Co nového přináší PHP4

Jiří Kosek

Co nového přináší PHP4

Jiří Kosek

Obsah

Seznam obrázků

Seznam příkladů

Tiráž

Tento dokument je dostupný ve dvou verzích:

- jako sada HTML stránek na adrese http://www.kosek.cz/clanky/php4/;
- jako PDF soubor vhodný pro tisk (http://www.kosek.cz/clanky/php4/php4.pdf).

Při přípravě dokumentu byl použit systém DocBook (http://www.docbook.org) a styly od Normana Walshe (http://www.nwalsh.com).

Poznámka: Dokument zatím není zdaleka kompletní, až budu mít čas, rozšířím ho.

Úvod

Necelé dva roky po uvedení PHP3 v létě 1998 je na světě další verze stále populárnějšího skriptovacího jazyka PHP. Na první pohled se PHP4 od předchozí verze neliší, největší změny byly provedeny v samotném jádře celého systému. Z toho plyne jeden velice důležitý a praktický důsledek — skripty napsané pro PHP3 budou v naprosté většině případů fungovat i v PHP4.

Interpret jazyka používaného v PHP byl kompletně přepsán a nese nyní jméno Zend (http://www.zend.com). Důvodů proč jádro přepsat bylo několik. Jedním z hlavních však byla snaha zvýšit rychlost provádění hodně složitých skriptů. Autoři Zendu (Zeev Suraski a Andi Gutmans) se proto rozhodli použít zcela jiný přístup. PHP3 bylo klasický interpret, který postupně četl zdrojový kód skriptu a prováděl ho. Zend nejprve načte celý skript, zkompiluje jej a poté spustí. U složitějších skriptů dojde k několikanásobnému zvýšení rychlosti jejich běhu, jednoduché skripty se provádí zhruba stejně rychle jako ve verzi 3.0.

Zatímco v PHP3 byl samotný interpret jazyka velice úzce provázán s jednotlivými funkcemi a s webovým serverem, resp. rozhraním CGI, PHP4 je vystaveno mnohem více modulárně. Interpret je realizován nezávislým modulem Zend. Díky tomu může být jazyk PHP použit i jinde než jen v PHP4. Uvažuje se o použití Zendu při implementaci uložených procedur v populární databázi MySQL.

Samotné jádro Zendu lze rozšiřovat o další moduly jako debugger, optimalizátor kódu apod. Výrazným způsobem se změnila i správa paměti. Podobně jako v Javě jsou počítány reference na každý objekt uložený v paměti. Pokud všechny reference zaniknou — například při skončení funkce, při přiřazení nové hodnoty do proměnné apod. —, je paměť alokovaná pro objekt automaticky uvolněna. Nestane se proto, že by větší skripty, které pracují s mnoha proměnnými a objekty, měly zbytečně velké paměťové nároky.

Aby šlo PHP4 snáze integrovat s různými webovými servery, obsahuje pro komunikaci s nimi nové rozhraní (SAPI). Přidání podpory pro nový server je pak mnohem jednodušší. Dnes existují rozhraní, které PHP4 umožňují provozovat například jako modul serveru Apache, jako ISAPI modul v serveru IIS, jako Java servlet, …. K dispozici jsou i další možnosti včetně klasického rozhraní CGI, se kterým si poradí snad každý webový server.

Obrázek 1. Architektura PHP4

V následujících odstavcích se podíváme, na změny a novinky, které se nám určitě budou hodit.

Kapitola 1. Drobná vylepšení jazyka

Ačkoli bylo PHP3 velice šikovný a kompaktní skriptový jazyk, některé věci v něm nebyly úplně dotažené nebo chyběly vůbec. PHP4 přináší několik nových vylepšení a odstraňuje některé nepříjemné rysy předchozí verze.

