**ლექცია № 1 -2. შესავალი PHP-ში. WEB-დაპროგრამების საფუძველი**

**PHP** არის **სერვერული სკრიპტების** ენა. იგი წარმოადგენს მძლავრი ინსტრუმენტს, რომლის საშუალებითაც სწრაფად მიიღება დინამიური და ინტერაქტიული ვებ-გვერდები.

PHP არის ფართოდ გამოიყენებადი, თავისუფალი - უფასო და ეფექტური ალტერ­ნატივა კონკურენტებისა, როგორიცაა Microsoft-ის ASP (**Active** **Server** **Pages** - აქტიური სერვერული გვერდები).

არცთუ დიდი ხნის წინ Web -კვანძებიდან - სერვერებიდან, მომხმარებელს ადგილზე ძირითადად სტატიკური HTML -ფაილები გადმოეცემოდა, რომელთაც იგი ბრაუზე­რის მეშვეობით მხოლოდ კითხულობდა. დღეს კი Web-სივრცეში უმეტესწილად დი­ნამიკური Web -ფურცლები ანუ DHTML -ფაილები გადაიგზავნება. მათი აწყობა ხდება Web-კვანძების სერვერზე განთავსებული Web -პროგრამების გაშვებით, რასაც მომხმარებელი ახორციელებს საკუთარი კომპიუტერიდან სისტემასთან დიალოგში.

ინტერნეტში პროგრამების შექმნისათვის დღეს ყველაზე პოპულარული Web-დაპ­როგრამების ენა გახლავთ PHP. იგი განსაკუთრებით აადვილებს სერვერებზე განლა­გებულ მონაცემთა ბაზებთან ურთიერთობას.

დასაწყისში PHP (პერსონალ ჰოუმ პეიჯ) შეიქმნა, როგორც მაკროსების კრებული, მარტივი სახის პირადი Web-ფურცლების ფორმირებისთვის. შემდეგ კი იგი იქცა სრულფასოვან დაპროგრამების ენად, რომელზეც დაწერილი კონსტრუქციები ჩაიდგმება სერვერზე მოთავსებულ HTML-ტექსტებში. Web -ფურცლის გამოძა­ხებისას PHP -ის პრეპროცესორის მიერ ადგილზე ხდება ამ კონსტრუქციების (პროგრამული ფრაგმენტების) დამუშავება და HTML-ის არსებულ ფრაგმენტებს შორის ჩაემატება

შესაბამისი ინფორმაცია, მაგალითად, ბაზიდან ამოღებული მონაცემები, სერვერზე კონსტრუირებული გრაფიკული გამოსახულებები და სხვ. PHP-ის ვერსია - PHP5 - მისაერთებელი მოდულის სახით კომპილირდება დღეისათვის ყველაზე პოპულა­რულ Web-სერვერ-პროგრამა Apache-სთვის. სერვერზე PHP -პროგრამები უმეტეს­წილად ურთიერთობენ MySQL მონაცემთა ბაზასთან (რომელიც უფასოდ ვრცელ­დება). ასე რომ, Apache, MySQL და PHP ტრიადას დღევანდელ დღეს კონკურენტი არ ჰყავს.

თუმცა, ცხადია, PHP-ს მუშაობა შეუძლია სხვა Web -გარემოში და სხვა ბაზებთანაც.

"მოვიყვანოთ მარტივი მაგალითი PHP-კოდისა HTML-ში.

**მაგალითად**

ლისტინგი 1

<!DOCTYPE html>  
<html>  
<body>  
<?php  
echo "ჩემი პირველი PHP სკრიპტი!";  
?>  
</body>  
</html>

PHP სკრიპტები მუშაობენ სერვერზე.

**PHP -ის შესასწავლად წინასწარ უნდა ვერკვეოდეთ ქვემოთ ჩამოთვლილ საკითხებში:**

* HTML
* CSS
* JavaScript

**რა არის PHP?**

* PHP არის აბრევიატურა და "PHP ჰიპერტექსტის Preprocessor"
* PHP არის ფართოდ გამოიყენებადი, ღია სკრიპტის ენის
* PHP სკრიპტები მუშაობენ სერვერზე
* PHP უფასოა, ჩამოტვირთეს და გამოიყენეთ

რას ნიშნავს **PHP - პრეპროცესორული ენაა?**

განმარტების თანახმად **პრეპროცესორი** წარმოადგენს პროგრამას, რომლის შესა­ვალზე მოიცემა მონაცემები, ხოლო გამოსავალზე მიიღება ასევე მონაცემები ოღონდ უკვე სხვა პროგრამისთვის, მაგალითად კომპილატორისთვის. პრეპროცესორის გამო­სავალზე არსებულ მონაცემებზე ამბობენ, რომ ისინი იმყოფებიან **პრეპროცესირებულ** ფორმაში და არიან მზად გამოყენებულ იქნან სხვა პროგრამების მიერ.

პრეპროცესორის დონეზეა დამოკიდებული მონაცემების დამუშავება და მათი სახე; ასე მაგალითად, ზოგიერთ პრეპროცესორს შეუძლია მხოლოდ ჩაანაცვლის ტექსტები, ზოგიერთს კი მაღალი დონის დაპროგრამირების ენის პრეტენზიაც კი გააჩნია. ძალზე ხშირად პრეპროცესორი გამოიყენება საწყისი კოდის დასამუშავებლად იმის წინ, როდესაც ეს კოდი უნდა გადასცენ კომპილაციისთვის. ასე მაგალითად **C**, **C++**, **TEX** იყენებენ პრეპროცესორს, რითაც იზრდება მათი საშუალებები. პრეპროცესორი C/ C++ -ში წაშლის კომენტარებს, გარდაქმნის საწყის კოდს მაკროსების შესაბამისად, და შეასრულებს იმ დირექტივებს, რომელიც იწყება #-სიმბოლოთი, მაგალითად # include, #define ან #pragma და სხვა.

