Тема 1. Основи на PHP

В тази тема:

- Въведение в PHP
- Променливи и константи
- Изрази, операции и връзки
- Конструкции на езика

1. Въведение в PHP

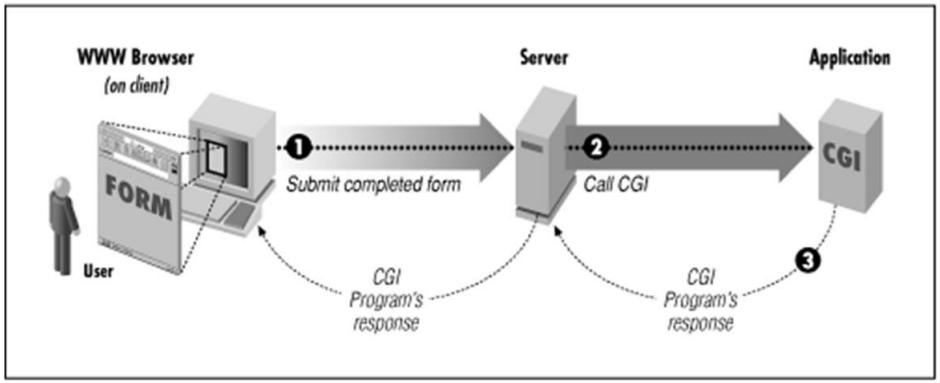
1.1. Въведение в CGI (Common Gateway Interface)

CGI е стандарт за интерфейс на външни програми с информационни сървъри чрез Интернет.

Основни разлики между CGI и обикновения HTML документ:

- HTML документът е статичен;
- CGI се изпълнява в реално време и генерира изходна динамична информация
- Програмата на CGI се изпълнява, докато HTML документът съдържа константен текст, който не може да бъде променен

Принцип на работа на CGI



1.2. Въведение в PHP

Език за програмиране на страната на сървъра проектиран за разработване на уеб страници, но също така се използва като език за програмиране с общо предназначение.

PHP (Personal Home Page) първоначално е създаден от Rasmus Lerdorf през 1994 г. Сега езикът се поддържа от PHP Group (изпълняван от Zend Engine (машина за изпълнение на скриптове с отворен код)).

Кодът на PHP

- Може лесно да се смеси с HTML кода.
- Може да се използва в комбинация с различни машини за шаблон и уеб платформи.

Интерпретаторът на PHP

- Явява като модул на уеб сървър.
- Или като CGI (Common Gateway Interface).

След като кодът на PHP се интерпретира и изпълни, уеб сървърът изпраща резултатния изход към своя клиент, обикновено във форма като част от генерирана уеб страница. Кодът на PHP генерира уебстраници с HTML код, изображения, или други видове данни.

Интерпретаторът на PHP, се поддържа от Zend машина, е свободен софтуер лицензиран при условията на PHP. PHP е широко пренесен и може да бъде безплатно разположен на повечето уеб сървъри за почти всички ОС и платформи.

1.3. Първа програма на PHP

primer-1-01.php

```
<?php
// Коментар на 1 ред: Отпечатване на екрана на текст
/* Многоредов коментар:
Първи варинат на програма на
PHP */
echo "<html><body>";
echo "<h1>Hello world!</h1>";
echo "</body></html>";
?>
```

- PHP кодът се поставя между таговете <?php ….. ?>. Всичко, което се намира извън тези тагове се показва като обикновен текст.
- Възможно е да се използва и съкратен синтаксис на тага:
- <? ….. ?>. Някои сървъри не изпълняват съкратения синтаксис. Всичко зависи от настройките на конкретния сървър.
- Показването на HTML кода се извършва с оператора **echo**, също така може да се използва функцията **print()**:
- print("текст");
- Всеки оператор завършва с ;

Стартиране на програмата

- Файлът primer-1-01.php се премества в папката wamp\www. В полето за адреси на браузъра се въвежда адреса http://localhost/ primer-1-01.php
- В прозореца на браузъра се показва текста: Hello world!
- Генерирания HTML код е:
- <html><body><h1>Hello world!<h1></body></html>

Втори вариант на програмата

HTML кодът е изваден от тага на PHP за да се покаже като обикновен текст:

primer-1-02.php

```
<html><body>
<?php
echo "<h1>Hello world! </h1>";
?>
</body></html>
```


2. Променливи и константи

В PHP пред името на всяка от променливите се поставя символ $.