Odkazy na proměnné uvnitř textových řetězců

V textových řetězcích, které jsou uzavřené do uvozovek, můžeme používat proměnné. V PHP3 proto nebyl problém odvolat se na hodnotu proměnné. Například skript

```
$pozdrav = "Ahoj";
echo "$pozdrav Karle";
```

zcela podle očekávání vypsal text „Ahoj Karle“. Tímto způsobem jsme se v řetězcích mohli odvolávat pouze na obyčejné proměnné nebo na jednorozměrná pole. PHP4 umožňuje v řetězcích používat zápis `{$`*`proměnná`*`}`, kde *`proměnná`* může být klidně vícerozměrné pole.

```
$pole[20]['CZ'] = "Česká republika";
echo "{$pole[20]['CZ']}";               // vypíše text Česká republika
```

„Here-Doc“ syntaxe pro zápis řetězců

Pokud uvnitř skriptu potřebujete vypsat delší kus textu nebo nějaký dlouhý text přiřadit do proměnné, můžete použít nový druh zápisu, který možná znáte z Perlu.

```
echo <<<EOT
Nějaký skutečně dlouhý text, který nechcete otrocky uzavírat
do uvozovek (") nebo apostrofů ('). Můžete se samozřejmě odvolávat
na proměnné. Např. hodnota proměnné \$x je $x. Škoda, že ta proměnná
je teď prázdná. Můžeme používat i escape sekvence jako \n apod.
EOT;
```

Jednoduše řešeno, řetězec můžeme uzavřít mezi sekvenci znaků `<<<`*`identifikátor`* a *`identifikátor`*. Oba dva výrazy přitom musí být na samostatném řádku. Místo `EOT` z naší ukázky, můžete použít libovolný řetězec, který se nevyskytuje v textu.

Nový datový typ boolean

PHP4 obsahuje pro logické hodnoty pravda/nepravda separátní datový typ. K dispozici máme konstanty `true` a `false`, jejichž názvy nejsou citlivé na velikost písmen. Můžeme proto klidně používat i `TRUE`, `True` nebo `FaLSe`.

Pokud nějakým relačním operátorem porovnáváme dvě hodnoty, a jedna z nich je typu boolean, je na tento typ konvertován i druhý výraz. To v praxi znamená, že například podmínka `10 == true` bude pravdivá, protože se hodnota 10 převede na typ boolean.

> **Varování**
>
> V PHP3 bylo `true` pouze konstantou s hodnotou jedna a proto výše uvedená podmínka neplatila. Chápala se jako `10 == 1`.
>
> Tato nová vlastnost PHP4 je jedním z mála možných zdrojů nekompatibility se skripty napsanými pro verzi PHP3.

Reference na proměnné

Pomocí referencí si můžeme jednu proměnnou pojmenovat více názvy. Reference na proměnnou se získá zapsáním znaku `&' před její název. Malá ukázka:

```
$a = 10;
$b = &$a;       // $b ukazuje na stejnou hodnotu $a
$b = 20;
echo $a;        // vypíše 20
```

Praktické uplatnění naleznou reference při práci s polem. Referenci totiž můžeme vytvořit i na prvek pole. Pokud jej potřebujeme použít v několika výrazech za sebou, je mnohem pohodlnější vytvořit si na něj referenci, než pořád dokola opisovat název pole a příslušný index. Navíc je to rychlejší, protože se nemusí opakovaně podle indexu prvku pole hledat jeho skutečné umístění v paměti.

```
$pole = array(...);
for ($i=0; $i<Count($pole); $i++)
{
  $x = &$pole[$i];      // místo $pole[i] stačí nyní psát $x
}
```

Příkaz foreach

Při práci s poli je velice častou operací průchod celého pole a jeho zpracování. V PHP3 šlo průchod polem realizovat několika různými způsoby, které byly více či méně elegantní. V PHP4 tyto problémy odpadají — k dispozici je nový příkaz `foreach`, který slouží k postupnému zpracování všech prvků pole. Pro vypsání všech prvků pole `$pole` můžeme použít následující kód:

```
$pole = array(...);
foreach ($pole as $hodnota)
{
  echo $hodnota;
}
```

Můžeme využít i alternativní syntaxi, kterou známe z ostatních příkazů pro větvení a cykly.

```
$pole = array(...);
foreach ($pole as $hodnota):
  echo $hodnota;
endforeach;
```

Jednotlivé prvky pole jsou předávány jako hodnota a ne jako reference. Pokud chceme při průchodu polem jednotlivé prvky pole modifikovat, musíme znát i index jednotlivých prvků. K tomu můžeme využít drobně modifikovanou podobu příkazu `foreach`.

```
$pole = array(...);
foreach ($pole as $index => $hodnota)
{
  echo $hodnota;                // vytiskneme obsah prvku
  $pole[$index] = 100;          // do prvku pole uložíme hodnotu 100
}
```

Příkaz `foreach` prochází pole jen přes jeden rozměr. Pokud chceme zpracovat všechny prvky vícerozměrného pole, musíme do sebe `foreach` několikrát vnořit.