PHP როგორც პრეპროცესორი გამოიყენება ვებ-გვერდების დასამუშავებლად. გვერდის ტექსტი წაიკითხება და იგი იმავე სახით გამოდის გამოსავალზე, მხოლოდ ერთი შესწორებით - ტექსტში არსებული ნაწილი **<?php** -სა და **?>**-ს შორის შესაბამისად შეიცვლება და მიეწოდება ბრაუზერს.

ვთქვათ გვაქვს საწყისი კოდი -

ლისტინგი 2

<!DOCTYPE html>

<**html**>

<**head**>

<**title**>მიმდინარე დრო</**title**>

<**head**>

<**body**>

<**h1**>>მიმდინარე დრო</**h1**>

<?php

print strftime('ამჟამად %H საათია, %M წუთი და %S წამია');

?>

</**body**>

</**html**>

PHP - პრეპროცესორის შემდეგ მივიღებთ:

<!DOCTYPE>

<**html**>

<**head**>

<**title**>მიმდინარე დრო</**title**>

<**head**>

<**body**>

<**h1**>>მიმდინარე დრო</**h1**>

ამჟამად 10 საათია, 12 წუთი და 31 წამია

</**body**>

</**html**>

**რა არის PHP ფაილი?**

* PHP ფაილი შეიძლება შეიცავდეს ტექსტს, HTML, CSS, JavaScript და PHP კოდს
* PHP კოდი სრულდება სერვერზე, და შედეგი ბრუნდება ბრაუზერში, როგორც უბრალო HTML
* PHP ფაილებს აქვთ გაფართოება ".php"

**რა შეიძლება გააკეთოს PHP-მ?**

* PHP -ს შეუძლია დინამიური გვერდის კონტენის გენერირება
* PHP-ით შეიძლება ფაილის შექმნა, გახსნა, წაკითხვა, ჩაწერა, წაშლა და დახურვა სერვერზე
* PHP-ს შეუძლია შეაგროვოს ფორმის მონაცემები
* PHP-ით შეგიძლიათ გააგზავნოთ და მიიღოთ cookies
* PHP -ით შეგიძლიათ დაამატოთ, წაშალოთ, ცვლილებები შეიტანოთ მონაცემები თქვენს მონაცემთა ბაზაში
* PHP-ს შეუძლება შეზღუდოს მომხმარებლების წვდომა თქვენი საიტის ზოგიერთ გვერდზე
* PHP-ით შეგიძლიათ დაშიფროთ მონაცემები

PHP-თი თქვენ არ შემოიფარგლებით მხოლოდ გამომავალი HTML-კოდით. თქვენ შეგიძლიათ მიიღოთ გამომავალი სურათები, PDF ფაილი, და Flash ფილმები. ასევე შეგიძლიათ გამომუშავდეს ნებისმიერი ტექსტი, როგორიცაა XHTML და XML.

**რატომ PHP?**

* PHP გაიშვება სხვადასხვა პლათფორმაზე (Windows, Linux, Unix, Mac OS X, და ა.შ.) - ანუ ის კროსპლათმორფულია
* PHP გაიშვება თითქმის ყველა იმ სერვერზე, რომელიც გამოიყენება დღეს (Apache, IIS და ა.შ.)
* PHP მხარს უჭერს ფართო სპექტრს მონაცემთა ბაზებისა
* PHP უფასოა. გადმოწეროთ ოფიციალური PHP რესურსი: [www.php.net](http://www.php.net/)
* PHP არის მარტივად შესასწავლი და ეფექტურად მუშაობს სერვერის მხარეს

**ვებ-დაპროგრამების საფუძვლები**

მაღალი დონის დაპროგრამების ენებზე დაწერილი პროგრამის რეალიზაციას აქვს მარტივი სახე: შეიყვანთ ბრძანებას (დააჭერთ ღილაკს) - იწყება პროგრამის რეალიზა­ცია - გამოდის შედეგი. და ეს ყველა პროცედურა ხდება ძირითადად ერთ კომპი­უტერზე.

ვებ-დაპროგრამებაში განსხვავებული სურათია, ასე მაგალითად:

1. ბრაუზერი **გახსნის კავშირს** სერვერთან;
2. ბრაუზერი აგზავნის სერვერზე მოთხოვნას გვერდის მიღებაზე;
3. სერვერი ახდენს პასუხის ფორმირებას (უფრო ხშირად HTML კოდის სახით, უგზავნის მას ბრაუზერს და **დახურავს მასთან კავშირს**;
4. ბრაუზერი შეასრულებს HTML-კოდს და ასახავს მონიტორზე ვებ-გვერდს.

მუშაობის ასეთ სქემას ეწოდება **კლიენტი - სერვერი**. კლიენტი აქ ბრაუზერია.

სერვერი

კლიენტი

ნახატი 1

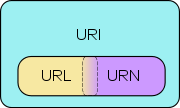
ყველა ეს ეტაპი ვებ-დაპროგრამებაში სასურველია გააზრებულ იქნას ღრმად. ამისთვის დაგვჭირდება ზოგიერთი ფაქტის გახსენება წინა კურსებიდან - HTML, JS, ქსელები.

ინტერნეტ ქსელის რესურსები შეიძლება დავყოთ **აბსტრაქტულ** (სოფტი) და **ფიზიკურ** რესურსებად. აბსტრაქტულ რესურს მიეკუთვნება HTML-დოკუმენტი, გრაფიკული გამოსახულება, აპლიკაცია, შემსრულებელი ფაილები და სამსახურები, ხოლო ფიზიკურ რესურსებს - კომპიუტერები და სერვერები.