2.1. Обявяване на променливи

```
$webdesigner = "Иванов";
$counter = 0;
$price = 0.00;
```

За разлика от езици за програмиране като C#, Visual Basic и др. в PHP при деклариране на променлива не се оказва нейния тип – той се приема автоматично, в зависимост от присвоената стойност.

Типове променливи в PHP

Тип данни	Описание
integer	Цяло число в диапазон -2 147 483 648 до +2 147 483 647
boolean	Логически тип данни
double	Число с плаваща запетая с повишена точност
string	Символен низ
array	Масив
object	Обект (достъп до свойствата се получава с оператора ->: $car->color

2.2. Функции за проверка на типа на променливата

```
is_integer($x)
is_double($x)
is_string($x)
is_array($x)
is_object($x)
is_boolean($x)
```

Връщат true, ако променливата $x е от посочения тип

`gettype($x)` – връща низ, описващ типа на променливата: integer, double, string и т.н.

- Тези функции са полезни за обработване на параметри, получени от PHP програмата.
- За принудително задаване на типа на променливата се използва функцията **settype()**

bool settype (mixed &$var , string $type)

```
$price2 = 0;
settype($price2, "double");
```


Пример за деклариране на променливи и показване на техните стойности:

primer-1-03.php

```
$userno = 1;
$username = "Ivanov";

echo $userno;
echo "<br>";
echo $username;
```

2.3. Константи

Дефинирането се извършва с функцията **define**, която има синтаксис:

```
bool define (string $name , mixed $value [, bool
$case_insensitive = false] )
```

Параметри:

- първия – име на променливата;
- втория – стойност, която да приеме;
- третия – дали да се отчита регистъра на буквите.

Пример (primer-1-04.php) за деклариране на константа:

```
define (“NUMBER_PI", 3.14159 , true);
echo NUMBER_PI;
```

След деклариране, константата се използва без $ пред нейното име:

```
echo NUMBER_PI;
```


В PHP има стандартни константи. Например:

- PHP_OS – показва версията на операционната система;
- PHP_VERSION – показва версията на PHP.

primer-1-05.php

2.4. Действия над променливите

Над променливите може да се изпълнят три действия:

- присвояване на стойност на променливата;
- проверка за съществуването на дадена променлива;
- унищожаване на променливата.

Проверка за съществуване на променлива

Функцията **isset()** проверява дали посочената променлива съществува и връща true, ако променливата съществува и false, ако не е декларирана.

Проверката за съществуване на променлива е полезно когато тя се получава от HTML форма (чрез методите GET и POST) или от URL (само с метод GET).

Пример:

```
// Проверка, дали променливата page е подадена чрез метода GET
if (isset($_GET[page])) {
// извършват се действия с променливата page
}
else echo "Променливата не е декларирана")
```


Унищожаване на променлива

Полезно е когато трябва да се освободи памет, особено в случаите, когато променливата заема много памет. Унищожаването се извършва с функцията **unset()**.

Пример:

```
// Унищожаване на променлива, която сме създали преди това
unset ($x);
//  Унищожаване на променлива, предадена чрез метода GET
unset ($_GET[page]);
```


Инициализация на променлива

В PHP декларирането и инициализирането на променливи се извършва с присвояване на стойност. Интерпретаторът автоматично определя типа на променливата от присвоената стойност.

primer-1-06.php

```
$quantity = 0;
$price = 0.00;

echo "Типът данни на \$quantity е: ";
echo gettype($quantity);
echo "<br>";
echo "Типът данни на \$price е: ";
echo gettype($price);
```

Резултатът от изпълнението е:

```
Типът данни на $quantity е: integer
Типът данни на $price е: double
```


Разширяваме primer-1-06.php с инициализиране на променлива $price2 със стойност о и принудително задаване на нейния тип – double.

```
$price2 = 0;
settype($price2, "double");
```


Когато се присвоява стойността на една променлива на друга, интерпретаторът автоматично присвоява и нейния тип.

