PHP ვიკიწიგნების ბიბლიოთეკიდან 2007

ღონღაძე გიორგი|Gnome

წიგნით, თავიდან ბოლომდე შეისწავლით PHP-ს საწყისებს. წიგნი განკუთვნილია დამწყებთათვის

2007

ნაწილი I

ბაზური PHP

PHP შესავალი

რა უნდა ვიცოდეთ?

სანამ ჩვენ დავიწყებთ php-ს შესწავლას, ჩვენ უნდა ვიცოდეთ შემდეგი :

- HTML / XHTML
- ცოტაოდენი სხვადასხვა სკრიპტები

HTML-ის შესასწავლად მოინახულეთ ბმული: http://ka.wikibooks.org/wiki/HTML

რა არის PHP?

- PHP იშიფრება, როგორც, **H**ypertext **P**reprocessor(ჰიპერტექსტული პრეპროცესორი)
- PHP არის სერვერული სკრიფტინგის ენა, როგორც ASP
- PHP სკრიფტები მუშაობენ სერვერზე
- PHP უზრუნველყოფს მრავალ მონაცემთა ბაზას (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC და ა.შ.)
- PHP არის ღია პროგრამული საშუალება (OSS)
- PHP - ს გადმოწერა და მოხმარება სრულიად უფასოა

რა არის PHP ფაილი?

- PHP ფაილები შეისაძლოა შეიცავდნენ ტექსტს, HTML ტეგებს და სკრიფტებს
- PHP ფაილები ბრუნდებიან ბრაუზერში, როგორც უბრალო HTML
- PHP ფაილებს აქვთ შემდეგი გაფართოებები : ".php", ".php3", ".php4", ".php5", ან ".phtml"

რა არის MYSQL?

- MySQL არის მონაცემთა ბაზის სერვერი
- MySQL იდეალურია დიდი და პატარა პროგრამებისათვის
- MySQL უზრუნველყოფს სტანდარტულ SQL-ს
- MySQL ეშვება პლადფორმების ნომრებზე
- MySQL-ის გადმოწერა და მოხმარება სრულიად უფასოა

PHP + MYSQL

- PHP და MySQL ერთად არის კროს-პლათფრომა (რაც იმას ნიშნავს რომ, ჩვენ შეგვიძლია დავაპროგრამოთ Windows-ზე და ვამუშავოთ Unix ფლათფრომაზე)

რატომ PHP?

- PHP ეშვება სხვადასხვა პლათფორმაზე(Windows, Linux, Unix და ა.შ.)
- PHP ეშვება თითქმის ყველა იმ სერვერზე, რაც დღეს-დღეობით გამოიყენება(Apache, IIS და ა.შ.)
- PHP სწავლა საკმაოდ ადვილია

როგორ დავიწყოთ?

- დავაყენოთ Apache server, Windows-ზე, ან Linux -ზე
- დავაყენოთ PHP , Windows-ზე, ან Linux -ზე
- დავაყენოთ MySQL , Windows-ზე, ან Linux -ზე

PHP INSTALL

რა გვჭირდება?

ეს ტუტორიალი არ გვასწავლის, თუ, როგორ უნდა დავაყენოთ PHP, MySQL, ან Apache Server.

თუ ჩვენი სერვერი უზრუნველყოფს PHP - ს, ჩვენ არ გვჭირდება არაფრის გაკეთება! ჩვენ არ გვჭირდება არაფრის დაყენება, უბრალოდ შევქმნათ რამოდენიმე .php ფაილი ჩვენ ვებ დირექტორიაში და სერვერი გაარჩევს მათ. უმეტესი ვებ ჰოსტინგი უზრუნველყოფს PHP-ს.

თუმცა, თუ ჩვენი სერვერი არ უზრუნველყოფს PHP-ს, მაშინ ჩვენ უნდა დავაყენოთ PHP. PHP-ს დაყენების ტუტორიალის სანახავად მიყევით ბმულს :

http://www.php.net/manual/en/install.php

გადმოვწეროთ PHP

გადმოვწეროთ PHP: http://www.php.net/downloads.php

გადმოვწეროთ MYSQL მონაცემთა ბაზა

გადმოვწეროთ MySQL მონაცემთა ბაზა: http://www.mysql.com/downloads/index.html

გადმოვწეროთ APACHE სერვერი

გადმოვწეროთ Apache სერვერი: http://httpd.apache.org/download.cgi

PHP სინტაქსი

ბაზური PHP სინტაქსი

PHP სკრიფტინგის ბლოკი ყველთვის იწყება შემდეგით : **<?php** და მთავრდება შემდეგით: **?>**. PHP ბლოკი შესაძლოა განთავსდეს დოკუმენტის ნებისმიერ ადგილზე.

სერვერებზე, რომლებიც უზრუნველყოფენ სტენოგრაფიას, ჩვენ შეგვიძლია დავიწყოთ სკრიფტინგის ბლოკი შემდეგით : <? და დავამთავროთ შემდეგით: ?>.

თუმცა, მაქსიმალური კომფორტულობისათვის რეკომენდირებულია დავიწყოთ შემდეგით: <?php.

```
<?php
?>
```

PHP ფაილი შეიცავს HTML ტეგებს, ისევე, როგორც HTML ფაილი და ზოგიერთი PHP სკრიფტინგის კოდი.

ქვემოთ მოცემულია PHP სკრიფტის უბრალო მაგალითი, რომელიც ბრაუზერში აბრუნებს "Hello World" :

```
<html>
<body>
<?php
echo "Hello World";
?>
</body>
</html>
```

თითოეული კოდის ხაზი PHP-ში უნდა დასრულდეს წერტილ-მძიმით. წერტილ-მძიმე არის სეპარატორი და გამოიყენება ერთი ინსტრუქციის, მეორესგან განსხვავებისათვის.

PHP-ში არსებობს ტექსტის დაბეჭდვის ორი ბაზური ცნობა: **echo** და **print**. წინა მაგალითში ჩვენ გამოვიყენეთ echo.

კომენტარები PHP-ში

PHP-ში, ჩვენ ვიყენებთ // -ს, რათა შევქმნათ ერთ ხაზიანი კომენტარი, ხოლო /* და*/ გამოიყენება დიდი კომენტარის ბლოკის გასაკეთებლად.

```
<html>
<body>
<?php
//This is a comment
/*
This is
a comment
block
*/
?>
</body>
</html>
```

PHP ცვლადები

ცვლადები PHP-ში

ცვლადები გამოიყენება მნიშვნელობების დაბრუნებისათვის, როგორც ტექსტური სტრინგი, რიცხვები, ან მასივები.

როდესაც სკრიპტები დანიშნულია, ის შეიძლება გამოყენებულ იქნას უსასრულოდ.

PHP-ში ყველა ცვლადი იწყება შემდეგით : $.

PHP-ში ცვლადების აღწერა:

```
$var_name = value;
```

შევქმნათ ცვლადი ტექსტთან ერთად და ცვლადი რიცხვთან ერთად:

```
<?php
$txt = "Hello World!";
$number = 16;
?>
```

PHP არის თავისუფლად საბეჭდი ენა

PHP-ში ცვლადს არ ჭირდება გამოცხადება

ზემოთ, მაგალითში, ჩვენ არ გვითქვია PHP-სთვის თუ რა ტიპისაა ცვლადი.

PHP ავტომატურად აკონვერტებს ცვლადებს მართებულ მონაცემთა ტიპებში.

ცვლადების სახელების წესები

- ცვლადის სახელი უნდა დაიწყოს ასოთი, ან ქვედა ტირეთი : "_"
- ცვლადის სახელი უნდა შეიცავდეს მხოლოდ ასოებს და რიცხვებს (a-Z, 0-9, და _)
- ცვლადის სახელი არ შეიცავს სივრცეებს. თუ ცვლადის სახელი შედგება ერთზე მეტი სიტყვისაგან, მაშინ ეს სიტყვები უნდა გამოიყოს ქვედა ტირეთი ($my_string), ან საწყისი ასოებით ($myString)

PHP სტრინგები

სტრინგები PHP-ში

სტრინგ ცვლადები გამოიყენებიან, იმ მნიშვნელობებისათვის რომლებიც შეიცავენ ასოებს.

სტრინგის შექმნის შემდეგ ჩვენ შევძლებთ მის მართვას. სტრინგი შესაძლოა გამოყენებულ იქნეს ფუნქციაში, ან დაბრუნდეს ცვლადად.

ქვემოთ, PHP სკრიფტი ქმნის სტრინგს "Hello World" $txt სტრინგ ცვლადში:

```
<?php
```

```
$txt="Hello World";
echo $txt;
?>
```

სკრიფტის შედეგი იქნება:

```
Hello World
```

ახლა ვცადოთ გამოვიყენოთ განსხვავებული ფუნქციები და ოპერატორები ჩვენი სტრინგის სამართავად.

გაერთიანების ოპერატორი

გაერთიანების ოპერატორი (.) გამოიყენება ორი სტინგ მნიშვნელობის შესაერთებლად.

მაგალითი:

```
<?php
$txt1="Hello World";
$txt2="1234";
echo $txt1 . " " . $txt2;
?>
```

სკრიფტის შედეგი იქნება:

```
Hello World 1234
```

თუ დავაკვირდებით სკრიფტს, ჩვენ შევამჩნევთ რომ გაერთიანების ოპერატორი გამოვიყენეთ ოჯერ. ეს იმიტომ რომ ჩვენ ჩავსვით მესამე სტრინგი. მესამე სტრინგი კი არის ბრჭყალებს შორის მოქცეული სივრცე.

STRLEN() ფუნქციის გამოყენება

strlen() ფუნქცია გამოიყენება სტრინგის სიგრძის შესამოწმებლად.

შევამოწმოთ "Hello world!" სტრინგის სიგრძე:

```
<?php
echo strlen("Hello world!");
?>
```

სკრიფტის შედეგი:

```
12
```

STRPOS() ფუნქციის გამოყენება

strpos() ფუნქცია გამოიყენება სტრინგში, სტრინგის ან ასოს საძებნელად.

თუ სტრინგში მოიძებნა დამთხვევა, ეს ფუნქცია დააბრუნებს პირველი დამთხვევის პოზიციას. თუ დამთხვევა არაა ნაპოვნი, მაშინ ის დააბრუნებს : FALSE.

მაგალითი:

```
<?php
echo strpos("Hello world!","world");
?>
```

სკრიფტის შედეგი:

```
6
```

როგორც ვხედავთ ჩვენს სტრინგში დამთხვევა "world" არის მეექვსე ადგილზე. მიზეზი იმისა რომ დამთხვევა არის მეექვსე ადგილზე და არა მეშვიდეზე, არის ის რომ ათვლა იწყება 0-დან და არა 1-დან.

PHP ოპერატორები

არითმეტიკული ოპერატორები

ოპერატორი	აღწერა	მაგალითი	შედეგი
+	შეკრება	X=2; x+2	4
-	გამოკლება	X=2; 5-x	3
*	გამრავლება	X=4; x*5	20
/	გაყოფა	15/5	3
%	მოდული	5%2	1
++	გაზრდა	X=5; x++	X=6
--	შემცირება	X=5; x--	X=4

დანიშვნის ოპერატორები

ოპერატორი	მაგალითი	...იგივეა, რაც
=	X=y	x=y
+=	X+=y	x=x+y
-=	x-=y	x=x-y
*=	X*=y	x=x*y
/=	X/=y	x=x/y
%=	X%=y	x=x%y

შედარების ოპერატორები

ოპერატორი	აღწერა	მაგალითი
==	უდრის	5==8, აბრუნებს False
!=	არ უდრის	5!=8, აბრუნებს True
>	მეტია	5>8, აბრუნებს False
<	ნაკლებია	5<8, აბრუნებს True
>=	მეტია, ან ტოლია	5>=8, აბრუნებს False
<=	ნაკლებია, ან ტოლია	5<=8, აბრუნებს True

ლოგიკური ოპერატორები

ოპერატორი	აღწერა	მაგალითი
&&	And	x=6 y=3 (x < 10 && y > 1) დააბრუნებს true
||	Or	x=6 y=3 (x==5 || y==5) დააბრუნებს false
!	Not	x=6 y=3 !(x==y) დააბრუნებს true

PHP IF...ELSE ოპერატორები

პირობითი კავშირის ოპერატორები

ძალიან ხშირად, როდესაც ჩვენ ვწერთ კოდს, ჩვენ უნდა შევასრულოთ განსხვავებული ქმედება განსხვავებული გადაწყვეტილებისათვის.

ამის გასაკეთებლად ჩვენ შეგვიძლია კოდში გამოვიყენოთ პირობითი კავშირის ოპერატორები.

- **if...else ოპერატორები** - გამოვიყენოთ ეს ოპერატორი მაშინ, როდესაც ერთი გადაწყვეტილება ჭეშმარიტია, ხოლო მეორე კი არა.
- **elseif ოპერატორები** - ეს ოპერატორი გამოვიყენოთ if...else-თან ერთად, თუ ერთ ერთი გადაწყვეტილება ჭეშმარიტია.

IF...ELSE ოპერატორი

სინტაქსი:

```
if (condition)
  code to be executed if condition is true;
else
  code to be executed if condition is false;
```

მაგალითი:

მიმდინარე მაგალითი დაბეჭდავს "Have a nice weekend!" თუ მიმდინარე დღეა პარასკევი, თუ არადა "Have a nice day!":

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
else
  echo "Have a nice day!";
```

თუ ოპერატორი შედგება ერთზე მეტი ხაზისაგან, მაშინ ის უნდა გამოიყოს ფიგურული ფრჩხილებით.

```
<html>
```

```
<body>
<?php
$d=date("D");
if ($d=="Fri")
  {
  echo "Hello!<br />";
  echo "Have a nice weekend!";
  echo "See you on Monday!";
  }
?>
</body>
</html>
```

ELSEIF ოპერატორი

სინტაქსი:

```
if (condition)
  code to be executed if condition is true;
elseif (condition)
  code to be executed if condition is true;
else
  code to be executed if condition is false;
```

მაგალითი:

მიმდინარე მაგალითი დაბეჭდავს "Have a nice weekend, თუ მიმდინარე დღეა Friday, და "Have a nice Sunday!", თუ მიმდინარე დღეა კვირა. სხვა შემთხვევაში დაბეჭდავს "Have a nice day!":

```
<html>
<body>
<?php
$d=date("D");
if ($d=="Fri")
  echo "Have a nice weekend!";
elseif ($d=="Sun")
  echo "Have a nice Sunday!";
else
  echo "Have a nice day!";
?>
</body>
</html>
```

PHP SWITCH ოპერატორი

თუ გვინდა მოვნიშნოთ კოდის ერთზე მეტი ბლოკი, გამოვიყენოთ Switch ოპერატორი.

switch ოპერატორი გამოიყენება გრძელი if..elseif..else ოპერატორების გრძელი კოდის თავიდან ასაცილებლად.

სინტაქსი:

```
switch (expression)
{
case label1:
  code to be executed if expression = label1;
  break;
case label2:
```

```
  code to be executed if expression = label2;
  break;
default:
  code to be executed
  if expression is different
  from both label1 and label2;
}
```

მაგალითი :

ეს მუშაობს შემდეგნაირად:

- ხშირად განმეორებადი ერთი და იგივე გამოთქმა გამოიყენება ერთხელ.
- გამოთქმის მნიშვნელობა შედარდება სტრუქტურის თითოეული ვარიანტის მნიშვნელობასთან.
- თუ მოხდება დამთხვევა, კოდი ამ ვარიანტებთან გააკეთებს ასოცირებას.
- როდესაც კოდი გაეშვება, **break** გამოიყენება შემდეგ ვარიანტში გაშვებული კოდის შესაჩერებლად.
- სტანდარტული(default) ცნობა გამოიყენება თუ არც ერთი ვარიანტი არ არის ჭეშმარიტი.

