მაგდა ცინცაძე

შესავალი PHP /MySQL-ში

რა არის PHP?

PHP იშიფრება, როგორც, Hypertext Preprocessor(ჰიპერტექსტული პრეპროცესორი) იგი სერვერული სკრიფტული ენაა, ისევე როგორც ASP უზრუნველყოფს მრავალი მონაცემთა ბაზის (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC და ა.შ.) მხარდაჭერას. შევნიშნოთ რომ PHP არის ღია პროგრამული საშუალება (OSS) და მისი გადმოწერა/მოხმარება სრულიად უფასოა. PHP ფაილები შეისაძლოა შეიცავდნენ ტექსტს, HTML ტეგებს და სკრიფტებს . PHP ფაილები ბრუნდებიან ბრაუზერში, როგორც უბრალო HTML და აქვთ შემდეგი გაფართოებები : ".php", ".php3", ".php4", ".php5", ან ".phtml"

რა არის MYSQL?

MySQL არის მონაცემთა ბაზის სერვერი, იდეალურია დიდი და პატარა პროგრამებისათვის, უზრუნველყოფს სტანდარტულ SQL-ის მხარდაჭერას, ეშვება სხვადასხვა პლადფორმაზე, მისი გადმოწერა და მოხმარება სრულიად უფასოა.

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

PHP სინტაქსი

PHP სკრიფტინგის ბლოკი ყველთვის იწყება**<?php** და მთავრდება **?>**. PHP ბლოკი შესაძლოა განთავსდეს დოკუმენტის ნებისმიერ ადგილზე.
სერვერებზე, რომლებიც უზრუნველყოფენ სტენოგრაფიას, ჩვენ შეგვიძლია დავიწყოთ სკრიფტინგის ბლოკი <? და დავამთავროთ ?>.
თუმცა, ექსცესების თავიდან ასარიდებლად რეკომენდირებულია დავიწყოთ <?php ქვემოთ მოცემულია PHP სკრიფტის უბრალო მაგალითი, რომელიც ბრაუზერში აბრუნებს "Hello World" :

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

ერთ ხაზიანი კომენტარის შესაქმნელად PHP-ში გამოიყენება // , ხოლო /* და*/ გამოიყენება დიდი კომენტარის ბლოკის გასაკეთებლად.
<html>

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

ცვლადები PHP-ში

ცვლადები გამოიყენება მნიშვნელობების დაბრუნებისათვის, როგორც ტექსტური სტრინგი, რიცხვები, ან მასივები. PHP-ში ყველა ცვლადი იწყება სიმბოლოთი $. PHP-ში ცვლადს არ ჭირდება გამოცხადება: PHP ავტომატურად აკონვერტებს ცვლადებს მართებულ მონაცემთა ტიპებში:

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

ცვლადისათვის სახელის განსაზღვრისას უნდა გავითვალისწინოთ:

- 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 ჰარის. თუ ცვლადის სახელი შედგება ერთზე მეტი სიტყვისაგან, მაშინ ეს სიტყვები უნდა გამოიყოს ქვედა ტირეთი ($my_string), ან საწყისი ასოებით ($myString)

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;
```

?>

კონკატენაციის (გაერთიანების) ოპერატორი

ოპერატორი (.) გამოიყენება ორი სტინგ მნიშვნეობის შესაერთებლად. განვიხილოთ მაგალითი:

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

STRLEN() და STRPOS() ფუნქციების გამოყენება

strlen() ფუნქცია გამოიყენება სტრინგის სიგრძის შესამოწმებლად. ხოლო 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. შევნიშნოთ რომ ათვლა იწყება 0-დან
განვიხილოთ მაგალითი:

```
<?php
echo strlen("Hello world!");  // დაწერს 12
echo strpos("Hello world!","world"); //დაწერს 6
?>
```

IF...ELSE 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.

სინტაქსი :

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

მაგალითი:

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

ELSEIF ოპერატორის სინტაქსი ასეთია:

```
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:

```
<?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!";
?>
```

თუ გვინდა მოვნიშნოთ კოდის ერთზე მეტი ბლოკი, გამოვიყენოთ Switch ოპერატორი. switch ოპერატორი გამოიყენება გრძელი if..elseif..else ოპერატორების გრძელი კოდის თავიდან ასაცილებლად.
სინტაქსი ასეთია:

```
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;
}
განვიხილოთ კონკრეტული მაგალითი:

```
<?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";
}
?>
```

Default გასაღები სიტყვა გამოიყენება თუ არც ერთი ვარიანტი არ არის ჭეშმარიტი. შესაბამისად თუ x-ის მნიშვნელობა არ იქნა 1,2 ან 3, გამოვა შეტყობინება **`"No number between 1 and 3"`**.

PHP მასივები

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(რიცხვითი მასივი) - მასივი რიცხობრივი ID გასაღებით

Associative array(ასოცირებული მასივი) - მასივი, სადაც თითორული ID გასაღები ასოცირებულია მნიშვნელობასთან.

მაგდა ცინცაძე

Multidimensional array(მულტისივრცული მასივი) - მასივი შეიცავს ერთ ან მეტ მასივს.
განვიხილოთ რიხცვითი მასივის მოცემის მაგალითები:

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

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

როდესაც ვაბრუნებთ სპეციფიური სახელების მონაცემებს, რიცხვითი მასივი ყოველთვის არ გამოგვადგება.
ასოცირებული მასივით ჩვენ შეგვიძლია მნიშვნელობები გამოვიყენოთ, როგორც გასაღებები და მივანიჭოთ მათ მნიშვნელობები.
განვიხილოთ მაგალითი:

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

მულტისივრცულ მასივს კი ექნება სახე:

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

მაგდა ცინცაძე

ციკლები PHP-ში

ძახშირად, კოდის წერისას, ერთი და იგივე ბლოკის გაშვება გვჭირდება რომოდენიმეჯერ.გამეორებების თავიდან ასაცილებლად ჩვენ შეგვიძლია გამოვიყენოთ ციკლის ოპერატორები.
PHP-ში არსებობს შემდეგი ციკლის ოპერატორები:

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

სინტაქსი ასეთია:
1.

```
while (condition)
code to be executed;
2.
 do
{
code to be executed;
}
while (condition);
3.
 for (initialization; condition; increment)
{
code to be executed;
}
4.
foreach (array as value)
{
code to be executed;
}
```

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 გახსნით "{" და დასრულდება ფიგურული ფრჩხილის დახურვით "}"
მაგალითი:

```
<html>
<body>
<?php
function writeMyName()
{
```

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

ფუნქციებს შესახლებელია გააჩნეთ პარამეტრი ან პარამეტრები, აგრეთვე შეუძლიათ დააბრუნონ მნიშვნელობები.
განვიხილოთ მაგალითი:

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

დააბრუნებს შეკრების შედეგს.

მონაცემების გამოქვეყნება ვებში

მთელი იდეა მონაცემთა ბაზაზე დამყარებული ვებ გვერდისა მდგომარეობს იმაში რომ გვერდის შიგთავსი მოთავსებული იყოს ბაზაში და მომხმარებლის მოთხოვნის შესაბამისად დინამიურად ხორციელდებოდეს ვებ გვერდის გენერირება ბრაუზერში ნახვისათვის. ანუ ერთის მხრივ თქვენ გყავთ ვიზიტორი რომელიც იყენებს ბრაუზერს იმისათვის რომ ნახოს გვერდი, ამასთან ის მოელის სტანდარტულ HTML დოკუმენტს. მეორეს მხრივ კი გაქვთ შიგთავსი რომელიც მოთავსებულია MySQL ბაზის ერთ ან რამოდენიმე ცხრილში და იგებს მხოლოდ მოთხოვნებს რომლებიც შეესაბამებიან SQL ბრძანებებს.

რა ხდება როდესაც ვიზიტორი შემოდის თქვენ მონაცემთა ბაზაზე დამყარებულ ვებ-გვერდზე?!

1. ვიზიტორის ვებ-ბრაუზერი მოითხოვს გვერდს სტანდარტული URL მისამართის გამოყენებით.

2. ვებ სერვერი (Apache, IIS, ან სხვა) ამოიცნობს რომ მოთხოვნილი ფაილი არის PHP სკრიპტი, ამდენად ახდენს ამ ფაილის ინტერპრეტაციას PHP plug-in-ის მეშვეობით მოთხოვნაზე პასუხამდე.
3. გარკვეული PHP ბრძანებები (რომლებიც ჩვენ უკვე განვიხილეთ - რომელი?) უკავშირდებიან MySQL ბაზას და მოითხოვენ ვებ-გვერდის შესაბამის შიგთავსს.
4. MySQL ბაზა პასუხობს რა მოთხოვნას უგზავნის შესაბამის შიგთავსს PHP სკრიპტს.
5. PHP სკრიპტი ათავსებს მოთხოვნილ შიგთავსს ერთ ან რამოდენიმე PHP ცვლადში, შემდგომ კი echo ან print-ის მეშვეობით გამოაქვს შესაბამისი შიგთავსი როგორც ვებ გვერდის ნაწილი.
6. PHP plug-in ამთავრებს სამუშაოს ახდენს რა HTML კოპიის შესრულებას რომელიც მან გააკეთა ვებ სერვერზე.
7. ვებ სერვერი უგზავნის HTML-ს ვებ ბროუზერს როგორც ჩვეულებრივ HTML ფაილს, რომელიც სინამდვილეში გენერირებული იქნა PHP plug- in -ის მეშვეობით.

განვიხილოთ მაგალითი

