RYCHLOKURZ PHP

Na jazyk PHP jste jistě slyšeli mnoho chvály. Pokud chcete s jeho pomocí zkusit vytvořit nějakou pěknou webovou aplikaci, třeba vám přijde vhod náš kurz základů PHP.

Úvod

V předchozím čísle Chipu se mohli uživatelé oblíbeného skriptovacího jazyka PHP seznámit s postupem instalace aktuální verze tohoto hypertextového preprocesoru, a sice s instalací na servery od společnosti Microsoft (IIS a PWS) provozované na operačních systémech téže společnosti. Od dnešního čísla přinášíme rychlokurz PHP určený čtenářům, kteří již slyšeli o přednostech a charakteristických vlastnostech tohoto jazyka a rozhodli se naučit nezbytné základy potřebné pro vytváření dynamických webových aplikací právě pomocí zmíněného nástroje (připomeňme, že s jazykem PHP jste se mohli seznámit v sérii tří článků od Jana Stoklasy, jež byly věnovány skriptovacím jazykům pro web a které vyšly v loňských Chipech 9 – 11).

V tomto úvodním díle budou popsány naprosté základy PHP – základy syntaxe a řídicí struktury. Další díly budou již zaměřeny na práci s webovými formuláři a na propojení PHP s různými databázemi.

Pozn.: Náskok budou mít ti čtenáři, kteří někdy v minulosti programovali v C/C++ (nebo v jazyce s podobnou syntaxí), neboť – jak brzy poznáte – PHP a C/C++ mají mnoho společného.

Začínáme

Předpokládám, že čtenáři se již s HTML (Hypertext Markup Language) setkali a zpočátku je bude logicky nejvíce zajímat, jak lze umístit PHP skript do HTML kódu. K tomuto účelu byly navrženy následující značky. Samozřejmě je zachována kompatibilita se standardem SGML (Standard Generalized Markup Language), tedy se speciální sekvencí znaků, jež v dokumentu oddělují příkazy od textu:

```
<? a  ?>
<?php a ?>
<script language=”php’> a </script>
<%  a  %>
```

Začlenění těchto značek je patrné z následujícího příkladu, ve kterém je pomocí příkazu echo vypsán v prohlížeči text: Ahoj, světe, začínáme! Komentáře jsou v textu odděleny pomocí dvojice znaků // (vše, co je napsáno za těmito znaky na daném řádku, bude považováno za komentář). Pro vložení delšího komentáře – víceřádkového – se používá uvozující znak /* a ukončující znak */.

```
<html>
<body>
<?php                          // začátek kódu
  echo ”Ahoj, světe, začínáme!”;   //výpis textu
?>
</body>
</html>
```

Tento soubor, pojmenujme jej například jako ”vypiš.php”, musíme samozřejmě nejprve umístit do adresáře, ve kterém je nastavena podpora PHP skriptů.

Jak již víme, hypertextový preprocessor PHP dynamicky vytvoří novou WWW stránku, ve které původní PHP kód nahradí požadovaným HTML kódem. Výsledkem předchozího skriptu bude následující vygenerovaný kód:

```
<html>
<body>
 Ahoj, světe, začínáme!
</body>
```

```
</html>
```

Vše se točí kolem proměnných

Téměř vše v programování se točí kolem proměnných (proměnná představuje datovou strukturu, jejíž hodnota se může v průběhu zpracování programu dynamicky měnit). Pomocí proměnných jsou uchovávány hodnoty, s nimiž v programu/skriptu operujeme. V PHP poznáme proměnné velice rychle, neboť všechny musí obsahovat povinný znak $, například proměnná a se ve skriptu skrývá za $a. Na rozdíl od již zmiňovaného jazyka C provádí PHP spoustu práce automaticky za nás – například deklarování proměnných v tomto jazyce odpadá (deklarace je příkaz, který udává typ proměnné a její jméno). Při prvním použití se proměnná nadeklaruje sama a je jí také přiřazen určitý typ.

Například příkaz

```
$a=10
```

přiřadí do proměnné $a číslo 10 (znak rovnítka "=" tedy slouží k přiřazení). $a bude automaticky celé číslo.