ინტერნეტ ქსელის ნებისმიერ რესურს გააჩნია თავისი იდენტიფიკატორი შემუშავე­ბული **URI**-ს (**Uniform Resource Identifier** - რესურსების უნიფიცირებული იდენტი­ფიკატორი) საფუძველზე. სწორედ იდენტიფიკატორის საშუალებით შეიძლება ამა თუ იმ რესურსის მოძიება. სახვადასხვა ტიპის ქმედებების განხორციელებისას (წვდომა, განახლება, ძებნა და სხვა) რესურსებზე, რესურსის იდენტიფიკატორი გამოიყენება როგორც ნაწილი ლინკის (მიმთითებლის) სამისამართო ნაწილისა.

გავეცნოთ უფრო დეტალურად **URI**-ს.

**URI**-ის გრამატიკის და სინტაქსისის გაფართოებული ვერსია აახლებს და განიხილავს თავის ქვესიმრავლეებად **URL**-ს (Uniform Resourse Locator - რესურსების ადგილმდებარეობის უნივერსალური მიმთითებელი) და **URN** (Uniform Resourse Name - რესურსის უნიფიცირებული სახელი).



ნახატი 2.

URL ეს იგივე URI-ა, სადაც რესურსის იდენტიფიკაციასთან ერთად მითითებულია ინფორმაცია ამ რესურსის ადგილმდებარეობაზე.

URN - ეს URI-ა რომელიც ახდენს რესურსის იდენტიფიკაციას სახელთა სივრცეში> მაგრამ არ უთითებს ამ რესურსის ადგილმდებარეობაზე. მაგალითად urn:ISBN:0-395-36341-1 ეს URI-ა, რომელიც მიუთითებს რესურსზე 0-395-36341-1 - წიგნზე ISBN რესურსთა სახელების სივრციდან

**URI** განსაზღვრავს იდენტიფიკატორის ორ ფორმას: აბსოლუტურს და ფარდობითს.

URI-ში ჩაუწერება მხოლოდ ლათინური ანბანის ასოები დ აზოგიერთი სიმბოლო. სხვა ენის ანბანის ასოები აქ შეიძლება გამოყენებულ იქნას მხოლოდ როგორც იუნიკოდში გადაყვანილი: მაგალითად სტრიქონი -

**ka.wikipedia.org/wiki/ივანე\_ჯავახიშვილი**

ტრანსფორმირდება (თითო ქართულ ასოს - 9 სიმბოლოს კომბინაცია შეესაბამება) URI-ში შემდეგნაირად (ე.წ. - UrlEncoded- სახით %XX - სადაც XX - სიმბოლოს თექვსმეტობითი კოდია)

http://ka.wikipedia.org/wiki/%E1%83%98%E1%83%95%E1%83%90%E1%83%9C%E1%83%94\_%E1%83%AF%E1%83%90%E1%83%95%E1%83%90%E1%83%AE%E1%83%98%E1%83%A8%E1%83%95%E1%83%98%E1%83%9A%E1%83%98

ეს ზოგიერთის მოსაზრებებით, არის სხვა ენების დისკრიმინაცია და ეწინააღმდეგება მსოფლიო აბლაბუდის ინტერნაციონალიზაციის პრინციპს, რომელიც აღიარებულია ინტერნეტის წამყვანი ორგანიზაციების, მათ შორის [**W3C**](https://ru.wikipedia.org/wiki/W3C) -ს და [**ISOC**](https://ru.wikipedia.org/wiki/ISOC) -ის მიერ. ამ პრობლემის დასაძლევად შემოღებულია სტანდარტი **IRI** (**Internationalized Resource Identifier** - რესურსის ინტერნაციონალური იდენტიფიკატორი).

ზოგადად აბსოლუტური,სქემაზე დამოკიდებული **URI** იდენტიფიკატორი შეიძლება ჩაიწეროს ასეთნაირად:

**<სქემა>:[//<<ჰოსტის სახელი | სამსახურის სახელი>,<ძირითადი DNS სუფიქსი>>|< IPv4 | IPv6>][<:პორტი>]/<გზა რესურსისკენ>?<მოთხოვნა>[<#ფრაგმენტი>]**

სადაც

* ”<” და ”>” სიმბოლოები ტეგის აღმნიშვნელი კი არ არის, არამედ ბეკუს-ნაურუს ფორმალური გრამატიკის ნოტაციის შემადგენელი ნაწილი და მათ შორის მოთავსებულია *სინტაქსიკური ერთეული*.
* კომპონენტი **<სქემა>** ორწერტილთან ”**:**”ერთად განსაზღვრავს, თუ რა საშუალე­ბით - მეთოდით მოხდება წვდომა. უმეტეს შემთხვევაში აქ ჩაიწერება პროტო­კოლები, თუმცა არა მხოლოდ ისინი, მაგალითად - [http](https://ru.wikipedia.org/wiki/Http), [ftp](https://ru.wikipedia.org/wiki/Ftp), [file](https://ru.wikipedia.org/wiki/File), [ldap](https://ru.wikipedia.org/wiki/Ldap), mailto, urn და სხვა;
* ორმაგი სლეში ”//” მიუთითებს ქსელურ გზას;
* **<<ჰოსტის სახელი | სამსახურის სახელი>,<ძირითადი DNS სუფიქსი>>** წაემოადგენს სრულ უფლებამოსილ დომენურ სახელს (FQDN);
* სიმბოლო “ | ” გამოიყენება ალტერნატივის არსანიშნავად , ხოლო კომპონენტი ”[]” - არა აუცილებლობის აღსანიშნავად, ანუ კვადრატულ ფრჩხილებში მითი­თებული პარამეტრების არსებობა **URI**- ში, არაა სავალდებულო;
* კომპონენტი **<გზა რესურსისკენ>** შეიძლება შედგებოდეს რამდენიმე სეგმენ­ტისგან, რიმლებიც სლეშით ”/” არიან გამოყოფილი;
* ზოგიერთი **URI** უთითებენ დამატებით ადგილს რესურსისი შიგნით, ამ შემ­თხვევაში **URI** მთავრდება ღუზით **#** - , რომლის შემდეგ მოდის საძებნი ფრაგ­მენტის იდენტიფიკატორი.