```
<html >
<head>
<title> LIST of NAMES</title>
</head>
<body>
<?php
$conn= @mysql_connect('localhost', 'root', ‘');
if (!$conn) {
exit('<p>Unable to connect to the ' .
'database server at this time.</p>');
}
if (!@mysql_select_db(‘baza')) {
exit('<p>Unable to locate the baza ' .
'database at this time.</p>');
}
?>
<p>Here are all the NAMES in our database:</p>
<blockquote>
<?php
$result = @mysql_query('SELECT NAMES FROM TABLE');
if (!$result) {
exit('<p> Error performing query: ' . mysql_error() . '</p>');
}
while ($row = mysql_fetch_array($result)) {
```

```
echo '<p>' . $row[‘firstname'] . '</p>';
}
?>
</blockquote>
</body>
</html>
```

მოთხოვნის სტრინგი

უმარტივესი მეთოდი რომელიც გამოიყენება ინფორმაციის გადასაგზავნად გვერდის მოთხოვნასთან ერთად არის URL მოთხოვნის სტრინგი
შევქმნათ welcome1.html შემდეგი შიგთავსით:
Hi, I'm George!
ის წარმოადგენს ლინქს welcome1.php-ზე რომლის შიგთავსი არის ვთქვათ ასეთი:

```
<?php
$name = $_GET['name'];
echo "Welcome to our Website, $name!";
?>
```

PHP ავტომატურად ქმნის ცვლადს (მასივს) $_GET რომელიც შეიცავს ნებისმიერ მნიშვნელობას რომელიც მიეწოდება მოთხოვნის სტრინგის მიერ. $_GET არის ასოციაციური მასივი,შესაბამისად, name ცვლადის მნიშვნელობაზე (რომელიც გადაეცემა მოთხოვნის სტრინგს მიერ) წვდომა ხდება $_GET['name']-ით.
ჩვენს სკრიპტში რა ხდება?
შესაძლებელია რამოდენიმე ცვლადის გადაცემა მოთხოვნის სტრინგის მეშვეობით:
<ahref=welcome2.php?firstname=George&lastname=Giorganashvili"> Hi,I'm George Giorganashvili!
როგორი უნდა იყოს შესაბამისი PHP ფაილი?

ფორმები

იმისათვის რომ მომხმარებელს თავად ქონდეს შესაძლებლობაინფორმაციის შეტანისა დაგვჭირდება ფორმები. განვიხილოთ მაგალითი index.htm:

```
<form action="welcome.php" method="get">
<label>First Name: <input type="text" name="firstname" />
</label><br />
<label>Last Name: <input type="text" name="lastname" />
```

```
</label><br />
<input type="submit" value="GO" />
</form>
```

URL -ს რომელიც გაიგზავნება სერვერზე ექნება ასეთი სახე:
http://www.test.com/index.htm?firstname=Giorgi&lastname=Gio rganashvili
როგორი უნდა იყოს welcome.php-სშიგთავსი?
ალბათ ასეთი არა?:)

```
<?php
$firstname = $_GET['firstname'];
$lastname = $_GET['lastname'];
echo "Welcome to our Website, $firstname $lastname!";
?>
```

GET და POST მეთოდები

არსებობს ორი საშუალება რომლითაც კლიენტის ბრაუზერს შეუძლია გაუგზავნოს ინფორმაცია სერვერს : ესენია GET და POST მეთოდები. პირველ მეთოდზე ჩვენ მეტნაკლებად ვისაუბრეთ ავღნიშნავთ მის ძირითად თავისებურებებს:

1. მეთოდს გააჩნია შეზღუდვა: შეუძია გააგზავნოს მხოლოდ 1024 სიმბოლო.
2. შეგიძლიათ გამოიყენოთ ეს მეთოდი ბინარული მონაცემების გადასაგზავნად, მაგალითად სურათი ან ვორდ დოკუმენტი.
3. არ გამოიყენოთ იგი პარილების ან სხვა საიდუმლო ინფორმაციის გადაგზავნისათვის. რადგან ინფორმაცია, გაგზავნილი ფორმიდან GET მეთოდით ჩანს ყველასათვის (ის გამოისახება ბრაუზერის მიმსამართების პანელზე)
4. მონაცემები რომლებიც იგზავნება ამ მეთოდის მეშვეობით მიიწვდომება QUERY_STRING ცვლადის მეშვეობით. შეიძლება მისი დაბუქმარკება
5. PHP გვაძლევს საშუალებას $_GETასოციაციური მასივის დახმარებით განვახორციელოთ წვდომა ინფორმაციაზე რომელიც იგზავნება GET მეთოდით.

POST მეთოდი

1. ეს მეთოდი ახდენს იმფორმაციის გადაცემას HTTP header-ების მეშვეობით, ხდება მისი კოდირება და GETმეთოდის მსგავსად თავსდება QUERY_STRING-ში.
2. მას არ გააჩნია არანაირი შეზღუდვ მონაცემების ზომასთან დაკავშირებით.
3. მეთოდი შეიძლება გამოიყენებოდეს როგორც ბინარული ისე ASCII მონაცემების გადასაგზავნად.
4. ინფორმაცია რომელიც იგზავნება მეთოდით POST მიდის HTTP header-ის მეშვეობით, ამდენად დაცულობის მომენტი დამოკიდებულია უკვე HTTP

პროტოკოლზე. დაცული HTTP-ის გამოყენება იქნება გარანტი გაგზავნილი ინფორმაციის დაცულობისა.
5. PHP გვაძლევს საშუალებას $_ POST ასოციაციური მასივის დახმარებით განვახორციელოთ წვდომა ინფორმაციაზე რომელიც იგზავნება POST მეთოდით. $_REQUEST გავარჩიოთ დამოუკიდებლად!

PHP სტრინგ-ფუნქცია „substr_replace“

ფუნქცია substr_replace გარკვეულ დამატებითფუნქციონალურობას გვთავაზობს str_replace-თან შედარებით. substr_replace მათემატიკურად უფრო გამართული ფუნქციაა რომელიც ეყრდნობა საქყის წერტილებს და სიგრძეებს რომ შეცვალოს სტრინგის ნაწილები განსხვავებით მეთოდისა „searching and replacing“.

არის 3 პარამეტრი რომელსაც მოითხოვს substr_replace ფუნქცია: (ორიგინალი სტრინგი, ჩასანაცვლებელი სტრინგი, საწყისი წერტილი და კიდევ ერთი არააუცილებელი სიგრძე
ორიგინალი სტრინგი- სტრინგი რომლის შეცვლაც გინდათ
ჩასანაცვლებელი სტრინგი- რითაც ცვლით დაწყებული საწყისი წერტილიდან დამტავრებული საბოლოო წერტილით (რასაც განსაზღვრავს „სიგრძე“)
საწყისი წერტილი - ორიგინალი სტრინგის ის ადგილი რომელიც მოინიშნება როგორც ჩანაცვლების დასაწყისი. უარყოფითი მნიშვნელობა გაიგება როგორც სიმბოლოების რაოდენობა დაწყებული სტრინგის ბოლოდან.
სიგრძე -განსაზღვრავს რამდენი სიმბოლო ჩანაცვლდეს ორიგინალი სტრინგიდან, 0 ნიშნავს რომ არცერთი არ ჩანაცვლდება დამოხდება insert. უარყოფითი მნიშვნელობა გაიგება როგორც სიმბოლოების რაოდენობა დაწყებული სტრინგის ბოლოდან.

განვიხილოთ მაგალითი :
PHP კოდი:

```
//შესაცვლელი სტრინგი
$original = "ABC123 Hello Mr. Cow! DEF321";

//საწყისი წერტილი 5
$sp5 = substr_replace($original, "Five", 5);
//საწყისი წერტილი 12
$sp12 = substr_replace($original, "Twelve", 12);
//საწყისი წერტილი 0
$sp0 = substr_replace($original, "Zero", 0);
//საწყისი წერტილი -1
$spneg1 = substr_replace($original, "Negative 1", -1);
```

```
//Echo-ები:
echo "Original String: $original <br />";
echo "Starting Point 5: $sp5 <br />";
echo "Starting Point 12: $sp12 <br />";
echo "Starting Point 0: $sp0 <br />";
echo "Starting Point -1: $spneg1 ";
```

გვექნება:

```
ორიგინალი სტრინგი: ABC123 Hello Mr. Cow! DEF321
საწყისი წერტილი 5: ABC12Five
საწყისი წერტილი 12: ABC123 HelloTwelve
საწყისი წერტილი 0: Zero
საწყისი წერტილი -1: ABC123 Hello Mr. Cow! DEF32Negative 1
```

როგორც ვხედავთ როდესაც არ ვუთითებთ მეოთზე პარამეტრს (სიგრძეს), ყველაფერი დაწყებული საწყისი წერტილიდან იცვლება მეორე პარამეტრით (ჩასანაცვლებელი სტრინგით).

შევნიშნოთ რომ ათვლა იწყება 0-დან.
ვთქვათ გვინდა მოვიშოროთ ABC123 და DEF321 სტრინგის დასაწყისუდან და ბოლოდან, რადგანაც ორივეს სიგრძე არის 6 და ვიცით რომ ერთი მდებარეობს სულ თავში და მეორე კი სულ ბოლოში შეგვიძლია ABC123-ისათვის გამოვიყენოთ საწყისი წერტილი 0 , ხოლო DEF321-ისთვის -6. ჩასანაცვლებელი სტრინგი კი ცარიელი „“. ანუ გვექნება:
PHP კოდი:

```
$original = "ABC123 Hello Mr. Cow! DEF321";
//წავშალოთ ABC123 და წავიღოთ $cleanedstr-ში
$cleanedstr = substr_replace($original, "", 0, 6);
//ახლა კი წავშალოთ DEF321 $cleanedstr-დან
$cleanedstr2 = substr_replace($cleanedstr, "", -6, 6);

//Echo
echo "Original String: $original <br />";
echo "Clean #1: $cleanedstr <br />";
echo "Clean #2: $cleanedstr2";
```

გვექნება:

```
Original String: ABC123 Hello Mr. Cow! DEF321
Clean #1: Hello Mr. Cow! DEF321
Clean #2: Hello Mr. Cow!
```

თუ კი სიგრძის განმსაძღვრელ პარამეტრს გავხდით 0-ს, მაშინ substr_replace ფუნქცია არაფერსაც არ შეცვლის ორიგინალ სტრინგში და განახორციელებს მიზმას:

PHP კოდი:
```
$original = "Hello Mr. Cow!";

// დავადგინოთ Mr. Cow-ს პოზიცია
$cowpos = strpos($original, "Mr. Cow");

// ვნახოთ სად მთავრდება Mr. Cow ამისათვის დავამატოთ Mr. Cow-ს სიგრძე
$cowpos_end = $cowpos + strlen("Mr. Cow");

// Mr. Cow-ს შემდეგ ჩავსვათ Mrs. Bear
$mrsbear = substr_replace($original, " and Mrs. Bear", $cowpos_end, 0);

// Mr. Cow-მდე ჩავსვათ Sensei Shark
$senseishark = substr_replace($mrsbear, "Sensei Shark, ", $cowpos, 0);


//Echo
echo "Original String: $original <br />";
echo "After Mrs. Bear: $mrsbear <br />";
echo "After Sensei Shark: $senseishark";
```
გვექნება:
```
Original String: Hello Mr. Cow!
After Mrs. Bear: Hello Mr. Cow and Mrs. Bear!
After Sensei Shark: Hello Sensei Shark, Mr. Cow and Mrs. Bear!
```

PHP - სტრინგის კაპიტალიზაციის (ზედა/ქვედა რეგისტრში გადაყვანის) ფუნქცია

PHP-ში არსებობს სამი პირდაპირი კაპიტალიზაციის ფუნქცია: strtoupper, strtolower და ucwords. განვიხილოთ თითოეული:

სტრინგის ზედა რეგისტრში გადაყვანის ფუნქცია strtoupper

მას გააჩნია ერთი არგუმენტი - სტრინგი, რომლის გადაყვანაც გინდათ და აბრუნებს გადაყვანილ სტრინგს. იცვლება მხოლოდ ასოები რიცხვები რჩება იგივე.

PHP კოდი:
```
$originalString = "String Capitalization 1234";
```

```
$upperCase = strtoupper($originalString);
echo "Old string - $originalString <br />";
echo "New String - $upperCase";
```
გვექნება:
Old string - String Capitalization 1234
New String - STRING CAPITALIZATION 1234

ისმის კითხვა რაში შეიძლება დაგვჭირდეს ეს ფუნქცია?

სტრინგის ქვედა რეგისტრში გადაყვანის ფუნქცია strtolower

ისიც იყენებს ერთ არგუმენტს:

PHP კოდი:
```
$originalString = "String Capitalization 1234";

$lowerCase = strtolower($originalString);
echo "Old string - $originalString <br />";
echo "New String - $lowerCase";
```
გვექნება:
Old string - String Capitalization 1234
New String - string capitalization 1234

პირველი აანბანის გადაყვანა ზედა რეგისტრში - ucwords

ინგლისურ ენაში სათაურში ხშირად გვხვდება ზედა რეგისტრით დაწერილი ყველა სიტყვის პირველი ანბანი ან ასე გვინდა რომ დავწეროთ ამისათვის გამოიყენება ფუნქცია ucwords-ს

PHP კოდი:
```
$titleString = "a title that could use some hELP";

$ucTitleString = ucwords($titleString);
echo "Old title - $titleString <br />";
echo "New title - $ucTitleString";
```
გვექნება:
Old title - a title that could use some hELP
New title - A Title That Could Use Some HELP

PHP – სტრინგის დახლეჩვა (String Explode)

PHP -ში ფუნქცია explode საშუალებას იძლევა სტრინგი გაიყოს პატარა ნაწილებად, მაგალითად გვინდა წინადადების დაყოფა სიტყვებად, ამისათვის კარგი ინსტრუმენტია explode ფუნქცია.

explode ფუნქცია

მას გააჩნია ორი არგუმენტი: პირველი არის დელიმიტერი (ასაფეთქებელი☺-ანუ სიმბოლო რომლის მიხედვითაც უნდა დაიშალოს მეორე არგუმენტი-სტრინგი). განვიხილოთ მაგალითი

PHP კოდი:

```
$rawPhoneNumber = "800-555-5555";

$phoneChunks = explode("-", $rawPhoneNumber);
echo "Raw Phone Number = $rawPhoneNumber <br />";
echo "First chunk = $phoneChunks[0]<br />";
echo "Second chunk = $phoneChunks[1]<br />";
echo "Third Chunk chunk = $phoneChunks[2]";
```

გვექნება:

```
Raw Phone Number = 800-555-5555
First chunk = 800
Second chunk = 555
Third Chunk chunk = 5555
```

explode ფუნქცია- ზღვარის დადგენა

თუ გვინდა დავუყენოთ დესტრუქციის მარჯა , ამისათვის შესაძლებელია მივუთითოთ დასახლეჩი ნაწილების რაოდენობა.

PHP კოდი:

```
$someWords = "Please don't blow me to pieces.";

$wordChunks = explode(" ", $someWords);
for($i = 0; $i < count($wordChunks); $i++){
        echo "Piece $i = $wordChunks[$i] <br />";
}

$wordChunksLimited = explode(" ", $someWords, 4);
for($i = 0; $i < count($wordChunksLimited); $i++){
        echo "Limited Piece $i = $wordChunksLimited[$i] <br />";
}
```

გვექნება:
Piece 0 = Please
Piece 1 = don't
Piece 2 = blow
Piece 3 = me
Piece 4 = to
Piece 5 = pieces.
Limited Piece 0 = Please
Limited Piece 1 = don't
Limited Piece 2 = blow
Limited Piece 3 = me to pieces.

PHP - Array implode

PHP ფუნქცია implode ოპერირებს მასივზე და ცნობილია როგორც "undo" ფუნქციისა explode. თუ გამოიყენეთ explode რომ დაგეხლიჩათ სტრინგი ნაწილებად ან უბრალოდ გაქვთ მონაცემების მასივი და გინდათ მისი ერთ სტრინგად გაკეთება მარტივი გზა არის ფუნქცია implode .

PHP implode – „ზარალის აღდგენა“

Implode-ის პირველი არგუმენტი არის სიმბოლოების სტრინგი რომელის გამოყენებაც გიდათ მასივის ნაწილების „შესაწებებლად“, მეორე კი თავად მასივი.
PHP კოდი:

```
$pieces = array("Hello", "World,", "I", "am", "Here!");

$gluedTogetherSpaces = implode(" ", $pieces);
$gluedTogetherDashes = implode("-", $pieces);
for($i = 0; $i < count($pieces); $i++){
        echo "Piece #$i = $pieces[$i] <br />";
}
echo "Glued with Spaces = $gluedTogetherSpaces <br />";
echo "Glued with Dashes = $gluedTogetherDashes";
```

გვექნება:
Piece #0 = Hello
Piece #1 = World,
Piece #2 = I
Piece #3 = am
Piece #4 = Here!
Glued with Spaces = Hello World, I am Here!

Glued with Dashes = Hello-World,-I-am-Here!

implode ფუნქცია მოახდენს მთლიანი მასივის სტრინგში გადაყვანას, მას არ გააჩნია სხვა არააუცილებელი არგუმენტი რომ განესაზღვროს ლიმიტი, როგორც ეს ქონდა explode ფუნქციას.

include() და require() ფუნქციები

ჩვენ შეგვიძლია რამოდენიმე ფაილის შემცველობა ჩავსვათ PHP ფაილში include() ან require() ფუნქციებით. ეს ორი ფუნქცია იდენტურია, მხოლოდ მათ გააჩნიათ განსხვავებული შეცდომების იდენტიფიკატორები. include() ფუნქცია გამოსახავს გაფრთხილებას (მაგრამს სკრიპტი გააგრძელებს მუშაობას) მაშინ როცა 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 menu.php მენიუს ფაილი შემდეგი შიგთავსით, რომელიც უნდა გამოვიყენოთ ყველა გვერდზე:

```
<a href="/default.php">Home</a>
<a href="/tutorials.php">Tutorials</a>
<a href="/references.php">References</a>
<a href="/examples.php">Examples</a>
<a href="/about.php">About Us</a>
<a href="/contact.php">Contact Us</a>
```

ვებ საიტის ყველა გვერდი უნდა შეიცავდეს ამ შიგთავსს, ამას კი მოვახერხებთ შემდეგნაირად:

```
<html>
<body>

<div class="leftmenu">
<?php include("menu.php"); ?>
</div>

<h1>Welcome to my home page.</h1>
<p>Some text.</p>

</body>
</html>
```

ბრაუზერში ამ გვერდის „source“--ს თუ შევხედავთ გვექნება:

```
<html>
<body>

<div class="leftmenu">
<a href="/default.php">Home</a>
<a href="/tutorials.php">Tutorials</a>
<a href="/references.php">References</a>
<a href="/examples.php">Examples</a>
<a href="/about.php">About Us</a>
<a href="/contact.php">Contact Us</a>
</div>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>

</body>
</html>
```

include() ფუნქციაც იდენტურად მუშაობს ოღონდ როგორც ზევით ავღნიშნეთ შეცდომის შემთხვევაში „სხვსნსირად“ იქცევა :
ვთქვათ გვაქვს შეცდომით მითითებული include() ფუნქციის შიგთავსი:

```
<html>
<body>

<?php
include("wrongFile.php");
echo "Hello World!";
?>

</body>
</html>
```

შეცდომის შეტყობინებას ექნება სახე:

```
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!
```

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

მსგავსი მაგალითი require() ფუნქციით კი გამოგვიტანს :

```
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
```

მაგდა ცინცაძე

ზოგადად რეკომენდირებულია require() ფუნქციის გამოყენება, რადგან წესით შეცდომის შემთხვევში სცრიპტი აგარ უნდა აგრძელებდეს მუშაობას.

ფაილებთან მუშაობა PHP-ში

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 უნდა გაიხსნას იგი:

```
<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.

მაგალითი:

```
 <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);
?>
```

ფაილის დასასრულის (END-OF-FILE)-ის შემოწმება

feof() ფუნქცია ამოწმებს, მიღწეულია თუ არა("end-of-file" (EOF)) ფაილის ბოლო.: **if** (feof($file)) echo "End of file";
feof() ფუნქცია გამოსადეგია უცნობი სიგრძის მონაცემთა ციკლირებისათვის.
შევნიშნოთ რომ ჩვენ ვერ წავიკითხავთ ფაილიდან, რომელიც გახსნილია w, a, და x რეჟიმებში.

ფაილის კითხვა PHP - File Read

სანამ შეგვეძლება ინფორმაციის ფაილიდან რაკითხვა, ის უნდა ჯერ გავხსნათ fopen ფუნქციის მეშვეობით:

```
$myFile = "testFile.txt";
$fh = fopen($myFile, 'r');
```

შევნიშნოთ რიმ სკრიპტი უნდა მდებარეობდეს იგივე დირექტორიაში სადაც "testFile.txt" (წინააღმდეგ შემთხვევაში უნდა მოვუთითოთ გზა ფაილამდე).
იმის შემდგომ რაც ფაილი გავხსენით წაკითხვის უფლებით შესაძლებელია fread ფუნქციის გამოყენება.

fread ფუნქციას უნდა გადავცეთ 2 პარამეტრი: ფაილის დამუშავების შესაბამისი და ინტეჯერი რომ ვუთხრათ მას რამდენი byte მონაცემის წაკითხვა გვინდა.
ერთი სიმბოლო ერთი byte-ს შესაბამისია. შესაბამისად თუ გვინდა პირველი 5 სიმბოლოს წაკითხვა შესაძლებელია მიუთითთ 5-იანი:

```
$myFile = "testFile.txt";
$fh = fopen($myFile, 'r');
$theData = fread($fh, 5);
fclose($fh);
echo $theData;
```

გამოიტანს: Flopp , რადგან testFile.txt-ის შიგთავსია: „Floppy Jalopy Pointy Pinto“.

პირველი 5 სიმბოლო ფაილიდან testFile.txt იქნება მოთავსებული ცვლადში $theData. შესაბამისად შესაძლებელია მისი გამოტანა echo $theData -თი, ან ჩავწეროთ ეს სტრინგი სხვა ფაილში.

იმისათვის რომ ფაილიდან მოვახდინოთ ყველა მონაცემის წაკითხვა, დაგვჭირდება ფაილის ზომის გაგება. ფუნქცია filesize აბრუნებს ფაილის სიგრძეს byte-ში. მას პარამეტრად უნდა გადაეცეს იმ ფაილის სახელი რომელსაც „ვზომავთ“:
PHP კოდს ექნება სახე:

```
$myFile = "testFile.txt";
$fh = fopen($myFile, 'r');
$theData = fread($fh, filesize($myFile));
fclose($fh);
echo $theData;
```

ახლა მთელი შიგთავსი testFile.txt ფაილისა მოთავსებულია $theData სტრინგ-ცვლადში.

ფაილში თითო-თითო ხაზის წაკითხვა

fgets() ფუნქცია გამოიყენება ფაილიდან ერთი ხაზის წაკითხვისათვის. 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
$file = fopen("welcome.txt", "r") or exit("Unable to open file!");
while(!feof($file))
  {  echo fgets($file). "<br />";  }
fclose($file);
```

```
?>
```

ფაილში თითო-თითო სიმბოლოს წაკითხვა

fgetc() ფუნქცია გამოიყენება ფაილიდან თითო-თითო სიმბოლოს წასაკითხად. 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
$file=fopen("welcome.txt","r") or exit("Unable to open file!");
while (!feof($file))
  {
  echo fgetc($file);
  }
fclose($file);
?>
```

ფაილში ჩაწერა PHP - File Write

ახლა როცა ვიცით ფაილის გახსნა დახურვა გადავიდეთ ფაილის მანიპულაციის ყველაზე უფრო გამოყენებით ნაწილზე: ჩაწერაზე! ისევე როგორც კითხვის შემთხვევაში ჩაწერის დროსაც ჯერ ფაილი უნდა გავხსნათ (ოღონდ ამჯერად ჩაწერის რეჟიმში):
PHP კოდი:

```
$myFile = "testFile.txt";
$fh = fopen($myFile, 'w');
```

ფუნქცია fwrite საშუალებას გვაძლევს მონაცემები ჩავწეროთ ნებისმიერი ტიპის ფაილში. Fwrite ფუნქციის პირველი პარამეტრი არის ფაილის დამუშავების მეორე კი სტრინგი რომლის ჩაწერაც გვინდა.
განვიხილოთ მაგალითი:

```
$myFile = "testFile.txt";
$fh = fopen($myFile, 'w') or die("can't open file");
$stringData = "Bobby Bopper\n";
fwrite($fh, $stringData);
$stringData = "Tracy Tanner\n";
fwrite($fh, $stringData);
fclose($fh);
```

ცვლადი $fh შეიცავს ფაილის დამუშავების შესახებ ინფორმაციას testFile.txt-ისათვის.

თუ გავხსნით testFile.txt ფაილს NOTEPAD-ით მისი შიგთავსი იქნება:

Bobby Bopper
Tracy Tanner

შევნიშნოთ რომ თუ იგივე ფაილში მოვინდომებთ ზემოღწერილი მეთოდით ახალი მონაცემების ჩაწერას, უკვე ჩაწერილი ინფორმაცია წაიშლება რადგანაც PHP File Write “W” mode-ში არის: Overwriting.

ფაილის ცვლილება PHP - File Append

თუ გვინდა ფაილის შიგთავსის ცვლილება ისე რომ არ მოხდეს მაშში არსებული ინფორმაციის დაკარგვა ანუ ისე ჩავამატოთ მონაცემები რომ წინაც დაგვრჩეს უნდა გამოვიყენოთ ფაილის გახსნა append mode-ში.
მაგალითი:

```
$myFile = "testFile.txt";
$fh = fopen($myFile, 'a') or die("can't open file");
$stringData = "New Stuff 1\n";
fwrite($fh, $stringData);
$stringData = "New Stuff 2\n";
fwrite($fh, $stringData);
fclose($fh);
```

ანუ ვიქცევით იგივენაირად როგორც ჩაწერის დროს იმ განსხვავებით რომ ფაილი ამ დროს გახსნილი გვაქვს არა ჩაწერის (w=write) არამედ ცვლილების (a=append) რეჯიმში(mode).