Tímto příkladem jsme se dostali k prvnímu typu, kterým je integer (celé číslo). Dalšími datovými typy jsou:

```
floating-point numbers (desetinné číslo)   – např. $b=10.25;
string (řetězec znaků)      – např. $c="text";
array (pole)                – např. $d[0]="abc";  //jednodimenzionální pole
                  – např. $d[0][0]="def";  //vícedimenzionální pole (dvoj-)
a object (objekt)
```

Řídicí struktury

Vysvětlení a pochopení práce s řídicími strukturami je nezbytnou podmínkou pro další psaní skriptů (ve všech jazycích), neboť jasně definují a řídí další běh programu. Proto ani v tomto rychlokurzu nezůstaneme "pozadu" a popíšeme si nejčastější struktury pomocí jednoduchých příkladů.

Příkazy pro větvení programu

Mezi nejpoužívanější struktury patří podmínky, pomocí kterých je podle nějaké zadané podmínky určován další běh (větvení) skriptu.

Příkaz **IF**

```
if ($aa==$bb)                          // když je $aa rovno $bb
  echo "aa je stejné jako bb";       //vypiš- aa je stejné jako bb
```

Nebo forma srozumitelnější pro "Céčkaře":

```
if ($aa==$bb) {
  echo "aa je stejné jako bb";
  }
```

Podmínka IF ELSE

```
if ($aa>$bb) {                         // když je $aa větší než $bb
  echo "aa je větší než bb";         //vypiš- aa je větší než bb
} else {                               //jinak
  echo "aa není větší než bb";       //vypiš- aa není větší než bb
}
```

Podmínka ELSEIF

```
if ($aa>$bb) {                         // když je $aa větší než $bb
  echo "aa je větší než bb";         // vypiš- aa je větší než bb
} elseif ($aa==$bb {                   // jinak když je $aa rovno $bb
```

```
  echo ”aa je rovno bb”;            //vypiš- aa je rovno bb
} else {                                        //jinak
  echo ”aa je menší než bb”;            //vypiš- aa je menší než bb
}
```

SWITCH – PHP také obsahuje příkaz přepínače neboli příkaz pro mnohonásobné větvení programu.

```
switch ($i) {                                   //když $i
  case 0:                                       // je 0
          print ”i rovno 0”;// vypiš i rovno 0
          break;                                //ukončení větve přepínače pomocí příkazu break
  case 1:                                       // je 1
          print ”i rovno 1”;// vypiš i rovno 1
          break;                                //ukončení větve přepínače pomocí příkazu break
}
```

Iterační příkazy – cykly

Mnozí programátoři postupem času narazí na požadavek, kterým je možnost/nutnost opakování jisté části skriptu podle zadané podmínky. Pro tento účel se v programování používají cykly, o kterých si dále povíme.

Velice často používaným je cyklus **WHILE** – tento iterační příkaz testuje podmínku cyklu před průchodem cyklem – cyklus tedy nemusí proběhnout ani jednou.

```
$a=20;          //přiřazení čísla 20 do proměnné $a
while ($a>0)            //dokud je $a větší než 0 pak dělej
{
  print $a;     // vypsání hodnoty proměnné $a
  $a--;         //dekrementování po použití=původní hodnota výrazu snížena o 1
}
```

Cyklus **DO-WHILE** – v tomto cyklu se podmínka testuje až po průchodu cyklem – cyklus proběhne nejméně jednou.

```
$a=20;          //přiřazení čísla 20 do proměnné $a
do                      //dělej (dokud $a>0)
{
  print $a;     // vypsání hodnoty proměnne $a
  $a--;         // dekrementování po použití=původní hodnota výrazu snížena o 1
} while ($a>0)  //dokud $a>0
```

Příkaz **FOR** – typický příkaz cyklu, který použijeme, když známe předem počet požadovaných průchodů cyklem.

```
for ($i = 1; $i <= 10; $i++) {          // pro $i od 1 do 10 inkrementuj $i
  print $i;                     // vypsání $i -> 12345678910
}
```

Příkaz **FOREACH** – další příkaz cyklu (jakási modifikace předchozího cyklu), který opět použijeme v případě, že dopředu známe požadovaný počet průchodů cyklem. Tento příkaz je v PHP zahrnut až od verze 4 a umožňuje nám iterovat přes pole.

Pozn.: Nemusíme použít příkaz reset(), protože ukazatel je před prvním průchodem cyklu automaticky nastaven na první prvek pole.

