განსაზღვრავს ჩანაწერების ანაკრებს, რომლიდანაც აიღებიან ჩანაწერები-ჩვენი სიის მომავალი პუნქტები. ავირჩიოთ მასში პუნქტი **Categories**-ჩვენი ჩანაწერების ანაკრები.
ჩამოსაშლელი სია **Values** განსაზღვრავს ჩანაწერების სიის ველს, რომლიდანაც მოხდება ეგრეთ წოდებული **სიის პუნქტების მნიშვნელობების** არჩევა. ეს მნიშვნელობები არ ჩნდებიან ეკრანზე, მაგრამ სწორედ მათი გაგზავნა ხდება ფორმიდან, მონაცემების მიმღები სერვერული პროგრამისათვის. ამ სიაში ავირჩიოთ პუნქტი **id**-ეს ველი შეიცავს უნიკალურ მნიშვნელობას, რომელიც ახდენს კატეგორიის იდენტიფიკაციას.
ჩამოსაშლელი სია **Labels** განსაზღვრავს ჩანაწერთა ანაკრების ველს, რომლიდანაც ხდება სიის პუნქტების დასახელებების აღება, სწორედ იმ პუნქტებისა, რომლებიც ეკრანზე ჩნდებიან. ბუნებრივია, ჩვენ უნდა ავიღოთ პუნქტი **name**.
შესატან ველში **Select value equal to** იწერება **PHP** კოდი, რომელიც აბრუნებს სიის გაჩუმებით მნიშვნელობას-ანუ, სიის იმ პუნქტის მნიშვნელობას, რომელიც თავდაპირველად უნდა იყოს არჩეული. მასში შეიძლება არც შევიტანოთ არაფელი და უბრალოდ დავაჭიროთ ამ ველის მარჯვნივ მდებარე ელვის ნიშნაკიან ღილაკს, ავირჩიოთ **Dynamic Data** იერარქიულ სიაში საჭირო არგუმენტი ან ველი და დავაჭიროთ ამავე ფანჯრის **OK** ღილაკს. ჩვენს შემთხვევაში იქ არჩეულ უნდა იქნას **GET** მეთოდით გადასაცემი არგუმენტი catid-სწორედ ის შეიცავს კატეგორიის ნომერს.

ყველა საჭირო მონაცემების განსაზღვრის შემდეგ, დავაჭიროთ ღილაკს **OK**. ხოლო, ამის შემდეგ, შეგვიძლია შევუდგეთ **Insert Record** სერვერული ქცევის შექმნას, ვინაიდან ყველა მოსამზადებელი მოქმედება ჩვენ უკვე შევასრულეთ. დიალოგურ ფანჯარაში **Insert Record** (იხ. ნახ. 9.10.) ჩვენ უნდა ავირჩიოთ:

-ფორმა, რომლიდანაც მოხდება ახალი ჩანაწერისათვის მონაცემების აღება (ჩამოსაშლელი სია **Submit values from)**-**item_add;**
-მონაცემთა ბაზასთან კავშირის სახელი (ჩამოსაშლელი სია **Connection**)-**Site**;
-ცხრილი, რომელსაც უნდა დაემატოს ჩანაწერი (ჩამოსაშლელი სია **Insert table**)-**items**;
-გვერდი, რომელზეც მოხდება გადასვლა ჩანაწერის დამატების მერე (შესატანი ველი **After inserting, go to**)-**Items_admin.php**.

ყოველივე აღნიშნულის გარდა, ჩვენ კიდევ გვჭირდება, **Column** სიაში არსებული, **items** ცხრილის ყოველი ველის რიგრიგობით არჩევა და მისთვის ზოგიერთი პარამეტრების განსაზღვრა. ისინი ყველა ჩამოთვლილია ცხრილში 9.1..

ცხრილი 9.1. items ცხრილის ველების პარამეტრები, Items_add.php ვებ-გვერდისათვის

ცხრილის ველი	**ფორმის მართვის ელემენტი (სია Value)**	**მონაცემების ტიპი (სია Submit as)**
id	არა (პუნქტი None)	მთელრიცხვა
author	author	სტრიქონული (პუნქტი Text)
name	name	სტრიქონული
added	added	თარიღი (პუნქტი Date)
href	href	სტრიქონული
catid	category	მთელრიცხვა (პუნქტი Integer)

შეტანის დასრულების შემდეგ ვაჭერთ ღილაკს OK. აღნიშნულის მერე, **Insert Record** სერვერული ქცევა შეიძლება ჩაითვალოს შექმნილად.
ახლა მოდით შევქმნათ **Items_admin.php** ვებ-გვერდზე მიმთითებელი ჰიპერმინიშნება. ცარიელი აბზაცის შესაქმნელად ტექსტური კურსორი დავაყენოთ ფორმის მარჯვნივ და დავაჭიროთ კლავიშს **<ENTER>**. ამ ცარიელ აბზაცში ავკრიფოთ ტექსტი

სტატიების სიაზე და გადავაქციოთ ის ასეთი ინტერნეტ-მისამართის მქონე ჰიპერმინიშნებად:

Items_admin.php?catid=<?php echo $_GET[“catid”] ; ?>

ახლა, გვრჩება მხოლოდ **Items_admin.php** ვებ-გვერდის გახსნა, **HTML** კოდის წარმოჩენის რეჟიმში გადასვლა, **დამატება** ჰიპერმინიშნების კოდის მოძებნა და ამ ჰიპერმინიშნებისათვის ასეთი ინტერნეტ-მისამართის მინიჭება:

Item_add.php?file=<?php echo $row_Category[‘file’] ; ?>&catid=<?php echo $row_Category[‘id’] ; ?>

(მასივი **$row_Category** შეიცავს ამ გვერდზე შექმნილი **Category**-ს ჩანაწერთა ანაკრების ველების მნიშვნელობებს. თვით ანაკრები **Category** კი შეიცავს მხოლოდ ერთ ჩანაწერს-არჩეულ კატეგორიას).
რაც შეეხება **სტატიების (ფაილების)** ჩასწორებისა და წაშლის ვებ-გვერდებს, მათი შექმნაც, კატეგორიების ჩასწორებისა და წაშლის ვებ-გვერდების ანალოგიურად ხდება. ამ ვებ-გვერდებისათვის ჩვენ მოგვიწევს სამი არგუმენტის გადაცემა **GET** მეთოდით: **file** საჭირო კატეგორიების ასარჩევად, **id** (შესასწორებელი ან წასაშლელი სტატიის ან ფაილის ასარჩევად) და **catid** სტატიების და ფაილების სიაში კორექტულად დაბრუნებისათვის). აგრეთვე თარიღის მნიშვნელობის გამოსატანად **MySQL**-ის მიერ მხარდაჭერილ ერთად-ერთ ფორმატში ”წელი-თვე-რიცხვი” , საჭირო იქნება **PHP**-ს ასეთი გამოსახულების გამოყენება:

<?php echo date(“Y-m-d” , strtotime($row_Item(‘added’])) ; ?>

თუმცა, **სტატიის (ფაილის)** წაშლის ვებ-გვერდზე **Item_delate.php**, თარიღის გამოტანა შესაძლებელია ჩვენთვის ჩვეული ფორმატითაც ”რიცხვი.თვე.წელი”-ჩვენ ხომ იქ მას აღარ შევცვლით. ესეიგი არ მოგვიწევს მისი ხელახლა ჩაწერა ცხრილში. ამიტომ აღნიშნული მიზნისათვის გამოვიყენებთ ჩვენთვის უკვე ნაცნობ სცენარს:

<?php echo date(“d.m.Y” , strtotime($row_Item(‘added’])) ; ?>

ვებ-გვერდზე **Item_delete.php**, ჩვენ ასევე დაგვჭირდება იმ კატეგორიის დასახელების გამოტანა, რომელსაც მიეკუთვნება წასაშლელი **სტატია (ფაილი).** ამისათვის ჩვენ უნდა შევქმნათ ჩანაწერთა ანაკრები, რომელიც **Categories** ცხრილის იმ ერთად-ერთ ჩანაწერს ამოიღებს, რომლის id ველის მნიშვნელობა, ტოლია იქნება **GET** მეთოდით გადაცემული, **catid** არგუმენტის მნიშვნელობის.
ამასთან არ უნდა დაგვავიწყდეს **შეცვლა** და **წაშლა** ჰიპერმინიშნებების შექმნა **Item_admin.php** ვებ-გვერდზე.
ცხადია აქ **Item_edit.php** და **Item_delete.php** ვებ-გვერდების შექმნის პროცესი სრულად არ არის აღწერილი-ჩავთვალოთ ის საშინაო დავალებად და დამოუკიდებლად შევასრულოთ.
უკანასკნელი, რაც ჩვენ უნდა გავაკეთოთ-ეს არის, ჩვენი საიტის მთავარ ვებ-გვერდზე-**default.htm**, **კატეგორიების სიის** ადმინისტრატიულ ვებ-გვერდზე **Categories_admin.php**, მიმთითებელი ჰიპერმინიშნებების განთავსება. ეს იმიტომ, რომ ვებ-დამთვალიერებლის მისამართების სტრიქონში, ამ ვებ-გვერდზე მიმთითებელი მისამართის აკრეფა ხომ ძალიან დამღლელია, ხოლო საიტის მისამართის აკრეფა და მასზე განთავსებულ ჰიპერმინიშნებაზე დაწკაპუნება, ბევრად უფრო მარტივი და სწრაფად განსახორციელებელია.

მონაცემების შეტანის კერძო შემთხვევა-საძიებო მანქანა.

სერვერული პროგრამების საფუძველზე შექმნილ ყოველ სერიოზულ ვებ-საიტს გააჩნია მონაცემების საძიებო საშუალება-ეგრეთ წოდებული **საძიებო მანქანა.**

საძიებო მანქანა მომხმარებლისაგან იღებს ძიების კრიტერიუმებს, ეძებს ამ კრიტერიუმების შესაბამის მონაცემებს და შედეგი გამოაქვს განსაკუთრებულ ვებ-გვერდზე **(ძიების შედეგების გამომტან ვებ-გვერდზე)**.
საძიებო მანქანის რეალიზაცია-ეს, შეიძლება ითქვას, რომ მონაცემების შეტანის კერძო შემთხვევაა. მართლაც, საიტის მომხმარებელმა ჯერ უნდა შეიტანოს მისთვის საჭირო საძიებო კრიტერიუმები, რათა საძიებო მანქანამ იცოდეს, თუ რა უნდა მოძებნოს.
მოდით, ჩვენი მარტივი საიტისათვის გამოვიყენოთ ძიების ერთად-ერთი კრიტერიუმი-გასაღები სიტყვა, რომელიც შესაძლოა შეგვხვდეს სტატიის ან ფაილის დასახელებაში. **PHP**-ს სცენარი, რომელსაც ჩვენ

შევქმნით (უფრო სწორად მას **Dreamweaver**-ი შექმნის), განახორციელებს **items** ცხრილის იმ ჩანაწერების ძებნას, რომელთა name ველში წარმოგდენილი იქნება აღნიშნული გასაღები სიტყვა. გასაღები სიტყვის შესატანად ჩვენ გამოვიყენებთ შესატანი ველისა და ღილაკის შემცველ ფორმას, რომელსაც მოვათავსებთ ჩვენი საიტის მთავარ ვებ-გვერდზე **default.htm**.
ამისათვის ვხსნით **default.htm** ვებ გვერდს, ვამატებთ მასში განმარტებითი ტექსტის შემცველ ახალ აბზაცს და ვსვამთ **search** სახელწოდების მქონე ფორმას. ამ ფორმაზე ვქმნით შესატან ველს **keyword** სახელწოდებით და მონაცემების გაგზავნის ღილაკს **submit**, წარწერით-**ძიება**.
ეხლა კი საჭიროა ყურადღება! ფორმის პარამეტრების განსაზღვრა, ჩვენ თვითონ მოგვიწევს-**DreamweaverMX**-ი მას არ გააკეთებს ჩვენს მაგივრად. მაშ ასე, ტექსტური კურსორი დავაყენოთ სადმე ფორმის შიგნით, ფორმის მოსანიშნად დავაწკაპუნოთ ტეგების სექციის **<form>** ღილაკზე და მივმართოთ თვისებათა რედაქტორს (იხ. ნახ. 9.5.). ფორმა **search**-ის პარამეტრები ასეთი იქნება:

-ფორმის სახელი (შესატანი ველი **Form name**)-**search** (ჩვენ იგი უკვე განვსაზღვრეთ);
-სერვერული გვერდი, რომელმაც უნდა მიიღოს მონაცემები ფორმიდან (შესატანი ველი **Action**)-**Result.php** (ჩვენ მას მერე შევქმნით);
-მონაცემების გადაცემის მეთოდი (ჩამოსაშლელი სია **Method**)-**GET** (სიის იმავე სახელის მქონე პუნქტი).

მონაცემების გადასაცემად ჩვენ გამოვიყენებთ **GET** მეთოდს-როგორც წესი გასაღები სიტყვები არც ისე დიდები არიან და არც საიდუმლოებას არ წარმოადგენენ.
შევინახოთ გადაკეთებული ვებ-გვერდი **default.htm** და დავხუროთ იგი. ახლა შევქმნათ ვებ-გვერდი **Result.php**, რომელზეც ძიების შედეგები უნდა იქნას გამოტანილი. მივანიჭოთ მას სახელი **ძიების შედეგები**, შევიტანოთ ასეთივე სათაური და რაიმე განმარტებითი ტექსტი. შევინახოთ ეს ვებ-გვერდი საიტის ძირეულ საქაღალდეში-მისდამი წვდომის შესაძლებლობა უნდა ჰქონდეთ ჩვეულებრივ მომხმარებლებს (სტუმრებს).
ამის შემდეგ, ჩვენ უნდა შევქმნათ ჩანაწერთა ანაკრები, რომელშიც მოთავსებული იქნება **items** ცხრილის ის ჩანაწერები, რომელთა **name**

ველის მნიშვნელობები შეიცავენ **search** ფორმაში ჩვენს მიერ შეტანილ გასაღებ სიტყვას. ჩანაწერთა ამ ანაკრებს ვუწოდოთ **Result**.
ამის შემდეგ, ჩვენ უნდა განვსაზღვროთ ჩანაწერების ფილტრაციის კრიტერიუმები. ფილტრაციას განვახორციელებთ **name** ველის მიხედვით, ასე რომ, ჩამოსაშლელ სიაში **Filter**, ავირჩიოთ პუნქტი **name**. **name** ველის მნიშვნელობა უნდა შეიცავდეს მომხმარებლის მიერ შეტანილ გასაღებ სიტყვას, ამიტომ მარჯვენა მხარეს მდებარე ჩამოსაშლელ სიაში ვირჩევთ პუნქტს **contains**. (ეს პუნქტი მოთხოვნაში ამატებს შედარების განსაკუთრებულ ოპერატორს **LIKE**, რომელიც აბრუნებს მნიშვნელობას **true**, თუ მისგან მარცხენა მხარეს მდებარე სტრიქონი შეიცავს მისგან მარჯვნივ მითითებულ სტრიქონს). თვით გასაღები სიტყვა აიღება **GET** მეთოდით გადაცემული არგუმენტიდან, ამიტომ ქვედა ჩამოსაშლელ სიაში ვირჩევთ პუნქტს **URL Parameter**. ხოლო, ამ არგუმენტის სახელი-**keyword**-შეგვაქვს მარჯვენა ქვედა მხარეს მდებარე შესატან ველში.
ახლა დაგვრჩა მხოლოდ სორტირების განსაზღვრა **added** ველის და კლებადობის მიხედვით-და ჩანაწერთა ანაკრები **Result** მზად იქნება. მაგრამ მოდით ეს ამოცანა რამდენადმე გავართულოთ. დავუშვათ, რომ აღნიშნულ ანაკრებს სხვებთან ერთად გამოაქვს იმ კატეგორიის დასახელება, რომელშიც მდებარეობს **სტატია** ან **ფაილი**. ამისათვის ჩვენ მოგვიწევს ერთ **SQL** მოთხოვნაში ორი ცხრილის ერთმანეთთან დაკავშირება (ცხრილების დაკავშირების შესახებ იხ. თავი-6).
დავაწკაპუნოთ **Recordset** დიალოგური ფანჯრის **Advanced** ღილაკზე-იგი გადაერთვება გაფართოებულ რეჟიმში (იხ. ნახ.8.15). **SQL**-ის შესატან ველში ჩვენ დავინახავთ **DreamweaverMX**-ის მიერ უკვე შექმნილი მოთხოვნის კოდს. მას ასეთი სახე ექნება:

```
SELECT * FROM items WHERE name LIKE '%colname% ORDER BY added DESC
```

აქ **colname**-არის **SQL** მოთხოვნის ცვლადი, რომლის ნაცვლადაც ჩასმული იქნება **Keyword** არგუმენტის მნიშვნელობა. ხოლო შაბლონი **%** აღნიშნავს ნებისმიერი სიმბოლოების, ნებისმიერ რაოდენობას (მათ შორის მათ არ არსებობასაც).
სამწუხაროთ **DreamweaverMX**-ი ჩვენ აქ ვერაფრით ვერ დაგვეხმარება. ჩვენ თვითონ მოგვიწევს ამ ველში ჩვენი **SQL**-

მოთხოვნის კოდის ხელით შეტანა. ჩვენს ახალ მოთხოვნას ასეთი სახე ექნება:

SELECT items. * , categories.name AS cat, categories.file
FROM items, categories WHERE items.catid=categories. id AND
Items.name LIKE '%colname%' ORDER BY items.added DESC

ყოველივე ამის შემდეგ შეგვიძლია დავაჭიროთ ღილაკს **OK.**
ახლა ვებ-გვერდზე **Result.php** შევქმნათ ხუთსვეტიანი ცხრილი. პირველ სვეტში მოვათავსოთ ავტორის სახელი, მეორეში-დასახელება, მესამეში-დამატების თარიღი, მეოთხეში-კატეგორიის დასახელება, ხოლო მეხუთეში სტატიის ან ფაილის გამხსნელი ჰიპერმინიშნება. ცხრილის პირველი სტრიქონის უჯრებში ჩავწეროთ სვეტების დასახელებები, ხოლო მეორეს უჯრედებში მოვათავსოთ ჩვენი ჩენაწერთა ანაკრების შესაბამისი ველების მნიშვნელობების გამომტანი დინამიური ტექსტი. შემდეგ შევქმნათ ცხრილის მეორე სტრიქონის შემცველი განმეორებადი არე-და საქმე დამთავრებულია.
ახლა პრინციპში, შესაძლებელია ვებ-გვერდის შენახვა და დასრულებული საძიებო მანქანის მოქმედებაში შემოწმება. მაგრამ, მოდით კიდევ რამდენადმე გავაუმჯობესოთ იგი. გავაკეთოთ ისე, რომ მეოთხე სვეტში კატეგორიის დასახელების შემდეგ ფრჩხილებში გამოიტანებოდეს სტრიქონი **სტატიები** ან **ფაილები**.

გადავერთოთ **HTML** კოდის წარმოჩენის რეჟიმში და მოვძებნოთ კატეგორიის გამომტანი **PHP** სცენარი. აგერ ისიც:

<?php echo $row_Result ['cat'] ; ?>

იმისათვის, რომ კატეგორიის დასახელებას დავუმატოთ სტრიქონი **სტატიები** ან **ფაილები**, ჩვენ დაგვჭირდება ლოგიკური ველის **file**-ის მნიშვნელობის შემოწმება. თუ ეს მნიშვნელობა იქნება 0 (**$row_Result** მასივის ელემენტებს ყოველთვის გააჩნიათ სტრიქონული ტიპი, ხოლ 0-არის **false** ლოგიკური ცვლადის რიცხვით წარმოდგენა), საჭიროა სტატიების გამოტანა, თუ კი იქნება 1-ფაილების. ამიტომ ადრე განხილულ სცენარს დავუმატოთ ფრაგმენტი და წარმოვადგინოთ იგი ასეთი სახით (დამატებული კოდის ფრაგმენტი გამოყოფილია მუქი შრიფტით):

<?php echo $row_Result['cat']. " (" . (($row_result['file] == "1") ? "ფაილები" : "სტატიები") . ") " ; ?>

ახლა კი ნამდვილად დასრულდა ყველაფერი! დასრულებული საძიებო მანქანა შეგვიძლია გავსინჯოთ "საბრძოლო" პირობებში.
რათქმა უნდა, ძიების შედეგების გამომტანი ვებ-გვერდი **Result.php**-შორს არის სრულყოფილებისაგან. მაგალითად, თუ ძიებამ შედეგი არ მოგვცა, ცხრილი ეკრანზე მაინც გამოიტანება. მაგრამ ეს სიტუაცია გასწორებადია-საკმარისია შეიქმნას ორი არაუცილებელი არე, ისევე როგორც **Items.php** ვებ-გვერდის შემთხვევაში (იხ. თავი 8). ასევე შესაძლებელია კატეგორიის დასახელების გარდაქმნა, მოცემულ კატეგორიაში შემავალი **სტატიების(ფაილების)** შემცველ **Items.php** ვებ-გვერდზე მიმთითებელ ჰიპერმინიშნებად. საერთოდ ბევრი რამის გაკეთებაა შესაძლებელი, მთავარია-დროზე გაჩერება!

რა იქნება ამის შემდეგ?

შეიძლება ითქვას, რომ ჩვენი საიტი **Site2** პრაქტიკულად მზად არის. ჩვენ შევქმენით მომხმარებლის გვერდები, რომელთაც ეკრანზე გამოაქვთ მონაცემთა ბაზებში არსებული ინფორმაცია. ჩვენ აგრეთვე შევქმენით ადმინისტრატიული ვებ-გვერდები, რომლებიც ახალი ჩანაწერების დამატების, არსებული ჩანაწერების ჩასწორების და გამოუსადეგარი ჩანაწერების წაშლის საშუალლებას იძლევიან. ჩვენ საკუთარი საძიებო მანქანაც კი შევქმენით, რითაც ვერ დაიტრაბახებს ბევრი საიტი. მაშ, კიდევ რა უნდა იყოს?