```
<html>
<body>
<?php
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
}
?>
</body>
</html>
```

PHP მასივები

რა არის მასივი?

როდესაც ვმუშაობთ PHP-ში, ადრე თუ გვიან, დაგვჭირდება შევქმნათ მრავალი მსგავსი ცვლადი.

იმის მაგივრად რომ შევქმნათ მრავალი ცვლადი, ჩვენ შეგვიძლია მოვაქციოთ ის მასივში.

მასივში თითოეულ ელემენტს აქვს საკუთარი ID, ამიტომ მასთან მიმართვა იქნება ძალიან ადვილი.

ქვემოთ მოყვანილია სამი განსხვავებული მასივი:

- **Numeric array(რიცხობრივი მასივი)** - მასივი რიცხობრივი ID გასაღებით
- **Associative array(ასოცირებული მასივი)** - მასივი, სადაც თითორული ID გასაღები ასოცირებულია მნიშვნელობასთან.
- **Multidimensional array(მულტისივრცული მასივი)** - მასივი შეიცავს ერთ ან მეტ მასივს.

რიცხობრივი მასივები

არსებობს სხვადასხვა გზა რიცხობრივი მასივის შექმნისა.

მაგალითი 1:

მაგალითი, სადაც ID გასაღები შექმნილია ავტომატურად:

```
$names = array("Peter","Quagmire","Joe");
```

მაგალითი 2:

მაგალითი, სადაც ID გასაღები ხელით უნდა მივუთითოთ:

```
$names[0] = "Peter";
$names[1] = "Quagmire";
$names[2] = "Joe";
```

ID გასაღებები შესაძლოა გამოვიყენოთ სკრიპტში:

```
<?php
$names[0] = "Peter";
$names[1] = "Quagmire";
$names[2] = "Joe";
echo $names[1] . " and " . $names[2] .
" are ". $names[0] . "'s neighbors";
?>
```

კოდი დაბეჭდავს:

```
Quagmire and Joe are Peter's neighbors
```

ასოცირებული მასივები

როდესაც ვაბრუნებთ სპეციფიური სახელების მონაცემებს, რიცხვითი მასივი ყოველთვის არ გამოგვადგება.

ასოცირებული მასივით ჩვენ შეგვიძლია მნიშვნელობები გამოვიყენოთ, როგორც გასაღებები და მივანიჭოთ მათ მნიშვნელობები.

მაგალითი 1:

ამ მაგალითში, მასივი სხვადასხვა პიროვნებებს ანიჭებს წლოვანებებს:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

მაგალითი 2:

ეს მაგალითი იგივეა რაც პირველი მაგალითი, მხოლოდ ჩვენებს მასივის შექმნის განსხვავებულ გზას:

```
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
```

ID გასაღებები შესაძლებელია გამოვიყენოთ სკრიპტში:

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

კოდი დაბეჭდავს:

```
Peter is 32 years old.
```

მულტისივრცული მასივები

მაგალითი:

ამ მაგალითში ჩვენ შევქმნით მულტისივრცულ მასივებს, ავტომატურად მინიჭებული ID გასაღებებით:

```
$families = array
(
  "Griffin"=>array
  (
  "Peter",
  "Lois",
  "Megan"
  ),
  "Quagmire"=>array
  (
  "Glenn"
  ),
  "Brown"=>array
  (
  "Cleveland",
  "Loretta",
  "Junior"
  )
);
```

ქვემოთ მოყვანილი მასივი გამოიყურება ასე, თუ დაწერილია ბეჭდვაზე:

```
Array
(
[Griffin] => Array
  (
  [0] => Peter
  [1] => Lois
  [2] => Megan
  )
[Quagmire] => Array
```

```
    (
    [0] => Glenn
    )
[Brown] => Array
    (
    [0] => Cleveland
    [1] => Loretta
    [2] => Junior
    )
)
```

მაგალითი 2:

შევეცადოთ გამოვსახოთ ერთი მნიშვნელობა:

```
echo "Is " . $families['Griffin'][2] .
" a part of the Griffin family?";
```

კოდი დაბეჭდავს:

```
Is Megan a part of the Griffin family?
```

PHP ციკლები

ციკლი

ძალიან ხშირად, როდესაც ვწერთ კოდს, კოდის ერთი და იგივე ბლოკის გაშვება გვჭირდება რომოდენიმეჯერ. ამისათვის ჩვენ შეგვიძლია გამოვიყენოთ ციკლის ოპერატორები.

PHP-ში არსებობს შემდეგი ციკლის ოპერატორები:

- **while** - მიმართავს კოდის ბლოკს სანამ სპეციფიური მითითება ჭეშმარიტია
- **do...while** - კოდის ბლოკს მიმართავს ერთხელ და იმეორებს ციკლს მანამ სანამ სპეციფიური მითითება ჭეშმარიტია
- **for** - კოდის ბლოკს მიმართავს n-ჯერ
- **foreach** - მიმართავს მასივში არსებული თითოეული ელემენტისათვის

WHILE ოპერატორი

სინტაქსი

```
while (condition)
code to be executed;
```

მაგალითი

მიმდინარე მაგალითი დემონსტრაციას უკეთებს ციკლს, რომელიც გაეშვება მანამ სანამ i ნაკლებია, ან ტოლი 5-ზე. i თითიეულ ციკლზე გაიზრდება 1-ით:

```
<html>
<body>
```

```
<?php
$i=1;
while($i<=5)
  {
  echo "The number is " . $i . "<br />";
  $i++;
  }
?>
</body>
</html>
```

DO...WHILE ოპერატორი

სინტაქსი

```
do
{
code to be executed;
}
while (condition);
```

მაგალითი

მიმდინარე მაგალითი i-ს მნიშვნელობას გაზრდის ერთხელ და ეს გაგრძელდება მანამ სანამ i-ს მნიშვნელობა არ იქნება 5:

```
<html>
<body>
<?php
$i=0;
do
  {
  $i++;
  echo "The number is " . $i . "<br />";
  }
while ($i<5);
?>
</body>
</html>
```

FOR ოპერატორი

სინტაქსი

```
for (initialization; condition; increment)
{
  code to be executed;
}
```

შენიშვნა: For ოპერატორს აქვს სამი პარამეტრი. პირველი პარამეტრი აღწერს ცვლადებს, მეორე პარამეტრი განსაზღვრავს პირობას და მესამე პარამეტრი შეიცავს ზრდას. თუ გვაქვს ერთზე მეტი ცვლადის, ან ზრდის პარამეტრი, ისინი უნდა გამოიყოს წერტილებით. პირობამ უნდა დააბრუნოს True, ან False.

მაგალითი

მიმდინარე მაგალითი დაბეჭდავს "Hello World!"-ს ხუთჯერ:

```
<html>
<body>
<?php
for ($i=1; $i<=5; $i++)
{
  echo "Hello World!<br />";
}
?>
</body>
</html>
```

FOREACH ოპერატორი

თითოეულ ციკლზე, მიმდინარე მასივის ელემენტის მნიშვნელობა ინიშნება $value - ასე რომ შემდეგ ციკლზე, ჩვენ დავინახავთ შემდეგ ელემენტს.

სინტაქსი

```
foreach (array as value)
{
    code to be executed;
}
```

მაგალითი

```
<html>
<body>
<?php
$arr=array("one", "two", "three");
foreach ($arr as $value)
{
  echo "Value: " . $value . "<br />";
}
?>
</body>
</html>
```

PHP ფუნქციები

PHP ფუნქციების შექმნა

ფუნქცია არის კოდის ბლოკი, რომლის გამოყენებასაც ჩვენ შევძლებთ სადაც გვინდა და როცა გვინდა.

PHP ფუნქციების შექმნა:

- ყველა ფუნქცია იწყება სიტყვით "function()"
- ფუნქციის სახელი - საშუალებას მოგვცემს მივხვდეთ რას ნიშნავს ფუნქცია. სახელი უნდა იწყებოდეს ასოთი.
- "{" - ფუნქციის კოდი იწყება ფიგურული ფრჩხილის დამატების შემდეგ.
- ჩავსვათ ფუნქციის კოდი

- "}" - ფუნქცია დასრულდება ფიგურული ფრჩხილის დახურვით

მაგალითი

```
<html>
<body>
<?php
function writeMyName()
  {
  echo "Kai Jim Refsnes";
  }
writeMyName();
?>
</body>
</html>
```

PHP ფუნქციების გამოყენება

```
<html>
<body>
<?php
function writeMyName()
  {
  echo "Kai Jim Refsnes";
  }
echo "Hello world!<br />";
echo "My name is ";
writeMyName();
echo ".<br />That's right, ";
writeMyName();
echo " is my name.";
?>
</body>
</html>
```

კოდი დაბეჭდავს:

```
Hello world!
My name is Kai Jim Refsnes.
That's right, Kai Jim Refsnes is my name.
```

PHP ფუნქციები - პარამეტრების დამატება

ჩვენი პირველი ფუნქცია (writeMyName()) არის ძალიან მარტივი ფუნქცია. ის წერს მხოლოდ სტატიკურ სტრინგს.

ფუნქციაზე უფრო მეტი ფუნქციონალურობის დამატებისათვის, ჩვენ შეგვიძლია დავამატოთ პარამეტრები. პარამეტრი არის, როგორც ცვლადი.

მაგალითი 1

მიმდინარე მაგალითი დაწერს განსხვავებულ სახელს, მაგრამ იგივე გვარს:

```
<html>
<body>
<?php
function writeMyName($fname)
  {
  echo $fname . " Refsnes.<br />";
  }
echo "My name is ";
writeMyName("Kai Jim");
echo "My name is ";
writeMyName("Hege");
echo "My name is ";
writeMyName("Stale");
?>
</body>
</html>
```

კოდი დაბეჭდავს:

```
My name is Kai Jim Refsnes.
My name is Hege Refsnes.
My name is Stale Refsnes.
```

მაგალითი 2

მიმდინარე ფუნქციას აქვს ორი პარამეტრი:

```
<html>
<body>
<?php
function writeMyName($fname,$punctuation)
  {
  echo $fname . " Refsnes" . $punctuation . "<br />";
  }
echo "My name is ";
writeMyName("Kai Jim",".");
echo "My name is ";
writeMyName("Hege","!");
echo "My name is ";
writeMyName("Ståle","...");
?>
</body>
</html>
```

კოდი დაბეჭდავს:

```
My name is Kai Jim Refsnes.
My name is Hege Refsnes!
My name is Ståle Refsnes...
```

PHP ფუნქციები - მნიშვნელობების დაბრუნება

მაგალითი

```
<html>
```

```
<body>
<?php
function add($x,$y)
  {
  $total = $x + $y;
  return $total;
  }
echo "1 + 16 = " . add(1,16)
?>
</body>
</html>
```

კოდი დაბეჭდავს:

```
1 + 16 = 17
```

PHP ფორმები და ველები

PHP ფორმა

ფორმის მაგალითი:

```
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
</body>
</html>
```

მაგალითი HTML გვერდზე შეიცავს ორ ველს და ერთ ღილაკს. როდესაც მომხმარებელი შეავსებს ველებს და იმოქმედებს ღილაკზე, ფორმის მონაცემები გაიგზავნება "welcome.php" ფაილში.

"welcome.php" ფაილი გამოიყურება ასე:

```
<html>
<body>
Welcome <?php echo $_POST["name"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old.
</body>
</html>
```

კოდი დაბეჭდავს:

```
Welcome John.
You are 28 years old.
```

PHP $_GET

THE $_GET ცვლადი

$_GET ცვლადი არის სახელების მასივი და HTTP GET მეთოდით გაგზავნილი მნიშვნელობები.

$_GET ცვლადი გამოიყენება ფორმა method="get"-დან მნიშვნელობების შესაკრებად. ინფორმაცია, გაგზავნილი ფორმიდან GET მეთოდით ჩანს ყველასათვის (ის გამოისახება ბრაუზერის მიმსამართების პანელზე) და მას აქვს ინფორმაციის გაგზავნის ლიმიტი (მაქს. 100 სიმბოლო).

მაგალითად

```
<form action="welcome.php" method="get">
Name: <input type="text" name="name" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

როდესაც მომხმარებელი იმოქმედებს ღილაკზე, გაგზავნილი URL გამოიყურება ამის მსგავსად:

```
http://geocg.myweb.ge/welcome.php?name=Peter&age=37
```

"welcome.php" ფაილს ახლა უკვე შეუძლია გამოიყენოს $_GET ცვლადი მონაცემების მისაღებად:

```
Welcome <?php echo $_GET["name"]; ?>.<br />
You are <?php echo $_GET["age"]; ?> years old!
```

$_REQUEST ცვლადი

PHP $_REQUEST ცვლადი შეიცავს ყველა ცვლადს : $_GET, $_POST, და $_COOKIE.

PHP $_REQUEST ცვლადი შესაძლებელია გამოვიყენით ფორმიდან მონაცემების შედეგების მისაღებად, რომლებიც გაიგზავნა ორივე GET და POST მეთოდებით.

მაგალითად:

```
Welcome <?php echo $_REQUEST["name"]; ?>.<br />
You are <?php echo $_REQUEST["age"]; ?> years old!
```

PHP $_POST

$_POST ცვლადი

$_POST ცვლადი არის სახელების მასივი და HTTP POST მეთოდით გაგზავნილი მნიშვნელობები.

$_POST ცვლადი გამოიყენება ფორმა method="post"-დან მნიშვნელობების შესაკრებად. ინფორმაცია, გაგზავნილი ფორმიდან POST მეთოდით არ ჩანს და მას არ აქვს ინფორმაციის გაგზავნის ლიმიტი.

მაგალითი

```
<form action="welcome.php" method="post">
Enter your name: <input type="text" name="name" />
Enter your age: <input type="text" name="age" />
<input type="submit" />
</form>
```

როდესაც მომხმარებელი იმოქმედებს ღილაკზე, გაგზავნილი URL გამოიყურება ამის მსგავსად:

```
http://geocg.myweb.ge/welcome.php
```

"welcome.php" ფაილს ახლა უკვე შეუძლია გამოიყენოს $_POST ცვლადი მონაცემების მისაღებად:

```
Welcome <?php echo $_POST["name"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old!
```

ნაწილი II

მოწინავე PHP

PHP თარიღი

PHP DATE() ფუნქცია

სინტაქსი:

```
date(format,timestamp)
```

პარამეტრი	აღწერა
ფორმატი	სავალდებულო. დროის ნიშნულზე ფორმატის სპეციფიკაცია
დროის ნიშნული	არასავალდებულო. დროის ნიშნულის სპეციფიკაცია. სტანდარტულად არის მიმდინარე დრო და თარიღი

PHP თარიღი - რა არის დროის ნიშნული?

