Списание .ком
Уеб училище
Bulgaria WebDev Magazine
издание за територията на България Урок 2/2005
списание за интернет технологии списание за интернет технологии списание за интернет технологии списание за интернет

Курс по PHP

"Първи стъпки в PHP"

Урок 4/2005

Съдържание

Какво научихме от предния урок?

Как да настроим PHP, така че да работим добре с него и да ни върши работа за следващите уроци. Ако не сте го направили или все още имате проблеми, питайте във форума или прочетете отново уроци от 1 до 3 за да направите така, че системата да работи добре.

А сега?

В този урок, ще разгледаме няколко основни неща:

1) Как се подават данни от една страница към друга;

2) Какво са масивите;

3) За какво е print_r();

4) Ще научите за някои новости и възможности и ще пишете доста код;

Отворете текстовия си редактор и нека да напишем един скрипт:

```
<html>
<head>
<title>Тест 1: начална страница</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>

<body>
<form name="form1" method="post" action="page2.php">
  <p>
    <input type="text" name="fname">
  </p>
  <p>
    <input type="submit" name="Submit" value="Изпрати">
  </p>
</form>
</body>
</html>
```

Нека да рагледаме някой елемнти на този файл и съответните им зависимости със следващата част на упражнението.

```
<form name="form1" method="post" action="page2.php">
```

Няма да се спирам на HTML обясненията на елементите на този HTML файл. Вие би трябвало да имате познания по този език. Затова ще премина направо на елементите в друг цвят (втори и трети елемент)

method="post" – определя метода по който се подават данните към следвашата страница посочена в action="page2.php". Малко по-надолу ще дам примери, какви точно са разликите на двата метода.

```
<input type="text" name="fname">
```

Всяко поле от една форма си има име, което в последствие предава на следващата страница, **заедно** със стойността въведена от потребителя.

Горния код, можете да запишете под името index.html в папка **urok 4** в папка **htdocs** на мястото, където сме инсталирали Apache.

Ако желаете можете да го сложите и във друга папка, но тя трябва да е в **htdocs.**

page2.php

Отворете нов документ и въведете в него само това:

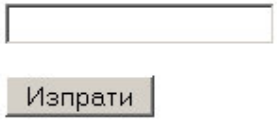
```
<?php

 print_r($_POST);

?>
```

Запишете този файл като **page2.php** в същата папка **urok4**.

След това, направете следното:

1) Стартирайте файла, който направихте преди малко **index.html,** по начина, по който стартирахме и предните упражнения: http://127.0.0.1/urok4/index.html

Нещото, което би трябвало да видите е следното:



2) Въведете нещо в полето и натиснете бутона "Изпрати". Аз написах "списание" и резултата който ТРЯБВА да се покаже е :

```
Array
(
    [fname] => списание
    [Submit] => Изпрати
)
```

За да продължим по нататък, обаче , ще направим малко прекъсване за да видим Какво са това масивите и как се работи с тях. Следващата част е написана от Боян Джумаков от php-bg.org. Тя дава добри начални познания за масивите, а ако искате да забълбочите, е добро начало да тръгнете от тук.

Какво са това масивите

Масивите в паметта на компютъра представляват, общо казано, подредени клетки с информация. В тези клетки може да има числени или знакови стойности. Възможно е, обаче елементите на масива да не са просто числа, а сложни стуктури от данни (обекти, символни низове, а дори и други масиви). Така може дасе организира списъчна или дървовидна структура на данните.

В PHP масивите са представени чрез последователност от двойки ключ-стойност. Това означава, че на всяка клетка от масива се задава уникално име (пр. число) и на него се провоява определена стойност. На практика масивите можете да разглеждате, като съвкупност от отделни променливи.

Пример:
```
$arr[0] = 5; $arr[1] = -6;
```

За ключове на елементите на масив в PHP можете да използвате, обаче и цели символни низове:

Пример:
```
$arr["name"] = "Boyan";
```

Това са т.нар. асоциирани списъци.

Приложенията на масивите са много - от обикновени списици до сложни речници, дървовидни структури и т.н.

За да създадем една променлива като масив, трябва да използваме ключовата дума Array ето така:

Пример:
```
<?php
$arr = array("foo" => "bar", 12 => true);// Създаваме масив

echo $arr["foo"]; // bar
echo $arr[12];    // 1
?>
```

В горният пример, променливата $arr е дефинирана като масив. Демонстрирано е също така, че може в един масив да има както цифрови ключове, така и символни такива (смесен тип масиви).

Когато се създава един масив, ако за дадена негова сойност не е зададен ключ, то автоматично му се създава ключ със стойност по-голяма с единица от най-голямата стойност на числен ключ.