დიახ, ჩვენ ვიცით, თუ როგორ უნდა შევქმნათ უმარტივესი ვებ-საიტები სერვერული ვებ-გვერდების საფუძველზე. მაგრამ სწორედ უმარტივესი! ჩვენ შევქმენით საიტი, რომელიც უზრუნველყოფს ყველაზე ძირითადი ფუნქციების რეალიზაციას: მონაცემების შეტანა-გამოტანას და უმარტივეს ძიებას. სინამდვილეში კი ჩვენ ძალიან ბევრი რამ ჯერ კიდევ არ ვიცით. მათ შორის საიტის უსაფრთხოების უზრუნველყოფა.
ასე, რომ მომდევნო მე-3 და მე-4 ნაწილებში ჩვენ ძირითად ყურადღებას გავამახვილებთ საიტის უსაფრთხოებაზე.

დასკვნა

მაშ ასე, წიგნის პირველი და მეორე ნაწილები დასრულებულია!..

-ჩვენ შევისწავლეთ ვებ-გვრდების საწერი ენა **HTML**;
-ვისწავლეთ თუ როგორ მუშაობს **DreamweaverMX**-ი;
-ჩვენ შევქმენით ჩვენი სულ პირველი და სულ მარტივი საიტი;
-ჩვენ გავიგეთ მონაცემთა ბაზების, მონაცემთა სერვერების და **SQL** მოთხოვნების ფორმირების ენის შესახებ;
-ჩვენ ვისწავლეთ დაპროგრამება **PHP** ენაზე;
-**HTML**, **PHP**, **MySQL** და **DreamveaverMX**-ის ერთობლივი გამოყენებით, ჩვენ შევქმენით ჩვენი გაცილებით უფრო რთული საიტი;
-რაც ყველაზე მთავარია-**DreamveaverMX**-ის მიერ შექმნილი **PHP** კოდის შესწავლის გზით, ჩვენ გავარკვიეთ, თუ როგორ იწერება სერვერული ვებ-გვერდები. ჩვენ ხომ სწორედ ეს გვინდოდა.
აი, ამდენი რამის გაკეთება და გაგება შევძელით ჩვენ.
მაგრამ კიდევ ბევრი დაგვრჩა შესასწავლი.
ჩვენ სულ არ განგვიხილავს ვებ-დიზაინთან - "საიტის ლამაზად გაფორმებასთან" დაკავშირებული საკითხები. ამის გამო ჩვენი საიტები გამოიყურებიან არცთუ ისე პრეზენტაბელურად. (ჩვენ არც კი დაგვისახავს მიზნად ვებ-დიზაინის შესწავლა-ჩვენ სწორედ ვებ-პროგრამირებთ ვიყავით დაკავებული).

-ჩვენ არ გამოგვიყენებია **MySQL**-ის ბევრი შესაძლებლობა, რომელთაც შეუძლიათ პროგრამისტების ცხოვრების, მნიშვნელოვანი გამარტივება. მაგრამ ყველა ისინი აღწერილია **MySQL**-ის დოკუმენტაციაში, რომლის მოძიება შესაძლებელია საიტზე **http://www.mysql.com.** კარგ პროგრამისტს კი უნდა შეეძლოს დოკუმენტაციის კითხვა.
-ასევე ჩვენ არ გამოგვიყენებია **PHP**-ს მრავალი ახალი შესაძლებლობები. პირველყოვლისა, იგულისხმება ობიექტურად-ორიენტირებული დაპროგრამების გაფართოებული მოდელი, რომელიც ხშირ შემთხვევაში ძალიან ამარტივებს პროგრამისტების ამოცანებას. მაგრამ ყოველივე ეს აგრეთვე აღწერილია **PHP**-ს დოკუმენტაციაში.

-ჩვენ არც ჩვენი **PHP** სცენარების ოპტიმიზაციის, და არც მათი უსაფრთხოებისა და მდგრადობის უზრუნველყოფის საკითხები

განგვიხილავს სერიოზულად. რათქმა უნდა ზოგიერთი რამ ამ მიმართულებით ჩვენ მაინც გავაკეთეთ, მაგრამ უფრო მეტის გაკეთება იყო საჭირო, და ამ ხარვეზის აღმოფხვრას შევეცდებით წინამდებარე წიგნის მომდევნო გამოშვებაში.

-**DreamweaverMX**-თანაც არ გვიმუშავია საკმარისად. ვინაიდან ის ხომ უმძლავრესი პროგრამული პაკეტია, რომელიც შეიძლება ისწავლო ძალიან, ძალიან დიდ ხანს, და სულ უფრო და უფრო გაოცებული დარჩე იმ შესაძლებლობებით, რაც მასში ჩადებულია მისი შემქმნელების მიერ.
-საერთოდ, ჩვენ ჯერ-ჯერობით მწირი გამოცდილება გაგვაჩნია. მაგრამ გამოცდილება პრაქტიკის საქმეა და მას ჩვენ აუცილებლად შევიძენთ.

სწორედ ამიტომ, საუბარი **MySQL**-ის, **PHP**-ს და **DreamweaverMX**-ის შესახებ ჯერ არ დასრულებულა. თუ ჩვენ გვსურს მისი გაგრძელება, ჩვენ სულ სხვა, უფრო სრული, პროფესიონალებზე ორიენტირებული ლიტერატურის წაკითხვა მოგვიწევს. ეს წიგნი იძლევა მხოლოდ ვებ-პროგრამირების შესწავლისათვის აუცილებელ საწყის ბიძგს, საწყის ცოდნას, პირველ გამოცდილებას.
ვებ-დაპროგრამების სწავლაში ჩვენ დაგვეხმარება არა მხოლოდ წიგნები, რომლებიც ჯერ სადმე უნდა მოვიძიოთ და მერე როგორმე უნდა შევიძინოთ, არამედ ინტერნეტის უფასო რესურსებიც. მე-3 ცხრილში ჩამოთვლილია ჰიპერმინიშნებები ზოგიერთ ასეთ რესურსებზე.

ცხრილი 3.1. ჰიპერმინიშნებები წიგნის თემატიკასთან დაკავშირებული ვებ-საიტებზე.

ჰიპერმინიშნება	აღწერილობა
http://www.macromedia.com	Dreamweaver-ის შემმუშავებლის-ფირმა Macromedia-ს საიტი. ამ ფირმის მიერ წარმოებული ყველა პროგრამის (მათ შორის თვით DreamweaverMX-ის) დისტრიბუტივები, დოკუმენტაცია, სასარგებლო

	რჩევები,გაფორთოების მოდულები, შემმუშავებლების ფორუმი.
http://www.phpmyadmin.net	phpMyAdmini-ის შემმუშავებელი ჯგუფის საიტი. დისტრიბუტივები, დოკუმენტაცია, სასარგებლო რჩევები, ამ პროგრამის შემმუშავებლებთან მიერთების მოწოდებები.
http://pear.php.net	PHP ენაზე დაწერილი კარგად დოკუმენტირებული მაგალითების, გამზადებული ფუნქციების და კლასების უზარმაზარი ბიბლიოთეკა. რეკომენდირებულია ყველა სერიოზული PHP-პროგრამისტებისათვის.
http://www.phpclub.ru	PHP-პროგრამისტების რუსული ჯგუფის საიტი. სტატიები, კოდების გამზადებული ფრაგმენტები, ღონისძიებების ანონსები, შემმუშავებლების ფრუმი.
http://www.phpinside.ru	ბრწყინვალე უფასო ელექტრონული ჟურნალი PHP-პროგრამირებაზე. ვრცელდება PDF ფორმატში (Portable Document Format-გადასატანი დოკუმენტების ფორმატი)
http://www.php.spb.ru	პეტერბურგელი PHP პროგრამისტების ჯგუფის საიტი. მრავალი ძალზე სასარგებლო სტატიები და რჩევები.
http://www.subscribe.ru	მასიური გზავნილების ძალზე

	პოპულარული, რუსული სამსახურის საიტი
Fido7.ru.php,	PHP-დაპროგრამებასთან დაკავშირებული სიახლეების ჯგუფი

დანართები

დანართი-1

ვებ-სერვერ Apache-ის დაყენება და გამართვა

Apache-პოპულარული უფასო ვებ-სერვერია, რომლის დისტრიბუტივი შეიძლება მოვიძიოთ Apache Group-ის საიტზე (**http://httpd.apache.org**)-ორგანიზაციის, რომელიც დაკავებულია მისი შემუშავებით, თანხლებით და გავრცელებით. ამ საიტზე ხელმისაწვდომია **Apache**-ის სხვადასხვა ვერსიები სხვადასხვა ოპერაციული სისტემებისათვის.

დაყენების პროცესი აღწერილია **Windows**-ის ოპერაციული სისტემებისათვის განკუთვნილი 2.0.55 ვერსიის მაგალითზე. დისტრიბუტივის მიწოდება ხდება **Microsoft Installer** (**msi** გაფართოების მქონე ფაილები) ფორმატში, ამიტომ **Windows 95, 98, Me** და **NT**-ს მომხმარებლებმა **http://www.microsoft.com** საიტიდან უნდა გადმოტვირთონ და თავიანთ კომპიუტერებზე უნდა დააყენონ შესრულებადი გარემო **Microsoft Installer**.

დაყენება

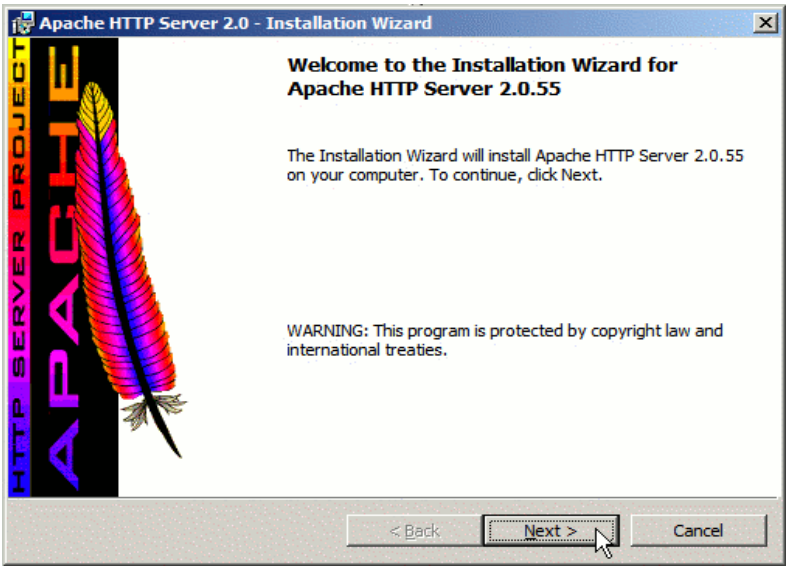
ფაილის **apache_2.0.55-win32-x86-no_ssl.msi** შესრულებაზე გაშვების შემდეგ ეკრანზე გაჩნდება შემოთავაზება **Apache**-ის დაყენებაზე (ნახ. დ1.1). ვაჭერთ ღილაკს **Next**.

ეკრანზე გამოჩენის რიგითობის მიხედვით, მომდევნო ფანჯარა-სალიცენზიო შეთანხმებაა (დ1.2). ვეთანხმებით მას გადამრთველის **I accept the terms in the license agreement** ჩართვით და ვაჭერთ ღილაკს **Next**.

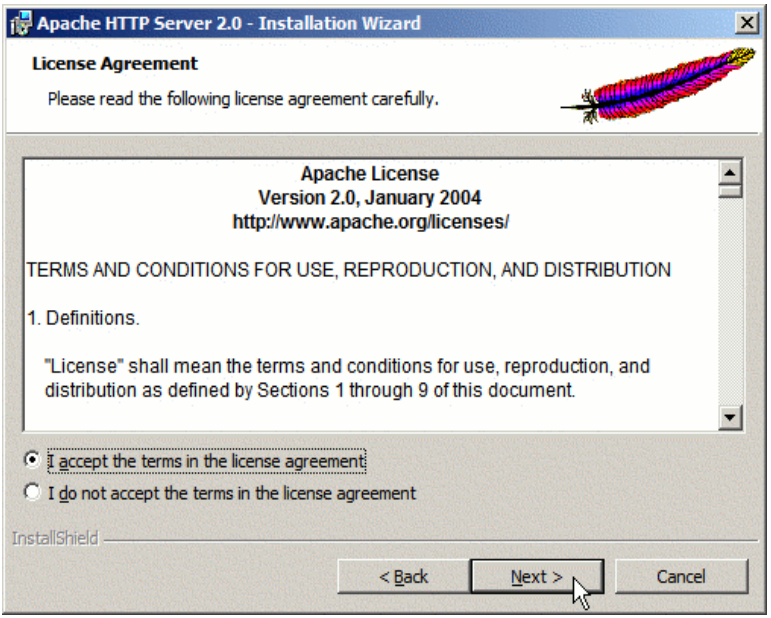
შემდეგ ჩვენ ვხედავთ სერვერის პარამეტრების განსაზღვრის ფანჯარას (დ1.4). ველში **Network Domain** შეგვყავს დომენი, რომელშიც ჩვენი კომპიუტერი მდებარეობს, ველში **Server Name**-ჩვენი კომპიუტერის დომენური სახელი, ხოლო ველში **Administrator's**

Email Address-ადმინისტრატორის საფოსტო მისამართი (ანუ ჩვენი მისამართი).

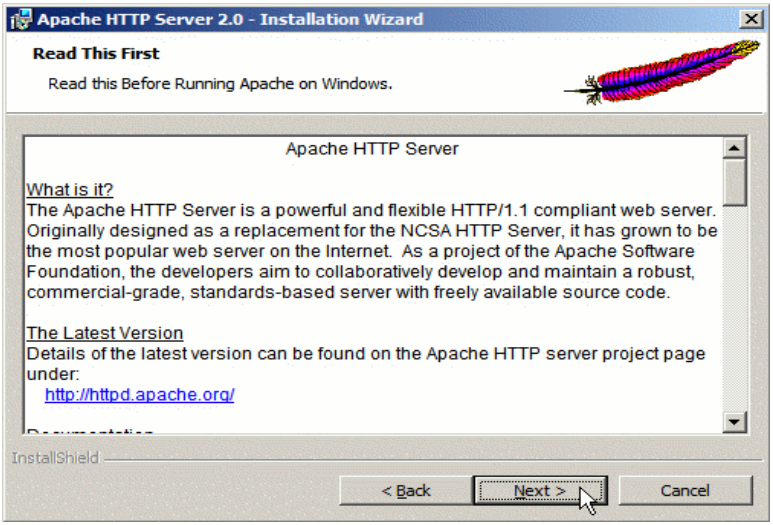
Windows NT, 2000, XP და **2003**-ის შემთხვევაში სერვერის დაყენებისას ამ ფანჯარაში წარმოდგენილი იქნება გადამრთველების ჯგუფი **Install Apache HTTP Server 2.0 programs and shortcuts for**. ვინაიდან სერვერი ჩვენ მხოლოდ ჩვენი საიტის ტესტირებისათვის გვჭირდება, ჩავრთოთ ალამი **only for the Current User, on Port8080, when started Manually**. ამის შემდეგა **Apache** დაყენებული იქნება არა როგორც მრავალმომხმარებლიანი სისტემა, არამედ მხოლოდ მიმდინარე მომხმარებლისათვის, იგი მორგებული იქნება პორტზე 8080, ხოლო მის გაშვებას განვახორციელებთ ხელით.



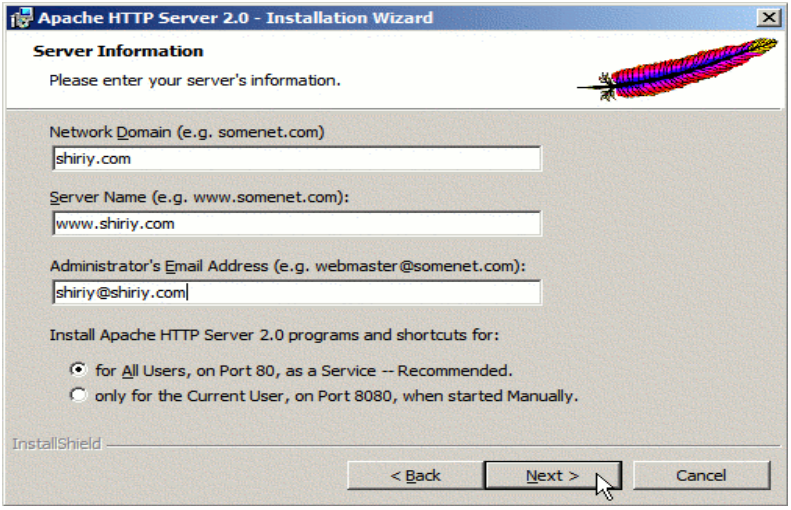
ნახ.დ1.1. Apache-ის დაყენების შემოთავაზება



ნახ.დ1.2.სალიცენზიო შეთანხმება



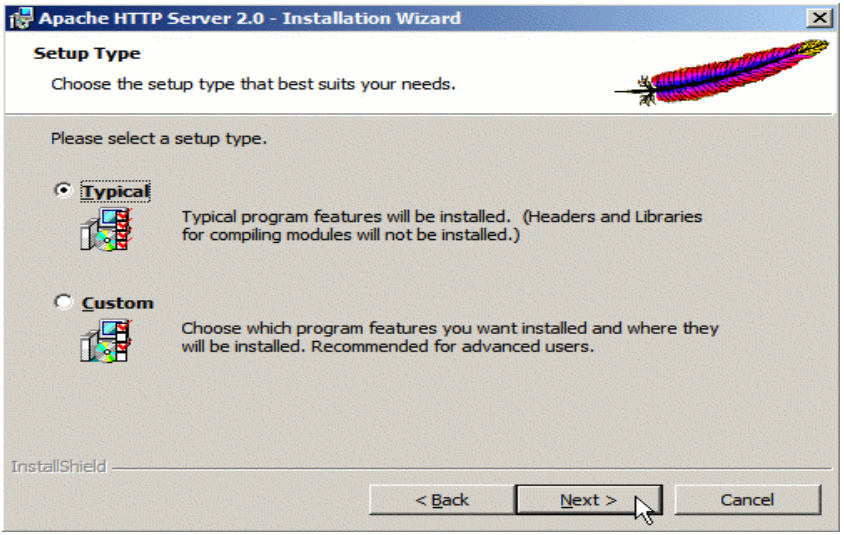
ნახ. დ1.3. ცნობები Apache სერვერის შესახებ



ნახ.დ1.4. სერვერის პარამეტრების განსაზღვრის ფანჯარა

შენიშვნა: *ალამი For All Users, on Port 80, as a Service უნდა ჩაირთოს მხოლოდ მაშინ, თუ ჩვენ ვაპირებთ, რომ ჩვენი საიტი გავხადოთ მუდმივმომქმედი და ქსელის ყველა მომხმარებლისათვის ხელმისაწვდომი. ამ შემთხვევაში Apache-ი გაიშვება ოპერაციულ სისტემასთან ერთად და მიერთებული იქნება პორტ 80-თან (WWW-სათვის წინასწარ განსაზღვრული პორტი).*

გამართვის დასრულებისთანავე, ვაჭერთ ღილაკს **Next**. შემდეგ ეკრანზე გამოჩნდება გამართვის ტიპის(ჩვეულებრივი თუ მომხმარებლის მიერ არჩეული) განმსაზღვრელი ფანჯარა (ნახ. დ1.5.). ჩავრთოთ გადამრთველი **Typical,** რომელიც განსაზღვრავს ჩვეულებრივი (ტიპიურ)სახის გამართვას-ჩვენი მიზნებისათვის ის სრულიად საკმარისი იქნება-და შემდეგ ვაჭერთ ღილაკს **Next**.

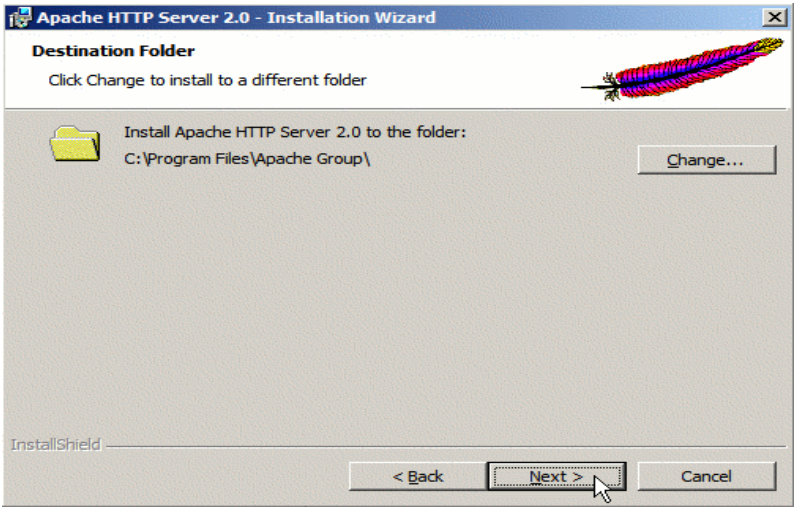


ნახ.დ1.5. გაწყობის ტიპის განმსაზღვრელი ფანჯარა

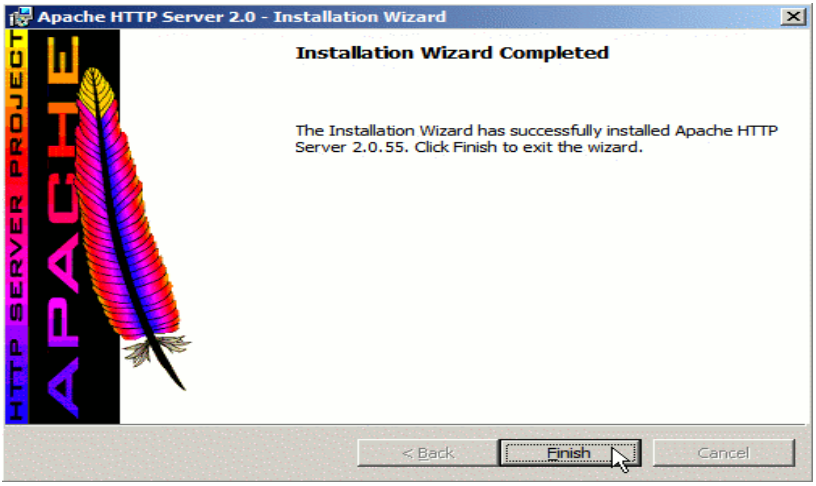
შემდეგ, ჩვენ დავინახავთ იმ საქაღალდის განმსაზღვრელ ფანჯარას, რომელშიც **Apache**-ი იქნება დაყენებული (ნახ.დ1.6). წინასწარი განსაზღვრით ეს საქაღალდე იქნება **Apache Group**-ი, რომელიც მოთავსებულია სისტემურ საქაღალდეში **Program Files**. როგორც წესი არ ჩნდება ამ საქაღალდეს შეცვლის რაიმე განსაკუთრებული აუცილებლობა, ამიტომ ვაჭერთ ღილაკს **Next**.