ფაილის წაშლა PHP - File Delete

დადგა დრო შევეხოთ ფაილის წაშლის საშუალებას. PHP -ში ამის განხორციელება ხდება unlink ფუნქციის გამოყენებით. როდესაც ათვალიერებთ დირექტორიის შიგთავსს, ხედავთ სიას იმ ფაილებისა რომლებიც მასში არიან მოთავსებულნი. შესაძლებელია ფაილების სახელები წარმოვიდგინოთ როგორც ლინქები იმ დირექტორიაზე რომელსაც ათვალიერებთ, შესაბამისად თუ მოახდენთ განლინქვას(Unlink), ამით სისტემას ავიწყებთ ამ ფაილს ანუ შლით მას.
სანამ მოახდენდეთ ფაილის განლინქვას (Unlink) უნდა დარწმუნდეთ რომ ის არ არის გახსნილი და თუ გახსნილია დახურეთ fclose ფუნქციის გამოყენებით.
წაშლის კოდი საკმაოდ მარტივია, unlink ფუნქციას ჭირდება მხოლოდ ნამსალინქი ფაილის სახელი:

```
 $myFile = "testFile.txt";
unlink($myFile);
```

დავალება: გავარჩიოთ PHP - String Explode და PHP - Array implode

მაგდა ცინცაძე

ფაილის ატვირთვა PHP - File Upload

საკმაოდ გამოსადეგარი თვისება PHP-ში არის სერვერზე ფაილების ატვირთვის შესაძლებლობა, თუმცა ეს არც თუ უსაფრთხო საქმეა, საჭიროა სიფრთხილე რომ ჩვენს სერვერზე არ მოხვდეს არასასურველი ფაილები.
სანამ დაიწყებთ PHP-ის გამოყენებას ატვირთვისათვის დაგვჭირდება HTML ფორმა რომელიც საშუალებას მისცემს მომხმარებელს ამოარჩიოს ასატვირთი ფაილი.
HTML კოდი:

```
<form enctype="multipart/form-data" action="uploader.php" method="POST">
<input type="hidden" name="MAX_FILE_SIZE" value="100000" />
Choose a file to upload: <input name="uploadedfile" type="file" /><br />
<input type="submit" value="Upload File" />
</form>
```

განვიხილოთ ფორმის ატრიბუტები დაწვრილებით:

- enctype="multipart/form-data" - აუცილებელია ჩვენი „შესაქმნელი“ PHP ფაილისათვის რომ იფუნქციონიროს გამართულად. 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.

- input type="hidden" name="MA... - განსაზღვრავს ფაილის მაქსიმალურ დასაშვებ ზომას ბაიტებში

- input name="uploadedfile" - uploadedfile არის ის სახელი რომლითაც PHP სკრიპტში განვახორციელებთ ფაილთან წვდომას.

დავიმახსოვროთ ფორმა upload.html-ში. როცა მომხმარებელი დააჭერს ღილაკს submit, მონაცემები გადაიგზავნება სერვერზე და მომხმარებელი გადამისამართდება uploader.php-თან. PHP ფაილმა უნდა გადაწყვიტოს მიიღოს ასატვირთი ფაილი თუ „გადააგდოს“ ის. „გადაგდების“ მიზეზი შეიძლება იყოს მრავალი : მაგალითად ასატვირთი ფაილის ზომა არის დიდი, არის გამშვები exe, და ასე შემდეგ... ან შეიქმნა სხვა პრობლემები ატვირთვის დროს

როდესაც მოხდება uploader.php ფაილის შესრულება, ასატვირთი ფაილი მოთავსებული იქნება დროებითი შენახვის ზონაში სერვერზე. შესაბამისად თუ არ მოხდება მისი სხვა მისამართზე გადატანა ის განადგურდება. თუ გვინდა ამ ფაილის შენახვა მაშინ უნდა გამოვიყენოთ $_FILES ასოციაციური მასივი.

- $_FILES მასივი არის ის ადგილი სადაც PHP ჩატვირთავს ფაილის შესახებ ყველა ინფორმაციას.
- $_FILES['uploadedfile']['name'] - name შეიცავს მისამართს მომხმარებლის ასატვირთ ფაილამდე
- $_FILES['uploadedfile']['tmp_name'] - tmp_name შეიცავს დროებითი ფაილის მისამართს რომელიც მდებარეობს სერვერზე (ფაილი უნდა არსებობდეს სერვერზე დროებით დირექტორიაში დროებითი სახელით)
- $_FILES['uploadedfile']['type'] - ასატვირთი ფაილის ტიპი
- $_FILES['uploadedfile']["size"] - ასატვირთი ფაილის ზომა ბაიტებში
- $_FILES['uploadedfile']["error"] - შეცდომის კოდი

ახლა უკვე შეგვიძლია შევუდგეთ PHP upload სკრიპტის წერას შევუდგეთ. ქვემოთ მოცემულია თუ როგორაა შესაძლებელი დროებით ფაილზე წვდომა და მისი გადატანა შესანახ დირექტორიაში.

PHP კოდი:

```
// ადგილი (დირექტორია) სადაც ფაილს მოვათავსებთ
$target_path = "uploads/";

/* დავამატოთ ფაილის სახელი მისამართს,გვექნება "uploads/filename.extension" */

$target_path = $target_path . basename( $_FILES['uploadedfile']['name']);

/* კომენტარი: ფუნქცია basename-ის მუშაობის მაგალითი
*)
<?php
$path = "/home/httpd/html/index.php";
$file = basename($path);         // $file მიიღებს მნიშვნელობას "index.php"
$file = basename($path, ".php"); // $file მიიღებს მნიშვნელობას "index"
?>
*/
```

შენიშვნა: დაგვჭირდება ახალი დირექტორიის შექმნა ("uploads" სახელწოდებით სადაც უნდა იყოს ატვირთული ფაილები) იმ დირექტორიაში სადაც მოთავსებულია ჩვენი uploader.php ფაილი.

ახლა მზად ვართ დავიმახსოვროთ ფაილი სერვერზე. $target_path შეიცავს მისამართს სადაც გვინდა ასატვირთი ფაილის დამახსოვრება.
ახლა კი დაგვჭირდება move_uploaded_file ფუნქციის გამოყენება.
move_uploaded_file ფუნქციას უნდა გადავცეთ 2 პარამეტრი: 1) დროებით ფაილამდე მისამართი და 2) მისამართი ფაილი სადაც უნდა გადავიტანოთ.

გვექნება:

```
<?php
$target_path = "uploads/";
$target_path = $target_path . basename( $_FILES['uploadedfile']['name']);

if(move_uploaded_file($_FILES['uploadedfile']['tmp_name'], $target_path))
 {
    echo "The file ".  basename( $_FILES['uploadedfile']['name']).
    " has been uploaded";
 }
 Else
{
    echo "There was an error uploading the file, please try again!";
}
?>
```

ეს არის ფაილების ატვირთვის ძალიან მარტივი გზა. დაცვისათვის, დავაწესოთ შეზღუდვები ასატვირთი ფაილების გაფართოებებზე. სკრიპტში ჩვენ ჩავამატებთ ფაილის ატვირთვის ზოგიერთ შეზღუდვებს. მომხმარებელს შეეძლება ატვირთოს მხოლოდ .gif, ან .jpeg ფაილები და ფაილის ზომა არ უნდა აღემატებოდეს 20 kb-ს:

```
if (($_FILES['uploadedfile']['type'] == "image/gif")
|| ($_FILES['uploadedfile']['type'] == "image/jpeg")
|| ($_FILES['uploadedfile']['type'] == "image/pjpeg")
&& ($_FILES['uploadedfile'][ 'size'] < 20000))
```

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

ფაილის ატვირთვის ფორმის შექმნა

ქვემოთ მოყვანილია ფაილების ატვირთვის HTML ფორმა:

```
<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

"uploader.php" შეიცავს ატვირთვის კოდს:

```
<?php
```

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

```
<?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

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

სესიები PHP -ში

სესიები და cookie-ები განსაზღვრულია მომხმარებლის შესახებ ინფორმაციის შენახვისათვის როდესაც ისინი გადადიან ერთი გვერდიდან მეორეზე.

სესიების გამოყენების დროს მონაცემები ინახება სერვერზე დროებით ფაილებში. სესია ქმნის უნიკალურ სახელს (UID) თითოეული ვიზიტორისათვის და აგროვებს ცვლადების ბაზას UID-ზე.
ხოლო cookies-ის ფაილები ინახება მომხმარებლის კომპიუტერზე და მოთხოვნიე შესაბამისად ბრაუზერის მიერ ეგზავნება სერვერს.

სესიებისა და cookies-ის გამოყენება საკმაოდ მოსახერხებელი და გამოყენებადია ფორუმებისათვის, ინტერნეტ-მაღაზიებისათვის როდესაც აუცილებელია მომხმარებლის შესახებ ინფორმაციის შენახვა და მეორეც საჭიროა მას დროულად მიეწოდოს ახალი ინფორმაცია.

HTTP პროტოკოლი წარმოადგენს პროტოკოლს რომელსაც არ შეუძლია „მდგომარეობის დამახსოვრება“, რაც იმას ნიშნავს რომ მას არ აქვს ტრანზაქციებს შორის მდგომარეობების დამახსოვრების შესაძლებლობა, ანუ როდესაც მომხმარებელი ხსნის საიტის ჯერ ერთ გვერდს და მერე გადადის მის მეორე გვერდზე, HTTP-ის ინფორმაციაზე დაყრდნობით შეუძლებელია იმის დადგენა ორივე მოთხოვნა ეკუთვნის თუ არა ერთიდაიგივე მომხმარებელს. შესაბამისად აუცილებელი ხდება ისეთი მეთოდის გამოყენება რომელიც საშუალებას მოგვცემს საიტზე მომხმარებლის ყოფნის მთლიანი სეანსის შესახებ ინფორმაციისათვის თვალის დევნების საშუალებას. ერთ-ერთ ასეთ მეთოდს წარმოადგენს სეანსების მარვა შესავამისი ფუნქციების მეშვეობით . ჩვენთვის მნიშვნელოვანია ის რომ სეანსი შინაარსობრივად წარმოადგენს ცვლადების ჯგუფს, რომლებიც ჩვეულებრივი ცვლადებისაგან განსხვავებით ინახებიან მას შემდეგაც რაც შესრულდება PHP სცენარი.