მაგალითები -ის გამოყენების:

* <http://www.phpfaq.ru/php5.ru/study/chapter1.html> - აქ სქემა არის [http](https://ru.wikipedia.org/wiki/Http) პროტოკოლი და ის უზრუნველყოფს რესურსის მიწოდებას ბრაუზერისადმი;
* <mailto:tvn@caucasus.net> - სქემა მეილის გაგზავნა, მხოლოდ ერთი მისამართია-მიმღების მითითებული;
* <mailto:tvn@caucasus.net,kereselidzenugzari@gmail.com?subject=Hi>! - სქემა მეილის გაგზავნა, მითითებულია ორი მისამართია მიმღებების თემის შეტყობინებით ”Hi!”;
* <telnet://towel.blinkenlights.nl/> - telnet სამსახურისათვის, აბონენტს აძლევს საშუალებას დაშორებულმა - როგორც ტერმინალმა იმუშაოს სხვა მანქანასთან. ამ მისამართის ჩაწერის შემდეგ ბრაუზერის სამისამართო სტრიქონში უნდა მოხდეს შეერთება, ანაც ვხსნით command prompt და იქ ავკრიფავთ telnet towel.blinkenlights.nl -ს
* <ftp://ftp.geo.gov.ge> – FTP პროტოკოლის სქემა, რომლითაც ხდება დიდი მოცულობისა და სხვადასხვა ტიპის მონაცემების გადაგზავნა;
* http://www.hmn.ru:8101/ - მითითებულია TCP-პორტის ნომერი, რომლითაც დავუკავშირდებით რესურს;

# 

* <news:comp.infosystems.www.servers.unix> - ტელეკონფერენციების სამსახურებისთვის.

ფარდობითი იდენტიფიკატორი URL-ისა არ შეიცავს ინფორმაციას სქემის შესახებ. გზა მიუთითებს იმ რესურსის მისამართზე, სადაც ამჟამად იმყოფება დოკუმენტი.

**სერვერზე მონაცემთა გაგზავნის მეთოდები**

რა ხდება მას შემდეგ, როცა მომხმარებელმა ბრაუზერის სამისამართო სტრიქონში შეიყვანა რესურსის URL და ის შესრულებაზე გაუშვა? ამის შემდეგ კლიენტის პროგრამა - ანუ ბრაუზერი ქმნის -მოთხოვნას, რომელიც უსათუოდ შეიცავს ამ რესურსის URL-ს.

http-პროტოკოლის ფარგლებში მოთხოვნისა და პასუხის შეტყობინებას აქვს საერთო ფორმატი. http-შეტყობინება არის უბრალო ტექსტი. ისევე როგორც მოთხოვნის, ასევე პასუხის წინ წარმოჩინდება საწყისი სტრიქონი (start line). ყველაფერი დანარ­ჩენი არის ფაკულტატიური - არა აუცილებელი. საწყისი სტრიქონის შემდეგ შეიძ­ლება იყოს სათაურის ერთი ან რამდენიმე ველი. სათაური და ტანი ერთმანეთისგან ცარიელი სტრიქონით გამოიყოფა.

საწყის სტრიქონს აქვს შენდეგი ფორმატი:

**<მეთოდი> <URI> <პროტოკოლის\_ვერსია>**

კერძოდ:

GET <http://www.sou.edu.ge/> HTTP/1.1

მეთოდი მიუთითებს იმაზე, თუ რა ქმედებების შესრულებაა მოხდება მოთხოვნილ რესურსზე. http-მეთოდის მაგალითებია: **GET** (მივიღოთ რესურსის კონტენტი-შემადგენლობა), **POST** (მონაცემების გააგზავნა დამუშავებაზე), **PUT** (მონაცემთა განთავსება სერვერზე), **DELETE** (რესურსის წაშლა), **HEAD** (რესურსის შესახებ ინფორმაციის მიღება თავად ამ რესურსის ჩამოტვირთვის გარეშე) და სხვა.

ამ მეთოდებიდან მხოლოდ ორს - **GET** და **POST** -ებს შეუძლიათ ვებ-გვერდის ჩამოტვირთვა ბრაუზერის ფანჯარაში. **GET** ამას აკეთებს უშუალოდ, ხოლო **POST** ჯერგააგზვნის დასამუშავებელ მონაცემებს, *(ეს მონაცემები ბევრი უნდა იყოს, რადგანაც მცირე მონაცემებისთვის GET -ის გამოიყენებაც შეიძლება, ამ დროს მონაცემები კითხვის ნიშნის შემდეგ ჩაიწერება*) სერვერი მათ დაამუშავებს და მიღებულ შედეგს უბრუნებს ბრაუზერს - ანუ საიტის ჩამოტვირთვა ხდება არა უშუალოდ, არამედ წინმსწრები ოპერაციების შემდეგ.