ბოლო ფანჯარა, რომელიც ეკრანზე გაჩნდება, გვამცნობს იმის შესახებ, რომ იწყება დაყენების პროცესი. დავაჭიროთ ღილაკს **Next** და დაველოდოთ, სანამ ის არ დამთავრდება.

ნიშანი იმისა, რომ სერვერიოს დაყენება დასრულებულია, იქნება ფანჯრის გახსნა, რომელიც ნაჩვენებია ნახ. დ1.7.-ზე. ამ ფანჯრის დასახურად დავაჭიროთ ღილაკს **Finish**.



ნახ.დ.1.6. სერვერის დასაყენებელი საქაღალდეს განმსაზღვრელი ფანჯარა.

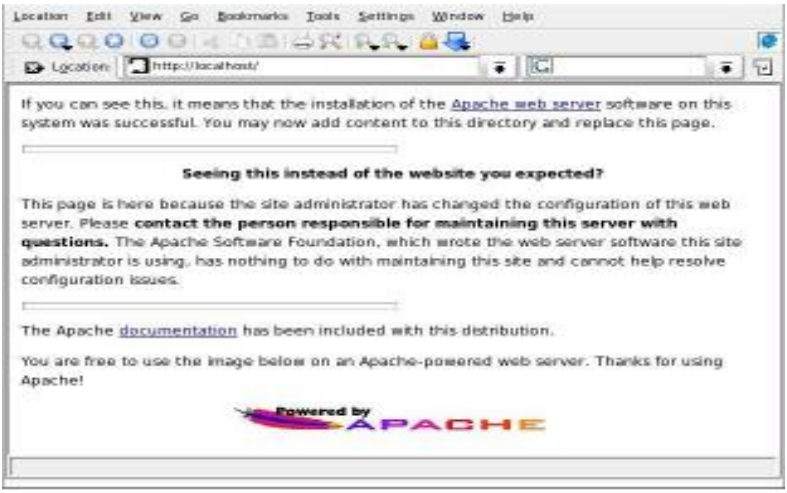


ნახდ1.7. ფანჯარა, რომელიც გვამცნობს, დაყენების პროცესის დასრულების შესაებ.

გაშვება და გაჩერება

Apache-ის გასაშვებად დავაჭიროთ ჩვენთვის კარგად ნაცნობ ღილაკს **Start** (გაშვება), მენიუში, რომელიც ამ დროს გაიხსნება ავირჩიოთ პუნქტი **Apache HTTP Server 2.0.55**, ეკრანზე გამოსულ ქვემენიუში ავირჩიოთ პუნქტი **Control Apache Server**, ხოლო მეორე ქვემენიუში-პუნქტი **Start Apach in Console**. ეკრანზე გამოჩნდება **MS-DOS** სეანსის ფანჯარა (**Windows 95, 98** და **ME**-ს შემთხვევაში) ან კონსოლის ფანჯარა (**Windows NT**, **2000, XP** ან **2003**-ის შემთხვევაში) სათაურით **Start Apache in Console**. ამით სერვერი გაშვებულია.

ახლა მოდით შევამოწმოთ ახლახანს დაყენებული ვებ-სერვერი მოქმედებაში. გავუშვათ ვებ-დამთვალიერებელი და მისამართის სტრიქონში ავირჩიოთ **http://localhost:8080/**. საპასუხოდ ვებ-დამთვალიერებელი გამოიყვანს მცირე ზომის ვებ-გვერდს, რომელიც გვამცნობს, რომ სერვერი ნორმალურად არის დაყენებული და ნორმალურად მუშაობს (ნახ.დ1.8.).



თუ თქვენ ამას ხედავთ, ეს ნიშნავს იმას, რომ ვებსერვერიApache-ის დაყენება ამ სისტემაძე დასრულდა წარმატებით.თქვენ უკვე შეგიძლიათ ამ დირექტორიაში შიგთავსის დამატება ამ გვერდის შეცვლა.

თქვენ ხედავთ ამას მოსალოდნელი გვერდის ნაცვლად?

ეს გვერდი აქ იმიტომ არის, რომ სისტემის ადმინისტრატორმა შეცვალა ამ ვებ-სერვერის კონფიგურაცია. დაუკავშირდით პიროვნებას, რომელიც პასუხისმგებელია ამ სერვერის მუშაობაზე, თქვენს კითხვებზე პასუხის მისაღებად. Apache Software Foundation-ვებსერვერის პროგრამული უზრუნველყოფის ავტორი, რომლითაც სარგებლობს ამ სისტემის ადმინისტრატორი არ არის დაკავშირებული ამ სისტემის მხარდაჭერასთან და მას არ შეუძლია დახმარება გაგიწიოთ კონფიგურაციასთან დაკავშირებული პრობლემების გადაწყვეტაში.

ვებ-სერვერ Apache-ის დოკუმენტაცია თან ერთვის პროგრამული უზრუნველყოფის კომპლექტს.
თქვენ თავისუფლად შეგიძლიათ გამოიყენოთ ქვემოთ ვებ-სერვერზე განთავსებული პიქტოგრამა. გმადლობთ Apache-ის გამოყენებისათვის.

ნახ.დ1.8. ვებ-გვერდი, რომელიც გვამცნობს ვებ-სერვერის ნორმალური მუშაობის შესახებ

ვებ-სერვერის გასაჩერებლად, პირველ რიგში უნდა დავხუროთ ვებ-დამთვალიერებლის ყველა ფანჯარა, რომელშიც ამ სერვერის გვერდებია გახსნილი, ხოლო შემდეგ დავხუროთ ფანჯარა **Start Apache in Console**. რამოდენიმე წამში ის გაქრება-სერვერი გაჩერდება.

წინასწარი გამართვა

პრინციპში **Apache**-ი არ მოითხოვს დამატებით გამართვას დაყენების შემდეგ, ვინაიდან ყველა საჭირო პარამეტრის მნიშვნელობა ჩვენ

უკვე განვსაზღვრეთ პარამეტრების განსაზღვრის ფანჯარაში (იხ.ნახ.დ1.4.), სერვერის დაყენების დროს. მაგრამ რაღაცა მინიმალური გაწყობა მაინც დაგვჭირდება.
მანამ **Apache**-ის გაწყობას მოვახდენდეთ, გავაჩეროთ იგი, თუ გაშვებული იყო. შემდეგ გავუშვათ **Windows**-ის გამცილებელი ან ფაილების მართვის სხვა პროგრამა და გავხსნათ საქაღალდე **Conf**, რომელიც მოთავსებულია იმავე საქაღალდეში, რომელშიც **Apache**-ი იყო დაყენებული. ნებისმიერ ტექსტურ რედაქტორში, მაგალითად **Blocknot**-ში გავხსნათ ფაილი **httpd.conf** და მასში მოვძებნოთ ასეთი სტრიქონი:

DirectoryIndex index.html index.html.var

ეს სტრიქონი განსაზღვრავს იმ ფაილის სახელს, რომელშიც ინახება გვერდი გაჩუმებით. ჩანს, რომ გაჩუმებითი სახელია **index.html**. მოდით დავამატოთ აქ გვერდების გაჩუმებითი სახელები, რომლებიც ამ წიგნშია გამოყენებული-**default.htm** და **default.php**. აქვე დაგვჭირდება კიდევ ერთი სახელის დამატება-index.php, რათა ნორმალურად იმუშაოს **phpMyAdmin**-მა. გადავწეროთ ეს სტრიქონი ასე:

DirectoryIndex default.htm default.php index.php index.html index.html.var

ამის შემდეგ Apache-ს უნდა დაემატოს PHP-ს მხარდამჭერი პროგრამა. ამისათვის httpd.conf ფაილის ნებისმიერ ადგილას დავამატოთ ასეთი სტრიქონები:

LoadModule php4_modul <ბილიკი საქაღალდემდე, სადაც დაყენებულია PHP>/php4apache.dll
AddType application/x-httpd-php.php

შენიშვნა: *Apache-ში (და ვებ-სერვერების სხვა პროგრამებშიც) PHP-ს მხარდამჭერი პროგრამების დამატება აღწერილია ფაილში install.txt, რომელიც მდებარეობს იმავე საქაღალდეში, რაც PHP.*

დაბოლოს, httpd.conf ფაილში მოვძებნოთ ასეთი სტრიქონები:

<Directory "C:/Program Files/Apache Group/Apache2/htdocs'>
Options Indexes FollowSymLinks
AllowOverride None
Order allow, deny
Allow from all
</Directory>

მივაქციოთ ყურადრება იმას, რომ ეს სტრიქონები გაზავებულია კომენტარებით (**Apache**-ის კონფიგურაციის ფაილში კომენტარები იწყება ასეთი ნიშნით #).
ჩვენ მოგვიწევს ზემოდან მესამე სტრიქონის შეცვლა, რათა ის გამოიყურებოდეს ასე (გამოყოფილია მუქი ფერით):

<Directory "C:/Program Files/Apache Group/Apache2/htdocs'>
Options Indexes FollowSymLinks
AllowOverride All
Order allow, deny
Allow from all
</Directory>

ეს აუცილებელია იმისათვის, რომ **Apache**-მა შეძლოს **.htaccess** ფაილების დამუშავება, რომლებიც ძირეულ საქაღალდეში ინახება. წინასწარი განსაზღვრით ის მათ უგულვებელყოფს.

ესეც ასე, დასრულდა. ახლა დავიმახსოვროთ ფაილი **httpd.conf** და დავხუროთ იგი. შემდეგი გაშვებისას **Apache**-ი გამოიყენებს პარამეტრების ჩვენს მიერ განსაზღვრულ მნიშვნელობებს.

ყურადღება: *სანამ რაიმე პარამეტრს შევცვლიდეთ Apache-ის კონფიგურაციის ფაილში httpd.conf, ის უნდა გავაჩეროთ. Apache-ი კითხულობს ფაილს httpd.conf გაშვებისას და მუშაობის პროცესში მას არ მიმართავს.*

Apache-ის საცნობარო სისტემაში შეღწევა

Apache-ს თან ახლავს საკმაოდ დაწვრილებითი საცნობარო სისტემა, რომელშიც აღწერილია მისი შესაძლებლობები და გამართვის ხერხები. მასში შეღწევისათვის საჭიროა;
1.თვით ვებ-სერვერის **Apache**-ის გაშვება;
2.ვებ-დამთვალიერებლის გაშვება;
3.ვებ-დამთვალიერებლის მისამართის სტრიქონში **http://localhost:8080/manual/** არჩევა.
სამწუხაროდ **Apache**-ის საცნობარო სისტემის სრული ვარიანტი ხელმისაწვდომია ინგლისურ ენაზე.

დანართი -2

MySQL მონაცემთა სერვერის დაყენება და გამართვა

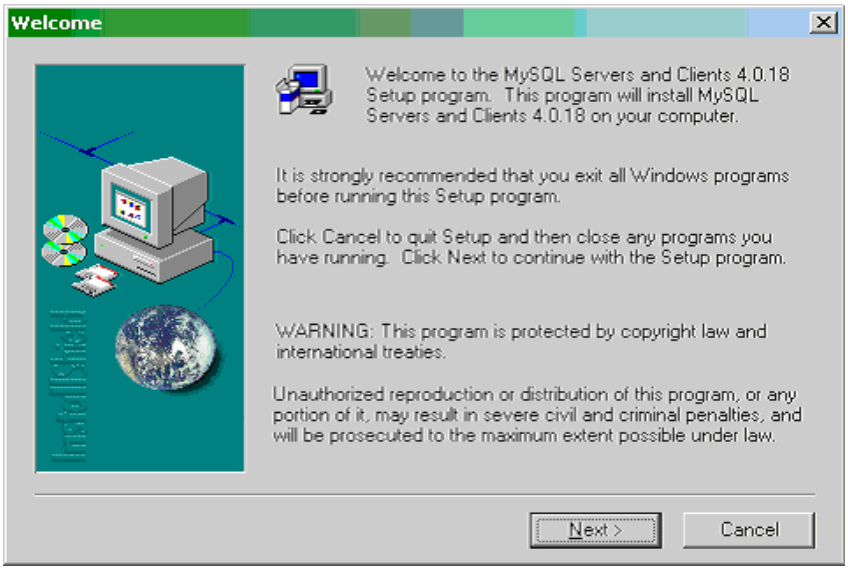
MySQL-პოპულარული და ამავე დროს უფასო სერვერია, რომლის დისტრიბუტივის მოძიება შესაძლებელია საიტზე **MySQL AB**-ს-იმ ორგანიზაციის საიტზე (**http://www.mysql.com**), რომელიც დაკავებულია **MySQL**-ის დამუშავებით, თანხლებით და გავრცელებით. ამ საიტზე ხელმისაწვდომია **MySQL**-ის სხვადასხვა ვერსიები, სხვადასხვა ოპერაციული სისტემებისათვის.

დაყენების პროცესი აღწერილია, **Windows** ოპერაციული სისტემისთვის განკუთვნილი 4.0.18 ვერსიის მაგალითზე. დისტრიბუტივის მიწოდება ხდება **ZIP** არხივის სახით, რომლის განპაკება(ამოლაგება) შესაძლებელია ნებისმიერ საქაღალდეში.

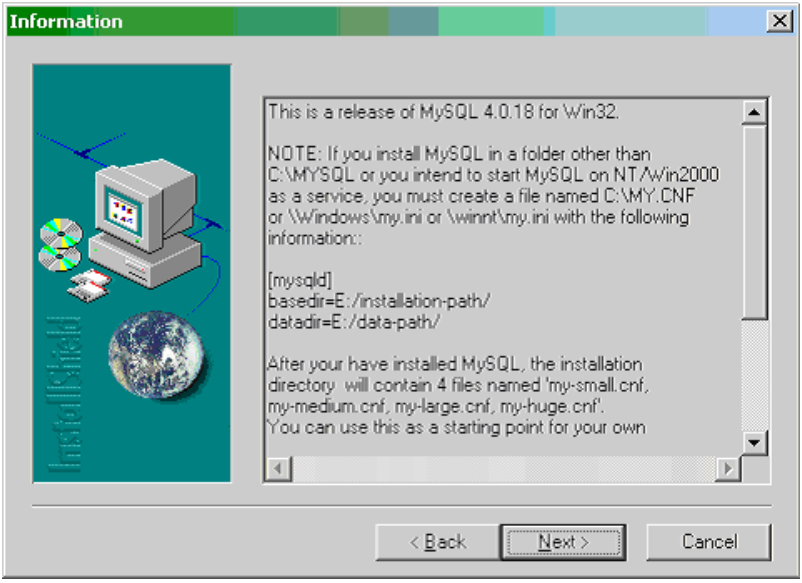
დაყენება

setup.exe ფაილის, შესრულებაზე გაშვების შემდეგ, ეკრანზე გამოჩნდება **MySQL**-ის დაყენებაზე მიწვევის ფანჯარა (ნახ. დ.2.1). დავაჭიროთ ამ ფანჯრის **Next** ღილაკზე.
ამის მერე, გამოჩნდება ფანჯარა, რომელიც შეიცავს სერვერის გამართვასთან (გაწყობასთან) დაკავშირებულ ცნობებს (ნახ. დ.2.2). ვკითხულობთ მათ და ისევ ვაჭერთ ღილაკს **Next**.
შემდეგი ფანჯრის დანიშნულება მდგომარეობს იმ საქაღალდეს განსაზღვრაში, რომელშიც უნდა მოვათავსოთ **MySQL** (ნახ. დ.2.3). გაჩუმებით ამ საქაღალდეს სახელია **mysql**, რომელიც დისკო C-ს ძირეულ საქაღალდეშია განთავსებული. რათქმა უნდა შეიძლება, რომ დავტოვოთ იგი უცვლელად, მაგარამ კარგი ტონის წესების მიხედვით, **Windows**-ის ყველ პროგრამა უნდა იყოს დაყენებული

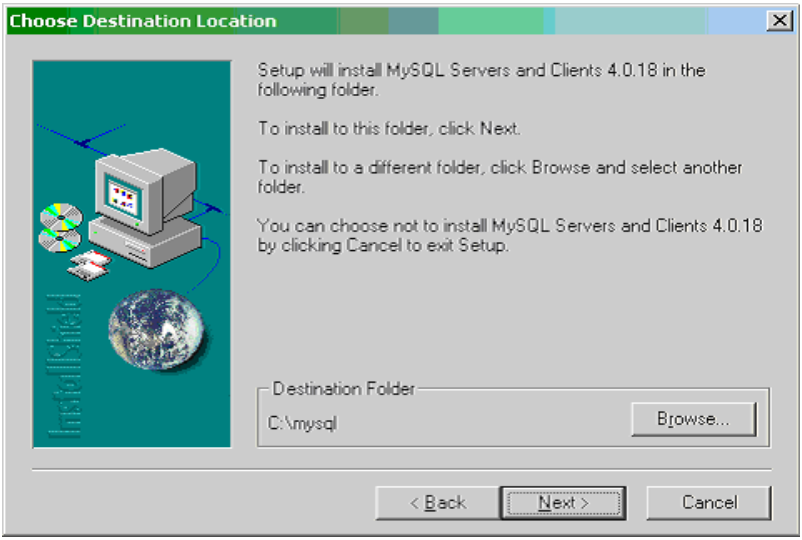
Program Files საქაღალდეში. ასე, რომ ჯობია შევცვალოთ სერვერის დასაყენებელი საქაღალდის მდებარეობა.
დავაჭიროთ ღილაკს Browse. ეკრანზე გაჩნდება მცირე ზომის დიალოგური ფანჯარა **Choose Folder**, რომელიც ნაჩვენებია ნახ. დ.2.4-ზე. ამ ფანჯრის შესატან ველში **Path** შევიტანოთ ბილიკი ფაილამდე **C:\Program Files\mysgl** და დავაჭიროთ ღილაკს **OK**. ისევ აღმოვჩნდებით დაყენების ბილიკის განმსაზღვრელ დიალოგურ ფანჯარაში (იხ. ნახ. დ.2.3), სადაც ვაჭერთ ღილაკს **Next**.
ამის შემდეგ ჩვენ დავინახავთ დაყენების ტიპის განსაზღვრის ფანჯარას: ტიპიური, კომპაქტური და არჩევითი (ნახ. დ.2.5). ჩავრთოთ გადამრთველი **Typical**, რომელიც განსაზღვრავს ტიპიურ დაყენებას-ის ჩვენი მიზნებისათვის სრულიად საკმარისია-და დავაჭიროთ ღილაკს **Next**.



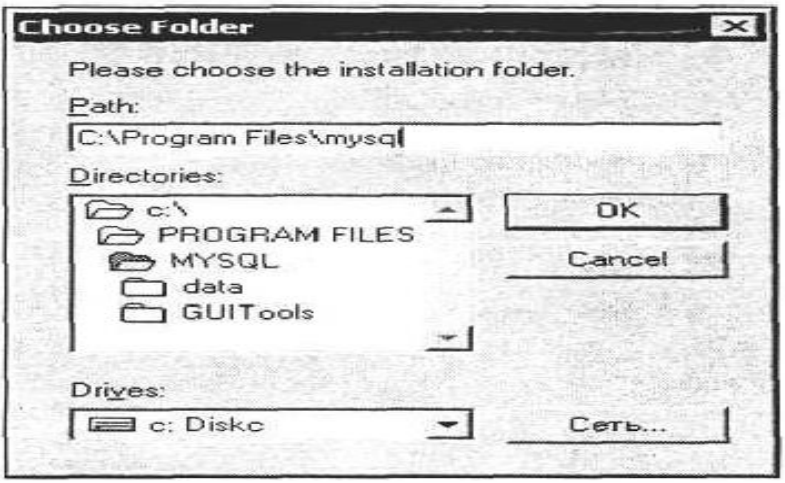
ნახ.დ.2.1. მიწვევა MySQL-ის დაყენებაზე.