დროის ნიშნული არის წამების რიცხვი, როგორც January 1, 1970 at 00:00:00 GMT. ეს ასევე ცნობილია როგორც Unix დროის ნიშნული.

PHP თარიღი - თარიღის ფორმატი

პირველი პარამეტრი date() ფუნქციაში არის დროის/თარიღის ფორმატის სპეციფიკაცია. დროის/თარიღის გამოსასახავად ის იყენებს ასოებს. ქვემოთ მოყვანილია ის ასოები, რომლებიც შესაძლოა გამოყენებულ იქნას:

- d - რიცხვი/დღე (01-31)
- m - მიმდინარე თვე (01-12)
- Y - მიმდინარე წელი

სხვა სიმბოლოები, როგორიცაა"/", ".", ან "-" შესაძლებელია ასევე ჩაისვას ასოებს შორის:

```
<?php
echo date("Y/m/d");
echo "<br />";
echo date("Y.m.d");
echo "<br />";
echo date("Y-m-d");
?>
```

კოდი დაბეჭდავს:

```
2007/07/11
2007.07.11
2007-07-11
```

PHP თარიღი - დროის ნიშნულის დამატება

წამების პარამეტრი date() ფუნქციაში არის დროის ნიშნულის სპეციფიკაცია. ეს პარამეტრი არასავალდებულოა. თუ ჩვენ არ ვიყენებთ დროის ნიშნულს, მაშინ გამოიყენება მიმდინარე დრო.

შემდეგ მაგალითში, მომდევნო დღის დროის ნიშნულის შესაქმნელად გამოვიყენებთ mktime() ფუნქციას.

mktime() ფუნქცია დააბრუნებს Unix დროის ნიშნულს სპეციფიური თარიღისათვის.

სინტაქსი:

```
mktime(hour,minute,second,month,day,year,is_dst)
```

მომდევნო დღეზე გადასასვლელად საჭიროა დაემატოს ერთი არგუმენტი mktime():

```
<?php
$tomorrow  =  mktime(0,0,0,date("m"),date("d")+1,date("Y"));
echo "Tomorrow is ".date("Y/m/d/", $tomorrow);
?>
```

კოდი დაბეჭდავს:

```
Tomorrow is 2006/07/12
```

PHP - ფაილის ჩასმა

სერვერული ჩამატებები

ჩვენ შეგვიძლია ფაილის შემცველობა ჩავსვათ PHP ფაილში სანამ სერვერი მიიღებს მას, include(), ან require() ფუნქციებით. ეს ორი ფუნქცია იდენტურია, მხოლოდ მათ გააჩნიათ განსხვავებული შეცდომების იდენტიფიკატორები. include() ფუნქცია გამოსახავს გაფრთხილებას (მაგრამს სკრიპტი გააგრძელებს მუშაობას) მანამ, სანამ require() ფუნქცია გამოსახავს გარდაუვალ შეცდომას (ამის მერე სკრიპტი შეწყვეტს მუშაობას).

ეს ორი ფუნქცია გამოიყენება ფუნქციების, სათაურების, ქვე კოლონტიტულების, ან ელემენტების შესაქმნელად, რომლებიც შესაძლოა გამოყენებულ იქნას შემცველობით გვერდებზე.

ეს აკეთებს დროის ეკონომიას. ეს ნიშნავს რომ ჩვენ შეგვიძლია შევქმნათ სტანდარტული სათაური, ან მენიუს ფაილი ყველა გვერდზე ერთდროულად. როდესაც საჭიროა სათაურის განახლება, ჩვენ შეგვიძლია განვაახლოთ მხოლოდ ჩამატებული ფაილი, ან როდესაც ვამატებთ ახალ გვერდს, ჩვენ შეგვიძლია ადვილად შევცვალოთ მენიუს ფაილი.

INCLUDE() ფუნქცია

include() ფუნქციას მთლიანი ტექსტი მიაქვს სპეციფიურ ფაილში და აკოპირებს იმ ფაილში რომელიც გამოიყენება ჩამატების ფუნქციით.

მაგალითი 1:

წარმოვიდგინოთ რომ გვაქვს სტანდარტული სათაურის ფაილი, რომელსაც ქვია "header.php". გვერდზე სათაურის ჩამატებისათვის, გამოვიყენოთ include() ფუნქცია შემდეგნაირად:

```
<html>
<body>
<?php include("header.php"); ?>
<h1>Welcome to my home page</h1>
<p>Some text</p>
</body>
</html>
```

მაგალითი 2:

ახლა ვთქვათ გვაქვს სტანდარტული მენიუს ფაილი რომელიც უნდა გამოვიყენოთ ყველა გვერდზე.

კოდი:

```
<html>
<body>
<a href="http://geocg.myweb.ge/index.php">Home</a> |
<a href="http://geocg.myweb.ge/about.php">About Us</a> |
<a href="http://geocg.myweb.ge/contact.php">Contact Us</a>
```

ეს სამი ფაილი, "index.php", "about.php", და "contact.php" ჩაემატება ფაილში "menu.php" .

კოდი:

```
<?php include("menu.php"); ?>
<h1>Welcome to my home page</h1>
<p>Some text</p>
</body>
</html>
```

თუკი ვიხილავთ კოდს ბრაუზერში მას ექნება შემდეგი სახე:

```
<html>
<body>
<a href="default.php">Home</a> |
<a href="about.php">About Us</a> |
<a href="contact.php">Contact Us</a>
<h1>Welcome to my home page</h1>
<p>Some text</p>
</body>
</html>
```

REQUIRE() ფუნქცია

თუ ჩვენ ფაილს ჩავსვავთ include() ფუნქციით და მოხდება შეცდომა, შესაძლოა მივიღოთ შემდეგი შეტყობინება.

PHP კოდი:

```
<html>
<body>
<?php
include("wrongFile.php");
echo "Hello World!";
?>
</body>
</html>
```

შეცდომის შეტყობინება:

```
Warning: include(wrongFile.php) [function.include]:
failed to open stream:
No such file or directory in C:\home\website\test.php on line 5
Warning: include() [function.include]:
Failed opening 'wrongFile.php' for inclusion
(include_path='.;C:\php5\pear')
in C:\home\website\test.php on line 5
Hello World!
```

გავითვალისწინოთ რომ echo ოპერატორი კვლავ მუშაობს! ეს იმიტომ რომ გაფრთხილება არ აჩერებს სკრიპტის მუშაობას.

ახლა გავუშვათ მსგავსი მაგალითი require() ფუნქციით.

PHP კოდი:

```
<html>
<body>
<?php
require("wrongFile.php");
echo "Hello World!";
?>
</body>
</html>
```

შეცდომის შეტყობინება:

```
Warning: require(wrongFile.php) [function.require]:
failed to open stream:
No such file or directory in C:\home\website\test.php on line 5
Fatal error: require() [function.require]:
Failed opening required 'wrongFile.php'
(include_path='.;C:\php5\pear')
in C:\home\website\test.php on line 5
```

echo პარამეტრმა აღარ იმუშავა რადგან გამოვიდა გარდაუვალი შეცდომის შეტყობინება.

რეკომენდირებულია გამოვიყენოთ require() ფუნქცია.

ფაილის გახსნა

fopen() ფუნქცია გამოიყენება PHP-ში ფაილების გასახსნელად.

ამ ფუნქციის პირველი პარამეტრი შეიცავს ფაილის სახელს, რომელიც უნდა გაიხსნას და მეორე პარამეტრი კი სპეციფიკაციას აკეთებს, თუ რა რეჟიმში გაიხსნას ფაილი:

```
<html>
<body>
<?php
$file=fopen("welcome.txt","r");
?>
</body>
</html>
```

ფაილი შესაძლებელია გაიხსნას ქვემოთ მოყვანილიდან ერთ-ერთ რეჟიმში:

რეჟიმი	აღწერა
r	კითხვა. იწყება ფაილის დასაწყისიდან
r+	კითხვა/ჩაწერა. იწყება ფაილის დასაწყისიდან
w	მხოლოდ ჩაწერა. ხსნის და ასუფთავებს ფაილის შემცველობას; ან ქმნის ახალ ფაილს თუ ის არ არსებობს
w+	კითხვა/ჩაწერა. ხსნის და ასუფთავებს ფაილის შემცველობას; ან ქმნის ახალ ფაილს თუ ის არ არსებობს
a	დამატება. ხსნის და წერს ფაილის ბოლოში, ან ქმნის ახალ ფაილს, თუ ის არ არსებობს
a+	კითხვა/დამატება. აკონსერვებს ფაილის შემცველობას ფაილის ბოლოში ჩაწერით
x	მხოლოდ ჩაწერა. ქმნის ახალ ფაილს. აბრუნებს FALSE და შეცდომას, თუ ფაილი უკვე არსებობს
x+	კითხვა/ჩაწერა. ქმნის ახალ ფაილს. აბრუნებს FALSE და შეცდომას, თუ ფაილი უკვე არსებობს

შენიშვნა: თუ fopen() ფუნქციას არ შეუძლია გახსნას სპეციფიური ფაილი, ის დააბრუნებს 0-ს.

მიმდინარე მაგალითი გამოსახავს შეტყობინებას, თუ fopen() ფუნქციას არ შეუძლია სპეციფიური ფაილის გახსნა:

```
<html>
<body>
<?php
$file=fopen("welcome.txt","r") or exit("Unable to open file!");
?>
</body>
</html>
```

ფაილის დახურვა

fclose() ფუნქცია გამოიყენება გახსნილი ფაილის დასახურად:

```
<?php
$file = fopen("test.txt","r");
//some code to be executed
fclose($file);
?>
```

END-OF-FILE-ის შემოწმება

feof() ფუნქცია ამოწმებს, მიღწეულია თუ არა("end-of-file" (EOF)) ფაილის ბოლოში.

feof() ფუნქცია გამოსადეგია უცნობი სიგრძის მონაცემთა ციკლისათვის.

შენიშვნა: ჩვენ ვერ წავიკითხავს ფაილიდან, რომელიც გახსნილია w, a, და x რეჟიმებში!

```
if (feof($file)) echo "End of file";
```

ფაილში თითო-თითო ხაზის წაკითხვა

fgets() ფუნქცია გამოიყენება ფაილიდან ერთი ხაზის წაკითხვისათვის.

შენიშვნა: ამ ფუნქციის გამოძახების შემდგომ ფაილის კურსორი გადავა მეორე ხაზზე.

მაგალითი

ქვემოთ მოყვანილი მაგალითი კითხულობს ფაილის თითო-თითო ხაზს, მანამ, სანამ არ გავა ფაილის ბოლოში:

```
<?php
$file = fopen("welcome.txt", "r") or exit("Unable to open file!");
//Output a line of the file until the end is reached
while(!feof($file))
  {
  echo fgets($file). "<br />";
  }
fclose($file);
?>
```

ფაილში თითო-თითო სიმბოლოს წაკითხვა

Fgetc() ფუნქცია გამოიყენება ფაილში თითო-თითო სიმბოლოს წასაკითხად.

შენიშვნა: ამ ფუნქციის გამოძახების შემდგომ ფაილის კურსორი გადავა შემდეგ სიმბოლოზე.

მაგალითი

ქვემოთ მოყვანილი მაგალითი ფაილში კითხულობს თითო-თითო სიმბოლოს, მანამ, სანამ ის არ მიაღწევს ფაილის დასასრულს:

```php
<?php
$file=fopen("welcome.txt","r") or exit("Unable to open file!");
while (!feof($file))
  {
  echo fgetc($file);
  }
fclose($file);
?>
```

PHP - ფაილის ატვირთვა

ფაილის ატვირთვის ფორმის შექმნა

ქვემოთ მოყვანილია ფაილების ატვირთვის HTML ფორმა:

```html
<html>
<body>
<form action="upload_file.php" method="post"
enctype="multipart/form-data">
<label for="file">Filename:</label>
<input type="file" name="file" id="file" />
<br />
<input type="submit" name="submit" value="Submit" />
</form>
</body>
</html>
```

გავითვალისწინოთ შემდეგი HTML ფორმისათვის:

- enctype ატრიბუტი <form> ტეგისათვის სპეციფიკაციას აკეთებს, თუ რომელი შემცველობითი ტიპი გამოიყენოს ფორმის გამოყენებისას.
- "multipart/form-data" გამოიყენება, როდესაც ფორმა მოითხოვს ბინარულ მონაცემებს, როგორიცაა ასატვირთი ფაილის შემცველობა.

შენიშვნა: მომხმარებელთათვის ფაილების ატვირთვის უფლების მიცემა წარმოადგენს დიდ რისკს.

ატვირთვის სკრიპტის შექმნა

"upload_file.php" შეიცავს ატვირთვის კოდს:

```php
<?php
if ($_FILES["file"]["error"] > 0)
  {
  echo "Error: " . $_FILES["file"]["error"] . "<br />";
  }
else
  {
  echo "Upload: " . $_FILES["file"]["name"] . "<br />";
  echo "Type: " . $_FILES["file"]["type"] . "<br />";
  echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
  echo "Stored in: " . $_FILES["file"]["tmp_name"];
  }
?>
```

გლობალური PHP $_FILES მასივების გამოყენებით ჩვენ შეგვიძლია კომპიუტერიდან სერვერზე ავტვირთოთ ფაილები.

პირველი პარამეტრი არის input სახელი, ხოლო მეორე პარამეტრი შესაძლოა იყოს ნებისმიერი "name", "type", "size", "tmp_name", ან "error". ამის მსგავსად:

- $_FILES["file"]["name"] - ატვირთული ფაილის სახელი
- $_FILES["file"]["type"] - ატვირთული ფაილის ტიპი
- $_FILES["file"]["size"] - ატვირთული ფაილის ზომა ბაიტებში
- $_FILES["file"]["tmp_name"] - სერვერზე დაბრუნებული ფაილის დროებითი ასლი
- $_FILES["file"]["error"] - შეცდომის კოდი

ეს არის ფაილების ატვირთვის ძალიან მარტივი გზა. დაცვისათვის, დავაწესოთ შეზღუდვები ასატვირთი ფაილების გაფართოებებზე.

შეზღუდვა ატვირთვაზე

სკრიპტში ჩვენ ჩავამატებთ ფაილის ატვირთვის ზოგიერთ შეზღუდვებს. მომხმარებელს შეეძლება ატვირთოს მხოლოდ .gif, ან .jpeg ფაილები და ფაილის ზომა არ უნდა აღემატებოდეს 20 kb-ს:

```
<?php
if ((($_FILES["file"]["type"] == "image/gif")
|| ($_FILES["file"]["type"] == "image/jpeg")
|| ($_FILES["file"]["type"] == "image/pjpeg"))
&& ($_FILES["file"]["size"] < 20000))
  {
  if ($_FILES["file"]["error"] > 0)
    {
    echo "Error: " . $_FILES["file"]["error"] . "<br />";
    }
  else
    {
    echo "Upload: " . $_FILES["file"]["name"] . "<br />";
    echo "Type: " . $_FILES["file"]["type"] . "<br />";
    echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
    echo "Stored in: " . $_FILES["file"]["tmp_name"];
    }
  }
else
  {
  echo "Invalid file";
  }
?>
```

შენიშვნა: IE-სათვის jpg ფაილების ცნობისათვის ტიპი უნდა იყოს pjpeg, ხოლო FireFox-სათვის ტიპი უნდა იყოს jpeg.