ძირითადად განვიხილავთ კლიენტის GET და POST- მოთხოვნებს. კლიენტის მოთ­ხოვნებს სერვერი ამუშავებს ერთიანი რიგის პირობებში, ამიტომ შემოსული მოთხოვ­ნები გროვდება *ბუფერში* - მის სპეციალურ მეხსიერებაში. თუ ბევრი მოთხოვნა იქნე­ბა შეიძლება სერვერმა ვეღარ უზრუნველყოს მათი დამუშავება, ამის მიზეზი კი ბუფერის გადავსებაა და შეწყვიტოს ფუნქციონირება. ბუფერის გადავსება შეძლება მოხდეს მიზანმიმართულად - როცა გარედან ამ კონკრეტულ სერვერს ერთდროუ­ლად ბევრი კლიენტი მიმართავს - ამას *DOS შეტევა* ჰქვია და მისი მიზანია საიტის მწყობრიდან გამოყვანა.

მაგრამ დავუბრუნდეთ კლიენტის მოთხოვნის დამუშავებას სერვერის მიერ. თუ კი რამე გაუთვალისწინებელი არ ხდება სერვერი უბრუნებს კლიენტს პასუხს, რომლის ტანშიც არის მოთხოვნილი რესურსი.

მაგრამ ხდება ხოლმე შეცდომები და ამის შესახებადაც უნდა იქნეს საუბარი სერვერის მხრიდან წამოსულ პასუხში. ამიტომაც სერვერის HTTP -პასუხის შეტყობინება ხასიათდება სტატუსის კოდით - სამნიშნა რიცხვით, რომელიც იწყება 1დან 5 დიაპაზონში. კოდები რომელიც იწყება:

1-ით, არის ინფორმაციული და მიუთითებს იმას, რომ კლიენტის მოთხოვნა მიღებუ­ლია და მუშავდება:

2-ით - წარმატებული კლიენტის-მოთხოვნა;

3-ით - მოთხოვნა არ შესრულდა და კლიენტმა უნდა მიმართოს დამატებით ღონის­ძიებებს;

4-ით - კლიენტის მოთხოვნა არ შესრულდა რადგანაც შეცდომა კლიენტის მხარესაა, მაგალითად არასწორედაა მითითებული URL;

5-ით - შეცდომა სერვერის მხარესაა და ამის გამო მოთხოვნა ვერ შესრულდა.

მაგალითად: 200 – (OK) კლიენტის მოთხოვნა სწორია, 404 ((Not Found) - რესურსი არ მოიძებნა - ყველაზე გავრცელებული პასუხებია.

**სერვერზე მონაცემთა გაგზავნის მეთოდები. HTML-ფორმები.**

გავიხსენოთ მონაცემთა შეყვანის HTML-ფორმები. მაგალითად:

ლისტინგი 3

<form method="GET" action="/cgi-bin/form\_handler.cgi">

შეიყვანეთ თქვენი სახელი: <input type="text" name="name">

<br>

<input type="submit" name="okbutton" value="OK">

</form>

თუ კი ამ HTML-კოდს ბრაუზერით გავხსნით, მივიღებთ:

შეიყვანეთ თქვენი სახელი: 

ok

განვიხილოთ ამ ლისტინგის ტეგები უფრო დაწვრილებით.

ტეგი **<form>**, რომელსაც აქვს დამხურავი **</form>** ტეგი, განსაზღვრავენ ფორმას. მისი ატრიბუტები არაა სავალდებულო **HTML5**-ში.

* **action** - მიუთითებს **URL**-ს, რომელზეც გაგზავნილია ფორმა - თავისი მნიშ­ვნელობებით. იგი ფაქტიურად დამმუშავებლის რესურსის იდენტიფიკატო­რია. ეს ატრიბუტი სავალდებულოა **HTML4**-ში. ფორმის გაგზავნა - ეს იგივე მოთხოვნაა სერვერისადმი. თუ ეს ატრიბუტი არ არის მითითებული ფორმა ამავე დოკუმენტზეა მიმართული და იქ გადაიგზავნება -ანუ, თავის თავზე.
* **method** - ფორმის გაგზავნის საშუალებებია. ისინი ორია
* **GET** - ფორმის მონაცემთა გაგზავნა სამისამართო სტრიქონში. გაგახსე­ნებთ, რომ URL ში ? ნიშნის შემდეგ იწერება ”პარამეტრი”=”მნიშვნელო­ბა”, სწორედ ”პარამეტრი” შეესაბამება ატრიბუტ name-ის მნიშვნელობას, ხოლო ”მნიშვნელობას” - ატრიბუტ value-ის მნიშვნელობა. ეს ყველაფე­რი კოდში არის მეოთხე სტრიქონში <input> ტეგში. ასე მაგალითად ”სხვა საქართველო სადაა” ინგლისურად ნათარგმნი არის ჩაწერილი yandex.ru -ში სამისამართო სტრიქონში და მიცემულია ძებნაზე. აქ საძებნი ფრაზა გაიგზავნა GET -ით, ”?” -კითხვის ნიშნის შემდეგ! http://yandex.ru/yandsearch?text=Where%20other%20Georgia&lr=10277
* **POST** - ფორმის მნიშვნელობები გაიგზავნება მოთხოვნის ტანში.

თუ მეთოდი მითითებული არაა იგულისხმება ჩუმათობის პრინციპით - **GET**.

**<input>** ტეგით მოიცემა ფორმის ელემენტები, რომელიც განისაზღვრება **type** ატრიბუტით:

* მნიშვნელობა ”text” ქმნის ერთსტრიქონიანი ტექსტური შეყვანის ველს
* მნიშვნელობა “submit” ქმნის ღილაკს, რომლის საშუალებით ხდება ფორმის სერვერზე გაგზავნა

რა მოხდება, როცა დავაჭერთ თითს ღილაკ OK-ის?