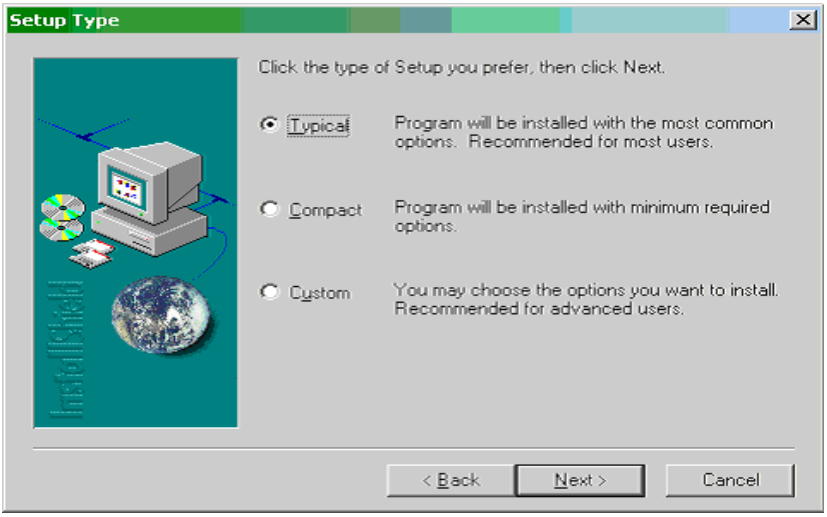
ნახ. დ.2.2. ცნობები გამართვის შესახებ



ნახ. დ.2.3. იმ საქაღალდეს განსაზღვრის ფანჯარა, რომელშიც სერვერი უნდა დადგეს.



ნახ. დ.2.4. დიალოგური ფანჯარა Choose Folder



ნახ. დ.2.5. დაყენების ტიპის განსაზღვრის ფანჯარა.

ამის შემდეგ, იმწამსვე დაიწყება თვით დაყენების პროცესი. დაველოდოთ მის დასრულებას და დავჭიროთ ეკრანზე გამოსული

უკანასკნელი ფანჯრის **Finish** ღილაკზე (ნახ. დ.2.6). ამით მონაცემთა სერვერის **MySQL** დაყენება დასრულებულია.

ნახ. დ.2.6. დაყენების დასრულების შესახებ შეტყობინების ფანჯარა.

სერვერის დაყენებისას ავტომატურად შეიქმნება მონაცემთა ბაზები **mysql** და **test**. პირველი მონაცემთა ბაზა შეიცავს დამხმარე მონაცემებს-მონაცემთა ბაზების, ცხრილების, ველების, ინდექსების, მომხმარებლების და მათი უფლებების სიებს-ასე რომ, მათთვის ხელის ხლება არ ღირს. მონაცემთა მეორე ბაზა-წარმოადგენს მაგალითს, ამიტომ საჭიროების შემთხვევაში მისი წაშლა შესაძლებელია. აგრეთვე, პროგრამის დაყენების დასრულებისთანავე შეიქმნება სამი მომხმარებელი. ორი მათგანი მაშინათვე უნდა წავშალოთ და დავტოვოთ მხოლოდ მომხმარებელი-**root@localhost**-ადმინისტრატორი, რომელსაც გააჩნია ლოკალური კომპიუტერიდან მიერთების უფლება. სწორედ მისი სახელით, შევქმნით ჩვენ შემდგომში ჩვენს მონაცემთა ბაზას **site**-ს.

წინასწარი გამართვა

Apache-ისაგან განსხვავებით **MySQL**-ს, გაშვების წინ, ჭირდება წინასწარი გამართვა. თუ კი ის არ შესრულდება, სერვერმა შეიძლება იმუშაოს შეცდომებით ან სულაც არი იმუშაოს.
გავუშვათ **Windows** გამცილებელი ან ფაილების მართვის სხვა პროგრამა და გავხსნათ საქაღალდე, რომელშიც მოთავსებულია თვით **Windows** სისტემა (როგორც წესი ეს საქაღალდე მოთავსებულია C-დისკოს ძირეულ საქაღალდეში და ატარებს სახელს **Windows** ან **WinNT**). ამ საქაღალდეში შევქმნათ ტექსტური ფაილი **my.ini**, გავხსნათ იგი ნებისმიერ ტექსტურ რედაქტორში, მაგალითად "ბლოკნოტ"-ში და ჩავწეროთ ასეთი ტექსტი:

[mysql]
basedir=<ბილიკი MySQL-მდე>
datadir=<ბილიკი MySQL-მდე>/data
default-character-set=cp1251

ჩვენს შემთხვევაში my.ini-ფაილის შიგთავსი ასეთი იქნება:

[mysqld]
basedir=C:/Program Files/mysql
datadir=C:/Program Files/mysql/data
default-character-set=cp1251

ყურადღება მივაქციოთ იმას, რომ აქ საქაღალდეების ერთმანეთისაგან გამოყოფისათვის გამოიყენება ნიშანი (/) ("პირდაპირი სლეში"), **Windows**-ში მიღებული (\) ("შებრუნებული სლეშ"-ის) ნაცვლად. საქმე იმაშია, რომ **my.ini** ფაილის იგივე სინტაქსისი გამოიყენება **Unix**-ოჯახის ოპერაციული სისტემებისთვის განკუთვნილი, **MySQL**-ის ვერსიების გამართვისათვის.
შევამოწმოთ, ხომ არ დაგვიშვია რაიმე შეცდომა, შევინახოთ ფაილი **my.ini** და დავხუროთ იგი. ამით ჩვენ, სერვერის ყველა ძირითადი პარამეტრი განვსაზღვრეთ.
მოკლედ განვმარტოთ ის, რაც დავწერეთ ამას წინათ. გასაგებია, რომ პარამეტრი **basedir** განსაზღვრავს საქაღალდეს, რომელშიც მოთავსებულია სერვერი **MySQL**; იგი საჭიროა იმისთვის, რომ სერვერმა შეძლოს თავისი დამხმარე კომპონენტების მოძებნა. პარამეტრი **datadir** განსაზღვრავს საქაღალდეს, რომელშიც ინახება

ყველა მონაცემთა ბაზა; გაჩუმებით ეს არის საქაღალდე **data**, რომელიც იმავე საქაღალდეშია მოთავსებული, რაც **MySQL**.
ახლა კი-განსაკუთრებული **ყურადღება!** ეს მომენტი ძალიან ცუდად არის დოკუმენტირებული **MySQL**-ის სახელმძღვანელოში, მაგრამ მისი გათვალისწინება აუცილებელია. ეს ეხება სტრიქონს

default-character-set=cp1251

რომლის გარეშეც სერვერი პრინციპში იმუშავებს, მაგრამ არც თუ მთლად კორექტულად.
როდესაც მონაცემთა სერვერი (ნებისმიერი-არა მხოლოდ **MySQL**) ასრულებს ჩანაწერების დახარისხებას სიმბოლური (სტრიქონული) ტიპის ველის მიხედვით, ის იყენებს დახარისხების იმ წესს, რომელიც მიღებულია მის პარამეტრებში განსაზღვრული ენისათვის. მაშ ასე, გამართვის ფაილი **my.ini** ჩვენ შევქმენით. ამის შემდეგ შეიძლება მონაცემთა სერვერის **MySQL** გაშვება.

MySQL-ის გაშვება და გაჩერება

MySQL სერვერის გაშვება და გაჩერება იმისდა მიხედვით, თუ **Windows**-ის რომელ ვერსიაშია იგი დაყენებული სხვადასხვანაირად ხორციელდება. ასე რომ, ჩვენ განვიხილავთ ამ პროცესს **Windows 95, 98, Me, Windows NT** და **Windows 2000, XP, 2003**-სთვის ცალცალკე.

MySQL-ის გაშვება და გაჩერება Windows 95, 98 და Me-ს გარემოში.

MySQL-ის გასაშვებად **Windows-ის 95, 98** და **Me** ვერსიებში საჭიროა **MS-DOS**-ის საკომანდე სტრიქონის გამოყენება-ეს მოკრძალებული, მაგრამ სასარგებლო უტილიტა შედის სისტემის კომპლექტაციაში. გავუშვათ იგი და მასში ყოველი სტრიქონის შემდეგ კლავიშა **<Enter>**-ის დაჭერით, ავკრიფოთ ასეთი ტექსტი:

```
cd \
cd Program Files
cd mysql
cd bin
```

ეს ბრძანებები ახორციელებენ **bin** საქაღალდეში გადასვლას, რომელიც იმავე საქაღალდეში მდებარეობს, რაც თვითონ სერვერი.

ამის შემდეგ უნდა ავკრიფოთ ასეთი ბრძანება (აქაც არ უნდა დაგვავიწყდეს მისი შეტანის შემდეგ კლავიშა **<Enter>**-ზე დაჭერა):

mysqld--standalone

სწორედ ის გაუშვებს ჩვენს სერვერს.
მხედველობაში უნდა ვიქონიოთ, რომ სერვერის გაშვებისას, ეკრანზე არანაირი ფანჯარა არ გამოჩნდება. **MySQL** მუშაობს "ჩუმად".
იმისათვის, რომ გავაჩეროთ **MySQL**, საჭიროა საკომანდე სტრიქონის გამოყენება. ისევ ავკრიფოთ საქაღალდე **bin**-ში გადსვლის, ადრე განხილული ოთხი ბრძანება, ხოლო მათ შემდეგ-ასეთი ბრძანება:

mysqladmin –u root shutdown

სერვერი იმ წუთშივე გაჩერდება და ამოიტვირთება მეხსიერებიდან.

შენიშვნა: *ადრე განხილულ, სერვერის გაჩერების ბრძანებაში, გამოიყენებოდა უტილიტა MySQLAdmin. ეს MySQL-ის კომპლექტაციაში შემავალი სერვისული უტილიტაა, რომელიც სერვერის ზოგიერთი პარამეტრის გაწყობის, მისი გაშვებისა და გაჩერებისთვის გამოიყენება. აგრეთვე მისი საშუალებით ხდება გახსნილი მონაცემთა ბაზების და ჩართული მომხმარებლების შესახებ ინფორმაციის დათვალიერება .*

MySQL-ის გაშვება და გაჩერება Windows NT-ს გარემოში

Windows NT-ს გარემოში **MySQL** შეიძლება გაშვებულ იქნას, როგორც სერვისი. **სერვისი**-ეს **Windows**-ის განსაკუთრებული პროგრამაა, რომელიც საერთოდ არ ურთიერთქმედებს მომხმარებელთან და დაკავებულია მხოლოდ იმით, რომ ემსახურება სხვა პროგრამებს.

ყურადღება! *სერვისების მხარდაჭერა ხდება მხოლოდ Windows NT, 2000, XP და 2003 სისტემების მიერ. Windows 95,98 და Me-ში ჯობია, რომ სერვისები არ გავუშვათ-ისინი მუშაობენ არასწორად.*

ნებისმიერი სერვისი გაშვების წინ სწორად უნდა იქნას დაყენებული. და ამის გაკეთება ჩვენ თვითონ მოგვიწევს. გავუშვათ სისტემის სტანდარტულ კომპლექტაციაში შემავალი უტილიტა "საკომანდო

სტრიქონი" და ავკრიფოთ მასში ოთხი, ჩვენთვის უკვე ნაცნობი და ერთი უცნობი ბრძანება (არ დაგვავიწყდეს თითოეული ბრძანების შეტანის შემდეგ **<Enter>** კლავიშაზე დაჭერა):

**cd **
cd Program Files
cd mysql
cd bin
mysqld-max-net --install

სწორად უკანასკნელი-უცნობი ბრძანება, დააყენებს სისტემაში **MySQL**-ს როგორც სერვისს.
სერვისის სახით დაყენებული **MySQL**-ის გაშვება შესაძლებელა ორნაირად.
თუ ჩვენ ჯერ კიდევ არა გვაქვს დახურული საკომანდო სტრიქონი, მაშინ შეგვიძლია მასში ასეთი ბრძანების აკრეფა:

net start MySQL

ეს არის პირველი ხერხი. მეორე ხერხი-ეს არის მართვის პანელის სპეციალური აპლეტის **სერვისები (Services)** გამოყენება. გავხსნათ მართვის პანელი, ორჯერ დავაწკაპუნოთ ნიშნაკზე **სერვისები (Services)** და ეკრანზე გამოჩნდება ფანჯარა **სერვისები (Services)**-აპლეტი გაშვებულია. დაყენებული სერვისების სიაში ავირჩიოთ პუნქტი **MySQL** და დავაწკაპუნოთ გაშვების ღილაკზე. ამის შემდეგ შეიძლება აპლეტის და თვით მართვის პანელის დახურვა.
MySQL-ის გაჩერება ასევე ორი ხერხით არის შესაძლებელი. **პირველი**-შეიძლება გავუშვათ საკომანდე სტრიქონი და მასში ავკრიფოთ ბრძანება: **net stop MySQL**
მეორე-შეიძლება გავუშვათ აპლეტი **სერვისები (Services),** სიაში ავირჩიოთ **MySQL** და დავაწკაპუნოთ სერვისის გაჩერების ღილაკზე.

MySQL-ის გაშვება და გაჩერება Windows 2000, XP და 2003 გარემოში.

Windows 2000, XP და **2003** სისტემებში **MySQL**-ის, როგორც სერვისის დაყენება ისეთნაირადვე ხორციელდება, როგორც **Windows NT**-ში. ზუსტად ისევე, საკომანდო სტრიქონის საშუალებით ხორციელდება მათში **MySQL**-ის სერვისების გაშვება და გაჩერება. ერთადერთი განსხვავება: **Windows**-ის ამ ვერსიებში სერვისებთან

სამუშაოდ გამოიყენება არა მართვის პანელის აპლეტი, არამედ მართვის სისტემის - **Microsoft Management Consol**, სპეციალური აღჭურვილობა-**Службы-Services-მომსახურებები**.
გავხსნათ მართვის პანელი, ორჯერ დავაწკაპუნოთ ნიშნაკზე **Administration** (ადმინისტრირება) და ეკრანზე გამოსულ ფანჯარაში **Administration**(ადმინისტრირება) ორჯერ დავაწკაპუნოთ ნიშნაკზე-Services(სამსახურები). გაიშვება იმავე დასახელების აღჭურვილობა და ეკრანზე მისი ფანჯარა გაჩნდება. აქ ისევე, როგორც **Windows NT**-ში, აპლეტ-**Services** (სამსახურები)-ის შემთხვევაში, დაყენებული სერვისების სიაში ავირჩიოთ **MySQL** და დავაჭიროთ გაშვების ღილაკს.
თუ ჩვენ დაგვჭირდება **MySQL**-ის გაჩერება, ისევ უნდა გავუშვათ აღჭურვილობა **Services** (სამსახურები), შესაბამის სიაში ავირჩიოთ **MySQL** და დავაჭიროთ სერვისის გაჩერების ღილაკს.

MySQL-ის საცნობარო სისტემის გამოყენება

MySQL-ის სტანდარტული კომპლექტაცია შეიცავს სრულ საცნობარო სახელმძღვანელოს ინგლისურ ენაზე. მის გამოსაყენებლად, უნდა მოვძებნოთ საქაღალდე Docs, რომელიც მოთავსებულია იმავე საქაღალდეშია, სადაც MySQL-ი არის დაყენებული და ვებ-დამთვალიერებელში გავხსნათ ვებ-გვერდი **manual_toc.html**. ეს გვერდი შეიცავს სახელმძღვანელოს სარჩევს.
MySQL-ის საცნობარო სახელმძღვანელო რუსულ ენაზე (საკმაოდ კარგი თარგმანი) მდებარეობს საიტზე **http://www.mysql.com**. ის აგრეთვე წარმოადგენს **ZIP** არქივში მოთავსებული ვებ-გვერდების ანაკრებს. ამ არქივის რომელიმე საქაღალდეში განპაკების შმდეგ, საჭიროა ვებ-დამთვალიერებელში **manual.ge_toc.html** გვერდის გახსნა, რომელიც სახელმძღვანელოს სარჩევს შეიცავს.

დამხმარე პროგრამები

თვით **MySQL** მონაცემთა სერვერის გარდა, საიტზე **http://www.mysql.com** ჩვენ შეგვიძლია მოვიძიოთ ორი დამხმარე პროგრამა, რომელთაც შეუძლიათ მნიშვნელოვანი დახმარება გაგვიწიონ მუშაობაში.
მოდით ვნახოთ, თუ რა პროგრამებია ისინი.
MySQL Control Center-ეს **MySQL** მონაცემთა ბაზების საკმაოდ მძლავრი და გამოსაყენებლად მოხერხებული კლიენტური

პროგრამაა. მისი საშუალებით შესაძლებელი ხდება მონაცემთა ბაზების შექმნასთან დაკავშირებული სამუშაოების სწრაფად შესრულება.

შენიშვნა: *ჩვენ, ჩვენი მონაცემთა ბაზის შესაქმნელად გამოვიყენებთ კლიენტს phpMyAdmin. მისი დაყენების და გამოყენების პროცედურები აღწერილია დანართში 4.*

MySQL Administrator-ეს არის უტილიტა (დამხმარე პროგრამა), რომლის დანიშნულება მდგომარეობს თვით **MySQL** სერვერის ადმინისტრირებაში და მომსახურებაში. მართალია მისგან სარგებელი ჯერ-ჯერობით არც თუ ისე დიდია, რაც ალბათ იმით აიხსნება, რომ ჯერჯერობით ხელმისაწვდომია მხოლოდ წინასწარი, საცდელი (ბეტა) ვერსია.

დანართი 3

PHP დამმუშავებლის დაყენება და გამართვა

აქ აღწერილი იქნება **PHP**-ს აქტიური სერვერული გვერდების დამმუშავებლის, ლოკალურ კომპიუტერზე დაყენების პროცესი. ამასთან, იგულისხმება, რომ ვებ-სერვერი უკვე დაყენებულია, გამართულია და ნორმალურად მუშაობს. (ვებ-სერვერ **Apache**-ის დაყენების პროცესი აღწერილია ***დანართ 1***-ში).

PHP-პოპულარული უფასო პლატფორმაა აქტიური სერვერული ვებ-გვერდების შესაქმნელად, რომლის დისტრიბუტივი შეიძლება მოძიებულ იქნას **PHP**-ს შემქმნელთა ჯგუფის საიტზე (**http://www.php.net**). ამ საიტზე ხელმისაწვდომია **PHP**-ს სხვადასხვა ვერსიები სხვადასხვა ოპერაციული სისტემებისათვის.
დაყენების პროცესი აღწერილია ოპერაციული სისტემა-**Windows**-ისათვის განკუთვნილი 4.3.6 ვერსიის მაგალითზე.

დაყენება

PHP-ს დაყენების პროცესი ძალიან მარტივია. **PHP**-ს დისტრიბუტივის მიწოდება ხდება **ZIP** არქივის სახით, რომლის განპაკებაც უნდა მოხდეს **Program Files** საქაღალდეში (**Windows**-ის ყველა პროგრამა ამ ფაილში უნდა იქნას დაყენებული-ეს თავისებური დე-ფაქტო სტანდარტია). აღნიშნულის შემდეგ **Program**

Files საქაღალდეში შეიქმნება **საქაღალდე-4.3.6-win32**, რომელშიც მოთავსებული იქნება **PHP**-ს კომპლექტში შემავალი ყველა ფაილი). ამის შემდეგ ჩვენ აუცილებლად დაგვჭირდება **php4ts.dll** ფაილის კოპირება საქაღალდიდან რომელშიც **PHP** არის დაყენებული, საქაღალდეში, რომელშიც **Windows**-ია დაყენებული. (როგორც წესი ეს საქაღალდე მდებარეობს **C**-დისკოს ძირეულ საქაღალდეში და მას **Windows** ან **WinNT** ეწოდება). ეს ფაილი შეიცავს **PHP**-ს დამმუშავებლის ბირთვს. რამდენადაც ყველა პროგრამა მათთვის საჭირო **DLL** ბიბლიოთეკების მოსაძებნად, სხვებთან ერთად ათვალიერებენ იმ საქაღალდესაც, რომელშიც **Windows**-ია დაყენებული, ვებ სერვერს ყოველთვის შეეძლება მისი მოძებნა და ჩატვირთვა.

ყურადღება! *PHP 4.3.6-ის დაყენების სახელმძღვანელო ამტკიცებს, რომ Apache 2.x-თან მუშაობისას ვებ-სერვერზე ფაილების გაგზავნა არ მოხდება ვებ-ინტერფეისის მეშვეობით. (ამის შესახებ იხილეთ თავი 15).*

წინასწარი გამართვა

პირველი გაშვების წინ, **PHP**-ს გამართვა ჭირდება. წინააღმდეგ შემთხვევაში ის შეიძლება არ გაიშვას და ვებ-სერვერი **PHP**-ს ნებისმიერი გვერდის გახსნის მცდელობისას, გამოსცემს შეტყობინებას შეცდომის შესახებ.
PHP-ს გამართვის დაწყებამდე, ჩვენ აუცილებლად უნდა გავაჩეროთ ვებ-სერვერი **Apache**, თუ ის გაშვებული იყო. ამის შემდეგ გავუშვათ **Windows**-გამცილებელი ან ფაილების მართვის სხვა პროგრამა და გავხსნათ საქაღალდე, რომელშიც **PHP** არის დაყენებული. მოვძებნოთ მასში ფაილი **php.ini-dist**, გადავაკოპიროთ იგი საქაღალდეში, რომელშიც დაყენებულია სისტემა **Windows** და შევცვალოთ მისი სახელი **php.ini**-ით. შემდეგ გავხსნათ ეს ფაილი ნებისმიერ ტექსტურ რედაქტორში, მაგალითად ბლოკნოტში და მოვძებნოთ მასში სტრიქონი:

```
;include_path=".;c:\php\includes"
```

შევცვალოთ იგი შემდეგნაირად:

```
Include_path="c:\Program Files\php-4.3.6-win\includes"
```

(აქ იგულისხმება, რომ ჩვენ **PHP** დავაყენეთ C დისკოზე, საქაღალდეში **Program Files\php-4.3.6-win32**). განსაკუთრებული ყურადრება მივაქციოთ იმას, რომ შესწორებული სტრიქონის დასაწყისში არ იყოს წერტილ-მძიმის ნიშანი.
შემდეგი სტრიქონი, რომელიც ჩვენ უნდა შევცვალოთ ასე გამოიყურება:

extension_dir = "./ "

ის კი ასე უნდა გამოიყურებოდეს:

extension_dir = "c:\program files\php-4.3.6-win32\extensions"

კიდევ ერთი, რიგით მე-3 სტრიქონი, რომელიც ჩვენ უნდა შევცვალოთ ასეთია:

;upload_tmp_dir =

შევასწოროთ იგი ანალოგიურად:

upload_tmp_dir = "C:\Program Files\php-4.3.6-win32\tmp"

ბოლო სტრიქონი:

;session.save_path = /tmp

ასე უნდა გამოიყურებოდეს:

session.save_path="C:\Program Files\php-4.3.6-win\tmp"

აღნიშნული ოთხი სტრიქონის შესწორების შემდეგ, შევინახოთ ფაილი php.ini და დავხუროთ იგი.