Kombinování polí a objektů

Parser v PHP3 měl značná omezení, co se týkalo kombinování polí a objektů. Mít pole objektů a přistupovat k polím v těchto objektech nebylo možné. PHP4 toto nepříjemné omezení odstraňuje a dovolí vám v libovolné míře do sebe zanořovat pole a objekty.

```
<?

class CCislo
{
  var $N = 0;

  function CCislo($n)
  {
    $this->N = $n;
  }
}

class CPokus
{
  var $x = array();

  function CPokus()
  {
    $this->x[0] = new CCislo(10);
    $this->x[1] = new CCislo(20);
    $this->x[2] = new CCislo(30);
  }
}

$y[0] = new CPokus();

echo $y[0]->x[0]->N;

?>
```

Tento skript byste si v PHP3 nespustili. Dostali byste místo něj hlášení, které každý zná, ale radost z něj rozhodně nikdo nikdy nemá.

```
Parse
error:  parse error, expecting `','' or `';'' in skript.php on line 27
```

Kapitola 2. Session proměnné

Bez session proměnných se neobejdeme v žádné větší aplikaci. Pomocí session proměnných můžeme odlišit jednotlivé uživatele, kteří s aplikací pracují. Každý návštěvník virtuálního obchodu musí mít vlastní nákupní košík, do kterého si ukládá zboží. Nákupní košík má přitom každý uživatel připojený k aplikaci — v tomto případě je nákupní košík právě session proměnnou.

Session proměnné jsou jedním z nejpohodlnějších způsobů, jak obejít bezstavovost protokolu HTTP. V protokolu HTTP jsou jednotlivé požadavky klientů zcela nezávislé a autonomní operace. Webový server proto neví, které požadavky přicházejí od jednoho uživatele a nemůže je proto předat dál ani PHP. Pokud vás napadne, že uživatele lze identifikovat pomocí IP adresy, tak vás zklamu. Mnoho firem a menších sítí je do Internetu připojeno přes proxy server a tváří se proto, že mají jednu společnou IP adresu.

> **Poznámka:** *Malá historka na oživení:* Tuto vlastnost si neuvědomila jedna nejmenová česká firma poskytující e-mail zdarma, a tak když se jeden ze zaměstnanců firmy přihlásil ke své poštovní schránce přes webové rozhraní, viděly jeho poštu i všichni ostatní ze stejné firmy, se stejným proxy serverem.

Fint, jak jednotlivé uživatele identifikovat, je několik. Nejpoužívanější je metoda, kdy si webový server, resp. aplikace označí každého uživatele jedinečným identifikátorem (třeba nějakým dlouhým číslem). Identifikátor se pak předává společně s každým požadavkem uživatele. Nejjednodušší je proto pro předávání identifikátoru využít cookies. Ne každý prohlížeč však cookies podporuje — s tím bychom měli počítat. V takových případech můžeme identifikátor předávat jako parametr v URL nebo skryté pole formuláře. To vyžaduje, abychom identifikátor přidávali za každý odkaz a do každého formuláře — je to dost pracné.

Pokud máme uživatele identifikovaného, máme vyhráno. Na serveru si můžeme vyhradit prostor — v paměti, na disku nebo v databázi, kam budeme pro každý identifikátor (tedy uživatele) ukládat proměnné. A session proměnné jsou na světě.

PHP4 obsahuje mechanismus, který umí uživatelům přidělovat jednoznačné identifikátory a umí označit vybrané proměnné jako session proměnné. Session proměnné se přitom mohou ukládat do sdílené paměti nebo do souborů. Pokud vám to nestačí, můžete si nadefinovat vlastní funkce pro ukládání a čtení session proměnných — můžete je pak ukládat třeba do databáze.