1. ბრაუზერი განიხილავს ფორმაში შემავალ ელემენტებს და მისი ატრიბუტების name value –ის მეშვეობით ქმნის ფორმის მონაცემებს. ვთქვათ შეყვანილის სახელი Beka, ამ შემთხვევაში ფორმის მონაცემებია- name=Beka & okbutton=OK
2. ბრაუზერი უკავშირდება სერვერს, აგზავნის იქ მოთხოვნას დოკუმენტზე-რესურსზე, რომელიც მითითებულია ტეგ **<form>** -ის ატრიბუტ action-ში, ამასთან მოთხოვნის გაგზავნის მეთოდად გამოიყენებს იმას, რომელიც მითითებულია ატრიბუტ **method** - ში (ამ შემთხვევაში **GET** ), მოთხოვნაში მითითებულია ფორმის მონაცემები.
3. სერვერი უკეთებს ანალიზს მიღებულ მოთხოვნას, ახდენს პასუხის ფორმირებას, უგზავნის მას ბრაუზერს და წყვეტს კავშირს
4. ბრაუზერი გამოსახავს მიღებულ პასუხს

იმავე მოთხოვნის გაგზავნა ხელით (telnet-ის საშუალებით), ასე გამოიყურება (ვთქვათ საიტის დომენური სახელია - [www.example.com](http://www.example.com)):

ბრაუზერის სამისამართო სტრიქონში ან command prompt ავკრიფოთ

telnet [www.example.com](http://www.example.com) 80

შემდეგ

**GET /cgi-bin/form\_handler.cgi?name=Beka&okbutton=OK HTTP/1.0**\r\n  
**Host: www.example.com**\r\n  
\r\n

აქ \r\n აღნიშნავს enter -ის დაჭერას.

ლისტინდგ 3-ში GET მეთოდით და დავაჭერთ ღილაკს submit-ს, მაშინ ეს ანალოგიური იქნება ბრაუზერის სამისამართო სტრიქონში შემდეგი URL -ის ჩაწერისა

**http://www.example.com/cgi-bin/form\_handler.cgi?name=Beka&okbutton=OK**

ეხლა შევცვალოთ ლისტინგ 3-ში პირველი სტრიქონი შემდეგნაირად:

<form method="POST" action="/cgi-bin/form\_handler.cgi">

ამ შემთხვევაში მონაცემთა გაგზავნა განსხვავებული სახით ხდება:

telnet [www.example.com](http://www.example.com) 80

შემდეგ

**POST /cgi-bin/form\_handler.cgi HTTP/1.0**\r\n  
**Host:** [www.example.com\r\n](http://www.example.com\r\n)

Content-Length: 21\r\n  
\r\n

**name=Beka&okbutton=OK**

**POST** მეთოდის გამოყენებისას, ფორმის მონაცემები გადაიგზავნება მას შემდეგ რაც ორჯერ დავაჭერთ enter -ს - ანუ გადავდივართ მოთხოვნის **ტანში**! ხოლო რაც მის მაღლაა, არის მოთხოვნის **სათაური**!

**ტექნოლოგია CGI**

როგორ ამუშავებს სერვერი კლიენტის მოთხოვნას? პრინციპში სერვერი თავის თავად მხოლოდ გასცემს მოთხოვნილ გვერდს, ხოლო თუ მოთხოვნაში კიდეც სხვა საკითხე­ბია დასმული, ისეთი როგორც ფორმის მონაცემების გაგზავნა და მათი გათვალისწი­ნება განახლებულ გვერდზე, მასინ ამისთვის გამოიყენება **CGI** (Common Gateway Interface - ერთიანი შლუზის ( კარიბჭის) ინტერფეისი).

განვიხილოთ **URL** –ის ეს კონკრეტული იდენტიფიკატორი

[**http://www.example.com/cgi-bin/form\_handler.cgi**](http://www.example.com/cgi-bin/form_handler.cgi)

ცხადია, რომ სერვერი ჩვეულებრივ შემთხვევაში გასცემს ფაილს form\_handler.cgi, რომელიც მოთავსებულია კატალოგში-საქაღალდეში cgi-bin . მაგრამ როცა გვაქვს **CGI** ტექნოლოგია , მაშინ საქმე სხვაგვარადაა - სერვერი **გაუშვებს პროგრამას** - form\_handler.cgi და გადასცემს მას ფორმის მონაცემებს. ეს პროგრამა ახდენს იმ ტექსტის ფორმირებას, რომელიც შემდეგში ბრაუზერს გადაუგზავნის, როგორც პასუხს მოთხოვნაზე!

პროგრამა form\_handler.cgi შეიძლება დაიწეროს დაპროგრამების სხვადასვა ენაზე, მთავარია დაცული იყოს **CGI** ტექნოლოგიის სტანდარტი. ჩვენ მოვიყვანთ მაგალი­თებს დაწერილს C ენაზე, მაგრამ მანამ გავეცნოთ, თუ როგორ ხდება მონაცემების გაცვლა სერვერსა და **CGI** -პროგრამას შორის:

* **CGI** -პროგრამის წინ სერვერი განსაზღვრავს გარემოს ცვლადების მნიშვნელო­ბებს, რომელთა წაკითხვა შეუძლია თითქმის ყველა დაპროგრამების ენას. **CGI** -სტანდარტი განსაზღვრავს გარემოს ცვლადების საკმაოდ დიდ ჩამონათვალს, რომლებიც განსაზღვრული უნდა იყვნენ - ანუ, მათ უნდა ჰქონდეს მინიჭებუ­ლი მნიშვნელობები. აქგანვიხილოთ მხოლოდ სამი მათგანი:
* **REQUEST\_METHOD** - მონაცემთა გადაცემის მეთოდი - GET ან POST (სხვა მეთოდებიც არსებობენ, მაგრამ ამჟამად ისინი ჩვენთვის საინტე­რესო არ არიან)
* **QUERY\_STRING** - შეიცავს URL -ის ნაწილს კითხვის ნიშნის შემდეგ, ანუ საქმე გვაქვს GET-ფორმის მონაცემები
* **CONTENT\_LENGTH** - მოთხოვნის ტანის სიგრძე ( POST-ფორმის მონა­ცემები)
* სერვერი გაუშვებს **CGI** -პროგრამას. მოთხოვნის ტანი პროგრამას გადაეცემა სტანდარტული შეყვანის (stdin) სახით - თითქოსდა ეს მონაცემები კლავიატუ­რიდანაა შეყვანილი
* პროგრამა ბრაუზერს პასუხს გასცემს სტანდარტული გამოტანით (stdout) –”ეკრანზე” - ეს გამოტანა მიიტაცება ვებ-სერვერის მიერ და მერე გადაეცემა ბრაუზერს.