სესიებთან მუშაობისას გვაქვს შემდეგი ეტაპები:

სესიის გახსნა
სესიის ცვლადების რეგისტრირება და მათი გამოყენება
სესიის დახურვა

სესიის გახსნა

სესიის გახსნის ყველაზე მარტივი მეთოდია ფუნქცია session_start -ის გამოყენება, რომელიც გამოიძახება PHP სცენარის დასაწყისში:

სინტაქსი:

```
session_start();
```

ეს ფუნქცია ამოწმებს არსებობს თუ არა სესიის იდენტიფიკატორი და თუ არ არსებობს ქმნის მას, ხოლო თუ არსებობს მოახდენს ამ სესიის დარეგისტირებულ ცვლადების ჩატვირთვას.

სესიის ცვლადების რეგისტრაცია

სესიის ინიციალიზაციის შემდეგ შესაძლებელია ინფორმაცია შევინახოთ სუპერგლობალური $_SESSION მასივში. ვთქვათ გვაქვს index.php ფაილი, სადაც $_SESSION მასივში ხდება ცვლადისა და მასივის შენახვა:

```
<?php
  // სესიის ინიციალიზაცია(დაწყება)
  session_start();
  // ვათავსებთ მნიშვნელობას სესიაში
  $_SESSION['name'] = "value";
  // ვათავსებთ მასივს სესიაში
  $arr = array("first", "second", "third");
  $_SESSION['arr'] = $arr;
  // გამოგვაქვს ლინქი სხვა გვერდზე
  echo "<a href='other.php'>other site</a>";
?>
```

გვერდებზე სადაც ხორციელდება session_start() ფუნქციის გამოძახება, მოცემული ცვლადების მნიშვნელობის ამოღება შესაძლებელია სუპერგლობალური $_SESSION მასივიდან. ქვემოთ მოყვანილია მაგალითი სადაც ხდება იმ მნიშვნელობების ამოღება რომლებიც განთავსდა გვერდზე index.php.

```
<?php
  // სესიის ინიციალიზაცია(დაწყება)
  session_start();
  // გამოგვაქვს სუპერგლობალური $_SESSION მასივის შიგთავსი
  echo "<pre>";
  print_r($_SESSION);
  echo "</pre>";
?>
```

სკრიპტის მუშაობის შედეგი ასე გამოიყურება:

```
Array
(
    [name] => value
    [arr] => Array
        (
```

```
        [0] => first
        [1] => second
        [2] => third
    )
)
```

სესიის დახურვა

სესიასთან მუშაობის დასრულების შემდეგ უნდა მოხდეს სესიის ყველა ცვლადების განრეგისტრირება , სინტაქსი ასეთია:

```
unset($_SESSION["username"]);
```

განვიხილოთ მარტივი სესიის მაგალითი რომელიც მუშაობს სამ გვერდთან. მომხმარებლის მიერ პირველი გვერდის გახსნისას იხსნება სესია და რეგისტრირდება ცვლადი $username. კოდს აქვს შემდეგი სახე:

```
<?

 session_start();
 $_SESSION['username'] = "vinme";
 echo 'hi, '.$_SESSION['username']."<br>";

?>

 <a href="page2.php"> next page </a>
```

ამის შემდეგ მომხმარებელი vinme აჭერს რა ბმულს ხვდება გვერდზე page2.php, რომლის კოდიც მოყვანილია ქვევით:

```
<?
 session_start();
 echo $_SESSION['username'].' , you've visited the next page!';
 echo("<br>");

?>
 <a href="page3.php"> next page </a>
```

ბმულზე დაჭერისას მომხმარებელი გადადის გვერდზე page3.php, ამ დროს ხდება განრეგისტრირება სეანსის ცვლადისა და სესიის განადგურება:

```
<?
 session_start();
 unset($_SESSION['username']); //სეანსის ცვლადის განრეგისტრირება
 echo 'hi, '.$_SESSION['username'];
```

```
  /* ამ შემთხვევაში მომხმარებლის სახელი აღარ გამოიტანება */
  session_destroy(); // სესიის განადგურება
?>
```

isset

ფუნქცია isset — ეს ფუნქცია განსაზღვრავს განსაზღვრულია თუ არა ცვლადი (მინიჭებული აქვს თუ არა არაცარიელი მნიშვნელობა)

სინტაქსი:
bool isset (mixed var [, mixed var [, ...]])

აბრუნებს TRUE, თუ var არსებობს და FALSE -ს წინააღმდეგ შემთხვევაში.

თუ ცვლადი გამოთავისუფლდა (unset ანუ) მაშინ ფუნქცია unset() isset() დააბრუნებს FALSE. ასევე მოიქცევა ის თუ ცვლადი არის NULL.

კოდის მაგალითი

```
<?php

$var = '';

// დააბრუნებს TRUE
if (isset($var))
{
   echo "This var is set so I will print.";
}

$a = "test";
$b = "anothertest";

var_dump(isset($a));     // TRUE
var_dump(isset($a, $b)); // TRUE

unset ($a);

var_dump(isset($a));     // FALSE
var_dump(isset($a, $b)); // FALSE

$foo = NULL;
var_dump(isset($foo));   // FALSE
?>
```

ქვემოთ მოყვანილ მაგალითში, შევქმნით უბრალო მთვლელს. isset() ფუნქცია შეამოწმებს "views" ცვლადი უკვე განსაზღვრულია თუ არა. თუ "views" უკვე გააჩნია მნიშვნელობა, 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 "SiteViews=". $_SESSION['views'];
?>
```

-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --- -- -- -- -- -- - -- -- -- -- -- -- -- -- -- -- -- --- --

cookies გამოყენება მოსახერხებელია როგორც პროგრამისტებისათვის ისე მომხმარებლებისათვის. მომხმარებელი იგებს იმაში რომ მას არ ჭირდება ყოველ ჯერზე თავის შესახებ ინფორმაციის შეყვანა, პროგრამისტები კი მას იყენებენ მომხმარებლის შესახებ ინორმაციის შესანახად.

Cookies განმარტება - ის წარმოადგენს ტექსტურ სტრიქონებს რომლებიც ინახება კლიენტის მხარეს და შეიცავს წყვილებს „სახელი-მნიშვნელობა“- რომელიც დაკავშირებულია URL რომლის მეშვეობითაც ბრაუზერი არკვევს საჭიროა თუ არა cookies-ს გადაგზავნა სერვერზე.
Cookies დაყენება ხორციელდება setcookie ფუნქციის მეშვეობით:

სინტაქსი

```
  bool setcookie (string name [, string value [, int expire [, string path
  [, string domain [, int secure]]]]])
```

ამ ფუნქციას გააჩნია შემდეგი არგუმენტები:
name – cookie სახელი;
value - მნიშვნელობა რომელიც ინახება cookie-ში სახელით $name;
expire - დრო წამებში რომლის გასვლის შემდეგაც მიმდინარე cookie ხდება ბათილი;
path - გზა cookie-ს წვდომისათვის;
domain - დომენი რომლიდზეც მდეანაც ხორციელდება cookie-ზე წვდომადомен, из которого доступен cookie;

secure - დირექტივა რომელიც განსაზღვრავს შესაძლებელია თუ არა cookie-ზე წვდომა არა HTTP მოთხოვნით. გაჩუმებით ამ დირექტივას აქვს მნიშვნელობა 0, რაც იმას ნიშნავს რომ cookie-ს გამოყენება შესაძლებელია ჩვეულებრივი HTTP მოთხოვნით.

cookies მაგალითი:

განვიხილოთ მარტივი cookies მაგალითი, რომელიც ითვლის კონკრეტული მომხმარებელი მერამდენედ შემოვიდა საიტზე:

```
<?
  $counter++;
  setcookie("counter",$counter);
  echo("You've visited this site $counter times");
?>
```

Cookie-ებთან მუშაობისას უნდა გავითვალისწინოთ ერთი მნიშვნელოვანი მომენტი, კერძოდ ის რომ ის უდა დავაყენოთ მანამ სანამ ბრაუზერს გადაეცემა რაიმე სათაური (content-type:text/html მაგალითად), რადგანაც თვითონ Cookie-ები ყენდება სათაურის სახით.
შენიშვნა: მაგალითში ჩვენ ცვლადს $counter-ს რომელშიც ინახება Cookie, მივმართეთ როგორც გლობალურს, ამისათვის კი register_globals უნდა იყოს დაყენებული ON-ზე. წინააღმდეგ შემთხვევაში უნდა გამოვიყენოთ გლობალური მასივი $_COOKIE["name"] ასე:

```
<?
  $_COOKIE['counter']++;
  setcookie("counter",$_COOKIE['counter']);
  echo 'You've visited this site '.$_COOKIE['counter'].' times ';
?>
```

აგრეთვე შევნიშნოთ კიდევ ერთი გარემოება: ზოგიერთ მომხმარებელს გამორთული აქვს ხოლმე cookie-ები ბრაუზერებისათვის, ამიტომ კორექტულობისათვის უნდა მოვახდინოთ ხოლმე მომხმარებელს ჩართული აქვს თუ არა cookie და თუ არ აქვს ვუთხრათ რომ კოდის მუშაობისათვის აუცილებელია ჩაურთოს cookie.