Пример:
```
<?php
// Този масив е същия като...
array(5 => 43, 32, 56, "b" => 12);

// ...този
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

spisanie.com

Ако създавате един масив, който има изцяло числени ключове, в PHP ви е предоставена възможността да оставите изцяло на интерпретатора задачата да определя ключовете, благодарение на горепосочените възможности:

```
<?php
$arr = array(6, 12); //$arr[0]->6; $arr[1]->12

$arr[] = 45; // $arr[2] = 45

?>
```

Този начин на задаване на стойности е особено полезен в практиката в някои задачи, в които динамично се въвеждат нови елементи в масива и следенето на брояча би затормозило програмиста.

Трябва да се отбележи, че при използването на тази функция PHP не следи за пропуснати стойности при ключовете. Например ако имаме

```
$arr = array(0=>6, 1=>12, 6 => 3);
```

и решим да добавиме по гореспоменатия начин нов елемент, ще получиме това:

```
$arr[] = 43; // $arr[7] = 43;
```

Докато в други съвременни езици от рода на PHP масивите с числени ключове и тези представляващи символни низове са отделни видове масиви, то в PHP те са обединени в един общ, с цел улеснение.

Съществуват много функции в PHP предназначени за работа с масиви.

Група от тези функции са тези за сортиране.

Съществъване функции за обхождане на целите масиви, за предвижване напред или назад по масива (пълзене), за извеждане на ключа на настоящия елемент (който разглеждаме в момента) и др. Съществуват функции също така и за конвертиране на масив в отделнни променливи и за обединяване на няколко променливи в един масив.

Ако имаме например променливата $arr = "a,b,c" можем да използваме функцията explode за да създадем масив, съдържащ отделните букви, разделени с зашпетая. Като първи параметър на функцията трябва да подадем разделителя - запетаята, а като втори - изходния символен низ ето така:

```
$new_array = explode(",", $arr);
```

Така ние ще имаме масив $new_array с 3 елемента - $new_array[0] ->"a", $new_array[1] ->"b" и $new_array[2] ->"c".

Както сте забелязали когато създаваме масив с числени ключове, то първият ключ винаги е 0 или по друг начин казано първият елемент на масив е нулевият.

spisanie.com

Обратно на кода

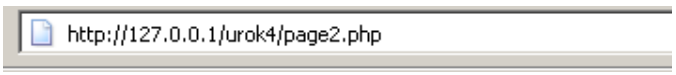
Нека да се върнем на резултата от  **page2.php**, който беше:

```
Array
(
    [fname] => списание
    [Submit] => Изпрати
)
```

Да , резултата е масив и можем да имаме достъп до всеки от елементите му, по следния начин:

```
<?
 echo $_POST["fname"];
?>
```

Ще изведе като резултат: **списание.** Тоест, това което сме въвели на предната страница. Нека да видим, обаче какъв е URL-а в браузера:



Нека сега да променим малко index.html и да го запишем по следния начин, но под името index2.html

```
<html>
<head>
<title>Тест 2: начална страница</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>

<body>
<form name="form1" method="get" action="page3.php">
  <p>
    <input type="text" name="fname">
  </p>
  <p>
    <input type="submit" name="Submit" value="Изпрати">
  </p>
</form>
</body>
</html>
```

Забелязахте ли разликата? Нека сега да създадем един файл, който да го запишем пак в директорията, която създадохме в началото на този урок и да го наречем  **page3.php** и да е със следното съдържание:

```
<?php

  print_r($_GET);

?>
```

Резултатът ще е същият, както и в първия пример, само че URL-а ще изглежда така:

Имената на полетата и техните стойости се предават 'видимо' чрез метода GET и съответно се търсят в масива $_GET. Когато те се предават, чрез метода POST, променливите и техните стойности се намират в масива ... $_ POST.

Важно:

Функцията print_r(); показва съдържанието на масива във вида, който видяхме по –горе. Повече за нея, можете да прочетете тук : http://www.php.net/print_r

Задача:
направете си един файл в който има и двата метода и POST и GET и подавайте различни данни за да видите, как се сменят даните в масивите, когато се ползва единият или другият метод. Ето малко код за начало:

```
<?php

  //тук масива ще има данни само, ако бъде използван метода GET в предния файл.
  echo "This is output of GET";
  print_r($_GET);

  //тук масива ще има данни само, ако бъде използван метода POST в предния файл.
  echo "<br>This is output of POST";
  print_r($_POST);

?>
```

С това този урок завършва, ако имате въпроси, моля пишете във форума:
http://spisanie.com/f/

Next >>>
Следващият урок ще продължи заниманието с комуникацията между страниците, като покаже интересни трикове и повече знания.

Други
От днес имаме вече нов курс, които може да продължи заниманията ви с PHP, повече за него можете да видите тук: http://spisanie.com/eshell/php3.php

Не забравяйте и за другият курс: http://spisanie.com/eshell/php2.php

Ако ви се хапва **сладко или солено,** вижте тук: http://spisanie.com/f/viewtopic.php?t=122

За всичко друго можете да пишете на
eshell@spisanie.com
http://spisanie.com