ყურადღება! *სანამ კონფიგურაციის ფაილში php.ini შევცვლიდეთ რაიმე პარამეტრებს, საჭიროა ვებ-სერვერის გაჩერება. ამ შემთხვევაში ვებ-სერვერი დაასრულებს მუშაობას და მეხსიერებიდან ამოტვირთავს PHP-ს დამმუშავებელს.*

უკანასკნელი, რაც ჩვენ დაგვრჩა გასაკეთებელი-ეს არის შემოწმება იმისა, საქაღალდეში, რომელშიც PHP არის განთავსებული, არსებობს თუ არა საქაღალდე tmp. თუ ასეთი საქაღალდე არ არსებობს, ჩვენ მისი შექმნა მოგვიწევს. ამ საქაღალდეში PHP-ს დამმუშავებელი შეინახავს ფაილებს, რომელთაც ექნებათ სესიების ცვალებადი დონეები (მათ შესახებ საუბარი გვქონდა მე-***11 თავში).***

გაშვება და გაჩერება

არ არის საჭირო **PHP** დამმუშავებლის სპეციალურად გაშვება. ვებ-სერვერი თვითონ ჩატვირთავს და გაუშვებს მას, როგორც კი ვებ-დამთვალიერებელში ჩვენ გავხსნით ნებისმიერ **PHP** გვერდს. გაშვებისას **PHP** დამმუშავებელი წაიკითხავს კონფიგურაციის ფაილს **php.ini**-ს და გამოიყენებს მასში მოცემულ პარამეტრებს.

PHP-ს დამმუშავებლის მუშაობის დასრულებისა და მისი მეხსიერებიდან ამოტვირთვისათვის, ჩვენ მოგვიწევს თვით ვებ-სერვერის გაჩერება. სამწუხაროდ ამის გაკეთების სხვა ხერხი არ არსებობს.

PHP-ს საცნობარო სისტემის გამოყენება

PHP-ს საცნობარო სახელმძღვანელო ცალკე უნდა ჩაიტვირთოს საიტიდან **http://www.php.net.** თვით პროგრამასთან ერთად მისი მიწოდება არ ხდება. ამ საიტზე შესაძლებელია სხვადასხვა ენებზე დაწერილი სახელმძღვანელოების მოძიება, მათ შორის რუსულ და ინგლისურ ენებზე.

PHP-ს საცნობარო სახელმძღვანელოს მიწოდება ხდება სხვადასხვა ფორმატებით: ვებ-გვერდების ანაკრები, **CHM** და PDF. ყველაზე მოხერხებულად **CHM** ფორმატი შეიძლება ჩაითვალოს. მისი მიწოდება ხდება **ZIP** არქივის სახით, რომელიც შეიცავს ფაილს **php_manual_en.chm** (ეს არის სახელმძღვანელო ინგლისურ ენაზე; რუსული კი მდებარეობს ფაილში: **php_manual_ru.chm**).

დანართი 4

მონაცემთა კლიენტის phpMyAdmin-ის დაყენება და გამოყენება

წინამდებარე დანართში აღწერილი იქნება MySQL მონაცემთა ბაზებთან სამუშაო კლიენტური პროგრამის, phpMyAdmin-ის ლოკალურ ვებ-სერვერზე დაყენების პროცესი.

პოპულარული, უფასო მონაცემთა კლიენტის **phpMyAdmin**-ის დისტრიბუტივი შეიძლება მოვიძიოთ მისი შემმუშავებელი ჯგუფის საიტზე **(http://www.phpmyadmin.net)**. მონაცემთა კლიენტი **phpMyAdmin**-ი წარმოადგენს **PHP**-ს სერვერული გვერდების ანაკრებს და ამიტომაც იგი უდგება ყველა ოპერაციულ სისტემას და ყველა ვებ-სერვერს, რომელთაც გააჩნიათ **PHP**-ს მხარდამჭერი საშუალებები.
ყველა უფასო, თუ ფასიან ვებ-სერვერებზე, რომლებიც უზრუნველყოფენ კლიენტებისთვის **MySQL** მონაცემთა ბაზებთან მუშაობის შესაძლებლობებს, როგორც წესი, წინასწარ არის დაყენებული **phpMyAdmin**. მისდამი წვდომის ხერხები აღწერილია მხარდამჭერ საიტებზე. თუკი რომელიმე ვებ-სერვერის ადმინისტრატორმა არ იზრუნა **phpMyAdmin**-ის დაყენებაზე, მაშინ ჩვენ თვითონ შეგვიძლია მისი დაყენება მარტივად.

დაყენება და გაწყობა.

მონაცემთა კლიენტის **phpMyAdmin**-ის დაყენებისა და გამოყენების პროცესი აღწერილია 2.6.1. ვერსიის მაგალითზე.
თუ ჩვენ ვაპირებთ **phpMyAdmin**-ის დაყენებას ლოკალურ კომპიუტერზე, მაშინ უნდა დავრწმუნდეთ, რომ მასზე უკვე დაყენებულია ვებ-სერვერი (მაგალითად **Appache**), თვით **MySQL**-ი და **PHP**-დამმუშავებელი. თუკი ეს ასე არ არის, მაშინ უნდა მივმართოთ ამ წიგნის ***დანართებს 1-3*** , სადაც აღწერილია აღნიშნული პროგრამების დაყენებისა და გამართვის პროცესები.
phpMyAdmin-ის დისტრიბუტივის მიწოდება ხდება **ZIP** არქივის სახით, რომლის განპაკებაც შესაძლებელია ნებისმიერ საქაღალდეში. ამასთან შეიქმნება საქაღალდე **phpMyAdmin-2.6.1-pll**, რომელშიც მოთავსებული იქნება **phpMyAdmin**-ის შემადგენლობაში შემავალი ყველა ფაილი.

პირველი, რაც უნდა გაკეთდეს არქივის განპაკებისთანავე-უნდა განხორციელდეს მისი გამართვა. ამისათვის უნდა შევიდეთ საქაღალდეში **phpMyAdmin-2.6.1-pll**, და ბლოკნოტში ან რომელიმე ანალოგიურ ტექსტურ რედაქტორში გავხსნათ ფაილი **config.inc.php.** ეს ფაილი შეიცავს **PHP** გვერდებზე გამოყენებადი იმ ცვლადების გამოცხადებებს, რომლებიც წარმოადგენენ **phpMyAdmin**-ს.
ამ ფაილში ჯერ მოვძებნოთ ასეთი გამოსახულება:

$cfg[' PmaAbsoluteUri '] = ' ' ;

ეს გამოსახულება განსაზღვრავს **phpMyAdmin**-ის სრულ ინტერნეტ-მისამართს. შევასწოროთ იგი ისეთნაირად, რომ მიღებული შედეგი ასე გამოიყურებოდეს:

$cfg [' PmaAbsoluteUri '] = ' <ჩვენი საიტის ინტერნეტ-მისამართი>/phpMyAdmin-2.6.1-pll/ ' ;

მაგალითად, თუ ჩვენ **phpMyAdmin**-ს დავაყენებთ ლოკალურ ვებ-სერვერზე, მაშინ ეს გამოსახულება ასეთ სახეს მიიღებს:

$cfg [' PmaAbsoluteUri '] = ' http://localhost:8080/phpMyAdmin-2.6.1.-pll/ ' ;

დაშორებულ ვებ-სრვერზე დაყენების შემთხვევაში (როდესაც ჩვენ დავაპირებთ ჩვენი საიტის გამოქვეყნებას ქსელში), მასში უნდა ჩავსვათ ჩვენთვის გამოყოფილი დაშორებული ვებ-სერვერის ინტერნეტ-მისამართი.
შემდეგ, გადავახვიოთ **config.inc.php** ფაილის შიგთავსი ბოლოსკენ და დავინახავთ ასეთ გამოსახულებას:

$cfg [' Servers '] [$i] [' auth_type '] = ' config ' ;

ის განსაზღვრავს **MySQL**-სერვერთან მომხმარებლის მიერთების ხერხს. თუ იქ მითითებულია მნიშვნელობა **config,** მაშინ მომხმარებლის სახელი და პაროლი აიღება იმავე **config.inc.php** ფაილიდან. იქ კი მითითებული იქნება მომხმარებელი **root-MySQL** სერვერის ადმინისტრატორი, ხოლო პაროლი -იქნება "ცარიელი"!

გასაგებია, რომ ეს დიდი "ხვრელია" უსაფრთხოების თვალსაზრისით. ჩვენი საიტის ნებისმიერ მომხმარებელს შეეძლება **phpMyAdmin**-ში შესვლა, სერვერთან root-სახელით ავტომატური მიერთება და ნებისმიერი რამის შეცლა. ამიტომ ჩვენ ეს "ხვრელი" უნდა ამოვავსოთ. აღნიშნული მიზნით საკმარისია ადრე განხილული გამოსახულების შესწორება ისეთნაირად, რომ მან მიიღოს ასეთი სახე:

$cfg [' Servers '] [$i] ['auth_type'] = 'http';

ამის შემდეგ **phpMyAdmin**-ში შესვლის მცდელობისას, ის ვებ-დამთვალიერებლის საშუალებით მოგვთხოვს მომხმარებლის სახელსა და პაროლს. ეს კი ნიშნავს იმას, რომ ვერავინ უცხო, ვერ შეძლებს ჩვენს მონაცემთა ბაზებში შეღწევას და იქ რაიმეს შეცვლას.
ამის შემდეგ შეიძლება **config.inc.php**-ს შენახვა და მისი დახურვა. ამით გამართვის პროცესი დასრულებული იქნება.
ახლა მთელი საქაღალდე **phpMyAdmin-2.6.1.-pll** გადავიტანოთ ჩვენი ვებ-სერვერის ძირეულ საქაღალდეში-ლოკალურში ან დაშორებულში (ამისათვის შესაძლებელია **FTP**-კლიენტ პროგრამის გამოყენება). ამ შემთხვევაში, ვინაიდან **phpMyAdmin**-ი საკმაოდ დიდია ჩვენ მოგვიწევს დალოდება, სანამ ის ჩაიტვირთება.

შენიშვნა: *სასურველია, რომ საქაღალდეს phpMyAdmin-2.6.1-pll დაერქვას უფრო მოკლე სახელი, მაგალითად pma. გაცილებით მარტივია მოკლე სახელის აკრეფა.*

გამოყენება

აქ კი ჩვენ აღვწერთ **phpMyAdmin**-ის გამოყენებას MySQL მონაცემთა ბაზებთან სამუშაოდ. ჩვენ შევისწავლით მონაცემთა ბაზების, ცხრილების, ველებისა და ინდექსების შექმნას და მათდამი წვდომის უფლებების მართვას.

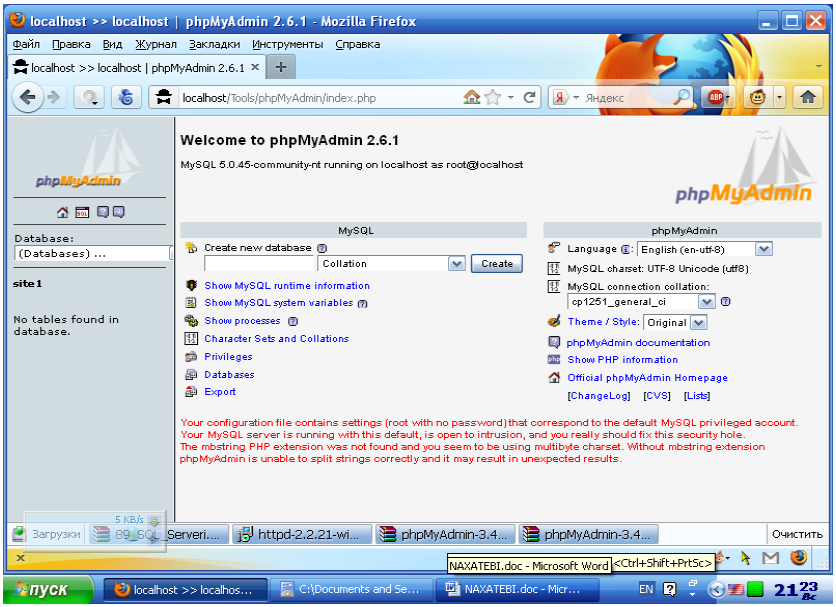
შესვლა

phpMyAdmin-ში შესვლა ძალზე მარტივია. ამისათვის საკმარისია ვებ-დამთვალიერებლის გაშვება და მისამართების სტრიქონში **http://localhost:8080/phpMyAdmin-2.6.1-pl1** აკრეფა.

ყურადღება! *აქ იგულისხმება, რომ phpMyAdmin დაყენებულია ლოკალურ ვებ-სერვერზე, ხოლო საქაღალდეს სახელი, რომელშიც იგი აყენია არ შეცვლილა. თუ ეს ასე არ არის, მაშინ ზემოთ მაყვანილ ინტერნეტ-მისამართში უნდა შევიდეს შესაბამისი ცვლილებები.*

ყოველივე აღნიშნულის შემდეგ ვებ-დამთვალიერებელი ეკრანზე გამოიტანს მომხმარებლის სახელისა და პაროლის შესატან სტანდარტულ დიალოგურ ფანჯარას. შევიტანოთ სახელი და პაროლი და დავაჭიროთ ამ ფანჯრის **OK** ღილაკს. თუ ყველაფერი სწორად იქნა გაკეთებული, რამოდენიმე წამში ვებ-დამთვალიერებელში გამოჩნდება **phpMyAdmin**-ის მთავარი გვერდი, რომელიც ნახ. დ.4.1-ზეა ნაჩვენები.
მონაცემთა კლიენტ **phpMyAdmin**-ს ერთი თავისებურება გააჩნია-ის ავტომატიურად ცნობს ოპერაციული სისტემის ენას და თავისი მთავარი გვერდი გამოაქვს ამ ენაზე. ასე, რომ თუ ჩვენ ვმუშაობთ **Windows**-ის რუსულ ვერსიაში, მაშინ **phpMyAdmin**-ის მთავარ გვერდს მივიღებთ რუსულ ენაზე.
თუ ჩვენ **MySQL** დაყენებული გვაქვს ლოკალურ კომპიუტერზე, მაშინ სერვერზე შესაღწევად უნდა გამოვიყენოთ მომხმარებელი **root** "ცარიელი" პაროლით. თუკი ჩვენ მონაცემთა ბაზას ვქმნით დაშორებულ ვებ-სერვერზე, მაშინ უნდა გამოვიყენოთ სახელი და პაროლი, რომელსაც გამოგვიყოფენ.

იმ შემთხვევაში, თუ ჩვენ შევცდებით მომხმარებლის სახელისა და პაროლის შეტანისას, ვებ-დამთვალიერებელი ისევ გამოიტანს მომხმარებლის სახელისა და პაროლის შესატან სტანდარტულ დიალოგურ ფანჯარას და მოგვეცემა ხელახლა ყველაფრის სწორად შეტანის შესაძლებლობა.



ნახ.დ.4.1. phpMyAdmin-ის მთავარი გვერდი

მონაცემთა ბაზის შექმნა

მაშ ასე, ჩვენ ვიმყოფებით **phpMyAdmin**-ის მთავარ ფანჯარაში (იხ. ნახ.დ.4.1). დროა შევუდგეთ ჩვენი მონაცემთა ბაზის შექმნას.

შევხედოთ ვებ-გვერდის ზედა მარცხენა კუთხეს. იქ უნდა იყოს სერვერზე უკვე შექმნილი და მოცემული მომხმარებლისათვის ხელმისაწვდომი მონაცემთა ბაზების ჩამონათვალი ჩამოსაშლელი სიის სახით. (რადგან ჩვენ შემოვედით root სახელით, ჩვენთვის ხელმისაწვდომია ყველა ბაზა). შევამოწმოთ, არჩეულია თუ არა ამ სიაში პუნქტი (**Database-Базы Данных-მონაცემთა ბაზები**) და თუ არა, ავირჩიოთ იგი.

phpMyAdmin-ში მონაცემთა ბაზის შექმნა ძალზე მარტივია. შესაქმნელი ბაზის სახელი შევიტანოთ შესატან ველში **Create New database-Создать новую БД-ახალი მბ-ის შექმნა** და დავაჭიროთ ღილაკს **Create-Создать-შექმნა**. ამის შემდეგ ჩვენს მიერ შექმნილი მონაცემთა ბაზა გაჩნდება ვებ-გვერდის მარცხენა სვეტში, ხოლო მის მარჯვენა ნაწილში დავინახავთ ორ შესატან ველს: **Name-Имя**

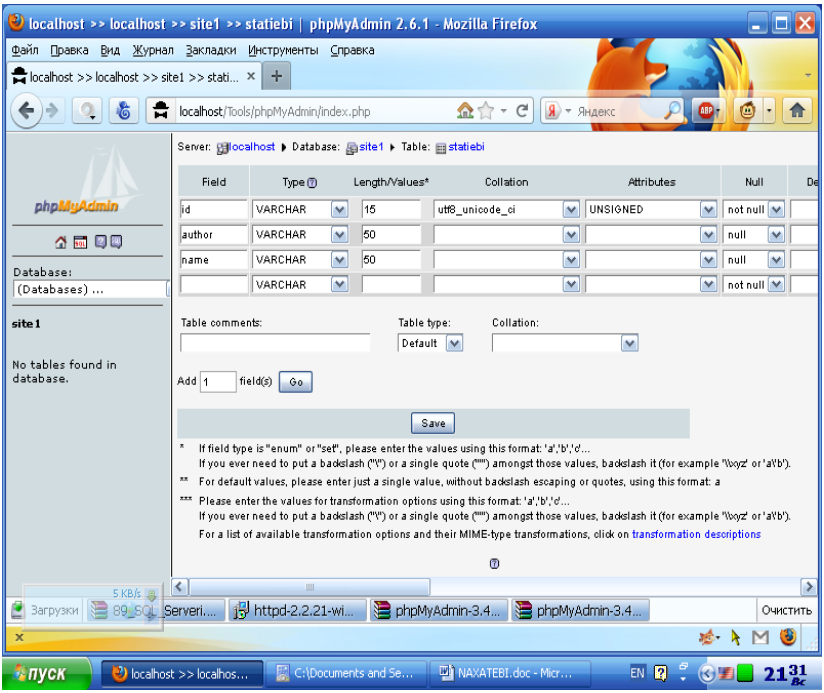
(შესაქმნელი ცხრილის სახელი) და **Field(s)-Поля** (ველების რაოდენობა).
ამის მერე შეიძლება შევუდგეთ ცხრილების შექმნას.

ცხრილების შექმნა

ცხრილის შესაქმნელად, ჯერ უნდა შევამოწმოთ, არჩეულია თუ არა ჩვენთვის სასურველი მონაცემთა ბაზა. ამისათვის შევხედოთ **phpMyAdmin** ვებ-გვერდის მარცხენა სვეტს, სადაც ჩამოსაშლელი სიდან არჩეული უნდა იყოს პუნქტი, რომლის სახელიც იმ მონაცემთა ბაზის სახელს ემთხვევა, რომელშიც მოთავსებულ უნდა იქნას ახალი ცხრილი. თუკი ეს ასე არ არის, ავირჩიოთ იგი.
მაშ ასე, ჩვენ ავირჩიეთ მონაცემტა ბაზა. ახლა ვებ-გვერდის მარჯვენა ნაწილში მდებარე შესატან ველში **Name-Имя-სახელი**, შევიტანოთ შესაქმნელი ცხრილის სახელი. შესატან ველში **Field(s)-Поля-ველები** შევიტანოთ აღნიშნული ცხრილის ველების რაოდენობა. დაბოლოს დავაჭიროთ ღილაკს **Go-Пошел-წავიდა**.
ღილაკის დაჭერისთანავე შეიქმნება ახალი ცხრილი და **phpMyAdmin**-ი გამოგვიტანს ახალ ვებ-გვერდს, სადაც ჩვენ შევძლებთ ცხრილის პარამეტრების განსაზღვრას.
შეიძლება ისეც მოხდეს, რომ **phpMyAdmin**-ის გვერდზე არ გამოჩნდნენ არც ველი **Name-Имя-სახელი** და არც ველი **Field(s)-Поля-ველები**. იმისათვის, რომ ისინი გამოჩნდნენ, ფანჯრის ზედა მარცხენა კუთხეში მდებარე ჩამოსაშლელ სიაში უნდა ავირჩიოთ პუნქტი **Database_Базы Данных_მონაცემთა ბაზები** და იმავდროულად-ჩვენი მონაცემთა ბაზაც. ადრე დასახელებული ველები იმწამსვე გაჩნდებიან ვებ-გვერდზე და ჩვენ მაშინათვე შეგვეძლება კიდევ ერთი ცხრილის შექმნა.

ველების შექმნა

როდესაც ჩვენ ვქმნიდით ახალ ცხრილს, შესატან ველში **Field-Поля-ველები** შევიტანეთ ველების ის რაოდენობა, რომელიც ამ ცხრილში უნდა ყოფილიყო. ამის შემდეგ **phpMyAdmin**-ს ვებ-გვერდის მარცხენა ნაწილში გამოაქვს ცხრილი ფორმის სახით, რომელშიც შეგვიძლია ველების პარამეტრების განსაზღვრა (დ.4.2).