```
<?
  if(!$cookie)
  {
    /* ვაგზავნით გადამისამართების  სათაურს იმ გვერდზე რომელზეც მოხდება
cookie -ს ჩართვის მცდელობა */
    header("Location: $PHP_SELF?cookie=1");
    /* ვაყენებთ cookie-ს სახელწოდებით "test" */
    setcookie("test","1");
```

```
  }
  else
  {
   if(!$test)
   {
    echo("კოდის კორექტული მუშაობისათვის აუცილებელია ჩართული
გქონდეთ cookie");
   }
   else
   {
    /* cookie ჩართულია გადავდივართ საჭირო გვერდზე */
    header("Location: http://localhost/test1.php");
   }
  }
?>
```

Cookie-ს „სიცოცხლის ვადის“ განსაზღვრა

გაჩუმებით cookie-ბი ყენდება ბრაუზერთან ერთი სეანსის განმავლობაში სამუშაოდ, თუმცა შესაძლებელია მათი „სიცოცხლისუნარიანობა“ გავზარდოთ, ანუ განვუსაზღვროთ მას „არსებობის ვადა“.

სინტაქსი:

```
  time();
```

გაუმჯობესებული ვარიანტი time ფუნქციისა არის ფუნქცია mktime:

სინტაქსი:

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

არგუმენტი is_dst განსაზღვრავს ეს თარიღი ხვდება თუ არა ზაფხულის დროის პერიოდში და შეუძლის მიიღოს მნიშვნელობები:
-1 (გაჩუმებით, ნიშნავს რომ თვისება არაა მოცემული);
0 (დროის ინტერვალი არ ხვდება ზაფხულის დროის პერიოდში);
1 (დროის ინტერვალი ხვდება ზაფხულის დროის პერიოდში);

განვიხილოთ cookie-ების ვადების განსაზღვრის მაგალითი:

```
<?
setcookie("name", $value, time() + 600);
  /* ეს cookie „ცოცხალია“ შექმნის შემდეგ 10 ჭუთის განმავლობაში */
   setcookie("name", $value, mktime(0,0,0,01,25,2011));
/* ეს cookie იცოცხლებს 2011 წლის 25 იანვრის შუაღამემდე */
```

```
  setcookie("name", $value, mktime(18,0,0,01,25,2011));
/* ეს cookie იცოცხლებს 2011 წლის 25 იანვრის 18.00-მდე */

?>
```

Cookie გაუქმება

cookie -ს გაუქმება საკმაოდ მარტივის, ამისათვის უნდა გამოვიყენოთ ფუნქცია setcookie და გადავცეთ მას იმ cookie -ს სახელი რომელიც უნდა განადგურდეს:

```
  setcookie("name");
```

უსაფრთხოების პრობლემები რომლებიც დაკავშირებულია cookie-ებთან

ხანდახან ხდება cookie-ში კონფიდენციალური მონაცემების შენახვა, შესაბამისად უნდა ვიზრუნოთ იმაზე რომ cookie-ში შენახული ინფორმაცია არ გადაეცეს მესამე პირებს. არსებობს რამოდენიმე მეთოდი ინფორმაციის დაცვისა:

1. cookie-ების დანახვის არის განსაზღვრა.
2. დომენებისათვის წვდომის არეალის შეზღუდვა
3. cookie-ების დაცული მოთხოვნით დაგაცემა.

საუკეთესო გადაწყვეტილებაა ყველა მეთოდის კომპლექსური გამოყენება. განვიხილოთ თითოეული მეთოდი ცალ-ცალკე:

cookie-ების დანახვის არის განსაზღვრა

რადგანაც გაჩუმებით cookie-ზე წვდომა ხდება ძირითადი კატალოგიდან, ამან შეიძლება გამოიწვიოს დაცვის სისტემაში ხვრელების გაჩენა, რადგანაც cookie-ზე წვდომა შესაძლებელი იქნება ამ კატალოგის ნებისმიერი ქვეკატალოგიდან. ამიტომაც უნდა მოხდეს შეზღუდვის დაწესება ყველა სხვა გვერდებისათვის და რომ მხოლოდ კონკრეტული კატალოგიდში არსებული გვერდებიდან ხორციელდებოდეს cookie-ზე წვდომა. მაგალითად /web კატალოგისთვის შეგვიძლია მოვიქცეთ ასე:

```
  setcookie("name", $value, "/web/");
```

მაგრამ ასეთ შემთხვევაშიც /web-ში არსებული გვერდებსაც ექნებათ წვდომა, მაგალითდ /web/index.php, /web1/page.html და ასე შემდეგ, შეზღუდვა შესაძლებელია დავიყვანოთ კონკრეტულ გვერდამდე:

```
  setcookie("name", $value, "/web/index.php");
```

თუმცა არც ეს მეთოდი წყვეტს პრობლემას, ინფორმაცია რომელიც განთავსებულია ზემოაღნიშნულ cookie-ში, შეუძლია მიიღოს შემდეგმა სკრიპტმა /web/index.php-script/anti_cookie.php. ამიტომ იყენებენ დამატებით დაშიფრის მეთოდს.

დაშიფრვა

Cookies დაშიფრვის სხვადასხვა მეთოდი არსებობს , განვიხილოთ ერთ-ერთი მათგანი:

```
<?
  //  დაშიფრვისათვის ვქმნით საწყისი მდგომარეობის ვექტორ
  $vector = mcrypt_create_iv(mcrypt_get_iv_size(MCRYPT_CAST_256,
                        MCRYPT_MODE_CFB), MCRYPT_RAND);
  $key = "qwe233jk312jx813893xk312"; // განშიფრვის გასაღები
  $cook_name = "maks";
  $cipher = mcrypt_encrypt(MCRYPT_CAST_256, $key, $cook_name,
                      MCRYPT_MODE_CFB, $vector);
  setcookie("username", $cipher, "/decrypt.php");
?>
```

Cookie-ს განშიფრვა ხდება decrypt.php-ის მეშვეობით, რომლის კოდიც ასეთია:

```
<?
  // საწყისი მდგომარეობის ვექტორი უცვლელი რჩება
  $vector = mcrypt_create_iv(mcrypt_get_iv_size(MCRYPT_CAST_256,
                        MCRYPT_MODE_CFB), MCRYPT_RAND);

  $key = "qwe233jk312jx813893xk312";

  $decrypt_name = mcrypt_ decrypt(MCRYPT_CAST_256, $key, $username,
                            MCRYPT_MODE_CFB, $vector);

  echo("$decrypt_name, we are glad to see you on our page!");
?>
```

დომენებისათვის წვდომის შეზღუდვა

დამატებითი უსაფრთხოებისათვის ახდენენ იმ დომენების სიაზე შეზღუდვის დაწესებას, რომლიდანაც შეიძლება cookie-ზე წვდომა:

```
  setcookie("name", $value, "/web/index.php", ".server.com");
```

cookie -ების დანახვის არის ასეთი შეზღუდვის პირობებში შემდეგ დომენებს შეეძლებათ მათი წვდომა: server.com, myservser.com, php.server.com და ასე შემდეგ (რადგანახ შეზღუდვის შემოწმება ხდება ბოლო ნაწილის შესაბამისობაზე დაყრდნობით).

cookie-ების დაცული მოთხოვნით დაგაცემა

ზედმეტი არ იქნება თუ cookie-ბს რომლებიც ინახავენ კონფიდენციალურ ინფორმაციას განვუსაზღვროთ რომ მას უფლება აქვს უპასუხოს მხოლოდ დაცულ HTTP მოთხოვნებს, რადგან ამ შემთხვევაში საკმაოდ რთულდება დაჭერა იმ მონაცემებისა რომლებსაც კლიენტი და სერვერი ერთმანეთში ცვლის. დაცული კავშირის უზრუნველყოფისათვის ფუნქციას setcookie გააჩნია მე-6 დამატებითი პარამეტრი მნიშვნელობით 1 :

```
setcookie("name", $value, time() + 600, "/web/", ".server.com", 1);
```

PHP გამონაკლისი

PHP 5-ში შემოვიდა ახალი ობიექტზე ორიენტირებული მეთოდი შეცდომებთან გამკლავებისათვის. „გამონაკლისები“ გამოიყენება იმისათვის რომ სპეციფიური მდგომარეობის შეცდომის მოხდენის შემთხვევაში შეიცვალოს კოდის მუშაობა. აღნიშნულ მდგომარეობას ქვია სწორედ გამონაკლისი.
უფრო დაზუსტებით გამონაკლისი არის სიგნალი წარმოქმნილი განსაკუთრებული სიტუაციის დროს, რომელიც ეგზავნება ინტერპრეტატორს პროგრამის მიერ კოდის შესრულების დროს
PHP-ში გამონაკლისებთან მუშაობის პრინციპები იგივეა რაც ყველგან. გამონაკლისის გენერირებისათვის გამოიყენება ოპერატორი throw (გატყორცნა), რომელსაც გადაეცემა გამონაკლისის ობიექტი, რომელსაც გააჩნია ორი არააუცილებელი პარამეტრი: გამონაკლისის შეტყობინება და გამონაკლისის კოდი