Uveďme příklad:

```
/* u cyklu foreach je v každém průchodu cyklem hodnota aktuálního prvku $a přiřazena do proměnné $v a index pole je zvýšen o 1 */
```

```
$a = array (1, 5, 6, 20);
foreach ($a as $v) {
  print "Současná hodnota z \$a: $v.\n";
}
```

Výsledkem bude následující vygenerovaný HTML kód (v souboru byly navíc pouze doplněny HTML značky, mezi něž byl skript vložen):

```
<HTML>
<BODY>
Současná hodnota z $a: 1.
Současná hodnota z $a: 5.
Současná hodnota z $a: 6.
Současná hodnota z $a: 20.
</BODY>
</HTML>
```

Pozn.: Pro násilné přerušení vykonávání cyklů (for, while a switch) se používá příkaz break. Příkaz continue slouží k přeskočení zbývajících příkazů cyklu a pro zahájení další iterace.

Funkce

Pomocí funkcí můžeme dosáhnout opakování určité části programu (stejné sekvence příkazů), aniž bychom museli danou část v kódu znovu opisovat.

Například následující funkce má dva vstupní parametry ($a a $b) a jejím výsledkem bude vrácení hodnoty většího z parametrů.

```
function Max ($a,$b)                //funkce Max se vstupními parametry $a a $b
{
  return $a>$b ? $a : $b;  //vrácení výsledku funkce Max
}
```

Pozn.: V tomto příkladu jsme použili podmíněný výraz, tzv. ternární operátor, který lze slovně interpretovat takto: když $a je větší než $b, bude vrácena hodnota $a, jinak $b.

Volání a výpis výsledku funkce ve skriptu může vypadat například takto:

```
echo  Max(10,12);  // vypíše se 12
```

Platnost proměnných

Pokud budeme pracovat s funkcemi, postupem času určitě narazíme na otázku rozsahu platnosti proměnných a na požadavek, jak zpřístupnit globální proměnnou uvnitř funkce. Pro tento případ se v PHP používá příkaz global, kterým danou proměnnou ve funkci zpřístupníme. Zde se PHP podstatně liší od jazyka C, ve kterém jsou globální proměnné automaticky přístupné uvnitř funkcí.

Příklad funkce napsané v PHP, ve které změníme hodnotu globální proměnné:

```
function JmenoF()
{
  global $a1;                //zpřístupnění proměnné $a1 pomocí příkazu global
  $a1++;            //post-inkrementování
}

$a1=100;                    // proměnné $a1 přiřazena hodnota 100
JmenoF();                           // volání funkce JmenoF
Print $a1;                          //výpis $a1 ==101
```

Závěr

V úvodním díle našeho rychlokurzu PHP jsme se seznámili s úvodními znalostmi nutnými pro další tvorbu dynamicky generovaných WWW stránek pomocí oblíbeného skriptovacího jazyka – PHP. V příštím díle postoupíme opět o krůček dále – povíme si, jak na webové formuláře.

[Milan Pinte I pinte@atlas.cz]

Příklad	Výsledek
$a == $b	True když $a je rovno $b
$a ===$b	True když $a je rovno $b a jsou stejného typu
$a != $b	True když $a není rovno $b
$a !== $b	True když $a není rovno $b a nejsou stejného typu
$a < $b	True když $a je menší než $b
$a > $b	True když $a je větší než $b
$a <= $b	True když $a je menší nebo rovno $b
$a >= $b	True když $a je větší nebo rovno $b

Tab. 1. Tabulka relačních operátorů

Příklad	Výsledek
$a + $b	Součet $a a $b
$a - $b	Rozdíl $a a $b
$a * $b	Součin $a a $b
$a / $b	Dělení $a a $b
$a % $b	Zbytek po dělení $a a $b

Tab. 2. Tabulka aritmetických operátorů

Příklad	Výsledek
$a and $b	True když jak $a tak $b jsou true
$a or $b	True když buï $a nebo $b je true
$a xor $b	True když buï $a nebo $b je true, ale ne oba
!$a	True když $a není true
$a && $b	True když jak $a tak $b jsou true
$a || $b	True když buď $a nebo $b je true

Tab. 3. Tabulka logických operátorů

infotipy:

PHP.CZ
www.php.cz

PHPBuilder.com
www.phpbuilder.com

Zend / Where PHP meets eBusiness
www.zend.com

PHP: php and mysql examples and resources from WeberDev.com
www.weberdev.com

Literaratura:

[1] Kosek, J.: PHP – tvorba interaktivních internetových aplikací Podrobný průvodce, 1. vydání, Grada Publishing, Praha 1999, 492 stran.

[2] PHP manuál – www.php.net/manual/cs