ნახ. დ.4.2. ახლად შექმნილი ცხრილის ველების პარამეტრების განსაზღვრის ფორმა

ამ ფორმა-ცხრილს გააჩნია შემდეგი სვეტები:

-**Fild-Поле-ველი**-ველის დასახელება;

-**Type_Тип-ტიპი**-ჩამოსაშლელი სია, ველში შესანახი მონაცემების ტიპების განსაზღვრისათვის;

-**Length/Value-Длины/Значения-სიგრძე/მნიშვნელობა**-მაქსიმალური სიგრძე (სტრიქონული და მთელრიცხვიანი ველებისათვის);

-**Attributes-Атрибуты-ატრიბუტები**-ჩამოსაშლელი სია ველის დამატებითი ატრიბუტების განსაზღვრისათვის. ჩვენთვის განსაკუთრებით სასარგებლო იქნება **UNSIGNED** (უნიშნო მთელრიცხვიანი მონაცემების ტიპი) და პუნქტი "**ცარიელი**" (ნიშნიანი, მთელრიცხვიანი ტიპი);

-**Null-Ноль-ნული**-ჩამოსაშლელი სია, რომელიც საშუალებას გვაძლევს განვსაზღვროთ, შეუძლება თუ არა ველის დატოვება მნიშვნელობის ჩაწერის გარეშე. ხელმისაწვდომი პუნქტებია: **not null**

(არ შეიძლება მიიღოს; არჩეულია გაჩუმებით) და **null** (შეიძლება მიიღოს);

-**Default-По умолчанию-გაჩუმებით**-ველის გაჩუმებითი მნიშვნელობა;

-**Additional-Дополнительно**-დამატებით-ჩამოსაშლელი სია, რომელიც მთვლელი ველის შექმნის საშუალებას იძლევა (ამისათვის საკმარისია **auto_increment** პუნქტის არჩევა);

-**Primary_Первичный-პირველადი**-გადამრთველი, რომელიც განსაზღვრავს მოცემული ველის საფუძველზე, გასაღები ინდექსის შექმნის შესაძლებლობას;

-**Index-Индекс-ინდექსი**-გადამრთველი, რომელიც განსაზღვრავს მოცემული ველის საფუძველზე ჩვეულებრივი ინდექსის შექმნის შესაძლებლობას;

-**Unique-Уникальное-უნიკალური**-გადამრთველი, რომელიც განსაზღვრავს მოცემული ველის საფუძველზე უნიკალური ინდექსის შექმნის შესაძლებლობას;

--- - გადამრთველი, რომელიც აუქმებს მოცემული ველის საფუძველზე ინდექსის შექმნის შესაძლებლობას;

-**Full Text-Полный Текст-სრული ტექსტი**-ალამი, რომელიც განსაზღვრავს სრულტექსტოვანი ინდექსის შექმნის შესაძლებლობას. ასეთი ინდექსი შეიცავს ყველა იმ სიტყვებს, რომლებიც გვხვდება მოცემული ველის მნიშვნელობაში და მისი გამოყენება შესაძლებელია ველებში სიტყვების მოსაძებნად. სრულტექსტოვანი ინდექსი შეიძლება შეიქმნას მხოლოდ სტრიქონული ველების საფუძველზე.

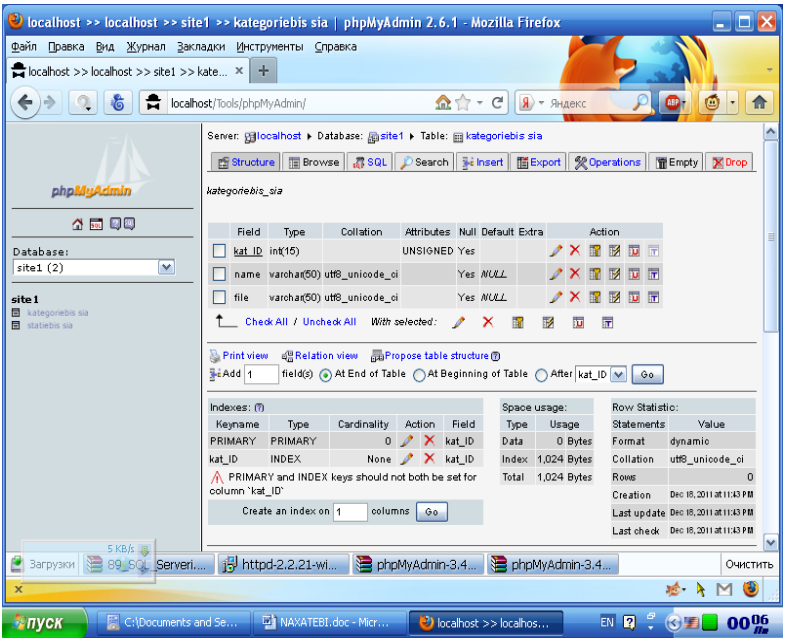
სამწუხაროდ, ჩამოსაშლელი სია **Type_Тип-ტიპი** არ შეიცავს პუნქტს **BINARI**-მონაცემთა ლოგიკურ ტიპს. ჩვენ მოგვიწევს მასში **TINYINT** პუნქტის არჩევა და **Length/Value-Длины/Значения-სიგრძე/მნიშვნელობა** შესატან ველში რიცხვი 1-ის შეტანა.

ფორმა-ცხრილში მონაცემების შეტანის დასრულების შემდეგ, დავაჭიროთ ღილაკს **Save-Сохранить-შენახვა.** ამის შემდეგ **phpMyAdmin** შექმნის ახალ ცხრილს, გამოიტანს მის სახელს ვებ-გვერდის მარცხენა სვეტში და ავტომატურად აირჩევს მას. ამ დროს ვებ-გვერდის მარჯვენა მხარეს ჩვენ დავინახავთ იმას, რაც ნაჩვენებია ნახ. დ.4.3-ზე.

თუკი ჩვენ დაგვჭირდება უკვე შექმნილ ცხრილში ახალი ველის ჩამატება, ჩვენ ჯერ უნდა ავირჩიოთ იგი. ამისათვის ვებ-გვერდის ზედა მარცხენა კუთხეში განლაგებულ ჩამოსაშლელ სიაში ავირჩიოთ

ჩვენი მონაცემთა ბაზის შესაბამისი პუნქტი. ამის შემდეგ ქვემოთ გაჩნდება ამ ბაზაში ჩვენს მიერ შექმნილი ცხრილების სია; დავაწკაპუნოთ საჭირო ცხრილს და ავირჩიოთ იგი.

ცხრილის არჩევის შემდეგ, ჩვენ შეგვიძლია ჩასამატებელი ველების რაოდენობის შეტანა შესატან ველში **Add New Field-Добавить новое поле-ახალი ველის დამატება,** რომელიც ვებ-გვერდის ქვედა ნაწილში მდებარეობს და დავაჭიროთ ღილაკს **Go-Пошел-წავედა**. ვებ-გვერდზე გაჩნდება ფორმა-ცხრილი, რომელიც ნახ. დ.4.2.-ზე ნაჩვენები ფორმა-ცხრილის ანალოგიურია და რომელში უნდა შევიტანოთ ჩასამატებელი ველის(ველების) პარამეტრები. ყოველივე აღნიშნულის შემდეგ დავაჭიროთ ღილაკს **Save-Сохранить-შენახვა**.



ნახ. დ.4.3. phpMyAdmi-ის მიერ გამოსახული, არჩეული ცხრილის სტრუქტურა.

ინდექსების შექმნა

ინდექსების შექმნა შესაძლებელია ველების პარამეტრების განმსაზღვრელ ფორმა-ცხრილში (იხ. ნახ. დ.4.2) შესაბამისი გადამრთველების ჩართვით. მაგრამ ასეთი ინდექსები მხოლოდ

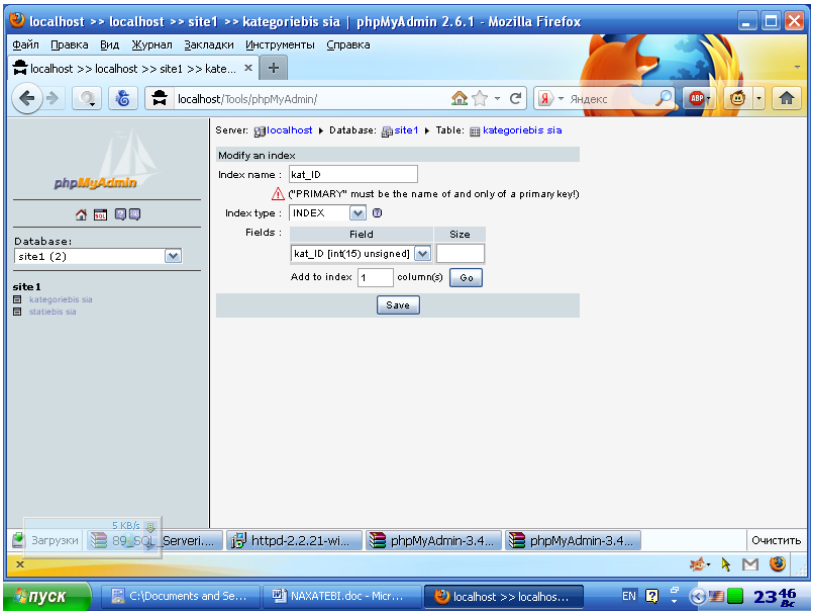
ერთი ველისაგან იქნებიან შემდგარი. თუკი ჩვენ გვჭირდება ისეთი ინდექსის შექმნა, რომელიც ორი ან მეტი ველის მნიშვნელობებისაგან იქნება შემდგარი (მაგალითად, ინდექსი **namefile** ცხრილში **categories**), მაშინ ჩვენ მოგვიწევს სხვა ხერხის გამოყენება.
შესატან ველში **Create Index for-Создать индекс на-ინდექსის შექმნა** ველების რაიმე რაოდენობაზე), შევიტანოთ ველების ის რაოდენობა, რომელიც ინდექსისათვის იქნება განსაზღვრული და დავაჭიროთ ღილაკს **Go-Пошел-წავიდა**. მონაცემთა კლიენტი **phpMyAdmin** გამოიტანს ფორმას, რომლის საშუალებითაც მოხდება ინდექსის შექმნა (ნახ. დ.4.4).

შესაქმნელი ინდექსის სახელი განისაზღვრება შესატან ველში, რომელსაც ასევე **Index Name-Имя индекса-ინდექსის სახელი** ეწოდება. თუკი გასაღები ინდექსის შექმნაა საჭირო, მაშინ ამ შესატან ველში უნდა შევიტანოთ სტრიქონი **PRIMARY** (ბრჭყალების გარეშე).
ჩამოსაშლელი სია **Index Type-Тип индекса-ინდექსის ტიპი**, განსაზღვრავს შესაქმნელი ინდექსის დამატებით ატრიბუტებს. ეს ხორციელდება მისი სამი პუნქტიდან ერთ-ერთის არჩევით:

-**INDEX**-მარტივი ინდეხსი;
-**UNIQUE**-უნიკალური ინდექსი;
-**FULLTEXT**-სრულტექსტოვანი ინდექსი.

ქვემოთ მოცემულია პატარა ცხრილი, რომლის საშუალებითაც ხდება ინდექსში შემავალი ველების განსაზღვრა. მას ასეთი სვეტები გააჩნია:

-**Field-Поле-ველი**-ჩამოსაშლელი სია, რომლიდანაც ხდება ველის არჩევა;
-**Size-Размер-ზომა**-ინდექსში შესატანი სტრიქონული ველის სიმბოლოების რაოდენობის განმსაზღვრელი ჩამოსაშლელი სია. **MYSQL** იძლევა ისეთი ინდექსების შექმნის შესაძლებლობას, რომლებიც შეიცავენ სტრიქონული ველების არა სრულ მნიშვნელობებს, არამედ მათ ფრაგმენტებს. (კერძოდ. შესაძლებელია ინდექსების შექმნა blob ტიპის ველების საფუძველზე).



ნახ. დ.4.4. ახალი ინდექსის შესაქმნელი ფორმა

ინდექსის პარამეტრების განსაზღვრის დასრულების შემდეგ დავაჭიროთ ღილაკს **Save-Сохранить-შენახვა.** შედეგად შეიქმნება ახალი ინდექსი და მის შესახებ ცნობები გამოჩნდება იმავე ვებ-გვერდზე, რომელზეც გამოსახულია ცხრილის სტრუქტურა (იხ. ნახ. დ.4.3).

იმისათვის, რომ დავამატოთ ახალი ინდექსი უკვე არსებულ ცხრილში, ჯერ ვებ-გვერდის ზედა მარცხენა კუთხეში მდებარე ჩამოსაშლელი სიიდან ავირჩიოთ ჩვენი მონაცემთა ბაზის შესაბამისი პუნქტი. ქვემოდ გაჩნდება ჩვენს მიერ ამ ბაზაში შექმნილი ცხრილების სია; მოვძებნოთ ის რომელიც გვჭირდება და დავაწკაპუნოთ მასზე. ამის შემდეგ, ვებ-გვერდზე გამოჩნდება ცნობა არჩეული ცხრილის შესახებ და აგრეთვე ჩვენთვის საჭირო შესატანი ველი **Create Index for Создать индекс- на-ინდეხსის შექმნა ველების გარკვეულ რაოდენობაზე** და ღილაკი **Go-Пошел-წავიდა**.

ველების, ინდექსების, ცხრილების და მონაცემთა ბაზების შესწორება და წაშლა.

სამწუხაროდ, ყველა ადამიანი უშვებს შეცდომებს. განსაკუთრებით მაშინ, როდესაც ისინი პირველად აკეთებენ რამეს. არც ჩვენ არა ვართ გამონაკლისები.

საბედნიეროდ, **phpMyAdmin** გვთავაზობს შეცდომების გასწორების საკმაოდ მძლავრ საშუალებებს. ჩვენ შეგვიძლია შევასწოროთ ნებისმიერი ველის, ნებისმიერი ინდექსის და ნებისმიერი ცხრილის პარამეტრები. ასევე, ჩვენ შეგვიძლია წავშალოტთ გამოუყენებელი ან ზედმეტი ველი, ინდექსი ან ცხრილი.

ველების შესწორება და წაშლა

იმისათვის, რომ შევასწოროთ ან წავშალოთ ცხრილის რომელიმე ველი, ჯერ უნდა ავირჩიოთ იგი. პროცესი იგივეა: ვებ-გვერდის ზედა მარცხენა კუთხეში მდებარე ჩამოსაშლელი სიიდან ვირჩევთ მონაცემტა ბაზას, რომელშიც მოთავსებულია ჩვენი ცხრილი და ვაწკაპუნებთ მასზე. ამის შემდეგ **phpMyAdmin** ვებ-გვერდზე გამოიტანს ცნობებს ამ ცხრილის სტრუქტურის შესახებ.

ველებისა და მათი პარამეტრების სია ნაჩვენები იქნება ვებ-გვერდის ზედა ნაწილში. მას ცხრილის სახე გააჩნია, რომლის სვეტებშიც ჩამოთვლილია ველების შექმნისას ჩვენს მიერ განსაზღვრული პარამეტრები.

იმისათვის, რომ შევასწოროთ რომელიმე ველი, დავაწკაპუნოთ ნიშნაკზე  , რომელიც მდებარეობს საჭირო ველის შესაბამის სტრიქონში. მონაცემთა კლიენტი **phpMyAdmin** გამოიტანს ველის პარამეტრების განსასაზღვრავ ფორმა-ცხრილს (იხ. ნახ.დ.4.2), სადაც ჩვენ შეგვეძლება მათი შეცვლა. საჭირო ცვლილებების შეტანის შემდეგ არ უნდა დაგვავიწყდეს **Save-Сохранить-შენახვა** ღილაკზე დაჭერა.

გამოუყენებელი ველის წაშლა ძალზე მარტივია. ამისათვის უნდა დავაჭიროთ ნიშნაკს  , რომელიც შესაბამის სტრიქონში მდეარეობს. ეკრანზე გამოჩნდება გამაფრთხილებელი ფანჯარა, რომელიც შეგვეკითხება, ნამდვილად გვსურს თუ არა აღნიშნული ველის წაშლა. დასადასტურებლად უნდა დავაჭიროთ ღილაკზე **OK**; **Cancel-Отмена-გაუქმება** ღილაკზე დაჭერა აუქმებს ველის წაშლის ოპერაციას.

ინდექსების შესწორება და წაშლა

იმისათვის, რომ შევცვალოთ ან წავშალოთ ცხრილის ინდექსი, ისევე, როგორც ველების შემთხვევაში, ჯერ უნდა ავირჩიოთ ეს ცხრილი. ვებ-გვერდის ზედა მარცხენა კუთხეში განთავსებული ჩამოსაშლელი სიიდან ავირჩიოთ მონაცემთა ბაზა და დავაწკაპუნოთ სასურველ ცხრილზე. აღნიშნულის შემდეგ **phpMyAdmin**-ი ვებ-გვერდზე გამოიტანს ცნობებს ამ ცხრილის სტრუქტურის შესახებ.

ინდექსებისა და მათი პარამეტრების სია მოცემული იქნება ვებ-გვერდის ქვედა ნაწილში. მას ცხრილის სახე ექნება, რომლის სვეტებშიც ჩამოთვლილი იქნება ამ ინდექსის შექმნის დროს ჩვენს მიერ განსაზღვრული პარამეტრები.

იმისათვის, რომ შევასწოროთ რომელიმე ინდექსი, დავაწკაპუნოთ ნიშნაკზე  , რომელიც არჩეული ინდექსის შესაბამის სტრიქონში მდებარეობს. მონაცემთა კლიენტი **phpMyAdmin**-ი გამოიტანს ინდექსის პარამეტრების განმსაზღვრელ ფორმა-ცხრილს (იხ. ნახ. დ.4.4), სადაც ჩვენ შეგვეძლება მათი შეცვლა.

ინდექსში ველის დასამატებლად, შესატან ველში **Add to Index-Добавить к индексу-დავამატოთ ინდექსს** შევიტანოთ დასამატებელი ინდექსების რაოდენობა და დავაჭიროთ ღილაკს **Go-Пошел-წავიდა**. ველების ცხრილში გაჩნდება კიდევ ერთი სტრიქონი, რომელშიც ჩვენ შეგვეძლება საჭირო ველის განსაზღვრა.

გამოუყენებელი ველის ამოშლა ინდექსიდან ხორციელდება არცთუ ისე გასაგებად. ცხრილში, სადაც ჩამოთვლილია ინდექსში შემავალი ველები, გამოვყოფთ იმას, რომლის წაშლაც გვსურს და ჩამოსაშლელ სიაში **Field-Поле-ველი** ვირჩევთ პუნქტს --**Ignor-Игнорировать-იგნორირება**--. აღნიშნული მოქმედების შემდეგ ეს ველი ამოღებული იქნება ინდექსიდან.

როგორც ყოველთვის, საჭირო ცვლილებების განხორციელების შემდეგ არ უნდა დაგვავიწყდეს ღილაკზე **Save-Сохранить-შენახვა** დაჭერა.

ცხრილიდან გამოუყენებელი ინდექსის ამოშლა ისევე ხდება, როგორც გამოუყენებელი ველისა-შესაბამის სტრიქონში მდებარე ნიშნაკზე დაჭერთ. ამ შემთხვევაში, ეკრანზე გამოჩნდება გამაფრთხილებელი ფანჯარა, რომელიც შეგვეკითხება, ნამდვილად გვსურს თუ არა ამ ინდექსის წაშლა. დასადასტურებლად ვაჭერთ

ღილაკს **OK; Cancel-Отмена-გაუქმება** ღილაკზე დაჭერა აუქმებს ინდექსის წაშლის ოპერაციას.

ცხრილების შესწორება და წაშლა

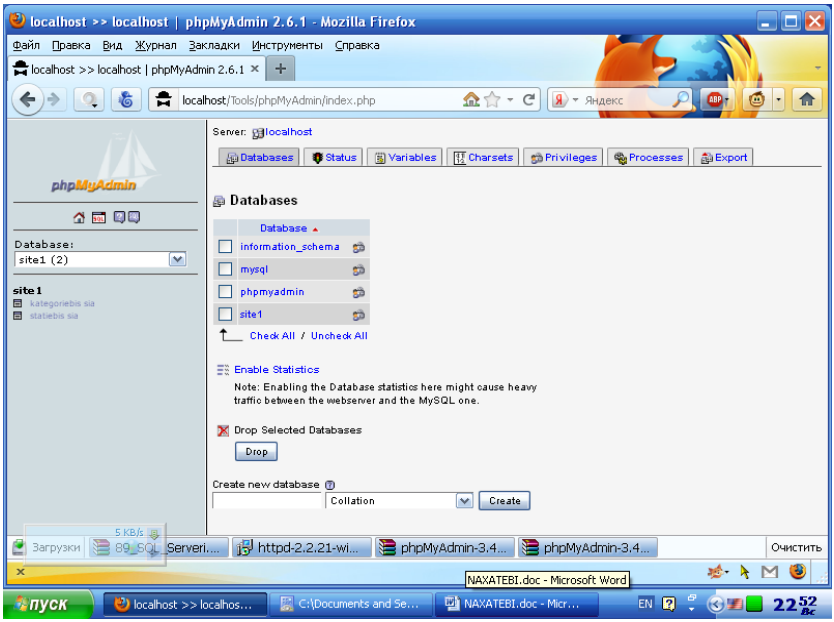
ცხრილების შესწორება მდგომარეობს მასში შემავალი ველების და ინდექსების დამატებაში, შესწორებაში და წაშლაში. თუ როგორ კეთდება ყოველი მათგანი ჩვენ უკვე ვიცით.