ატვირთული ფაილის დამახსოვრება

ქვემოთ მოყვანილია მაგალითები, ატვირთული ფაილების დროებითი ასლის შექმნისა.

ფაილის დროები ასლები ქრებიან, როდესაც სკრიპტი ასრულებს მუშაობას. ატვირთული ფაილის შესანახად ჩვენ გვჭირდება მისი სხვა ადგილზე კოპირება:

```php
<?php
if (($_FILES["file"]["type"] == "image/gif")
|| ($_FILES["file"]["type"] == "image/jpeg")
|| ($_FILES["file"]["type"] == "image/pjpeg")
&& ($_FILES["file"]["size"] < 20000))
  {
  if ($_FILES["file"]["error"] > 0)
    {
    echo "Return Code: " . $_FILES["file"]["error"] . "<br />";
    }
  else
    {
    echo "Upload: " . $_FILES["file"]["name"] . "<br />";
    echo "Type: " . $_FILES["file"]["type"] . "<br />";
    echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
    echo "Temp file: " . $_FILES["file"]["tmp_name"] . "<br />";
    if (file_exists("upload/" . $_FILES["file"]["name"]))
      {
      echo $_FILES["file"]["name"] . " already exists. ";
      }
    else
      {
      move_uploaded_file($_FILES["file"]["tmp_name"],
      "upload/" . $_FILES["file"]["name"]);
      echo "Stored in: " . "upload/" . $_FILES["file"]["name"];
      }
    }
  }
else
  {
  echo "Invalid file";
  }
?>
```

PHP COOKIES(ბმულები)

რა არის ბმული(COOKIE)?

ბმული არის ხშირად გამოყენებადი მომხმარებლის იდენტიფიკაციისათვის. ბმული არის პატარა ფაილი, რომელიც სერვერმა ჩადგა მომხმარებლის კომპიუტერში. რამდენჯერაც კომპიუტერი მოითხოვს მისამართს ბრაუზერში, იმდენჯერ გაეგზავნება მონაცემები ბმულს. PHP-თ, ჩვენ შეგვიძლია შევქმნათ და მივიღოთ ბმულები.

როგორ შევქმნათ ბმული?

setcookie() ფუნქცია გამოიყენება ბმულების დასასმელად.

შენიშვნა: setcookie() ფუნქცია უნდა დაიწეროს <html> ტეგამდე.

სინტაქსი:

```
setcookie(name, value, expire, path, domain);
```

მაგალითი:

ქვემოთ მოყვანილ მაგალითში ჩვენ შევქმნი ბმულს სახელით "user" და მივანიჭებთ მნიშვნელობას "Alex Porter":

```
<?php
setcookie("user", "Alex Porter", time()+3600);
?>
<html>
<body>
</body>
</html>
```

როგორ შევასწოროთ ბმული?

PHP $_COOKIE ცვლადი გამოიყენება ბმულის მნიშვნელობების შესასწორებლად.

ქვემოთ მოყვანილ მაგალითში, ჩვენ შევასწორებთ ბმულს სახელად "user" და გამოვსახავთ მას გვერდზე:

```
<?php
// Print a cookie
echo $_COOKIE["user"];
// A way to view all cookies
print_r($_COOKIE);
?>
```

მიმდინარე მაგალითში ჩვენ გამოვიყენებთ isset() ფუნქციას, რათა ვიპოვოთ დასმული ბმული:

```
<html>
<body>
<?php
if (isset($_COOKIE["user"]))
  echo "Welcome " . $_COOKIE["user"] . "!<br />";
else
  echo "Welcome guest!<br />";
?>
</body>
</html>
```

როგორ წავშალოთ ბმული?

წაშლის მაგალითი:

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time()-3600);
?>
```

თუ ბრაუზერი არ უზრუნველყოფს ბმულებს

თუ ბრაუზერი არ უზრუნველყოფს ბმულებს, ჩვენ უნდა გამოვიყენოთ სხვა მეთოდი, რათა ინფორმაცია გადავცეთ ერთი გვერდიდან მეორე გვერდზე.

ქვემოთ მოცემული ფორმა გადასცემს მომხმარებლის ინფორმაციას "welcome.php"-ს როდესაც მომხმარებელი იმოქმედებს "Submit" ღილაკზე:

```
<html>
<body>
<form action="welcome.php" method="post">
Name: <input type="text" name="name" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
</body>
</html>
```

შევასწოროთ მნიშვნელობა "welcome.php"-ში შემდეგის მსგავსად:

```
<html>
<body>
Welcome <?php echo $_POST["name"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old.
</body>
</html>
```

PHP სესიები

PHP სესიის ცვლადები

როდესაც ვმუშაობთ პროგრამასთან, ჩვენ ვხსნით მას, ვაკეთებთ ცვლილებებს და შემდგომ ვხურავთ მას. ეს გავს სესიას. კომპიუტერმა იცის ვინ ხარ შენ. მან იცის ჩვენ როდის გავუშვით პროგრამა და როდის დავხურეთ. მაგრამ ინტერნეტში არის ერთი პრობლემა: ვებ სერვერმა არ იცის ვინ ხარ შენ და რას აკეთებ.

PHP სესია ჭრის ამ პრობლემას. თუმცა, სესიის ინფორმაცია დროებითია და წაიშლება მას შემდეგ რაც მომხმარებელი დატოვებს ვებ გვერდს.

სესია ქმნის უნიკალურ სახელს (UID) თითოეული ვიზიტორისათვის და აგროვებს ცვლადების ბაზას UID-ზე.

PHP სესიის დაწყება

სანამ ჩვენ შევაგროვებთ ინფორმაციას PHP სესიაზე, ჩვენ უნდა დავიწყოთ სესია.

შენიშვნა: session_start() ფუნქცია იწერება <html> ტეგამდე:

```
<?php session_start(); ?>
<html>
<body>
</body>
</html>
```

ზემოთ მოყვანილი კოდი დაარეგისტრირებს მომხმარებლის სესიას, საშუალებას მოგვცემს შევინახოთ მომხმარებლის ინფორმაცია და მივანიჭოთ UID ამ მომხმარებლის სესიას.

სესიის ცვლადების შეგროვება

სწორი გზა სესიის ცვლადების შეგროვებისა და ჩასწორების არის PHP $_SESSION ცვლადის გამოყენება:

```
<?php
session_start();
// store session data
$_SESSION['views']=1;
?>
<html>
<body>
<?php
//retrieve session data
echo "Pageviews=". $_SESSION['views'];
?>
</body>
</html>
```

დაბეჭდავს:

```
Pageviews=1
```

ქვემოთ მოყვანილ მაგალითში, ჩვენ შევქმნით უბრალო მთვლელს. isset() ფუნქცია შეამოწმებს "views" ცვლადი უკვე დასმულია თუ არა. თუ "views" უკვე დასმულია, ჩვენ შეგვიძლია გავზარდოთ ჩვენი მთვლელი. თუ "views" არ არსებობს, ჩვენ ვქმნით"views" ცვლადს და ვაყენებთ მას 1-ზე:

```
<?php

session_start();
if(isset($_SESSION['views']))
  $_SESSION['views']=$_SESSION['views']+1;

else
  $_SESSION['views']=1;
echo "Views=". $_SESSION['views'];
?>
```

სესიის განადგურება

თუ გვსურს სესიის მონაცემთა წაშლა,ჩვენ შეგვიძლია გამოვიყენოთ unset(), ან session_destroy() ფუნქციები.

unset() ფუნქცია გამოიყენება სესიის ცვლადის გასანთავისუფლებლად:

```
<?php
unset($_SESSION['views']);
?>
```

ჩვენ ასევე შეგვიძლია საბოლოო გავანადგუროთ სესიის ცვლადი session_destroy() ფუნქციის გამოყენებით:

```
<?php
session_destroy();
?>
```

შენიშვნა: session_destroy() ფუნქცია წაშლის ყველა მონაცემს ცვლადში.

PHP ელ. ფოსტის გაგზავნა

PHP MAIL() ფუნქცია

PHP mail() ფუნქცია გამოიყენება სკრიპტიდან ელ. ფოსტის გასაგზავნად.

სინტაქსი

```
mail(to,subject,message,headers,parameters)
```

პარამეტრი	აღწერა
to	აუცილებელი. ელ.ფოსტის მიმღების/მიმღებების სპეციფიკაცია
subject	აუცილებელი. ფოსტის თემის სპეციფიკაცია. **შენიშვნა:** ეს პარამატრი არ უნდა შეიცავდეს რაიმე სიმბოლოებს.
message	აუცილებელი. განსაზღვრავს გასაგზავნ შეტყობინებას. თითოეული ხაზი გამოიყოფა LF (\n)-ით.
headers	არა აუცილებელი. სათაურების სპეციფიკაცია, როგორც From, Cc და Bcc. სათაურები გამოიყოფა CRLF (\r\n)-ით
parameters	არა აუცილებელი. დამატებითი პარამეტრების სპეციფიკაცია

PHP უბრალო ელ. ფოსტა

უბრალო გზა PHP-ს დახმარებით ელ. ფოსტის გაგზავნისა არის ტექსტური მეილის გაგზავნა.

ქვემოთ მოყვანილ მაგალითში ჩვენ გამოვაცხადებთ ცვლადებს ($to, $subject, $message, $from, $headers), შემდეგ ამ ცვლადებს გამოვიყენებთ mail() ფუნქციაში:

```
<?php
$to = "someone@example.com";
$subject = "Test mail";
$message = "Hello! This is a simple email message.";
$from = "someonelse@example.com";
$headers = "From: $from";
mail($to,$subject,$message,$headers);
echo "Mail Sent.";
?>
```

PHP ფოსტის ფორმა

PHP-თ ვებ გვერდზე ჩვენ შეგვიძლია შევქმნათ უკუკავშირი-ფორმა. ქვემოთ მოყვანილი მაგალითი აგზავნის ფოსტას სპეციფიურ მისამართზე:

```
<html>
<body>
```

```
<?php
if (isset($_REQUEST['email']))
//if "email" is filled out, send email
  {
  //send email
  $email = $_REQUEST['email'] ;
  $subject = $_REQUEST['subject'] ;
  $message = $_REQUEST['message'] ;
  mail( "someone@example.com", "Subject: $subject",
  $message, "From: $email" );
  echo "Thank you for using our mail form";
  }
else
//if "email" is not filled out, display the form
  {
  echo "<form method='post' action='mailform.php'>
  Email: <input name='email' type='text' /><br />
  Subject: <input name='subject' type='text' /><br />
  Message:<br />
  <textarea name='message' rows='15' cols='40'>
  </textarea><br />
  <input type='submit' />
  </form>";
  }
?>
</body>
</html>
```

როგორ მუშაობს ზემოთ მოყვანილი მაგალითი:

- პირველი, დავრწმუნდეთ რომ ელ. ფოსტის ველი გვაქვს
- თუ ის არ არის დასმული; შევქმნათ HTML ფორმა
- თუ ის დასმულია; გავგზავნოთ ფოსტა ფორმიდან
- როდესაც გავგზავნით, ფორმა გაიწმინდება, გვერდი გადაიტვირთება, შემოწმდება ყველა მოთხოვნილი ველი და გაიგზავნება

PHP ელ.ფოსტის დაცვა

PHP ელ. ფოსტის ინექციები

პირველი, შევხედოთ PHP კოდს წინა პარაგრაფიდან:

```
<html>
<body>
<?php
if (isset($_REQUEST['email']))
//if "email" is filled out, send email
  {
  //send email
  $email = $_REQUEST['email'] ;
  $subject = $_REQUEST['subject'] ;
  $message = $_REQUEST['message'] ;
  mail("someone@example.com", "Subject: $subject",
  $message, "From: $email" );
  echo "Thank you for using our mail form";
  }
else
//if "email" is not filled out, display the form
```

```
  {
  echo "<form method='post' action='mailform.php'>
  Email: <input name='email' type='text' /><br />
  Subject: <input name='subject' type='text' /><br />
  Message:<br />
  <textarea name='message' rows='15' cols='40'>
  </textarea><br />
  <input type='submit' />
  </form>";
  }
?>
</body>
</html>
```

ზემოთ მოყვანილი კოდის პრობლემა არის ის რომ, არა ავტორიზებულ მომხმარებლებს შეუძლიათ ფორმის მეშვეობით ჩასვან მონაცემები ფოსტის სათაურებში.

რა მოხდება თუ მომხმარებელი შეიყვანს მიმდიანრე ტექსტ ფოსტის ველში?

```
someone@example.com%0ACc:person2@example.com
%0ABcc:person3@example.com,person3@example.com,
anotherperson4@example.com,person5@example.com
%0ABTo:person6@example.com
```

mail() ფუნქცია ზემოთ მოყვანილ ტექტს წერს ფოსტის სათაურების ველში და ახლა სათაურს აქვს ექსტრა Cc:, Bcc: და To: ველი. როდესაც მომხმარებელი იმოქმედებს "Submit" ღილაკზე, ფოსტა გაიგზავნება ზემოთ მოყვანილ ყველა მისამართზე.

PHP ელ. ფოსტის ინექციების შეჩერება

საუკეთესო გზა ფოსტის ინექციების შეჩერებისა არის გაგზავნის დასტური.

ქვემოთ მოყვანილი კოდი იგივეა რაც წინა პარაგრაფში, მაგრამ ახლა ჩვენ ჩავამატეთ გაგზავნის დამადასტურებელი, რომელიც ამოწმებს ფოსტის ველს:

```
<html>
<body>
<?php
function spamcheck($field)
  {
//eregi() performs a case insensitive regular expression match
  if(eregi("to:",$field) || eregi("cc:",$field))
    {
    return TRUE;
    }
  else
    {
    return FALSE;
    }
  }
//if "email" is filled out, send email
if (isset($_REQUEST['email']))
  {
  //check if the email address is invalid
  $mailcheck = spamcheck($_REQUEST['email']);
  if ($mailcheck==TRUE)
    {
    echo "Invalid input";
```

```
    }
  else
    {
    //send email
    $email = $_REQUEST['email'] ;
    $subject = $_REQUEST['subject'] ;
    $message = $_REQUEST['message'] ;
    mail("someone@example.com", "Subject: $subject",
    $message, "From: $email" );
    echo "Thank you for using our mail form";
    }
  }
else
//if "email" is not filled out, display the form
  {
  echo "<form method='post' action='mailform.php'>
  Email: <input name='email' type='text' /><br />
  Subject: <input name='subject' type='text' /><br />
  Message:<br />
  <textarea name='message' rows='15' cols='40'>
  </textarea><br />
  <input type='submit' />
  </form>";
  }
?>
</body>
</html>
```

PHP - შეცდომა

PHP შეცდომის შეტყობინება

როდესაც ვქმნით სკრიპტებს და ვებ პროგრამებს, შეცდომის გამოტანა არის საკმაოდ მნიშვნელოვანი. თუკი ჩვენ კოდს არ აქვს კოდში შეცდომების შემოწმების საშუალება, ჩვენი პროგრამა იქნება არაპროფესიონალური და შესაძლოა დაუცველი.

არსებობს შეცდომის გამოტანის რამოდენიმე საშუალება:

- უბრალო "die()" ოპერატორები
- ინდივიდუალური შეცდომები და შეცდომის კვანძები
- შეცდომის ანგარიშები

ბაზური შეცდომის შეტყობინება: DIE() ფუნქციის გამოყენება

პირველი მაგალითი გვაჩვენებს უბრალო სკრიპტს, რომელიც ხსნის ტექსტურ ფაილს:

```
<?php
$file=fopen("welcome.txt","r");
?>
```

თუ ფაილი არ არსებობს ჩვენ მივიღებთ მსგავს შეცდომას:

```
Warning: fopen(welcome.txt) [function.fopen]: failed to open stream:
No such file or directory in C:\webfolder\test.php on line 2
```

იმისათვის რომ მომხმარებელმა არ მიიღოს მსგავსი შეტყობინება, მანამ, სანამ მოვითხოვთ ფაილს, უნდა დავტესტოთ არსებობს იგი თუ არა:

```
<?php
if(!file_exists("welcome.txt"))
 {
 die("File not found");
 }
else
 {
 $file=fopen("welcome.txt","r");
 }
?>
```

ახლა უკვე თუ ფაილი არ არსებობს, გამოვა მსგავსი შეტყობინება:

```
File not found
```

ეს კოდი უფრო ეფექტურია ვიდრე პირველი, რადგან ეს იყენებს სკრიპტის გაჩერებისათვის შეცდომის გამოტანის უბრალო მექანიზმს.

თუმცა, სკრიპტის შეჩერება ყოველთვის არაა სასურველი. განვიხილოთ შეცდომის გამოტანის ალტერნატიული PHP ფუნქციები.

ინდივიდუალური შეცდომის შეტყობინების შექმნა

ინდივიდუალური შეცდომის შეტყობინების შექმნა საკმაოდ ადვილია. ჩვენ უბრალოდ შევქმნით სპეციალურ ფუნქციას , რომელიც გამოიძახება PHP-ში შეცდომის შემთხვევაში.

სინტაქსი:

```
error_function(error_level,error_message,
error_file,error_line,error_context)
```

პარამეტრი	აღწერა
error_level	აუცილებელი. შეცდომის დონის სპეციფიკაცია. მნიშვნელობა უნდა იყოს რიცხვი
error_message	აუცილებელი. შეცდომის შეტყობინების სპეციფიკაცია
error_file	არა აუცილებელი. ფაილის სახელის სპეციფიკაცია, რომელშიც აღმოჩნდა შეცდომა
error_line	არა აუცილებელი. ხაზის ნომრის სპეციფიკაცია, რომელშიც აღმოჩნდა შეცდომა
error_context	არა აუცილებელია. ყველა ცვლადის და მათი მნიშვნელობების შემცველი მასივის სპეციფიკაცია, რომელთა გამოყენების დროსაც აღმოჩნდა შეცდომა

შეცდომის დონე

შეცდომის დონეები:

მნიშვნელობა	მუდმივი	აღწერა
2	E_WARNING	არა გარდაუვალი ”რან-თაიმ” შეცდომა. სკრიპტის მუშაობა არ შეჩერდება
8	E_NOTICE	”რან-თაიმ” განცხადება. სკრიპტმა იპოვა რაღაც, რომელის შესაძლოა იყოს შეცდომა
256	E_USER_ERROR	სამომხმარებლოს გარდაუვალი შეცდომა. ეს მსგავსია პროგრამისტის მიერ PHP ფუნქცია trigger_error()-ის გამოყენებით დასმული E_ERROR-სა
512	E_USER_WARNING	არა გარდაუვალი სამომხმარებლო გაფრთხილება. ეს მსგავსია პროგრამისტის მიერ PHP ფუნქცია trigger_error()-ის გამოყენებით დასმული E_WARNING-სა
1024	E_USER_NOTICE	სამომხმარებლო განცხადება. ეს მსგავსია პროგრამისტის მიერ PHP ფუნქცია trigger_error()-ის გამოყენებით დასმული E_NOTICE -სა
4096	E_RECOVERABLE_ERROR	დამჭერი გარდაუვალი შეცდომა. ეს მსგავსია E_ERROR-სა, მაგრამ შესაძლოა გამოიწვიოს მომხმარებლის განსაზღვრულმა ქმედებამ
8191	E_ALL	ყველა შეცდომა და გაფრთხილება, გარდა E_STRICT -სა

ახლა შევქმნათ შეცდომის შეტყობინების ფუნქცია:

```
function customError($errno, $errstr)
 {
 echo "<b>Error:</b> [$errno] $errstr<br />";
 echo "Ending Script";
 die();
 }
```

ზემოთ მოყვანილი კოდი არის უბრალო შეცდომის გამოტანის ფუნქცია. როდესაც ის გაეშვება, ის იღებს შეცდომის დონეს და შეცდომის შეტყობინებას. ის შემდეგ დაბეჭდავს შეცდომის დონეს და შეტყობინებას და შეაჩერებს სკრიპტის მუშაობას.

ახლა, როდესაც ჩვენ შევქმენით შეცდომის გამოტანის ფუნქცია, ჩვენ უნდა გადავწყვიტოთ თუ როდის გავუშვათ იგი.

შეცდომის შეტყობინების დასმა

PHP-სთვის სტანდარტული შეცდომის შეტყობინება არის ჩაშენებული შეცდომის გამომტანი. ზემოთ ჩვენ ვცდილობთ შევქმნათ სტანდარტული შეცდომის გამოტანის ფუნქცია სკრიტის მუშაობის დროისათვის.

შესაძლებელია შეიცვალოს შეცდომის გამომტანი, რათა ის გამოვიყენოთ მხოლოდ რამოდენიმე შეცდომაზე. ამ გზით სკრიპტს შეუძლია მართოს სხვადასხვა შეცდომები,

სხვადასხვა გზით. თუმცა, ამ მაგალითში შევქმნით ჩვენთვის სასურველ შეცდომას ყოველი შეცდომისათვის:

```
set_error_handler("customError");
```

მას შემდეგ, რაც ჩვენ მოვინდომებთ რომ ჩვენმა ფუნქციამ გამოიტანოს ყველა შეცდომა, set_error_handler()-ს სჭირდება მხოლოდ ერთი პარამეტრი, ხოლო მეორე პარამეტრი დასჭირდება შეცდომის დონის სპეციფიკაციისათვის.

მაგალითი:

დავტესტოთ შეცდომის გამომტანი, იმ ცვლადის დაბეჭდვით, რომელიც არ არსებობს:

```
<?php
//error handler function
function customError($errno, $errstr)
 {
 echo "<b>Error:</b> [$errno] $errstr";
 }
//set error handler
set_error_handler("customError");
//trigger error
echo($test);
?>
```

კოდი დაბეჭდავს მსგავსს:

```
Custom error: [8] Undefined variable: test
```

შეცდომის გაშვება

სკრიპტში სადაც მომხმარებელს შეუძლია შეიყვანოს მონაცემები, საკმაოდ გამოყენებადია გავუშვათ შეცდომა, როდესაც აღმოჩენილ იქნება არალეგალური მონაცემები. PHP-ში ეს ხდება trigger_error() ფუნქციით.

მაგალითი:

მაგალითში გამოვა შეცდომა, თუ "test" ცვლადი მეტია, ვიდრე "1":

```
<?php
$test=2;
if ($test>1)
{
trigger_error("Value must be 1 or below");
}
?>
```

კოდი დაბეჭდავს მსგავსს:

```
Notice: Value must be 1 or below
in C:\webfolder\test.php on line 6
```

შეცდომის გაშვება შესაძლებელია სკრიპტში, ნებისმიერ ადგილზე და მეორე პარამეტრის დამატებით, ჩვენ შეგვიძლია შეცდომის დონის სპეციფიკაციის გაკეთება.

შესაძლო შეცდომის ტიპები:

- E_USER_ERROR - გარდაუვალი სამომხმარებლო რან-თაიმ შეცდომა. სკრიპტის მუშაობა შეჩერდება.
- E_USER_WARNING - არაგარდაუვალი სამომხმარებლო რან-თაიმ გაფრთხილება. სკრიპტის მუშაობა არ ჩერდება.
- E_USER_NOTICE - სტანდარტული. სამომხმარებლო რან-თაიმ შენიშვნა. სკრიპტმა იპოვა რაღაც, რომელიც შესაძლოა იყოს შეცდომა.

მაგალითი:

ეს არის E_USER_WARNING მაგალითი. გამოვა შეცდომა, თუ "test" ცვლადი მეტია, ვიდრე "1". თუ E_USER_WARNING გამოვა ჩვენ გამოვიყენებთ ჩვენ მიერ შექმნილ შეცდომის გამოტანას და შევაჩერებთ სკრიპტის მუშაობას:

```
<?php
//error handler function
function customError($errno, $errstr)
 {
 echo "<b>Error:</b> [$errno] $errstr<br />";
 echo "Ending Script";
 die();
 }
//set error handler
set_error_handler("customError",E_USER_WARNING);
//trigger error
$test=2;
if ($test>1)
 {
 trigger_error("Value must be 1 or below",E_USER_WARNING);
 }
?>
```

კოდი დაბეჭდავს მსგავსს:

```
Error: [512] Value must be 1 or below
Ending Script
```

ახლა უკვე როდესაც ვისწავლეთ საკუთარი შეცდომების შექმნა და მათი გაშვება, შევისწავლოთ შეცდომის აღრიცხვა.

შეცდომის აღრიცხვა

სტანდარტულად, PHP აგზავნის შეცდომის აღრიცხვას სერვერის აღრიცხვის სისტემაზე, ან ფაილს დამოკიდებული, თუ როგორი error_log კონფიგურაცია არის დასმული php.ini ფაილში. error_log() ფუნქციის გამოყენებით ჩვენ შეგვიძლია გავგზავნოთ შეცდომის აღრიცხვები სპეციფიურ ფაილში, ან მთავარი დანიშნულების ადგილას.

შეცდომების შეტყობინებების გაგზავნა ელ-ფოსტით შესაძლოა იყოს კარგი გზა სპეციფიური შეცდომების შეტყობინებისათვის.

შეცდომის შეტყობინების გაგზავნა ელ-ფოსტაზე

ქვემოთ მოყვანილ მაგალითში, თუ მოხდება სპეციფიური შეცდომა ჩვენ გავგზავნით ფოსტას შეცდომის შეტყობინებით და დავასრულებთ სკრიპტის მუშაობას:

```
<?php
//error handler function
function customError($errno, $errstr)
 {
 echo "<b>Error:</b> [$errno] $errstr<br />";
 echo "Webmaster has been notified";
 error_log("Error: [$errno] $errstr",1,
 "someone@example.com","From: webmaster@example.com");
}
//set error handler
set_error_handler("customError",E_USER_WARNING);
//trigger error
$test=2;
if ($test>1)
 {
 trigger_error("Value must be 1 or below",E_USER_WARNING);
 }
?>
```

კოდი დაბეჭდავს მსგავსს:

```
Error: [512] Value must be 1 or below
Webmaster has been notified
```

და ელ-ფოსტაზე მიიღება მსგავსი:

```
Error: [512] Value must be 1 or below
```

ეს არ გამოიყენება ყველა შეცდომაზე. რეგულარული შეცდომები აღირიცხება სერვერზე, PHP სტანდარტული აღრიცხვის სისტემის გამოყენებით.

PHP გამონაკლისი

რა არის გამონაკლისი?

PHP 5-ში შემოვიდა ახალი შეცდომებთან გარიგებაზე ორიენტირებული ობიექტი.

გამონაკლისის გამომტანი გამოიყენება კოდის ნორმალური მუშაობის შესაცვლელად, თუ გამოვა სპეციფიური შეცდომა. ამ პირობას ქვია გამონაკლისი.

ასე ხდება როდესაც გამონაკლისი გაეშვება:

- მიმდინარე კოდი დამახსოვრდება
- კოდის შესრულება შეუერთდება წინასწარ განსაზღვრულ გამონაკლისის გამომტან ფუნქციას.
- სიტუაციაზე დამოკიდებულად, გამომტანი შესაძლოა ამუშავდეს დამახსოვრებული კოდიდან,შეაჩეროს სკრიპტის მუშაობა, ან გააგრძელოს იგი კოდის სხვა ადგილიდან.

ქვემოთ ნაჩვენებია შეცდომის გამოტანის სხვადასხვა მეთოდები:

- ბაზური გამოყენების გამონაკლისები
- ინდივიდუალური გამონაკლისების შექმნა
- რამოდენიმე გამონაკლისი
- გამონაკლისის ასხლეტვა
- ზე-დონის გამონაკლისის გამომტანის კონფიგურაციები

შენიშვნა: გამონაკლისების გამოყენება შესაძლოა მხოლოდ შეცდომისთვის და ის არ გამოიყენება კოდის სხვა წერტილში გადასახტომად.

ბაზური გამოყენების გამონაკლისები

როდესაც გამონაკლისი გასროლილია, მასზე გაყოლილი კოდი არ იმუშავებს და PHP შეეცდება იპოვოს "დაჭერა" ბლოკის დამთხვევა.

თუ გამონაკლისი არ იქნება დაჭერილი, გამოვა გარდაუვალი შეცდომა "Uncaught Exception" შეტყობინებით.

მაგალითი:

```
<?php
//create function with an exception
function checkNum($number)
 {
 if($number>1)
  {
  throw new Exception("Value must be 1 or below");
  }
 return true;
 }

//trigger exception
checkNum(2);
?>
```

ზემოთ მოყვანილი კოდი მიიღებს მსგავს შეცდომას:

```
Fatal error: Uncaught exception 'Exception'
with message 'Value must be 1 or below' in C:\webfolder\test.php:6
Stack trace: #0 C:\webfolder\test.php(12):
checkNum(28) #1 {main} thrown in C:\webfolder\test.php on line 6
```

ცდა, გასროლა და დაჭერა

ზემოთ მოყვანილ მაგალითში შეცდომისგან თავის ასარიდებლად, ჩვენ გვჭირდება შევქმნათ გამონაკლისის გამოტანის ჩვეული კოდი.

გამონაკლისის ჩვეული კოდი შეიცავს:

1. ცდა - ფუნქცია იყენებს გამონაკლისს, რომელიც იქნება "try" ბლოკში. თუ გამონაკლისი არ გაეშვება, კოდი გააგძელებს ნორმალურად მუშაობას. თუმცა, თუ გამონაკლისი გაეშვება, გამონაკლისი იქნება"გასროლილი".
2. გასროლა - ეს არის, თუ როგორ ვუშვებთ გამონაკლისს. თითოეულ "გასროლას" უნდა ჰქონდეს სულ მცირე ერთი "დაჭერა".

3. დაჭერა- "დაჭერა" ბლოკი იპოვის გამონაკლისს და შექმნის ობიექტს, რომელიც შეიცავს გამონაკლისის ინფორმაციას.

მაგალითი:

```
<?php
//create function with an exception
function checkNum($number)
 {
 if($number>1)
  {
  throw new Exception("Value must be 1 or below");
  }
 return true;
 }

//trigger exception in a "try" block
try
 {
 checkNum(2);
 //If the exception is thrown, this text will not be shown
 echo 'If you see this, the number is 1 or below';
 }

//catch exception
catch(Exception $e)
 {
 echo 'Message: ' .$e->getMessage();
 }
?>
```

ზემოთ მოყვანილი კოდი მიიღებს შეცდომას:

```
Message: Value must be 1 or below
```

მაგალითის ახსნა:

ზემოთ მოყვანილი კოდი ისვრის და იჭერს გამონაკლისებს:

1. checkNum() ფუნქცია შექმნილია. ის ამოწმებს, რიცხვი მეტია, თუ არა 1-ზე. თუ ის მეტია, გამონაკლისი გაისროლება.
2. checkNum() ფუნქცია გამოძახებულია "try" ბლოკში
3. გამონაკლისი checkNum() ფუნქციაში გასროლილია
4. "catch" ბლოკი პოულობს გამონაკლისს და ქმნის ობიექტს($e) , რომელიც შეიცავს გამონაკლისის ინფორმაციას
5. შეცდომის შეტყობინება გამონაკლისიდან დაბეჭდილია $e->getMessage()-ის გამოძახებით გამონაკლისი ობიექტიდან.

თუმცა, არსებობს ერთი გზა "ყველა გასროლა დაჭერილ უნდა იქნას" წესის თავიდან ასარიდებლად. ესაა შეცდომებზე ზე-დონის გამონაკლისების გამომტანის დანიშვნა, რომელიც გვერდს აუვლის მას.

ინდივიდუალური გამონალკისების კლასის შექმნა

ინდივიდუალური გამონაკლისების გამოტანის შექმნა საკმაოდ ადვილია. ჩვენ უბრალოდ ვქმნით სპეციუალურ კლასის ფუნქციებს, რომელთა გამოძახება შესაძლებელი იქნება,

როდესაც PHP-ში გამოჩნდება გამონაკლისი. კლასი უნდა იყოს გამონაკლისის კლასის გაფართოება.

გამონაკლისის ინდივიდუალური კლასი მემკვიდრეობით მიიღებს თვისებებს PHP-ს გამონაკლისების კლასიდან და ჩვენ შეგვეძლება დავამატოთ მას ინდივიდუალური ფუნქციები.

შევქმნათ გამონაკლისის კლასი:

```
<?php
class customException extends Exception
 {
 public function errorMessage()
  {
  //error message
  $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
  .': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
  return $errorMsg;
  }
 }
$email = "someone@example...com";
try
 {
 //check if
 if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE)
  {
  //throw exception if email is not valid
  throw new customException($email);
  }
 }
catch (customException $e)
 {
 //display custom message
 echo $e->errorMessage();
 }
?>
```

ახალი კლასი არის ძველი გამონაკლისის კლასის ასლი errorMessage() ფუნქციით. მას შემდეგ რაც ის გახდა ძველი კლასის ასლი და მას მემკვიდრეობით მიენიჭა ძველი კლასის თვისებები და მეთოდები, ჩვენ შეგვიძლია გამოვიყენოთ შემდეგი გამონაკლისების კლასის მეთოდები - getLine() და getFile() და getMessage().

მაგალითის ახსნა:

ზემოთ მოყვანილი კოდი გაისვრის გამონაკლისს და დაიჭერს მას ინდივიდუალური გამონაკლისის კლასით:

1. customException() კლასი შექმნილია, როგორც ძველი გამონაკისის კლასის გაფართოება. ამ მეთოდით ის მემკვიდრეობით იღებს ძველი კლასის თვისებებს და მეთოდებს
2. errorMessage() ფუნქცია შექმნილია. ფუნქცია დააბრუნებს შეცდომის შეტყობინებას, თუ ელ-ფოსტის მისამართი არასწორია
3. "try" ბლოკი ამუშავდა და გამონაკლისი გაისროლა, მას შემდეგ რაც ელ-ფოსტის მისამართი არასწორი აღმოჩნდა
4. "catch" ბლოკი იჭერს გამონაკლისს და გამოსახავს შეცდომის შეტყობინებას

სკრიპტს შეუძლია გამოიყენოს რამოდენიმე გამონაკლისი, რამოდენიმე პირობის შესამოწმებლად.

შესაძლებელია გამოვიყენოთ რამოდენიმე if..else ბლოკი, გამანაწილებელი, ან რამოდენიმე გამონაკლისის ბუდე. ამ გამონაკლისებს შეუძლიათ გამოიყენონ რამოდენიმე სხვა გამონაკლისის კლასები და და დააბრუნონ სხვადასხვა შეცდომის შეტყობინებები:

```
<?php
class customException extends Exception
{
public function errorMessage()
{
//error message
$errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
.': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
return $errorMsg;
}
}

$email = "someone@example.com";

try
 {
 //check if
 if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE)
  {
  //throw exception if email is not valid
  throw new customException($email);
  }
 //check for "example" in mail address
 if(strpos($email, "example") !== FALSE)
  {
  throw new Exception("$email is an example e-mail");
  }
 }

catch (customException $e)
 {
 echo $e->errorMessage();
 }
catch(Exception $e)
 {
 echo $e->getMessage();
 }
?>
```

მაგალითის ახსნა:

ზემოთ მოყვანილი კოდი ტესტირებას უკეთებს ორ პირობას და გაისვრის გამონაკლისს, თუ არ შეხვდა რომელიმე პირობა:

1. customException() კლასი შექნილია როგორც ძველი გამონაკლისის გაფართოება.
2. errorMessage() ფუნქცია შექმნილია. ფუნქცია დააბრუნებს შეცდომის შეტყობინებას, თუ ელ-ფოსტის მისამართი არასწორია
3. "try" ბლოკი გაშვებულია და გამონაკლისი გასროლილია პირველ პირობაზე
4. მეორე პირობა უშვებს გამონაკლისს მას მერე, რაც ელ-ფოსტა შეიცავს სტრინგს "example"

5. "catch" ბლოკი იჭერს გამონაკლისს და გამოსახავს შეცდომის შეტყობინებას

თუ არ მოხდება ინდივიდუალური გამონაკლისის დაჭერა, დაიჭირება მხოლოდ ბაზური გამონაკლისი, გამონაკლისი გამოვა იქ.

გამონაკლისების ასხლეტა

ზოგჯერ, როდესაც გამონაკლისი გაისროლება, ჩვენ შესაძლოა ვისურვოთ მისი გამოტანა სტანდარტული გზისგან განსხვავებულად. ეს შესაძლებელია გამონაკლისის მეორეჯერ გასროლით "catch" ბლოკში.

სკრიპტი დამალავს შეცდომას მომხმარებლებისაგან. სისტემური შეცდომები შესაძლებელია მნიშვნელოვანი იყოს პროგრამისტისათვის, მაგრამ ეს არ აინტერესებდეს მომხმარებელს. იმისათვის რომ გავაკეთოთ მსგავსი რამ მომხმარებლისათვის ჩვენ შეგვიძლია გავაკეთოთ გამონაკლისის ასხლეტა "მეგობრული" შეტყობინებით:

```
<?php
class customException extends Exception
 {
 public function errorMessage()
  {
  //error message
  $errorMsg = $this->getMessage().' is not a valid E-Mail address.';
  return $errorMsg;
  }
 }
$email = "someone@example.com";
try
 {
 try
  {
  //check for "example" in mail address
  if(strpos($email, "example") !== FALSE)
   {
   //throw exception if email is not valid
   throw new Exception($email);
   }
  }
 catch(Exception $e)
  {
  //re-throw exception
  throw new customException($email);
  }
 }
catch (customException $e)
 {
 //display custom message
 echo $e->errorMessage();
 }
?>
```

მაგალითის ახსნა:

ზემოთ მოყვანილი მაგალითი ტესტავს ელ-ფოსტის მისამართს, რომელიც შეიცავს სტრინგს "example", თუ არადა, გამონაკლისი აისხლიტება:

1. customException() კლასი შექმნილია როგორც ძველი გამონაკლისის გაფართოება

2. errorMessage()ფუნქცია შექმნილია. ფუნქცია დააბრუნებს შეცდომის შეტყობინებას, თუ ელ-ფოსტის მისამართი არასწორია
3. "try" ბლოკი შეიცავს სხვა "try" ბლოკს, რათა შესაძლებელი იყოს გამონაკლისის ასხლეტა
4. გამონაკლისი გაშვებულია მას შემდეგ, რაც ელ ფოსტა შეიცავს სტრინგს "example"
5. "catch" ბლოკი იჭერს გამონაკლისს და ისხლიტავს "customException"
6. "customException" დაჭერილია და ის გამოსახავს შეცდომის შეტყობინებას

თუ გამონაკლისი არ არის დაჭერილი მიმდინარე "try" ბლოკში, ის მოიძებნება დაჭერის ბლოკში "უფრო მაღალი დონის" საშუალებით.

ზე-დონის გამონაკლისების გამოტანის დასმა

```
<?php
function myException($exception)
{
echo "<b>Exception:</b> " , $exception->getMessage();
}
set_exception_handler('myException');
throw new Exception('Uncaught Exception occurred');
?>
```

კოდი დაბეჭდავს შემდეგს:

```
Exception: Uncaught Exception occurred
```

ზემოთ მოყვანილი კოდი არ იყო "catch" ბლოკი. ეს ფუნქცია გამოიყენება დაუჭერელი გამონაკლისების დასაჭერად.

წესები გამონაკლისებისათვის

- კოდი შესაძლოა მოექცეს ცდის ბლოკში, პოტენციალური გამონაკლისების დასაჭერად
- თითოეული ცდის ბლოკს, ან "გასროლას" უნდა ქონდეს სულ მცირე ერთი შესაბამისი დაჭერის ბლოკი
- რამოდენიმე დაჭერის ბლოკი შესაძლებელია გამოყენებულ იქნას გამონაკლისების სხვადასხვა კლასებში
- გამონაკლისები შესაძლოა გასროლილ(ასხლეტილ) იქნან ცდის ბლოკში არსებული დაჭერის ბლოკიდან

მარტივი წესი: თუკი ვისვრით რამეს, ჩვენ უნდა დავიჭიროთ ის.

PHP ფილტრი

რა არის PHP ფილტრი?

PHP ფილტრი გამოიყენება არა საიმედო წყაროებიდან მოსული მონაცემების დასამტკიცებლად და გასაფილტრად.

რატომ გამოვიყენოთ ფილტრი?

თითქმის ყველა ვებ პროგრამა დაცულია გარე შეყვანისაგან. საერთოდ ეს მოდის მომხმარებლისაგან, ან სხვა პროგრამისაგან. ფილტრების გამოყენებით ჩვენ შევძლებთ დავრწმუნდეთ იმაში, რომ ჩვენი პროგრამა იღებს სწორ შეყვანის ტიპს.

ჩვენ ყოველთვის უნდა გავფილტროთ გარე მონაცემები!

რა არის გარე მონაცემები?

- ფორმიდან შეყვანილი ინფორმაცია
- ბმულები
- ვებ სერვისის მონაცემები
- სერვერის ცვლადები
- მონაცემთა ბაზების მოთხოვნის შედეგები

ფუნქციები და ფილტრები

იმისათვის რომ გავფილტროთ ცვლადი, გამოვიყენოთ ერთ-ერთი ფილტრაციის ფუნქცია:

- filter_var() - სპეციფიური ფილტრით, ფილტრავს თითო ცვლადს
- filter_var_array() - ფლიტრავს რამოდენიმე ცვლადს, ამ, ან სხვა ფილტრით
- filter_input - მივიღოთ ერთი ცვლადი და გავფილტროთ ის
- filter_input_array - მივიღოთ რამოდენიმე ცვლადი და გავფლიტროთ იგი ამ, ან სხვა ფილტრით

ქვემოთ მოყვანილ მაგალითში, ჩვენ ვამტკიცებთ რიცხვის filter_var() ფუნქციას:

```
<?php
$int = 123;
if(!filter_var($int, FILTER_VALIDATE_INT))
 {
 echo("Integer is not valid");
 }
else
 {
 echo("Integer is valid");
 }
?>
```

ზემოთ მოყვანილი კოდი იყენებს "FILTER_VALIDATE_INT" ფილტრს. კოდი დაბეჭდავს: "Integer is valid".

თუ ვცდით ცვლადს, რომელიც არ არის მთელი (როგორც "123abc"), კოდი დაბეჭდავს: "Integer is not valid".

დამტკიცება და სანიტარია

აქ მოყვანილია ფილტრის ორი სახეობა:

ფილტრების დამტკიცება:

- გამოიყენება მომხმარებლის მიერ შეყვანილი ინფორმაციის გასაფილტრად
- წესების მკაცრი ფორმატი

ფილტრების სანიტარია:

- გამოიყენება სტრინგის სპეციფიური სიმბოლოების დასაშვებად ან ასაკრძალად
- არ გააჩნია მონაცემთა ფორმატის წესები
- ყოველთვის აბრუნებს სტრინგს

პარამეტრები და დროშები

პარამეტრები და დროშები გამოიყენება სპეციფიურ ფილტრებზე დამატებითი ფილტრაციის პარამეტრების დამატებისთვის.

სხვადასხვა ფილტრებს გააჩნიათ სხვადასხვა პარამეტრები და დროშები.

ქვემოთ მოყვანილ მაგალითში, ჩვენ ვამტკიცებთ filter_var(), "min_range" და "max_range" პარამეტრების მომხმარებელ რიცხვს:

```
<?php
$var=300;
$int_options = array(
"options"=>array
 (
 "min_range"=>0,
 "max_range"=>256
 )
);
if(!filter_var($var, FILTER_VALIDATE_INT, $int_options))
 {
 echo("Integer is not valid");
 }
else
 {
 echo("Integer is valid");
 }
?>
```

ზემოთ მოყვანილი კოდის მსგავსად, პარამეტრები უნდა ჩაიდოს ასოციაციურ მასივში ”options” სახელით. თუ დროშა გამოიყენება მას არ დასჭირდება მასივი.

მას შემდეგ რაც მთელი რიცხვი მიაღწევს "300"-ს, ის არ არის სპეციფიურ მწკრივში და კოდი დაბეჭდავს შემდეგს: "Integer is not valid".

შეტანის დამტკიცება

პრიველი რაც უნდა გავაკეთოთ, არის ის რომ დამატკიცოთ შეტანილი მონაცემები.

შემდეგ გავფლიტროთ შეტანილი მონაცემები filter_input() ფუნქციის გამოყენებით.

ქვემოთ მოყვანილ მაგალითში, შეყვანილი ცვლადი "email” გაგზავნილია PHP გვერდზე:

```
<?php
if(!filter_has_var(INPUT_GET, "email"))
 {
 echo("Input type does not exist");
 }
```