ლისტინგი 4

#include <stdio.h>

#include <stdlib.h>

int main(void)

{

// ვკითხულობთ გარემოს ცვლადებს, რომელიც განსაზღვრა ვებ-სერვერმა

char \*query\_string = getenv("QUERY\_STRING");

char \*request\_method = getenv("REQUEST\_METHOD");

char \*post\_data; // POST-მოთხოვნის მონაცემების ბუფერი

int post\_length = 0; // მოთხოვნის ტანის სიგრძე

if (strcmp(request\_method, "POST") == 0) { // თუ მიღებულია POST-მოთხოვნა,

post\_length = atoi(getenv("CONTENT\_LENGTH")); // თავიდან ვკითხულობთ

// გარემოს ცვლადებიდან მის სიგრძეს,

if (post\_length) { // თუ კი ის ნულის ტოლი არაა,

post\_data = (char\*)malloc(post\_length+1); // გამოვყოფთ მეხსიერებას ბუფერისთვის,

fread(post\_data, post\_length, 1, stdin); // ვკითხულობთ სტანდარტული შეყვანით მოთხოვნის ტანიდან,

post\_data[post\_length] = 0; // სტრიქონს ვამთავრებთ ნულოვანი ბაიტით.

}

}

// გამოგვყავს პასუხის სათაური...

printf("Content-type: text/html\r\n\r\n");

// და მისი ტანი:

printf("<h1>გამარჯობა!</h1>\r\n");

if (strlen(query\_string)) {

printf("<p>GET-ფორმის პარამეტრები: %s\r\n", query\_string);

}

if (post\_length) {

printf("<p>POST-ფორმის პარამეტრები: %s (მოთხოვნის ტანის სიგრძე: %d)\r\n", post\_data, post\_length);

free(post\_data); // არ დაგვავიწყდეს მე-17-ე სტრიქონში გამოყოფილი მეხსიერების განთავისუფლება

}

return 0;

}

ესაა უმარტივესი CGI-პროგრამა C-ზე, რომელსაც გამოჰყავს სერვერიდან მიღებული ფორმის პარამეტრები. ამის შედეგაც ბრაუზერი მიიღებს შემდეგ კოდს:

<h1>გამარჯობა!</h1>

<p>POST-ფორმის პარამეტრები: name=Beka&okbutton=OK (მოთხოვნის ტანის სიგრძე:21)

როგორც ჩანს, პარამეტრების გამოტანის უმარტივესი პროგრამაც გარკვეულ სირთუ­ლეებს შეიცავს - ასე მაგალითად, HTML-სტანდარტით თითქმის ყველა არა ანბანურ-ციფრული სიმბოლოები (მათ შორის ქართული) გადაიცემა UrlEncoded-სახით, და თუ კი დავუმატებს მოყვანილ C-პროგრამას UrlEncoded-ის გაშიფრვის კოდს, იგი უკვე საკმაოდ გაიზრდდება და მონიტორზეც ერთიანად ვერ გამოვა. ეს მხოლოდ საბაზო ინფორმაციაა და ადვილი წარმოსადგენია, რა მდგომარეობა შეიქმნება - როგორ გაიზ­რდება C-პროგრამა, თუ კი საქმე გვექნება მონაცემთა ბაზებთან.

მაგრამ CGI-პროგრამა C-ზე -იშვიათობაა, უფრო ხშირად ამას აკეთებენ Perl დაპროგ­რამების ენაზე, რომელიც სპეციალურად ტექსტური მონაცემების დამუშავებისთვისაა შექმნილი. ხოლო ის, რომ არსებობს CGI-მოდული, ამით CGI-სკრიპტების დაწერა მარტივდება.

იმ სერვერზე სადაც არსებობს Apache - ანუ გვაქვს Apache სერვერი, უფრო მოსახერ­ხებელია სკრიპტების ენა, რომელიც სპეციალურად ვებისთვისაა შექმნილი. ასეთი ენაა სწორედ PHP.

**PHP: ჰიპერტექსტის პრეპროცესორი**

1994 წელს პროგრამისტმა ***რასმუს ლენდორფმა*** (Rasmus Lendorf), რომელიც შეწუხე­ბული იყო CGI-ის Perl -ის მოდულების შექმნით, გადაწყვიტა დაეწერა რამდენიმე საკუთარი Perl-სკრიპტი, რათა ადვილი ყოფილიყო პერსონალური მთავარი გვერდის შექმნა და მათ უწოდა Personal Home Page (პერსონალური მთავარი გვერდი - PHP). შემდეგ მან ეს პროგრამები გადაწერა C-ზე, როცა დასჭირდა ფორმების დამუშავება - ასე წარმოიშვა Personal Home Page/Forms Interpreter (PHP/FI) 2.0. თავისი ნაღვაწი რასმუს ლენდორფმა განათავსა ღიად ინტერნეტში - Open Source -ის პრინციპის შესაბამისად.