ხოლო, გამოუყენებელი ცხრილის წასაშლელად, ჯერ მისი არჩევაა საჭიროა (ჩვენ აღარ აღვწერთ ამ პროცესს-ის არაერთხელ იყო აღწერილი). როდესაც **phpMyAdmin**-ი გამოიტანს ცნობებს ამ ცხრილის სტრუქტურის შესახებ, ვებ-გვერდის ზედა მარჯვენა კუთხეში მოვძებნოთ ჰიპერმინიშნება **Delate-Уничтожить-განადგურება.** (იგი გამოყოფილია ღია წითელი ფერით და შეუძლებელია მისი არ დანახვა). თუ დავაწკაპუნებთ მასზე, ეკრანზე გამოჩნდება გამაფრთხილებელი ფანჯარა, რომელიც შეგვეკითხება, ნამდვილად გვსურს თუ არა ამ ცხრილის წაშლა. ღილაკზე **OK** დაჭერა წაშლის მას, ხოლო **Cancel-Отмена-გაუქმება** ღილაკზე დაჭერა დატოვებს "ცოცხლად".

მონაცემთა ბაზების შესწორება და წაშლა.

მონაცემთა ბაზების შესწორება მდგომარეობს მის შემადგენლობაში შემავალი ცხრილების დამატებაში, შესწორებაში და წაშლაში. ასე, რომ გადავიდეთ პირდაპირ წაშლაზე.

გამოუყენებელი ცხრილის წასაშლელად, ჯერ უნდა ავირჩიოთ ვებ-გვერდის ზედა მარცხენა კუთხეში მდებარე ჩამოსაშლელი სიიდან პუნქტი **Database-Базы Данных-მონაცემტა ბაზები**. როდესაც phpMyAdmin-ი გამოიტანს თავის მთავარ გვერდს, დავაწკაპუნოთ ჰიპერმინიშნებაზე **Database-Базы Данных-მონაცემტა ბაზები**. ეკრანზე გამოჩნდება ვებ-გვერდი, რომელზეც განთავსებული იქნება მონაცემთა ბაზების სამართავი ფორმა (ნახ. დ.4.5).



ნახ. დ.4.5. მონაცემთა ბაზების სამართავი ფორმა.

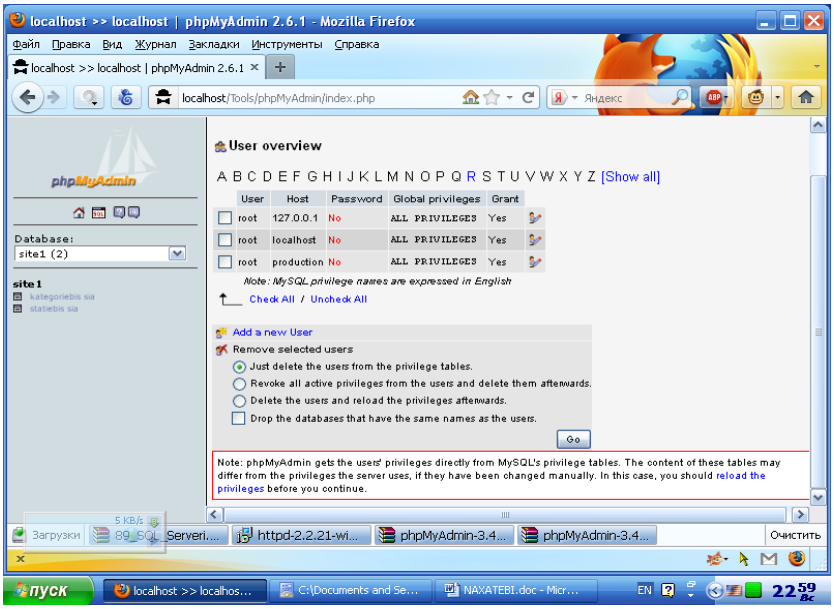
როგორც ვხედავთ ამ ფორმას ცხრილის სახე აქვს. მის მარცხენა უკიდურეს სვეტში მდებარეობს ალამი, რომლის მონიშვნითაც შესაბამისი მონაცემთა ბაზა ჩართული იქნება ჩვენს მიერ შესრულებად ოპერაციაში (მოცემულ შემთხვევაში-წაშლის ოპერაციაში). ამის შემდეგ უნდა დავაჭიროთ ღილაკს **Go-Пошел-წავიდა**, რომელიც **Drop Selected Databases-Удалить выбранные БД-არჩეული მონაცემთა ბაზების წაშლა,** წარწერის ქვემოთ მდებარეობს. ზოგადად, ასეთ შემთხვევებში ეკრანზე ჩნდება **Windows**-ის ჩვენთვის კარგად ნაცნობი გამაფრთხილებელი ფანჯარა. მაგრამ, მოცემულ შემთხვევაში **phpMyAdmin**-ი ოდნავ განსხვავებულად მოიქცევა და გამოიტანა ვებ-გვერდს, რომელზეც განთავსებულია ფორმა გამაფრთხილებელი ტექსტით და ორი ღილაკი-**Yes-Да-კი** და **No-Нет-არა**. გასაგებია, რომ **Yes-Да-კი** ღილაკზე დაჭერა წაშლის ჩვენს მრავალტანჯულ მონაცემთა ბაზას, ხოლო **No-Нет-არა** ღილაკზე დაჭერა-გააუქმებს წაშლის ოპერაციას.

მომხმარებლების მართვა

ჩვენ უკვე შევქმენით ჩვენთვის საჭირო ყველა ცხრილი. ახლა კი დროა გავერკვეთ მომხმარებლებში და მათ უფლებებში.

phpMyAdmin-ის მომხმარებლების მართვის საშუალებები

იმისათვის, რომ გადავერთოთ მომხმარებლების სიაში, საჭიროა დავაწკაპუნოთ ჰიპერმინიშნებაზე **Home Page-Домашняя страница-მთავარი გვერდი**, რომელიც მუდმივად ვებ-გვერდის ზედა მარცხენა კუთხეში, მონაცემთა ბაზების ჩამოსაშლელი სიის ზემოთ მდებარეობს. მას შემდეგ, რაც **phpMyAdmin** გამოიტანს მთავარ ვებ-გვერდს, დავაწკაპუნოთ ჰიპერმინიშნებაზე **Privileges-Привилегии-პრივილეგიები** და პირდაპირ აღმოვჩნდებით ვებ-გვერდზე, რომელზეც განთავსებულია მომხმარებლების სამართავი ფორმა (ნახ. დ.4.6).



ნახ. დ.4.6. მომხმარებლების სამართავი ფორმა

ეს ფორმა შეიცავს რამოდენიმე სვეტიან ცხრილს. მოდით ჩამოვთვალოთ ისინი.

-მარცხენა უკიდურესი სვეტი შეიცავს ალამს, რომლის მონიშვნაც მოცემულ მომხმარებელს ჩართავს რაიმე ოპერაციაში (მაგალითად, წაშლის ოპერაციაში).

-**User-Пользователь-მომხმარებელი**-მომხმარებლის სახელი. თუ იქ წერია **Any-Любой-ნებისმიერი**, იგულისხმება მომხმარებელი ნებისმიერი სახელით. (**MySQL**-ის მომხმარებელთა სიაში ასეთი მომხმარებლის სახელია -).

-**Host-Хост-ჰოსტი**-იმ კომპიუტერის ინტერნე-მისამართი, რომლიდანაც მოცემულ მომხმარებელს უფლება აქვს შევიდეს MySQL-ში. ნიშნაკი აღნიშნავს ნებისმიერ ინტერნეტ-მისამართს, ანუ ნებისმიერ კომპიუტერს.

-**Password-Пароль-პაროლი**-მომხმარებლის პაროლი. თუ აქ წერია **Нет** (არა), მაშინ მომხმარებელს გააჩნია "ცარიელი" პაროლი.

-**Grant-Предоставить-მინიჭება**-შეუძლია თუ არა მოცემულ მომხმარებელს სხვა მომხმარებლების უფლებების მართვა (სიტყვა **Yes-Да-კი**) ან (სიტყვა **No-Нет-არა**).

-**Action-Действие-მოქმედება**-ჰიპერმინიშნება **Edit-Правка-შესწორება**, რომელსაც გადავყავართ მომხმარებლის პარამეტრების შესწორების ვებ-გვერდზე.

MySQL-ის დაყენებისთანავე მის სიაში ჩნდება ოთხი მომხმარებელი:

- **%@%** - ნებისმიერი სახელის მატარებელი მომხმარებელი, რომელსაც ნებისმიერი კომპიუტერიდან შეუძლია ჩართვა და რომელსაც ძალზე შეზღუდული უფლებები გააჩნია;

-**%@localhost** - მომხმარებელი ნებისმიერი სახელით, რომელსაც შეუძლია ლოკალური კომპიუტერიდან ჩართვა და რომელსაც პრაქტიკულად მონაცემთა ბაზების შექმნასთან და გამოყენებასთან დაკავშირებული ყველა უფლება გააჩნია;

- **root@%** - მომხმარებელი სახელით root, რომელსაც შეუძლია ნებისმიერი კომპიუტერიდან ჩართვა და რომელსაც ადმინისტრატორის უფლებები გააჩნია;

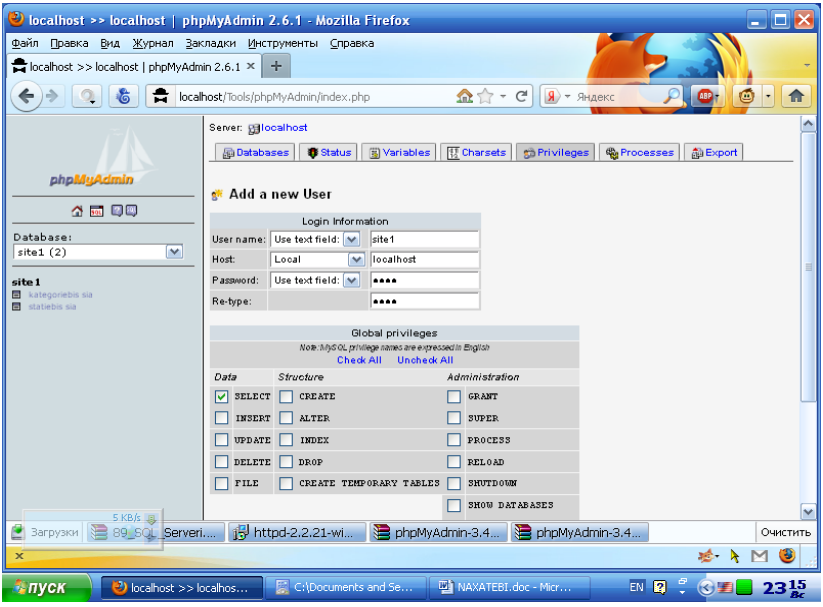
-**root@localhost**-მომხმარებელი სახელით root, რომელსაც შეუძლია ლოკალური კომპიუტერიდან ჩართვა და რომელსაც ადმინისტრატორის უფლებები გააჩნია.

ყველა ამ მომხმარებლებს გააჩნიათ "ცარიელი" პაროლები. მაგრამ რისთვის არიან ისინი საჭირო?

იმისთვის, რომ **MySQL**-ის დაყენებისთანავე ჩვენ შეგვეძლოს მასთან მიერთება და მონაცემთა ბაზებისადმი წვდომის ისეთი უფლებების განსაზღვრა, რომლებიც ჩვენ გვჭირდება. **MySQL** გვიხსნის "კარებს", რათა ჩვენ შევძლოთ მასში შესვლა და "ბოქლომის" შიგნიდან დადება. თუკი ჩვენ ამას არ გავაკეთებთ, ყველას ვისაც არ დაეზარება შეეძლება "კერებში" შესვლა და ჩვენი მონაცემთა ბაზებისთვის ნებისმიერი სახის ზიანის მიყენება.
მაშასადამე, პირველი, რაც ჩვენ უნდა გავაკეთოთ-ეს არის ის, რომ სიაში დავამატოთ მომხმარებელი, რომლის სახელი იქნება site და პაროლიც იქნება site. მეორე ჩვენი ნაბიჯი უნდა იყოს, ყველა მომხმარებლის ამოშლა, გარდა **root@localhost** და **site@localhost** მომხმარებლებისა.

მომხმარებლის შექმნა

ახალი მომხმარებლის დასამატებლად უნდა დავაწკაპუნოთ ჰიპერმინიშნებას **Add a New User-Добавить нового пользователя-ახალი მომხმარებლის დამატება**. აღნიშნულის საპასუხოდ **phpMyAdmin** გამოიტანს ვებ-გვერდს, დასამატებელი მომხმარებლის პარამეტრების განმსაზღვრელი ფორმით (ნახ. დ.4.7), რომელიც ჩვენ უნდა შევავსოთ.



ნახ. დ.4.7. მომხმარებლის პარამეტრების განმსაზღვრელი ფორმა

მომხმარებლის სახელის განსასაზღვრავად, საჭიროა ჩამოსაშლელ სიაში **User Name-Имя пользователя-მომხმარებლის სახელი**, **Use text fild-Использовать текстовое поле-ტექსტური ველის გამოყენება** პუნქტის არჩევა და მარჯვენა მხარეს მდებარე შესატან ველში საჭირო სახელის შეტანა. იმისათვის, რომ განვსაზღვროთ მომხმარებელი ნებისმიერი სახელით (%), საკმარისია ამ სიაში უბრალოდ პუნქტ **AnyUser-Любой пользователь-ნებისმიერი მომხმარებელი**-ს არჩევა.
კომპიუტერი, რომლიდანაც მოცემულ მომხმარებელს MySQL-თან მიერთების უფლება აქვს ანალოგიურად განისაზღვრება. ჩამოსაშლელ სიაში **Хост(ჰოსტი)** ვირჩევთ პუნქტს:

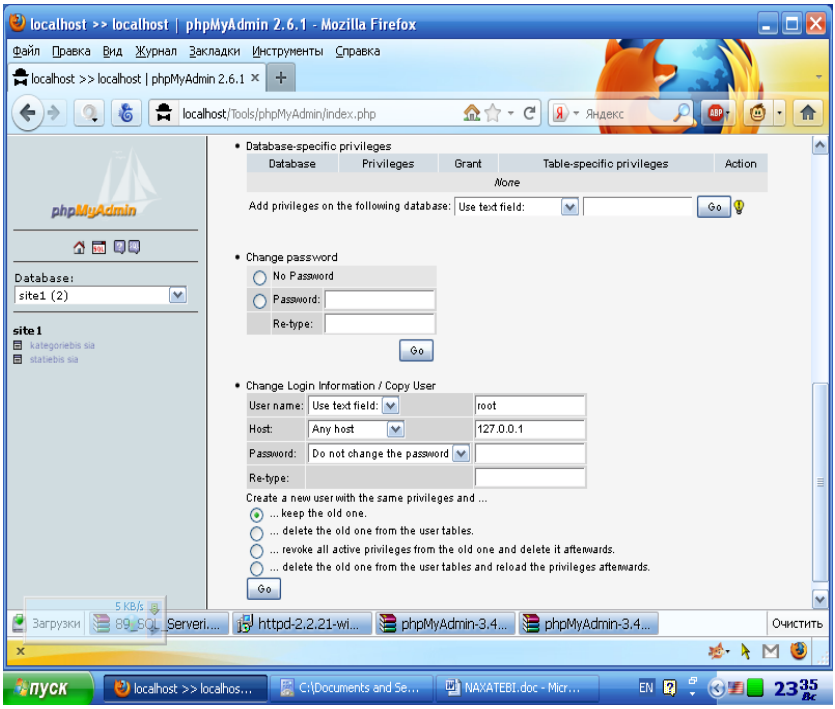
-**Any host_Любой хост-ნებისმიერი ჰოსტი**-ნებისმიერი კომპიუტერის განსასაზღვრავად (%);
-**Local-локальный-ლოკალური-ლოკალური** კომპიუტერის განსასაზღვრავად (localhost);
-**Use text fild-Использовать текстовое поле**-ტექსტური ველის გამოყენება-ნებისმიერი ინტერნეტ-მისამართის განსასაზღვრავად. ამის შემდეგ საჭირო მისამართი შეიყვანება მარჯვენა მხარეს განთავსებულ შესატან ველში.

ახლა ჩვენ დაგვრჩა პაროლის განსაზღვრა. ჩამოსაშლელ სიაში **Password-Пароль-პაროლი** ვირჩევთ პუნქტს **Use text fild-Использовать текстовое поле-ტექსტური ველის გამოყენება**, ამ სიის მარჯვენა მხარეში განთავსებულ შესატან ველში შეგვაქვს საჭირო პაროლი, ხოლო შემდეგ იგივე პაროლი შეგვაქვს შესატან ველში **Re-Type-Подтверждение-დამოწმება.** თუ საჭიროა "ცარიელი " პაროლის შეტანა, მაშინ საკმარისი იქნება სიაში **Password-Пароль-პაროლი**, პუნქტის **Without password- Без пароля-პაროლის გარეშე** არჩევა.
ყველა ამ სიებისა და შესატანი ველების ქვემოთ განთავსებულია ალმების დიდი ანაკრები, რომლებიც განსაზღვრავენ სერვერისადმი, ანუ მისი ყველა ბაზებისადმი მომხმარებლის წვდომის უფლებებს. ამ ალმებს გააჩნიათ თავიანთი დასახელებები, რომლებიც ემთხვევა SQL ენის ანალოგიური გასაღები სიტყვების დასახელებებს (ეს ენა

მოკლედ არის აღწერილი მე-**6 თავში).** მაშასადამე, იმისათვის, რომ მომხმარებელს მიეცეს ყველა ბაზებიდან მონაცემების წაკითხვის შესაძლებლობა, საჭიროა ალმის **SELECT(არჩევა)** ჩართვა.

რადგანაც ჩვენ არ ვაპირებთ ჩვენი ახალი **site** მომხმარებლისათვის არცერთი ბაზიდან, გარდა site ბაზისა მონაცემების წაკითხვის შესაძლებლობის მიცემას, მაშინვე შეგვიძლია დავაჭიროთ ღილაკს **Go-Пошел-წავიდა**. აღნიშნულის შემდეგ **phpMyAdmin** გამოიტანს ახალ გვერდს, სადაც ჩვენ შეგვეძლება ცალკეულ ბაზებში წვდომის უფლებების განსაზღვრა (აგრეთვე სერვერისადმი წვდომის უფლებების, სახელებისა და პაროლების შეცვლა, თუკი ჩვენ ასეთი სურვილი გაგვიჩნდება).

ეს გვერდი, ანუ მასზე განთავსებული, ცალკეულ მონაცემთა ბაზებისათვის, სპეციფიკური წვდომის უფლებების განმსაზღვრელი ფორმა ნაჩვენებია ნახ. დ.4.8-ზე.

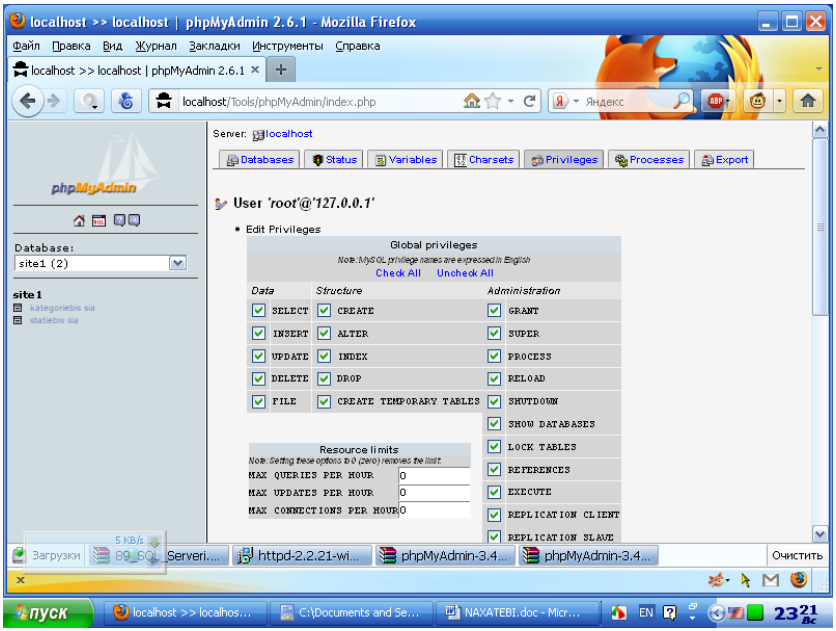


ნახ. დ.4.8. ცალკეული მონაცემთა ბაზებისათვის სპეციფიკური წვდომის უფლებების განმსაზღვრელი ფორმა.

ეს ფორმა ორგანიზებულია ცხრილის სახით, რომლის სვეტებსაც ჩვენ ახლა ჩამოვთვლით:

-**Database-БД-მონაცემთა ბაზები** - განსაკუთრებული უფლებების მქონე მონაცემთა ბაზის სახელი;
-**Privileges-Привилегии-პრივილეგიები**-SQL ენის გასაღები სიტყვების სია, რომელთა საშუალებითაც იქმნებიან მომხმარებლისათვის ნებადართული ბრძანებები და მოთხოვნები **(SELECT, INSERT, UPDATE, DELETE** და სხვა);
-**Grant-Предоставлять-მინიჭება**-შეუძლია თუ არა მოცემულ მომხმარებელს სხვა მომხმარებლების უფლებების მართვა (სიტყვა **Yes-Да-კი**) ან (სიტყვა **No-Нет-არა**);
-**Table-specific privileges-Привилегии, специфичные для таблицы-ცხრილებისათვის სპეციფიკური პრივილეგიები**- მონაცემთა ბაზის ცალკეული ცხრილებისათვის განსაზღვრული უფლებები;
-**Action-Действие-მოქმედება-** ჰიპერმინიშნებები **Edit-Правка-ჩასწორება** და **Cancel-Отменить-გაუქმება**. მათი დანიშნულება ისედაც ცხადია და არ საჭიროებს განმარტებას.

თავდაპირველად ეს ფორმა-ცხრილი ცარიელია, ვინაიდან ჩვენ ჯერ კიდევ არა გვაქვს განსაზღვრული **site** მონაცემთა ბაზებისადმი წვდომის არანაირი უფლებები. ამიტომ ჩამოსაშლელ სიაში **Add privileges on the following Database-Добавить привилегии на следующую базу-შემდეგ მონაცემთა ბაზაში პრივილეგიების დამატება**-ვირჩევთ პუნქტს **site** (ჩვენს მონაცემთა ბაზას) და მაშინათვე ვხვდებით მონაცემთა ბაზებისადმი წვდომის უფლებების განსაზღვრის ვებ-გვერდზე (ნახ. დ.4.9).