При присвояване на няколко променливи на една и съща стойност може да се използва синтаксиса:

$променлива = $променлива = стойност;

```
$x = $y = 5;
```


3. Изрази, операции и връзки

Същност на изразите

Елементи на езика, които се използват за изчисление.

Състоят се от константи, променливи и функции, свързани със съответния знак за операция.

```
$amount = ($quantity * $price);
```

Аритметични операции

+, -, *, /, % (деление с остатък)

Побитови операции

Предназначени са за промяна (от 0 на 1 и обратно) на групи битове от целочислени променливи. Всяко число може да се разглежда като последователност от битове. Целите числа в PHP са 32 разрядни, ето защо за представяне на едно число се използват 32 бита.

0000 0000 0000 0000 0000 0000 0000 0000 - числото 0

0000 0000 0000 0000 0000 0000 0000 0001 - числото 1

0000 0000 0000 0000 0000 0000 0000 0010 - числото 2

0000 0000 0000 0000 0000 0000 0000 0011 - числото 3

Побитови операции:

- a & b – ще се вдигнат (ще станат 1) само тези битове, които са установени (са единица) в a и b едновременно:
  a = 1111 (15); b = 1100; Резултат = 1100 (12).
- a | b – ще се вдигнат само тези битове, които са установени в a или b:
  a = 0100 (4); y = 0101 (5); Резултат = 0101 (5).
- ~a – инвертиране на битовете (единицата става 0 и обратно):
  a = 1001; Резултат 0110.

- a << b – изместване вляво. Всички битове на a се изместват вляво на b позиции:
  a = 0010; b = 0001 (1 разряд); Резултат 0100.
- a >> b – изместване надясно. Всички битове на a се изместват надясно на b позиции:
- a = 0010; b = 0001 (1 разряд); Резултат = 0001.

Логически изрази и операции

Оператори за сравнение:

== равно

!= неравно

<, >, <=, >=

Логически операции

!a (NOT)

a && b (AND)

a || b (OR)

Низ и низови стойности

Низовите стойности могат да бъдат заградени с апострофи или с двойни кавички:

$s1 = 'Hello';

$s2 = "world";

Има голяма разлика между двата вида ограждане:

- Низ заграден в апострофи – всички символи се третират каквито са. Изключение прави само последователността \' и \\. Първата се третира като обикновена наклонена черта, а втората като обратно наклонена черта.
- Низ заграден в кавички – позволява освен останалото да се покажат и стойностите на променливите и да се използват специални символи.

echo '$s1, world'; // показва: $s1, world

echo "$s1, world"; // показва: Hello world

Низовете, заградени с двойни кавички, могат да съдържат следните специални символи:

\n – символ за нов ред

\r – символа за връщане на каретката

\t – символ за табулация

\$ - знак за $

\" – кавички

\\ - обратно наклонена черта

\xNN – символ с код NN (в шестнадесетична система)

Низовите операции в PHP са две:

$s1.$s2 – конкатенация (сливане) на два низа;

$s1[n] – обръщение към символ с номер n от низа.

Пример: Конкатенация на низове. Обръщение към символи от низа.

primer-1-07.php

```
$user_first_name = "John";
$user_last_name = "Smith";

echo "Hello ".$user_first_name." ".$user_last_name;
echo "<br>";
echo "Your initials are: ".$user_first_name[0].$user_last_name[0];
```

Резултатът е:

Hello John Smith
Your initials are : JS

Връзки

В PHP няма указатели, подобно на езика C, а има връзки. Те са аналог на указателите в C. Връзките в PHP приличат повече на връзките във файловата система UNIX.