1997 წელს PHP/FI-მა მიიქცია ***ენდი გუტმანსი*** (Andi Gutmans) და ***ზეევ სურასკის*** (Zeev Suraski) ყურადღება - მათ ეს იდეა მოეწონათ, მაგრამ მისი რეალიზაცია - არა, ამიტო­მაც PHP თავიდან გადაწერეს - ფაქტიურად თავიდან შექმნეს. შედეგად მიიღეს უნი­ვერსალური ენა და მას უწოდეს **P**HP: **H**ypertext **P**reprocessor. შემოკლებით მას ისევ PHP უწოდებენ და სხვადასხვა ვერსიებიც იქმნება. ჩვენ განვიხილავთ PHP5-ის ერთ-ერთ მოდიფიკაციას, რომელშიც არ გამოსული PHP6-ის ზოგიერთ ელემენსაც შეი­ცავს.

ეხლა ვაჩვენოთ თუ როგორ შეცვლის PHP-ის გამოყენების საშუალება ლისტინგ 3-ს და მასთან დაკავშირებულ საკითხებს -

ლისტინგი 5

<form method="POST" action="form\_handler.php">

შეიყვანეთ თქვენი სახელი: <input type="text" name="name">

<br>

<input type="submit" name="okbutton" value="OK">

</form>

ქვემოთ კიდეც ჩავწეროთ form\_handler.php -ის კოდი

ლისტინგი 6

 <html>

 <body>

**<?**

echo "<h1>გამარჯობა, <b>" . $\_POST['name'] . "</b></h1>!";

**?>**

 </body>

 </html>

PHP-სკრიპტი არის პრინციპში HTML-გვერდი, C და Perl -ისგან განსხვავებით. თავად სკრიპტი ჩაწერილია სპეციალურ ტეგებს შორის <? და ?>, რომელთა შიგნით ჩვენ გა­მოვიყენეთ ტექსტის გამოტანის ოპერატორი echo. ასეთი სკრიპტები-ბლოკები შეიძ­ლება რამდენიმე იყოს, მათ შორის კი ჩაწერილია აღიქმება ჩვეულებრივ html-ით.

ცვლადები GET-მოთხოვნისა მოთავსდება $\_GET მასივში, ცვლადები POST-მოთხოვ­ნისა მოთავსდება $\_POST მასივში, სერვერულო ცვლადევი IP-მისამართის, სკრიპტის სახელის და მათი მსგავსი მოთავსდება $\_SERVER მასივში, ოპერატოტი წერტილი (”.”) - სტრიქონების გაერთიანებაც მოთავსდება შესაბამის მასივში და ა.შ. ამასთან ყვე­ლა სამსახურეობრივი ოპერაციები(stdin წაკითხვა და გარემოს ცვლადები, Url-დეკოდირება) შეასრულა თავად PHP-მა. როგორც ვხედავთ ყველაფერი ეს ძალზე მოსახერხებელია.

მაგრამ რაღა საჭიროა ორი ბოლო ფაილი- HTML-ი ფორმით და PHP-სკრიპტი? მო­დით ისინი გავაერთიანოთ და გვექნება ერთი, შემდეგი სახის სკრიპტი:

ლისტინგი 7

 <html>

 <body>

**<?**

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

  echo "<h1>გამარჯობა, <b>" . $\_POST['name'] . "</b></h1>!";

  }

**?>**

 <form method="POST">

შეიყვანეთ თქვენი სახელი: <input type="text" name="name">

<br>

<input type="submit" name="okbutton" value="OK">

 </form>

 </body>

</html>

ტეგი <form>-დან ამოვიღეთ ატრიბუტი action - რაც ნიშნავს იმას, რომ ფორმა გაიგ­ზავნება ”თავის თავზე” ანუ მიმდინარე Url-ზე. ამას უწოდებენ ”postback form” - ფორმის დაბრუნებას. მეოთხე სტრიქონში მოწმდება if ოპერატორით - გამოყენებულ იქნს თუ არა დოკუმენტის ჩასატვირთად მეთოდი POST ( **C**-ზე დაწერილი პროგ­რამის მე-13-ე სტრიქონის ანალოგი) და თუ ეს ასეა - შემდეგი სტრიქონით გამოდის მისალმება. აი, ასეთი მარტივი მაგალითი მოვიყვანეთ PHP სკრიპტის CGI ტექნოლო­გიისთვის რაც შეიძლება ჩავთვალოთ “Hello Word”-ის ვებ-ვარიანტად!!!

და ბოლოს, PHP-სკრიპტის გამოცხადდება შეიძლება ერთ-ერთი ქვემოთ ჩამოთვლილი სამიდან ვარიანტიდან:

ვარიანტი 1.

**<?php**

**............**

**?>**

ვარიანტი 2.

**<?**

**........**

**?>**

ვარიანტი 3.

**<script language="php">**

**.....**

**</script>**

*ილუსტრაცია - მოცემულია PHP კოდი:*

ლისტინგი 8

<!DOCTYPE>

<html >

<head>

<title>Web გვერდი </title>

</head>

<body>

<p>

<?php

echo "ეს PHP -ს საბაზო დოკუმენტია";

?>

</p>

<p>

<?

print "PHP – ეს მშვენიერია!";

?>

</p>

<p>

<script language="php">

$myvar = "Hello World! ";

echo $myvar;

</script>

</p>

</body>

</html>

*სერვერი მას დაამუშავებს და მონიტორზე გამოდის:*

ეს PHP -ს საბაზო დოკუმენტია

PHP – ეს მშვენიერია!

Hello World!

*ხოლო ამ Web-დოკუმენტის წყარო - html-კოდი - ასეთია*

ლისტინგი 9

<!DOCTYPE html>

<html >

<head>

<title>Web გვერდი </title>

</head>

<body>

<p>

ეს PHP -ს საბაზო დოკუმენტია</p>

<p>

PHP – ეს მშვენიერია!</p>

<p>

Hello World! </p>

</body>

</html>