```
else
 {
 if (!filter_input(INPUT_GET, "email", FILTER_VALIDATE_EMAIL))
  {
  echo "E-Mail is not valid";
  }
 else
  {
  echo "E-Mail is valid";
  }
 }
?>
```

მაგალითის ახსნა:

ზემოთ მოყვანილ მაგალითს გააჩნია ველი, გაგზავნილი "GET" მეთოდით:

1. შევამოწმოთ, თუ არსებობს "GET" ტიპის "email" ცვლადი
2. თუ ცვლადი არსებობს, შევამოწმოთ ელ-ფოსტის მისამართის სისწორე

შეტანის სანიტარია

ჯერ დავრწმუნდეთ რომ შეტანის ველი არსებობს.

შემდეგ გავუკეთოთ სანიტარია შეტანილ მონაცემებს filter_input() ფუნქციის დახმარებით.

ქვემოთ მოყვანილ მაგალითში, ცვლადი "url" გაგზავნილია PHP გვერდზე:

```
<?php
if(!filter_has_var(INPUT_POST, "url"))
 {
 echo("Input type does not exist");
 }
else
 {
 $url = filter_input(INPUT_POST,
 "url", FILTER_SANITIZE_URL);
 }
?>
```

მაგალითის ახსნა:

ზემოთ მოყვანილ მაგალითს აქვს ველი, გაგზავნილი "POST" მეთოდით:

1. შევამოწმოთ, თუ არსებობს "POST" ტიპის "url" ცვლადი
2. თუ ცვლადი არსებობს, გავუკეთოთ სანიტარიზაცია და მოვაგროვოთ ის $url ცვლადში

თუ სტრინგი ამის მსგავსია : "http://geocg.myweb.ge/", $url ცვლადი სანიტარიის მერე გამოიყურება ასე:

```
http://geocg.myweb.ge/
```

რამოდენიმე შეტანის ფილტრაცია

ფორმა თითქმის ყოველთვის შეიცავს ერთზე მეტ შეტანის ველს. filter_var ან filter_input ფუნქციების გამოძახების თავიდან ასარიდებლად, გამოვიყენოთ filter_var_array, filter_input_array ფუნქციები.

ამ მაგალითში ჩვენ ვიყენებთ filter_input_array() ფუნქციას, რათა გავფილტროთ სამი GET ცვლადი. მიღებული GET ცვლადები არის სახელი, წლოვანება და ელ-ფოსტის მისამართი:

```
<?php
$filters = array
 (
 "name" => array
  (
  "filter"=>FILTER_SANITIZE_STRING
  ),
 "age" => array
  (
  "filter"=>FILTER_VALIDATE_INT,
  "options"=>array
   (
   "min_range"=>1,
   "max_range"=>120
   )
  ),
 "email"=> FILTER_VALIDATE_EMAIL,
 );
$result = filter_input_array(INPUT_GET, $filters);
if (!$result["age"])
 {
 echo("Age must be a number between 1 and 120.<br />");
 }
elseif(!$result["email"])
 {
 echo("E-Mail is not valid.<br />");
 }
else
 {
 echo("User input is valid");
 }
?>
```

ნაწილი III

PHP მონაცემთა ბაზები

PHP MYSQL შესავალი

რა არის MYSQL?

MySQL არის მონაცემთა ბაზა. მონაცემთა ბზა განსაზღვრავს ინფორმაციის მოგროვების სტრუქტურას.

მონაცემთა ბაზაში არის ცხრილები. ისევე როგორც HTML ცხრილები, მონაცემთა ბაზის ცხრილები შეიცავენ რიგებს, სვეტებს და უჯრედებს.

მონაცემთა ბზაზების გამოყენება სასარგებლოა, მაშინ, როცა ვაგროვებთ კატეგორიულ ინფორმაციას. კომპანიას შესაძლოა ქონდეს მონაცემთა ბაზა მიმდინარე ცხრილებით: "მოსამსახურეები", "პროდუქტები", "მყიდველები" და "შეკვეთები".

მონაცემთა ბაზის ცხრილები

მონაცემთა ბაზა ყველაზე ხშირად შეიცავს ერთ, ან რამდენიმე ცხრილს. თითოეულ ცხრილს გააჩნია საკუთარი სახელი. თითოეული ცხრილი შეიცავს მონაცემთა ჩანაწერებს.

ქვემოთ მოყვანილია სახელად "Persons" ცხრილის მაგალითი:

გვარი	სახელი	მისამართი	ქალაქი
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

ზემოთ მოყვანილი ცხრილი შეიცავს სამ ჩანაწერს (თითოს, თითო პიროვენებისათვის) და ოთხ სვეტს (გვარი, სახელი, მისამართი და ქალაქი).

მოთხოვნები

მოთხოვნა არის შეკითხვა, ან მოთხოვნა.

MySQL-თან ერთად, ჩვენ შეგვიძლია მოვითხოვოთ მონაცემთა ბაზა სპეციფიური ინფორმაციისათვის და და გვქონდეს დაბრუნებული ჩანაწერთა ბაზა.

შევხედოთ მიმდინარე მოთხოვნას:

```
SELECT LastName FROM Persons
```

ზემოთ მოყვანილი მოთხოვნა მონიშნავს პიროვნებათა მონაცემთა ბაზაში, გვარის სვეტში არსებულ ყველა მონაცემს და დააბრუნებს მას ჩანაწერთა ბაზის სახით:

გვარი
Hansen
Svendson
Pettersen

MYSQL - შეერთება მონაცემთა ბაზასთან

MYSQL მონაცემთა ბაზასთან შეერთება

სანამ ჩვენ გვექნება უფლება მონაცემთა ბაზის მონაცემებთან მუშაობისა, ჩვენ უნდა შევქმნათ კავშირი მონაცემთა ბაზებთან.

PHP-ში, ეს კეთდება mysql_connect() ფუნქციით.

სინტაქსი:

```
mysql_connect(servername,username,password);
```

პარამეტრი	აღწერა
servername	არააუცილებელი. სტანდარტული მნიშვნელობა არის "localhost:3306"
username	არააუცილებელი. მომხმარებლის სახელი სპეციფიკაცია
password	არააუცილებელი. მომხმარებლის პაროლის სპეციფიკაცია.

მაგალითი

მიმდინარე მაგალითში ჩვენ შევაგროვებთ კავშირებს ცვლადში ($con) მოგვიანებით სკრიპტში გამოსაყენებლად. "die" ნაწილი გაეშვება თუ კავშირი გაწყდება:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
```

```
  {
  die('Could not connect: ' . mysql_error());
  }
// some code
?>
```

კავშირის დახურვა

კავშირი დაიხურება, მაშინ როდესაც სრიპტი დაამთავრებს მუშაობას. მანამდე კავშირის დახურვისათვის გამოვიყენოთ the mysql_close() ფუნქცია.

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
// some code
mysql_close($con);
?>
```

MYSQL - მონაცემთა ბაზების და ცხრილების შექმნა

მონაცემთა ბაზის შექმნა

CREATE DATABASE ოპერატორი გამოიყენება MySQL მონაცემთა ბაზის შესაქმნელად.

სინტაქსი

```
CREATE DATABASE database_name
```

იმისათვის რომ PHP-მ ამუშავოს ზემოთ მოყვანილი ოპერატორი, უნდა გამოვიყენოთ mysql_query() ფუნქცია. ეს ფუნქცია გამოიყენება მოთხოვნის, ან ბრძანების MySQL კავშირზე გასაგზავნად.

მაგალითი

მიმდინარე მაგალითში ჩვენ შევქმნით ბაზას სახელად "my_db":

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
if (mysql_query("CREATE DATABASE my_db",$con))
  {
  echo "Database created";
  }
else
  {
  echo "Error creating database: " . mysql_error();
  }
mysql_close($con);
?>
```

ცხრილის შექმნა

CREATE TABLE ოპერატორი გამოიყენება MySQL-ში მონაცემთა ბაზების ცხრილების შესაქმნელად.

სინტაქსი

```
CREATE TABLE table_name
(
column_name1 data_type,
column_name2 data_type,
column_name3 data_type,
.......
)
```

აუცილებელია დავამატოთ CREATE TABLE ოპერატორი mysql_query() ფუნქციაში, რათა შესრულდეს ბრძანება.

მაგალითი

მაგალითი გვიჩვენებს როგორ შევქმნათ ცხრილი სახელად "person", სამი სვეტით. სვეტების სახელები იქნება "FirstName", "LastName" და "Age":

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
// Create database
if (mysql_query("CREATE DATABASE my_db",$con))
  {
  echo "Database created";
  }
else
  {
  echo "Error creating database: " . mysql_error();
  }
// Create table in my_db database
mysql_select_db("my_db", $con);
$sql = "CREATE TABLE person
(
FirstName varchar(15),
LastName varchar(15),
Age int
)";
mysql_query($sql,$con);
mysql_close($con);
?>
```

აუცილებელი: სანამ შეიქმნება ცხრილები უნდა აირჩეს მონაცემთა ბაზა. მონაცემთა ბაზა ირჩევა mysql_select_db() ფუნქციით.

შენიშვნა: როდესაც ვქმნით varchar ტიპის მონაცემთა ბაზის ველს, ჩვენ უნდა მივუთითოთ ველის მაქსიმალური ზომა, მაგ.: varchar(15).

MYSQL მონაცემთა ტიპები

ქვემოთ მოყვანილია სხვადასხვა MySQL მონაცემთა ტიპები:

რიცხობრივი მონაცემთა ტიპები	აღწერა
int(size) **smallint(size)** **tinyint(size)** **mediumint(size)** **bigint(size)**	შეიცავს მხოლოდ მთელ რიცხვებს. ციფრთა მაქსიმალური რაოდენობის სპეციფიკაცია შესაძლებელია size პარამეტრში
decimal(size,d) **double(size,d)** **float(size,d)**	შეიცავს არამთელ რიცხვებს. ციფრთა მაქსიმალური რაოდენობის სპეციფიკაცია შესაძლებელია size პარამეტრში. რიცხვის მარჯვენა მხარეს ციფრთა მაქსიმალური რაოდენობის სპეციფიკაცია შესაძლებელია d პარამეტრში

ტექსტური მონაცემთა ტიპები	აღწერა
char(size)	შეიცავს ფიქსირებულ სიგრძის სტრინგს. ფიქსირებული ზომის სპეციფიკაცია ხდება ფრჩხილებში
varchar(size)	შეიცავს ცვლადი სიგრძის სტრინგს. მაქსიმალური ზომის სპეციფიკაცია ხდება ფრჩხილებში
tinytext	შეიცავს ცვლად სტრინგს, მაქსიმუმ 255 სიმბოლოს
text **blob**	შეიცავს ცვლად სტრინგს, მაქსიმუმ 65535 სიმბოლოს
mediumtext **mediumblob**	შეიცავს ცვლად სტრინგს, მაქსიმუმ 16777215 სიმბოლოს
longtext **longblob**	შეიცავს ცვლად სტრინგს, მაქსიმუმ 4294967295 სიმბოლოს

თარიღის მონაცემთა ტიპები	აღწერა
date(yyyy-mm-dd) **datetime(yyyy-mm-dd hh:mm:ss)** **timestamp(yyyymmddhhmmss)** **time(hh:mm:ss)**	შეიცავს თარიღს და/ან დროს

Misc. მონაცემთა ტიპები	აღწერა
enum(value1,value2,ect)	ENUM არის ENUMERATED სიის შემოკლებული ვარიანტი. შეუძლია შეაგროვოს 65535 მნიშვნელობა. თუ მნიშვნელობა ჩამატებულია, ის არ არის სიაში, ჩაისმება ცარიელი მნიშვნელობა
set	SET ანალოგიურია ENUM-სა. თუმცა, SET შეუძლია ქონდეს 64 სია.

მონაცემთა ბაზის ცხრილში მონაცემის ჩასმა

INSERT INTO ოპერატორი გამოიყენება მონაცემთა ბაზის ცხრილში ახალი ჩანაწერის ჩასამატებლად.

სინტაქსი

```
INSERT INTO table_name
VALUES (value1, value2,....)
```

ასევე შესაძლებელია სვეტების სპეციფიკაცია, თუ სად გვსურს მონაცემთა ჩასმა:

```
INSERT INTO table_name (column1, column2,...)
VALUES (value1, value2,....)
```

შენიშვნა: SQL ოპერატორების დაწერა შესაძლებელია, როგორც პატარა, ასევე დიდი ასოებით.

იმისათვის რომ PHP-მ შეასრულოს ეს ოპერატორი უნდა გამოვიყენოთ mysql_query() ფუნქცია.

მაგალითი

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
mysql_select_db("my_db", $con);
mysql_query("INSERT INTO person (FirstName, LastName, Age)
VALUES ('Peter', 'Griffin', '35')");
mysql_query("INSERT INTO person (FirstName, LastName, Age)
VALUES ('Glenn', 'Quagmire', '33')");
mysql_close($con);
?>
```

მონაცემთა ბაზაში მონაცემების ფორმიდან ჩამატება

ახლა ჩვენ შევქმნით HTML ფორმას, რომელსაც გამოვიყენებთ ცხრილში ახალი ჩანაწერის ჩამატებისათვის.

HTML ფორმა:

```
<html>
<body>
<form action="insert.php" method="post">
Firstname: <input type="text" name="firstname" />
Lastname: <input type="text" name="lastname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
</body>
</html>
```

როდესაც მომხმარებელი იმოქმედებს submit ღილაკზე, ფორმაში არსებული მონაცემები გაიგზავნება "insert.php" ფაილში. "insert.php" ფაილი შეუერთდება მონაცემთა ბაზას და იპოვის მნისვნელობებს ფორმიდან, PHP $_POST ცვლადებთან ერთად. შემდეგ, mysql_query() ფუნქცია გაუშვებს INSERT INTO ოპერატორს და მონაცემთა ბაზის ცხრილს დაემატება ახალი ჩანაწერი.

ქვემოთ მოყვანილია "insert.php" გვერდის კოდი:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
mysql_select_db("my_db", $con);
$sql="INSERT INTO person (FirstName, LastName, Age)
VALUES
('$_POST[firstname]','$_POST[lastname]','$_POST[age]')";
if (!mysql_query($sql,$con))
  {
  die('Error: ' . mysql_error());
  }
echo "1 record added";
mysql_close($con)
?>
```

MYSQL - მონიშვნა

მონაცემთა ბაზის ცხრილიდან მონაცემის მონიშვნა

SELECT ოპერატორი გამოიყენება მონაცემთა ბაზის ცხრილში მონაცემის მოსანიშნად.

სინტაქსი

```
SELECT column_name(s)
FROM table_name
```

მაგალითი

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
mysql_select_db("my_db", $con);
$result = mysql_query("SELECT * FROM person");
while($row = mysql_fetch_array($result))
  {
  echo $row['FirstName'] . " " . $row['LastName'];
  echo "<br />";
  }
mysql_close($con);
?>
```

ზემოთ მოყვანილი მაგალითი აგროვებს mysql_query() ფუნქციით დაბუნებულ მონაცემებს $result ცვლადში. შემდეგ, ჩვენ ვიყენებთ mysql_fetch_array() ფუნქციას პირველი რიგის ჩანაწერთა ბაზდიდან დასაბრუნებლად, როგორც მასივი.