První kroky

Použití session proměnných je v PHP velice jednoduché. Musíme si však v konfiguračním souboru `php.ini` zkontrolovat, zda máme vše správně nastaveno. Standardní nastavení většině uživatelů vyhoví, musíme však zkontrolovat, zda je parametr `session.save_path` nastaven na nějaký existující adresář, do kterého má webová aplikace práva zápisu. Ukládají se do něj soubory se session proměnnými jednotlivých uživatelů.

Pokud chceme na stránkách používat session proměnné, měli bychom na začátku stránky použít funkci `session_start()`. Ta nejprve zkontroluje, zda už má uživatel přidělen identifikátor. Pokud ne, přidělí mu ho. Pro existující identifikátor načte všechny existují session proměnné a zpřístupní je jako běžné proměnné skriptu.

Pokud chceme z nějaké proměnné udělat session proměnnou, poslouží nám k tomu funkce `session_register()`. Jako parametr se předává název proměnné (ne samotná proměnná).

Následující jednoduchý příklad ukazuje, jak můžeme na jedné stránce session proměnnou zaregistrovat a na druhé použít její hodnotu.

Příklad 2-1. Zaregistrování session proměnné — `session_prvni.php`

```
<?
session_start();
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head><title>První stránka se session proměnnou</title></head>
<body>
<?
    session_register("x");
    $x = 10;
?>
Proměnná x je zaregistrována a má hodnotu <?echo $x?>.
Podívejte se na <a href="session-dalsi.php">další stránku</a>,
kde uvidíte, zda zůstane obsah proměnné $x zachován.
</body>
</html>
```

Příklad 2-2. Využití session proměnné — `session_dalsi.php`

```
<?
session_start();
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>Testovací stránka</title>
</head>
<body>
Proměnná x má nyní hodnotu <?echo $x?>.
</body>
</html>
```

A co když někdo nemá rád koláčky

Pokud má někdo vypnuté cookies, nebude mu bohužel výše zmíněný příklad fungovat. Identifikátor musíme předávat pomocí parametrů v odkazech. K dispozici máme naštěstí konstantu `SID`, která obsahuje kompletní nastavení parametru (ve tvaru `PHPSESSID=`*`identifikátor`*). Náš příklad proto musíme upravit tak, aby se identifikátor předával v URL.

Příklad 2-3. Zaregistrování session proměnné — `session_prvni.php`

```
<?
session_start();
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
```

```
<head><title>První stránka se session proměnnou</title></head>
<body>
<?
    session_register("x");
    $x = 10;
?>
Proměnná x je zaregistrována a má hodnotu <?echo $x?>.
Podívejte se na <a href="session-dalsi.php?<?echo SID?>">další stránku</a>,
kde uvidíte, zda zůstane obsah proměnné $x zachován.
</body>
</html>
```

Pokud session proměnné využíváme opravdu intenzivně, může být neustále doplňování odkazů o identifikátor session zdlouhavé a otravné. PHP4 umí za nás tuto práci udělat samo — stačí při kompilaci aktivovat volbu `--enable-trans-id`. Všechna URL v generovaných stránkách pak budou podle potřeby automaticky doplněna o identifikátor session. Tato volba však sníží výkon celého systému, protože se musí všechny stránky prohledávat na výskyt odkazů.

Kapitola 3. Integrace s dalšími prostředími

Informační technologie se vyvíjejí nebývalou rychlostí, vznikají neustále nové jazyky, protokoly atd. Na druhou stranu lze tyto i velice různorodé technologie čím dál, tím lépe vzájemně kombinovat a propojovat. Ani PHP se tomuto trendu nevyhnulo, a tak ve své čtvrté verzi umožňuje přímo využívat COM objekty a javové třídy. V budoucnu možná ještě přibude podpora Corby.

COM

Rozhraní COM je dobře známé všem programátorům, kteří se pohybují ve světe Windows. COM umožňuje vytvářet binární komponenty, které lze využívat v mnoha různých jazycích právě díky tomu, že používají jednotné rozhraní. Komponenty jsou však binární, takže je lze provozovat pouze na platformě Win32 (Windows 95/98/NT/2000).