Връзките са два вида: твърди и символични.

Твърдата връзка може да се разглежда като синоним на променлива. Те се създават с оператора &.

primer-1-08.php

```
$var1 = 10;
$link_to_var1 = &$var1;

$link_to_var1 = 20;
echo $var1;        // 20
```


Прекъсване на свързаността между две променливи се извършва с функцията **unset()**;

```
unset ($link_to_var1);
```

Забележка: Променлива, към която има връзка не може да се изтрие (премахне).

4. Конструкции на езика

4.1. Оператор if-else

```
if (логически израз)
     оператор_1;
else
     оператор_2;
```


Записване на няколко оператора на 1 ред:

```
if (логически израз) { оператор_1; оператор_2; ....; оператор_n; }
```

За проверка на няколко логически условия се използва elseif:

```
if (логически израз 1)
     оператор_1;
elseif (логически израз 2)
     оператор_2
else
     оператор_3;
```


4.2. Конструкция switch-case

Конструкция за избор.

```
switch (израз) {
case стойност1 : оператор_1; [break;]
...
case стойностN : оператор_N; [break]
[default: оператор_по_подразбиране; [break]]
}
```


primer-1-10.php

```
$year = 2011;
switch ($year)
{
case 2010: echo "Годината е 2010\n";
case 2011: echo "Годината е 2011\n";
case 2012: echo "Годината е 2012\n";
case 2013: echo "Годината е 2013\n";
case 2014: echo "Годината е 2014\n";
default: echo "Неразпозната година\n";
}
```

Резултатът ще бъде:

```
Годината е 2011
Годината е 2012
Годината е 2013
Годината е 2014
Неразпозната година
```

Primer-1-10.php: Правилно използване – изисква break.

```
$year = 2011;
switch ($year)
{
case 2010: echo "Годината е 2010"; break;
case 2011: echo "Годината е 2011"; break;
case 2012: echo "Годината е 2012"; break;
case 2013: echo "Годината е 2013"; break;
case 2014: echo "Годината е 2014"; break;
default: echo "Неразпозната година"; break;
}
```

Резултатът ще бъде:
Годината е 2011

4.3. Цикли

Пример: Отпечатване на числата от от 0 до 9 на екрана.

Цикъл for

Използва се когато тялото на цикъла трябва да се изпълни точно определен брой пъти.

```
for (инициализиращи_команди; условие;
команди_след_итерациите)
{ тяло на цикъла }

for ($i=0; $i<10; $i++) echo $i;
```


Цикъл while

Цикълът изчислява логическия израз и изпълнява тялото на цикъла на докато логическия израз е истина.

```
while ( логически израз )
     оператор;

$1=0;
while ($i++ < 10) echo $i;
```

Цикъл do while

Отначало се изпълняват операторите (тялото на цикъла), а след това се проверява условието. Следващата итерация започва, ако условието е истина.

```
do
     {
     // тяло на цикъла
     }
while (условие)
```

```
$i = 1;
do echo $i;
while ($i++ < 10);
```


Цикъл foreach

Използва се за работа с масиви.

Оператори break и continue

break прекъсва работата на цикъла.

continue прекъсва само текущата итерация.

4.4. Инструкции include и require

Използват за добавяне на допълнителните сценарии. Require добавя съответния код преди да е започнало изпълнението на сценария, докато include добавя код по време на изпълнение на сценария.

Добра практика е да се използва require. В повечето случаи това ще бъде възможно – добавяне на сценарии, допълнителни класове и др.

Инструкциите require_once и include_once работят по същия начин като require и include, с тази разлика, че преди да добавят сценария интерпретатора проверява дали вече той е добавен веднъж. Ако е бил добавен веднъж, не го добавя втори път.

Пример за добавяне на файла confog.php.

```
require "config.php";
```


Препоръчителна литература

Денис Колисниченко

PHP & MySQL

практическо програмиране

Издателство АСЕНЕВЦИ

2014 г.

ISBN: 978-954-8898-37-9