ნახ.დ.4.9. ცალკეულ მონაცემთა ბაზაში შეღწევის უფლებების განმსაზღვრელი ფორმა.

აქ ჩვენ ისევ ვხედავთ ალმების ანაკრებს, რომლებიც განსაზღვრავენ უფლებებს SQL-ის შესაბამისი მოთხოვნების შესრულებაზე. ჩვენი site მომხმარებლისათვის უნდა ჩავრთოთ **Data-Данные-მონაცემები** ჯგუფში შემავალი ყველა ალამი და ასევე ჯგუფში **Administration-Администрирование-ადმინისტრირება** შემავალი ალამი **LOCK TABLES**. რატომ? ამაზე ჩვენ ქვემოთ ვისაუბრებთ.

ჯგუფ **Data-Данные-მონაცემები**-ში შემავალი ალმები მომხმარებელს აძლევენ მონაცემების მიღებასთან, დამატებასთან, ჩასწორებასთან და წაშლასთან დაკავშირებული მოთხოვნების შესრულების შესაძლებლობებს. ესენია ჩვენთვის მე-6 თავიდან ნაცნობი ალმები **SELECT**, **INSERT**, **UPDATE** და **DELETE**. ვინაიდან მომხმარებელ **site**-ს მოუწევს მონაცემებთან მუშაობა, მას უნდა გააჩნდეს უფლება ყველა ზემოთ აღნიშნული მოთხოვნების შესრულებაზე.

ალმების ჯგუფი **Structure-Структура-სტრუქტურა** უშვებს ან კრძალავს მონაცემთა ბაზების, ცხრილების და ინდექსების შექმნაზე,

აგრეთვე ჩასწორებაზე ან წაშლაზე **SQL** მოთხოვნების შესრულების შესაძლებლობებს.
ჩვენი მონაცემთა ბაზა **site** ყველა მისი ცხრილებისა და ინდექსების ჩათვლით ჩვენ შევქმენით root მომხმარებლის-სერვერის ადმინისტრატორის სახელით, რომელსაც აბსოლუტურად ყველაფრის უფლება გააჩნია. მომხმარებელს **site**-ს არ მოუწევს ცხრილებისა და ინდექსების შექმნა-ის მხოლოდ მათში მოთავსებულ ინფორმაციასთან იმუშავებს. ეს კი ნიშნავს იმას, რომ ყველა ამ მონაცემთა ბაზების, ცხრილების და ინდექსების შექმნასთან, ჩასწორებასთან და წაშლასთან დაკავშირებული უფლებები მას არ გამოადგება.
ალმების ჯგუფი **Administration-Администрирование-ადმინისტრირება** განსაზღვრავს უფლებებს დამხმარე ოპერაციების შესრულებაზე. ამ ჯგუფიდან ჩვენ დაგვჭირდება უფლება მხოლოდ **LOCK TABLES**(ცხრილების ბლოკირება) მოთხოვნის შესრულებაზე და შესაბამისად მხოლოდ ამ ალმის ჩართვა დაგვჭირდება.

შენიშვნა! *მონაცემთა ბაზებში წვდომის უფლებების განმსაზღვრელი ალმების ჯგუფის ქვემოთ მდებარეობს ფორმა-ცხრილი, რომლის საშუალებითაც შესაძლებელია ამ ბაზაში შემავალი ცალკეული ცხრილებისადმი წვდომის უფლებების მართვა. ის ნახ.დ.4.8.-ზე ნაჩვენები ფორმის ანალოგიურია და ისეთნაირადვე გამოიყენება. თვით ცხრილებისადმი წვდომის უფლებები კი განისაზღვრებიან მონაცემთა ბაზებისადმი წვდომის უფლებების განმსაზღვრელი ფორმის მსგავსი ფორმის საშუალებით (იხ. ნახ.დ.4.9).*

მაშ ასე, უფლებები განსაზღვრულია. დავაჭიროთ ღილაკს **Go-Пошел-წავიდა**. **phpMyAdmin**-ი **MySQL**-ს გაუგზავნის **SQL**-ის შესაბამის მოთხოვნებს, ხოლო **MySQL** შეასრულებს უფლებების განსაზღვრასთან დაკავშირებულ ყველა ოპერაციას. ამის შემდეგ ვებ-გვერდის ზედა ნაწილში ვეძებთ ჰიპერმინიშნებას **Privileges-Привилегии-პრივილეგიები**, ვაწკაპუნებთ მასზე და ვხვდებით მომხმარებლების მართვის ვებ-გვერდზე (იხ. ნახ.დ.4.6.)

მომხმარებლების ჩასწორება და ამოშლა.

ახლა ჩვენ გვჭირდება ყველა მომხმარებლის, გარდა **root@localhost** და **site@localhost** ამოშლა. ამის გაკეთება ძალზე მარტივია: მომხმარებლების მართვის ფორმა-ცხრილის მარცხენა სვეტში (იხ.

ნახ. დ.4.6) ჩავრთოთ შესაბამისი ალმები და დავაჭიროთ ღილაკს **Go-Пошел-წავიდა.**

ყურადღება! *მომხმარებლების, აგრეთვე ცალკეულ მონაცემთა ბაზებთან და ცხრილებთან წვდომის უფლებების ამოშლა, ხორციელდება გაფრთხილების გარეშე.*

რომელიმე მომხმარებლის პარამეტრების შეცვლა შესაძლებელია მომხმარებლების მართვის ფორმა-ცხრილის სვეტში **Action-Действие-მოქმედება** ჰიპერმინიშნების **Edit-Правка-ჩასწორება** დაწკაპუნებით. მომხმარებლის პარამეტრების ჩასწორება ხორციელდება ჩვენთვის კარგად ნაცნობი, ნახ. დ.4.7-დ.4.9-ზე ნაჩვენები ფორმების საშუალებით.

მონაცემებთან მუშაობა

სამწუხაროდ **phpMyAdmin**-ს არ გააჩნია თვით მონაცემებთან მუშაობის-დამატების, ჩასწორების და ჩანაწერების წაშლის, ასევე მათი ამოკრეფის მოხერხებული საშუალებები. უფრო ზუსტად, რაღაც საშუალებები მას რათქმა უნდა მაინც გააჩნია, მაგრამ ჩვენ თვითონ მოგვიწევს საჭირო **SQL** მოთხოვნების შეტანა შესაბამის ფორმაში.

მაშ ასე, ვებ-გვერდის ზედა მარცხენა კუთხეში განთავსებული ჩამოსაშლელი სიიდან ვირჩევთ ჩვენს მონაცემთა ბაზას. მას შემდეგ, რაც **phpMyAdmin**-ი გამოიტანს ცხრილების სიას, ვებ-გვერდის ზედა ნაწილში ვპოულობთ ჰიპერმინიშნებას **SQL** და ვაწკაპუნებთ მასზე. აღნიშნულის შედეგად გამოიტანება SQL მოთხოვნების შესატანი ფორმა (ნახ.დ.4.10).
აქ ყველაფერი მარტივად არის. შეგვაქვს **SQL**-მოთხოვნის კოდი რედაქტირების დიდ არეში და ვაჭერთ ღილაკს **Go Пошел-წავიდა**. მონაცემთა კლიენტი **phpMyAdmin**-ი გადასცემს ამ მოთხოვნას **MySQL**-ს და მისგან იღებს მოთხოვნის წარმატებით შესრულების დამადასტურებელ ან შეცდომის შესახებ შეტყობინებას. რადგანაც ჩანაწერის დამატებაზე ჩვენს მიერ შეტანილი მოთხოვნა (იხ. ნახ. დ.4.10) არ შეიცავს შეცდომებს, იგი შესრულდება.

MySQL მონაცემთა კლიენტების სხვა პროგრამები

პროგრამა **phpMyAdmin**-ი არ არის ერთად ერთი ხელმისაწვდომი კლიენტური პროგრამა **MySQL** მონაცემთა ბაზებთან სამუშაოდ. არსებობენ სხვა პროგრამებიც, რომელთაც ჩვენ აქ განვიხილავთ.

ამ წიგნის *დანართ 2*-ში ნახსენები იყო პროგრამა MySQL **Control Center**. ეს საკმაოდ მძლავრი მონაცემთა კლიენტია, რომლის საშუალებითაც შესაძლებელია მონაცემთა ბაზების, ცხრილების და ინდექსების შექმნასთან, აგრეთვე მათი მონაცემებით "შევსებასთან", დაკავშირებული ოპერაციების შესრულება. კერძოდ, **MySQL**-ის ცხრილებში მონაცემთა შესატანად ყველაზე მოხერხებულია სწორედ **MySQL Control Center.**

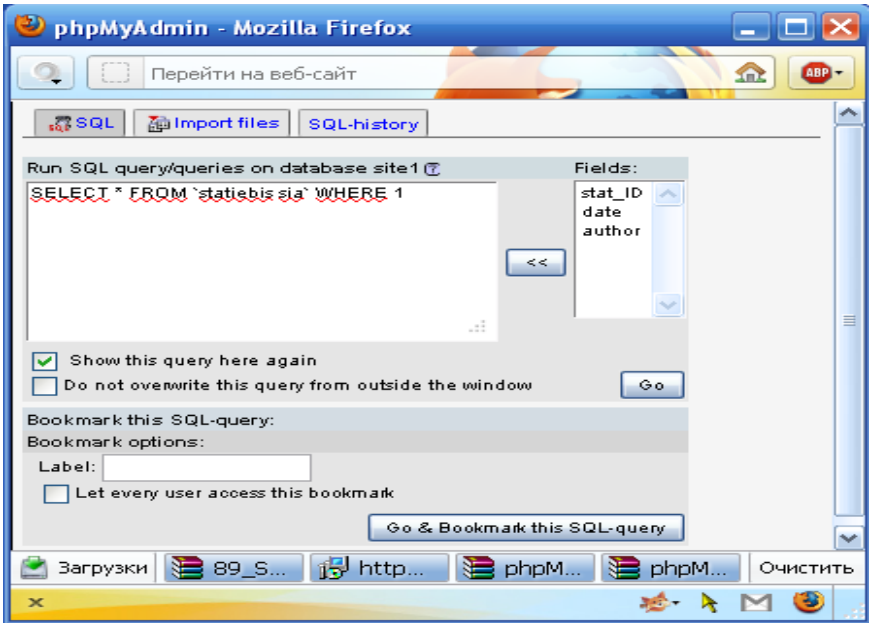
MySQL Control Center-ეს **Windows**-ის ჩვეულებრივი პროგრამაა, რომელიც მუშაობისათვის არ მოითხოვს არავითარ სხვა პროგრამებს (ცხადია, გარდა თვით **MySQL** სერვერისა). მისი დისტრიბუტივის მოძიება შესაძლებელია საიტზე **MySQL AB(http://www.mysql.com**).

არსებობს **MySQL** მონაცემთა კიდევ ერთი პროგრამა-კლიენტი-**EMS MySQL Manager**. მას პრაქტიკულად იგივე შესაძლებლობები გააჩნია, რაც **MySQL Control Center**-ს. მაგრამ ამავე დროს, გააჩნია აგრეთვე ზოგიერთი დამატებითი შესაძლებლობები და რამდენადმე უფრო მიმზიდველი ინტერფეისი. ამ პროგრამის მოძიება შესაძლებელია მისი შემმუშავებელი ჯგუფის საიტზე: **http://www.mysqlmanager.com.**

თუმცა ორივე დასახელებულ პროგრამას გააჩნია ერთი დიდი ნაკლოვანება. საქმე იმაშია, რომ მათი ნორმალური ფუნქციონირებისათვის კომპიუტერზე უნდა დაყენდეს **MySQL** სერვერის კლიენტური ნაწილიც. ხოლო ეს კლიენტური ნაწილი **MySQL** სერვერთან დასაკავშირებლად იყენებს პორტს 3306 (**TCP/IP** პორტების შესახებ იხ. *თავი1*). თუ ჩვენ "ვურთიერთქმედებთ" ლოკალურ კომპიუტერზე დაყენებულ სერვერთან, არავითარი პრობლემა არ გვექნება. მაგრამ თუკი ჩვენ ვცდილობთ დაშორებულ სერვერთან მიერთებას, მაშინ შეიძლება წავაწყდეთ ისეთ სიტუაციას, როდესაც ინტერნეტ-პროვაიდერს, უსაფრთხოების მიზნებიდან გამომდინარე დაბლოკილი ექნება აღნიშნული პორტი. (გამოუყენებელი პორტების ბლოკირება ჩვეულებრივი პრაქტიკაა).

phpMyAdmin-ის შემთხვევაში ასეთი პრობლემა არ არსებობს. როგორც უკვე ვიცით, ის წარმოადგენს **PHP**-ს სერვერული ვებ-გვერდების ანაკრებს, ყენდება იმავე სერვერზე, რაც **MySQL** და "ურთიერთქმედებს" მომხმარებელთან მე-80 პორტის საშუალებით,

რომლისადმი წვდომის შესაძლებლობა ყოველთვის ღიაა. ამასთანავე, როგორც წესი, **phpMyAdmin** უკვე დაყენებულია იმ დაშორებულ ვებ-სერვერებზე, რომლებიც კლიენტებს აძლევენ **MySQL**-ის გამოყენების შესაძლებლობას, ამიტომ მასთან მუშაობის გამოცდილება შემდგომშიც გამოგვადგება.



ნახ. დ.4.10. SQL-მოთხოვნების შესატანი ფორმა.

ამგვარად ჩვენ შეგვეძლება ჩვენს მიერ შექმნილი **site** მონაცემთა ბაზის ცხრილებში, დავამატოთ რამოდენიმე დამხმარე ჩანაწერი, რომელთაც შემდგომში სერვერული გვერდების გასამართავად გამოვიყენებთ. ასევე, ჩვენ შეგვიძლია მონაცემების **არჩევის(SELECT)**, **შეცვლის(UPDATE)**, **წაშლის(DELETE)** ნებისმიერი მოთხოვნის შესრულება.

პრინციპში, ამ ფორმის საშუალებით შესაძლებელია MySQL სერვერის მიერ მხარდაჭერილი ნებისმიერი **SQL** მოთხოვნის შესრულება, მათ შორის ცხრილების, ინდექსების, მომხმარებლების, უფლებების მართვის და სხვა მოთხოვნების შესრულება. თუმცა, რათქმა უნდა, ამ

მიზნებისათვის **phpMyAdmin**-ს სხვა, უფრო მოხერხებული ინსტრუმენტებიც გააჩნია.

გამოსვლა

მონაცემთა ბაზებთან და მომხმარებლებთან მუშაობის დასრულების შემდგომ, საჭიროა გამოსვლის ოპერაციის შესრულება. ამასთან **phpMyAdmin**-ი გაწყვიტავს ყველა კავშირს მონაცემთა სერვერთან და გაათავისუფლებს კომპიუტერის ოპერატიულ მეხსიერებას.

phpMyAdmin-იდან გამოსასვლელად, უნდა დავაწკაპუნოთ ჰიპერმინიშნებაზე **Home Page-Начальная страница-მთავარი გვერდი**, რომელიც მისი ნებისმიერი ვებ-გვერდის ზედა მარცხენა კუთხეში მდებარეობს. ამის შემდეგ **phpMyAdmin**-ი გამოიტანს თავის მთავარ ვებ-გვერდს, რომელზეც, ჰიპერმინიშნება **LogOut-Выйти из системы-სისტემიდან გამოსვლა**-ზე უნდა დავაწკაპუნოთ. ვებ-დამთვალიერებელი ეკრანზე გამოიტანს მომხმარებლის სახელისა და პაროლის შესატან სტანდარტულ დიალოგურ ფანჯარას, რომელშიც უბრალოდ ღილაკ **Cansel-Отмена-გაუქმება**-ზე დაჭერა დაგვჭირდება.

თუ აღნიშნულის შემდეგ ვებ-დამთვალიერებლის ფანჯარაში გამოჩნდება სტრიქონი **Wrong username / password. Access denied./Ошибочный логин/пароль. В доступе отказано./ არასწორია მომხმარებლის სახელი/პაროლი. შესვლა აკრძალულია.**, ეს იმას ნიშნავს, რომ გამოსვლის ოპერაცია წარმატებით შესრულდა.

გამოყენებული ლიტერატურა:

1.Tarin Towers,Abie Hadjitarkhani,Sasha Magee-Macromedia DreamweaverMX, Advanced for Windows and Macintosh: Visual Quick Pro Guide. Изд.: Peachpit Press; 2002г., 512 стр..

2. Горев А.,Ахаян Р., Макашаринов С. Эффективная работа с СУБД.-СПб.: Питер, 1997.

3. Карпова Т.С. Базы данных: модели, разработка, реализация.-СПб.: Питер, 2001.

4.Тим Конверс, Джойс Парк и Кларк Морган. PHP5 и MySQL. Библия пользователя. Изд.: Диалектика, 2006г, 1217стр.

5. Ларри Ульман-Основы программирования на PHP. Изд.: ДМК пресс, 2001, 288стр..

6. Бурцов М.В., Доменко В.М., Гаврилин Д.А. Николаев Д.Г.-Основы работы с HTML-редактором Dreamweaver. Изд.: СПб: СПб ГИМО(ТУ), 2002, 104стр..

7.Э.В. Фуфаев, Д.Э. Фуфаев. Базы Данных. –Москва, Изд.центр «Академия», 2007

8. Джеффри Ульман, Дженифер Уидом- Введение в системы баз данных

9.Abiteboul,S., R. Hull and V.Vianu, Foundations of Databases, Addison-Wesley, Reading, MA 1995

10. Д. Кренке. Теория и практика построения Баз данных, 8-е издание; Питер, 2003

11.Лещев Д. Создание интерактивного Web-сайта: учебный курс-СПб.: Питер, 2003. -544 стр.

12.Вин Дж. Искусство Web-дизайна: Самоучитель. СПб.: Питер, 2002.

13.Нидерст Дж. Web-мастеринг для профуссионалов. СПб.: Питер, 2001.

14.Федорчук А.В. Как создаются Web-сайты: Краткий курс. СПб.: Питер,2000.

15. Холмогоров В. Основы веб-мастерства: Учебный курс.2-е изд. СПб.: Питер, 2003.

16. Якушин Е.В. Изучаем Интернет, создаем Web-страничку. СПб.: Питер, 2001

17. Гончаров А.Ю. Самоучитель HTML. СПб.: Питер 2001.

18. Холмогоров В., Солоницын Ю. Интернет: Энциклопедия. 3-е изд. СПб.: Питер, 2002.

19. Денисов А.Р. Internet Explorer 5.5: Справочник. СПб.: Питер, 2001.
20. Хольцер С. Dynamic HTML: руководство разработчика: пер. с англ.-К.: Издательская группа BHV, 2003.
21. Айзек С. Dynamic HTML. Пер. с англ.-СПб.: BHV-Санкт-Петербург, 2004.
22. Хилайер С., Мизик Д. Программирование Active Server Pages. Пер. с англ.-3-е изд., доп.-М.: Издательско-торговый дом «РусскаяРедакция», 2003.
23.Токарев С. Самоучитель Micromedia Dreamweaver MX, Изд. BHV-2003; 544стр..
24.Michael Doyle,Macromedia Dreamweaver e-Learning Toolkit, Изд.Wiley, 2003г; 547 стр..
25.А.Божко, Dreamweaver MX, Базовый курс; Изд.: Кудиц-образ, 2003г, 576стр..
26.Джанин Уорнер, Сюзанна Гарднер-DreamweaverMX2004 для чайников; Изд “Вильямс”, 2004g, 352стр..
27. Дронов В. –PHP, MySQL и Dreamweaver MX 2004, Разработка Интерактивных Web-сайтов. Санкт-Петербург, «БХИ-Петербург», 2005.-448 с.
28. Дронов В. –Macromedia Dreamweaver MX, Наиболее полное руководство Санкт-Петербург,Изд.: «БХИ-Петербург», 2005,448стр.
29. Петюшкин А. HTML в Web-дизайне 400с.
30.Дюбуа Поль, My SQL: Уч. пос; пер. с английского М.: Издательский дом “Вильямс”, 2001-816 стр.
31.Ланга Томсон, Люк Веллинг-Разработка Web-приложений на PHP и MY SQL. Изд.: ДиаСофтЮП, 2003г, 672стр..

რამაზ შამუგია
ტექნიკურ მეცნიერებათა კანდიდატი (აკადემიური დოქტორი).

გამოქვეყნებული აქვს 18 სამეცნიერო ნაშრომი.

1953 - დაიბადა გალის რაიონში.

1970 - დაამთავრა თბილისის ე.კომაროვის სახელობის ფიზიკა-მათემატიკური სკოლა.

1976 - დაამთავრა თსუ-ს ფიზიკის ფაკულტეტი, ბირთვული ფიზიკის სპეციალობით.

1977-1984 - მუშაობდა თბილისის ავტომატიზაციის საშუალებათა სამეცნიერო კვლევით ინსტიტუტში წამყვან მეცნიერ მუშაკად.

1984-1995 - მუშაობდა სპი-ს მანქანათმშენებლობის ფაკულტეტის პროგრამული მართვის კათედრაზე, დოცენტად.

1995-2004 - მუშაობდა ტრანსპორტისა და კომუნიკაციების სამინისტროში სხვადასხვა საპასუხისმგებლო თანამდებობებზე.

2005-2012 - მუშაობს სოხუმის სახელმწიფო უნივერსიტეტის მონვეულ ლექტორად.

2010-2012 - მუშაობს ი.ვეკუას სახელობის სოხუმის ფიზიკა-ტექნიკურ ინსტიტუტში უფროს მეცნიერ თანამშრომლად.