კოდი დაბეჭდავს:

```
Peter Griffin
Glenn Quagmire
```

რეზულტატების HTML ცხრილში გამოსახვა

მიმდინარე მაგალითი მონიშნულ მონაცემებს გამოსახავს HTML ცხრილში:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

$result = mysql_query("SELECT * FROM person");

echo "<table border='1'>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>";
while($row = mysql_fetch_array($result))
  {
  echo "<tr>";
  echo "<td>" . $row['FirstName'] . "</td>";
  echo "<td>" . $row['LastName'] . "</td>";
  echo "</tr>";
  }
echo "</table>";
mysql_close($con);
?>
```

კოდი დაბეჭდავს:

Firstname	Lastname
Glenn	Quagmire
Peter	Griffin

MYSQL - სადაც წინადადებაა

სადაც წინადადებაა

მხოლოდ იმ მონაცემთა მოსანიშნად, რომელიც ემთხვევა სპეციფიურ კრიტერიუმებს, დავამატოთ WHERE clause, SELECT ოპერატორს.

სინტაქსი

```
SELECT column FROM table
WHERE column operator value
```

მიმდინარე ოპერატორები შესაძლებელია გამოვიყენოთ WHERE clause-თან ერთად:

ოპერატორი	აღწერა
=	უდრის
!=	არ უდრის
>	მეტია ვიდრე
<	ნაკლებია ვიდრე
>=	მეტია ან ტოლი
<=	ნაკლებია ან ტოლი
BETWEEN	შუა, ან შემცველი რიგი
LIKE	მსგავსის ძებნა

მაგალითი

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

$result = mysql_query("SELECT * FROM person
WHERE FirstName='Peter'");

while($row = mysql_fetch_array($result))
  {
  echo $row['FirstName'] . " " . $row['LastName'];
  echo "<br />";
  }

?>
```

კოდი დაბეჭდავს:

```
Peter Griffin
```

MYSQL - საკვანძო სიტყვებით მითითებული

ORDER BY

ORDER BY გამოიყენება ჩანაწერთა ბაზაში მონაცემების დასალაგებლად.

სინტაქსი

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name
```

მაგალითი

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

$result = mysql_query("SELECT * FROM person ORDER BY age");

while($row = mysql_fetch_array($result))
  {
  echo $row['FirstName'];
  echo " " . $row['LastName'];
  echo " " . $row['Age'];
  echo "<br />";
  }
mysql_close($con);
?>
```

კოდი დაბეჭდავს:

```
Glenn Quagmire 33
Peter Griffin 35
```

ზრდადობით, ან კლებადობით დალაგება

თუ ვიყენებთ ORDER BY-ს, ჩანაწერთა ბაზის დალაგება მითითებულია სტანდარტულად.

გამოვიყენოთ DESC კლებადობით დალაგების მისათითებლად:

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name DESC
```

ორი მწკრივით მითითება

შესაძლებელია ერთზე მეტი მწკრივით მითითება. როდესაც ვუთითებთ ერთზე მეტი მწკრივით, მეორე მწკრივი გამოიყენება მხოლოდ მაშინ, თუ პირველი მწკრივის მნიშვნელობები იდენტურია:

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name1, column_name2
```

MYSQL - განახლება

მონაცემთა ბაზებში მონაცემთა განახლება

UPDATE ოპერატორი გამოიყენება მონაცემთა ბაზების ცხრილში მონაცემთა განახლებისათვის.

სინტაქსი

```
UPDATE table_name
SET column_name = new_value
WHERE column_name = some_value
```

მაგალითი

FirstName	LastName	Age
Peter	Griffin	35
Glenn	Quagmire	33

მიმდინარე მაგალითი აახლებს მონაცემებს "Person" ცხრილში:

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
mysql_select_db("my_db", $con);

mysql_query("UPDATE Person SET Age = '36'
WHERE FirstName = 'Peter' AND LastName = 'Griffin'");
mysql_close($con);
?>
```

განახლების შემდეგ, "Person" ცხრილი გამოიყურება ასე:

FirstName	LastName	Age
Peter	Griffin	36
Glenn	Quagmire	33

MYSQL - წაშლა

მონაცემთა წაშლა

DELETE FROM ოპერატორი გამოიყენება მონაცემთა ბაზაში მონაცემების წაშლისათვის.

სინტაქსი

```
DELETE FROM table_name
WHERE column_name = some_value
```

მაგალითი

FirstName	LastName	Age
Peter	Griffin	35
Glenn	Quagmire	33

მიმდინარე მაგალითი ცხრილში "Person" წაშლის ყველას მონაცემს, ვისი LastName='Griffin':

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
mysql_select_db("my_db", $con);

mysql_query("DELETE FROM Person WHERE LastName='Griffin'");
mysql_close($con);
?>
```

წაშლის შემდეგ ცხრილს ექნება სახე:

FirstName	LastName	Age
Glenn	Quagmire	33

ნაწილი IV

PHP XML

PHP XML EXPAT PARSER

რა არის XML?

XML გამოიყენება მონაცემთა აღწერისათვის და ფოკუსირებისათვის. XML ფაილი აღწერს მონაცემთა სტრუქტურას.

XML-ში, ტეგები არ არიან განსაზღვრულნი. ჩვენ უნდა განვსაზღვროთ საკუთარი ტეგები.

რა არის EXPAT?

წასაკითხად და განახლებისათვის - შევქმნათ და მანიპულაცია გავუკეთოთ - XML დოკუმენტს, ჩვენ დაგვჭირდება XML გამრჩეველი.

აქ არის ორი ბაზური ტიპი XML გამრჩეველისა:

- ხე-ბაზური გამრჩეველი: ეს გამრჩეველი გარდაქმნის XML დოკუმენტს ხისებრ სტრუქტურაში. ის ანალიზს უკეთებს მთლიან დოკუმენტს და უზრუნველყოფს ხის ელემენტებთან წვდომას. მაგ.: Document Object Model (DOM)
- შემთხვევა-ბაზური გამრჩეველი: ათვარიელებს XML დოკუმენტს, როგორც სერიების შემთხვევას. როდესაც სპეციფიური შემთხვევა მოხდება, ის გამოიძახებს ფუნქციას მის გამოსატანად

გამრჩეველის გაძევება არის შემთხვევა-ბაზური გამრჩეველი.

შემთხვევა-ბაზური გამრჩეველები ფოკუსირებას აკეთებენ XML დოკუმენტების შემცველობაზე და არა მათ სტრუქტურაზე.

შევხედოთ მიმდინარე XML ნაწილს:

```
<from>Jani</from>
```

ზემოთ, შემთხვევა-ბაზური გამრჩეველი ანგარიშს აბარებს XML-ს, როგორც სერიის სამ შემთხვევას:

- საწყისი ელემენტი: from
- საწყისი CDATA სექცია, მნიშვნელობა: Jani
- სასრული ელემენტი: from

ზემოთ მოყვანილი XML მაგალითი შეიცავს well-formed XML-ს. თუმცა, მაგალითი არასწორი XML-ია, რადგან აქ არ არის განსაზღვრული დოკუმენტის ტიპი(Document Type Definition (DTD)).

ინსტალაცია

XML გამრჩეველის გაძევება ფუნქციები არიან PHP-ს შემცველობაში. ამ ფუნქციების გამოყენებისათვის ინსტალაცია საჭირო არ არის.

XML ფაილი

ქვემოთ მოყვანილი XML ფაილი გამოყენებულ იქნება ჩვენ მაგალითში:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

XML გამრჩეველის ინიციალიზება

ჩვენ გვსურს XML გამრჩეველის ინიციალიზება PHP-ში, ზოგიერთი გამომტანის განსაზღვრა სხვადასხვა XML შემთხვევებისათვის და XML ფაილის გარჩევა.

მაგალითი

```
<?php
//Initialize the XML parser
$parser=xml_parser_create();
//Function to use at the start of an element
function start($parser,$element_name,$element_attrs)
  {
  switch($element_name)
    {
    case "NOTE":
    echo "-- Note --<br />";
    break;
    case "TO":
    echo "To: ";
    break;
    case "FROM":
    echo "From: ";
    break;
    case "HEADING":
    echo "Heading: ";
    break;
    case "BODY":
    echo "Message: ";
    }
  }
```

```
//Function to use at the end of an element
function stop($parser,$element_name)
  {
  echo "<br />";
  }
//Function to use when finding character data
function char($parser,$data)
  {
  echo $data;
  }
//Specify element handler
xml_set_element_handler($parser,"start","stop");
//Specify data handler
xml_set_character_data_handler($parser,"char");
//Open XML file
$fp=fopen("test.xml","r");
//Read data
while ($data=fread($fp,4096))
  {
  xml_parse($parser,$data,feof($fp)) or
  die (sprintf("XML Error: %s at line %d",
  xml_error_string(xml_get_error_code($parser)),
  xml_get_current_line_number($parser)));
  }
//Free the XML parser
xml_parser_free($parser);
?>
```

კოდი დაბეჭდავს:

```
-- Note --
To: Tove
From: Jani
Heading: Reminder
Message: Don't forget me this weekend!
```

როგორ მუშაობს:

1. გავაკეთოთ XML გამრჩეველის ინიციალიზება xml_parser_create() ფუნქციით
2. შევქმნათ ფუნქციები სხვადასხვა გამომტანების გამოსაყენებლად
3. დავამატოთ xml_set_element_handler() ფუნქცია, იმის სპეციფიკაციისათვის, თუ რომელი ფუნქცია გაეშვება, როცა გამრჩეველი შეხვდება ტეგების გახსნას და დახურვას
4. დავამატოთ xml_set_character_data_handler() ფუნქცია, იმის სპეციფიკაციისათვის, თუ რომელი ფუნქცია გაეშვება, როდესაც გამრჩეველი შეხვდება სიმბოლოების მონაცემებს.
5. გავარჩიოთ "test.xml" ფაილი xml_parse() ფუნქციით
6. შეცდომის შემთხვევაში, დავამატოთ xml_error_string() ფუნქცია XML შეცდომის კონვერტირებისათვის ტექსტურ აღწერაში
7. გამოვიძახოთ xml_parser_free() ფუნქცია, რათა განვათავისუფლოთ მახსოვრობა, რომელიც გაწაწილებულია xml_parser_create() ფუნქციასთან

PHP XML DOM

რა არის DOM?

W3C DOM უზრუნველყოფს სტანდარტულ ობიექტებს HTML და XML დოკუმენტებისათვის და სტანდარტულ ინტერფეისს მათზე წვდომისა და მანიპულაციისათვის.

W3C DOM გამოყოფილია განსხვავებულ ნაწილებში (Core, XML და HTML) და განსხვავებულ დონეებში (DOM დონე 1/2/3):

- Core DOM - განსაზღვრავს სტანდარტულ ობიექტებს დოკუმენტის ნებისმიერი სტრუქტურისათვის
- XML DOM - განსაზღვრავს სტანდარტულ ობიექტებს XML დოკუმენტისათვის
- HTML DOM - განსაზღვრავს სტანდარტულ ობიექტებს HTML დოკუმენტისათვის

DOM გარჩევა

DOM გამრჩეველი არის ხე-ბაზური გამრჩეველი.

შევხედოთ XML დოკუმენტის ნაწილს:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<from>Jani</from>
```

XML DOM ხედავს XML-ს, როგორც ხისებრ სტრუქტურას:

- დონე 1: XML დოკუმენტი
- დონე 2: მთავარი ელემენტი: <from>
- დონე 3: ტექსტური ელემენტი: "Jani"

XML ფაილი

ქვემოთ მოყვანილ XML ფაილს გამოვიყენებთ მაგალითში:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

XML-ის ჩატვირთვა და დაბეჭდვა

ჩვენ გვინდა XML გამრჩეველის ინიციალიზაცია, xml-ის ჩატვირთვა და მისი დაბეჭდვა:

მაგალითი

```
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("note.xml");

print $xmlDoc->saveXML();
?>
```

კოდი დაბეჭდავს:

```
Tove Jani Reminder Don't forget me this weekend!
```

თუ მოვნიშნავთ "View source" ბრაუზერში, ჩვენ დავინახავთ შემდეგ HTML კოდს:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

ზემოთ მოყვანილი კოდი ქმნის DOM დოკუმენტ-ობიექტს და ტვირთავს XML-ს "note.xml"-დან.

შემდეგ saveXML() ფუნქცია დებს შიდა XML დოკუმენტს სტრინგში, ისე რომ ჩვენ შეგვეძლოს მისი დაბეჭდვა.

XML ციკლი

ჩვენ გვინდა XML გამრჩეველის ინიციალიზება, XML-ის ჩატვირთვა და <note> ელემენტებს შორის ციკლი:

მაგალითი

```
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("note.xml");
$x = $xmlDoc->documentElement;
foreach ($x->childNodes AS $item)
  {
  print $item->nodeName . " = " . $item->nodeValue . "<br />";
  }
?>
```

კოდი დაბეჭდავს:

```
#text =
to = Tove
#text =
from = Jani
#text =
heading = Reminder
#text =
body = Don't forget me this weekend!
#text =
```

PHP SIMPLEXML

რა არის SIMPLEXML?

SimpleXML არის ახალი PHP 5-ში. ეს არის ადვილი გზა ელემენტის ატრიბუტებისა და ტექსტის მისაღებად, თუ ჩვენ ვიცით XML დოკუმენტის სქემა.

DOM, ან Expat parser-თან შედარებით, SimpleXML უბრალოდ იღებს კოდის რამოდენიმე ხაზს, ელემენტიდან მონაცემთა ტექსტის წასაკითხად.

SimpleXML აკონვერტებს XML დოკუმენტს ობიექტში, ამის მსგავსად:

- ელემენტები - კონვერტირებულნი არიან SimpleXMLElement ობიექტის თითო ატრიბუტად. როდესაც აქ არის ერთ დონეზე, ერთზე მეტი ელემენტი, ისინი განთავსდებიან მასშივში
- ატრიბუტები - აქვთ წვდომა ასოციაციური მასივების გამოყენებაზე, სადაც ინდექსი შეესაბამება ატრიბუტის სახელს
- ელემენტის მონაცემები - ტექსტური მონაცემები ელემენტებიდან კონვერტირებულნი არიან სტრინგში.

SimpleXML არის სწრაფი და ადვილი გამოსაყენებელი, როდესაც სრულდება ბაზური ამოცანები:

- XML ფაილების წაკითხვა
- XML სტრინგებიდან მონაცემთა ამოღება
- ტექსტური კვანძების, ან ატრიბუტების რედაქტირება

SIMPLEXML-ის გამოყენება

ქვემოთ მოყვანილია XML ფაილი:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

ვაკეთებთ შემდეგს:

1. ჩვატვირთოთ XML ფაილი
2. მივიღოთ პირველი ელემენტის სახელი
3. შევქმნათ ციკლი, რომელიც გაეშვება თითოეულ კვანძზე, children() ფუნქციის გამოყენებით
4. დავბეჭდოთ ელემენტის სახელი და მონაცემები თითოეული კვანძისათვის

```
<?php
$xml = simplexml_load_file("test.xml");
echo $xml->getName() . "<br />";
foreach($xml->children() as $child)
  {
  echo $child->getName() . ": " . $child . "<br />";
  }
?>
```

კოდი დაბეჭდავს:

```
note
to: Tove
from: Jani
heading: Reminder
body: Don't forget me this weekend!
```