Kromě toho, že si každý může vytvářet komponenty jaké chce, je mnoho komponent standardní součástí Windows, případně webového serveru IIS. Většina aplikací jako třeba Word a Excel nabízí pomocí COM rozhraní většinu svých funkcí ostatním aplikacím. Na využití COM objektů je postaven i vývoj webových aplikací v ASP (Active Server Pages). Vzhledem k tomu, že v ASP se standardně používají velice „chudé“ jazyky jako JScript nebo VBScript, je potřeba i na tak triviální věci jako je přístup k databázi nebo odeslání e-mailu volat speciální komponenty.

Pokud PHP provozujete pod Windows můžete si ve skriptech vytvářet instance jednotlivých komponent a volat jejich metody a vlastnosti. Tuto možnost jste v omezené míře měli již v PHP3 pomocí funkcí jako `COM_Load()`, `COM_Invoke()` apod. Kdo s těmito funkcemi někdy pracoval však potvrdí, že mnoho věcí nefungovalo a občas to spadlo.

PHP4 je díky novému jádru schopno pracovat s COM objekty zcela stejně jako se svými vlastními objekty. Po vytvoření COM objektu můžeme pomocí běžné notace (`->`) volat jednotlivé metody a přistupovat k vlastnostem.

Instance COM objektu se vytváří velice jednoduše. Stačí znát identifikátor objetu:

```
$objekt = new COM("ProgId");
```

Na následujícím příkladě je vidět, jak lze z PHP spustit o ovládat MS Excel. Moc užitečná aplikace to není, ale je vidět, co všechno lze pomocí COM udělat.

```
<?
// Vytvoříme si instanci Excelu
$excel = new COM("Excel.Application");

// Excel má být vidět
$excel->Visible = true;

// Varování se nebudou zobrazovat
$excel->DisplayAlerts = false;

// Otevřeme si nový sešit
$excel->WorkBooks->Add();
```

```
// Do aktuální buňky vložíme aktuální čas
$excel->ActiveCell->Value = "Aktuální čas: " . Date("H:i:s");

// Chvilku počkáme, aby si to všichni prohlédli
Sleep(3);

// Excel ukončíme
$excel->Quit();
?>
```

Pěkná hračka, že. COM objekty samozřejmě většina z nás asi ve spojení s PHP používat nebude, protože je nelze využívat v unixových systémech. Nicméně se podpora COM může hodit. Pokud potřebujeme napsat webové rozhraní k existující aplikaci, která má COM rozhraní, může pro nás být pohodlnější využít k tomu PHP než ASP.

Java

Java se stává stále populárnějším a používanějším jazykem. Není proto divu, že i PHP nyní nabízí možnost využívání kódu, který je napsán v Javě. Díky novému jádru PHP, si lze přímo v PHP skriptu vytvořit instanci javové třídy. Další práce s ní je pak stejná, jako kdyby se jednalo o třídu napsanou v PHP.

Následující ukázka ilustruje použití Javy pro vypsání aktuálního údaje o datu a čase ve skriptu.

```
$formatter = new Java("java.text.SimpleDateFormat",
                        "EEEE, MMMM dd, yyyy 'at' h:mm:ss a zzzz");

echo $formatter->format(new Java("java.util.Date"));
```

Pro vypisování časových údajů nám samozřejmě nabízí mnohem efektivnější nástroje přímo PHP. Možnost využívat javové třídy se však může hodit v mnoha jiných situacích. Některé funkce nemusí být v PHP k dispozici, nebo například potřebujeme přistupovat k funkcím podnikového informačního systému, který je napsaný v Javě.

Někomu může připadat, že spouštění javového kódu z PHP skriptů bude příliš pomalé. Ale není to pravda. JVM (Java Virtual Machine), která se stará o spouštění javového byte-code, se zavádí pouze při prvním požadavku na vytvoření instance javové třídy. Pak už je stále v paměti a opakované spouštění skriptu už není zpomalováno poměrně dlouhým startem JVM.

Kapitola 4. PEAR nejsou jen hrušky

Kapitola 5. Moduly