```
throw new Exception(\\\"This is exception message\\\", $exception_code);
```

განვიხილოთ თუ რა ხდება კონკრეტულად როცა გამონაკლისი „ამუშავდება“ :

- ინახება კოდის მიმდინარე მდგომარეობა;
- კოდის გაშვება გადაირთვება გამონაკლისის დასამუშავებელ წინასწარ განსაზღვრულ ფუნქციაზე (მომხმარებლის), რომელიც სიტუაციიდან გამომდინარე განაახლებს კოდის დამახსოვრებული მდგომარეობიდან, ან შეაჩერებს სკრიპტის შესრულებას ან განაახლებს სკრიპტის შესრულებას კოდის სხვა ადგილიდან.

განვიხილოთ შეცდომებთან გამკლავების მეთოდები:

გამონაკლისების გამოყენების ძირითადი გზა
გამონაკლისების დასამუშავებელი სამომხმარებლო ფუნქციები
მრავალჯერადი გამონაკლისები
გამონაკლისის ასხლეტვა
ზე-დონის გამონაკლისის დასამუშავებელი კონფიგურაციები

შენიშვნა: გამონაკლისების გამოყენება შესაძლებელია მხოლოდ შეცდომის პირობებთან და არა იმისათვის რომ კოდის ერთი ადგილიდან მეორეზე გადავხტეთ

გამონაკლისების გამოყენების ძირითადი გზა

როდესაც მოხდება გამონაკლისის გაშვება, მისი შემდგომი კოდი არ შესრულდება და PHP შეეცდება იპოვოს შესაბამისი „დამჭერი“ ბლოკი. თუ ვერ მოხდა გამონაკლისის დაჭერა, გამომუშავდება „fatal error“ -ი შეტყობინებით „Uncaught Exception“.

ვცადოთ „გავისროლოთ“ გამონაკლისი ისე რომ არ მოვახდინოთ მისი შემდგომი დაჭერა:

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

//გამოვიძახოთ გამონაკლისი
checkNum(2);
?>
```

ზემოაღნიშნული კოდის შედეგად მივიღებთ შემდეგ შეცდომაზე შეტყობინებას:

```
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
```

მეთოდი Try, throw and catch

ზემოაღნიშნული შეცდომა რომ თავიდან ავიცილოთ უნდა შევქმნათ უფრო სრულყოფილი კოდი, რომელიც უნდა შეიცავდეს:

Try - ფუნქცია რომელიც იყენებს გამონაკლისს უნდა იყოს "try" ბლოკში. თუ არ მოხდა გამონაკლისის გაშვება კოდი გაგრძელდება ჩვეულებრივად, ხოლო წინააღმდეგ შემთხვევაში მოხდება გამონაკლისის გატყორცნა (throw)
Throw - ეს წარმოადგენს როგორ უნდა მოხდეს გამონაკლისის „გატყორცნა“, თითოეულ „გატყორცნა“-ს უნდა ქონდეს შესაბამისი ერთი მაინც „დაჭერა“
Catch - "catch" ბლოკი იღებს გამონაკლისს და ქმნის ობიექტს რომელიც შეიცავს ინფორმაციას გამონაკლისზე
მარტივად რომ ვთქვათ: გამონაკლისი რომ დავიჭიროთ ვიყენებთ კონსტრუქციას try...catch. ბლოკში try სრულდება ოპერაციები რომლებმაც შეიძლება მიგვიყვანონ განსაკუთრებულ (გამონაკლის) სიტუაციებამდე , ხოლო ბლოკი catch საშუალებას გვაძლევს მივიღოთ გადაწყვეტილება თუ რა გაკეთდეს იმ შემთხვევაში თუ გამონაკლისი იქნა გატყორცნილი(thrown).

განვიხილოთ კოდი (სრულყოფილი გაშვებისათვის- „მუშა“ კოდი ანუ ☺):

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

//გამონაკლისის გაშვება "try" ბლოკში
try
 {
 checkNum(2);
 //თუ მოხდა გამონაკლისის გატყორცნა (thrown),ეს ტექსტი არ გამოჩნდება
 echo 'If you see this, the number is 1 or below';
 }

//გამონაკლისის დაჭერა
catch(Exception $e)
```

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

ზემოაღნიშნული კოდი გამოიტანს შეცდომის შემდეგ შეტყობინებას:
Message: Value must be 1 or below

მაგალითის ახსნა: ზემოთ მოყვანილი კოდი ისვრის და იჭერს გამონაკლისებს:

checkNum() ფუნქცია ამოწმებს, რიცხვი მეტია, თუ არა 1-ზე. თუ ის მეტია, გამონაკლისი გაიტყორცნება.
checkNum() ფუნქცია გამოძახებულია "try" ბლოკში
გამონაკლისი checkNum() ფუნქციაში გატყორცნილია შესაბამისად "catch" ბლოკი პოულობს გამონაკლისს და ქმნის ობიექტს($e) , რომელიც შეიცავს გამონაკლისის ინფორმაციას
შეცდომის შეტყობინება გამონაკლისიდან იბეჭდება $e->getMessage()-ის გამოძახებით.

თუმცა, არსებობს ერთი გზა "ყველა გატყორცნა დაჭერილ უნდა იქნას" წესის თავიდან ასაცილებლად. ესაა შეცდომებზე ზე-დონის გამონაკლისების დანიშვნა.

გამონაკლისების სამომხმარებლო კლასების შექმნა

გამონაკლისის სამომხმარებლო კლასის შექმნა საკმაოდ მარტივია. უბრალოდ ვქმნით სპეციალურ კლასს იმ ფუნქციებით რომლებიც შესაძლებელი იქნება რომ გამოვიძახოთ როდესაც მოხდება გამონაკლისი PHP-ში. კლასი უნდა იყოს გამონაკლისის კლასის გაფართოება.
გამონაკლისის სამომხმარებლო კლასს მემკვიდრეობით გადაეცემა PHP-ს Exception კლასის თვისებები (მეთოდები) და შესაძლებელია მისთვის მომხმარებლის ფუნქციების დამატება.
Exception ობიექტს გააჩნია შემდეგი ფინალური (final) მეთოდები:
final function getMessage(); // გამონაკლისის შეტყობინება
final function getCode(); // გამონაკლისის კოდი
final function getFile(); // ფაილი საიდანაც მოხდა გამონაკლისის გატყორცნა
final function getLine(); // გატყორცნის სტრიქონი
final function getTrace(); // გამოძახებების სტეკის მასივი
final function getTraceAsString(); //გამოძახებების სტეკის მასივი დაფორმატებული სტრიქონში.

შევქმნათ გამონაკლისის კლასი:

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

$email = "someone@example...com";

try
 {
 //შევამოწმოთ if პირობა
 if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE)
  {
  //გამონაკლისის გატყორცნა თუ ასეთი მაილი არ არსებობს
  throw new customException($email);
  }
 }

catch (customException $e)
 {
 //გამოაქვს მომხმარებლის შეტყობინება
 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().

ზემოთ მოყვანილი მაგალითის ახსნა:

customException() კლასი შექმნილია და ის მემკვიდრეობით იღებს ძველი კლასის თვისებებს და მეთოდებს

errorMessage() ფუნქცია დააბრუნებს შეცდომის შეტყობინებას, თუ ელ-ფოსტის მისამართი არასწორია
"try" ბლოკი ამუშავდა და გამონაკლისი გაიტყორცნა, მას შემდეგ რაც ელ-ფოსტის მისამართი არასწორი აღმოჩნდა
"catch" ბლოკი იჭერს გამონაკლისს და გამოსახავს შეცდომის შეტყობინებას .

მრავალჯერადი გამონაკლისი

სკრიპტს შეუძლია გამოიყენოს რამოდენიმე გამონაკლისი, რამოდენიმე პირობის შესამოწმებლად. (ვრცლად კონსპექტში)

გამონაკლისების ასხლეტა

ზოგჯერ, როდესაც გამონაკლისი გაისროლება, ჩვენ შესაძლოა ვისურვოთ მისი გამოტანა სტანდარტული გზისგან განსხვავებულად. ეს შესაძლებელია გამონაკლისის მეორეჯერ გატყორცნით "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();
}
?>
```

ისმის კითხვა როდისაა მოსახერხებელი გამონაკლისების გამოყენება? რათქმა უნდა ყოველთვის როცა ფუნქციას ან მეთოდს შესაძლოა მივყავდეთ შეცდომამდე. ბაზებთან მიმართებაში მაგალითად ფუნქცია mysql_connect() და mysql_select_db() გამოყენებისას თუნდაც:
კოდს ექნებოდა შემდეგი სახე:

```
try {
   mysql_connect($hostname, $username, $password);
   mysql_select_db($dbname);
} catch (Exception $e) {
   echo $e->getMessage(); //გამოიტანს შეტყობინებას ან დაკავშირების ანდა ამორჩევის შეცდომაზე
 }
```

წესები გამონაკლისებისათვის

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

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