# PHP v praxi - 1. díl

(05.06. 2001)

Možná se zeptáte, proč je zde další seriál o PHP, když už jich lze najít na Internetu hned několik. Důvodem je to, že většinou se jedná o opis manuálu, tzn. autor si stáhne z http://www.php.net/ manuál, umí anglicky a pokusí se manuál interpretovat. Ale podle toho se začátečník může něco naučit jen velmi těžko. Cílovou skupinou tohoto seriálu jsou tvůrci stránek, kteří si chtějí práci ulehčit a ne ztížit, lidé, kteří nemusí vědět nic o programování, stačí jim základní znalost html.

PHP je scriptovací jazyk (podobně jako JavaScript), ale třeba právě od JavaScriptu se nezpracovává u vás v prohlížeči, nýbrž na serveru, kde máte stránky uloženy. Výhodou je, že ať už má člověk jakýkoliv prohlížeč, vždycky bude výsledek stejný. S JavaScriptem je věčný problém - stránka se zobrazuje u každého jinak a člověk nemá jistotu shodného výsledku. Nevýhodou je menší interaktivita, ke zpracování dojde vždy až po odeslání požadavku na server.

Abyste mohli začít vytvářet stránky s php, musíte také někde vidět výsledek. Pro začátek vám může stačit některý z free hostingů s podporou php. (např. http://www.kgb.cz/ nebo http://www.yo.cz/ a spousta dalších). Je však důležité vědět, kterou verzi php mají - momentálně se můžete setkat s verzemi PHP3 a PHP4, přičemž vřele doporučuji verzi novější (však také většina serverů na čtvrtou verzi již přešla či se chystá v nejbližší době). Na určitém stupni znalostí ale bude nutné nainstalovat si server doma, přece jen je pak práce trošku rychlejší. K tomu se vrátíme někdy později, dovíte se, jak nainstalovat asi nejpoužívanější kombinaci - server Apache a PHP modul.

Co byste měli vědět, než začnete psát: každý soubor, ve kterém je uložen nějaký PHP příkaz, musí mít příponu - pro třetí verzi nazev_souboru**.php3** a pro čtvrtou stačí nazev_souboru**.php**. Proto je důležité znát verzi, která na serveru běží. Samotný kód je ohraničený například značkami **<?** tady je nějaký příkaz **?>**. Nenechte se vyvést z míry, když uvidíte **<?php**, je to stále to samé. A za poslední - každý příkaz končí **středníkem**.

Nejprve si ukážeme příklad nejjednodušší, abyste viděli základ, pak se již podíváme přímo na funkci, která vám usnadní život a několikanásobně vrátí čas vložený do přečtení tohoto článku. Jsme tedy již ve stadiu, že máte někde možnost umístit vaše první stránky v PHP, takže si v poznámkovém bloku (nebo jiném jednoduchém editoru) otevřete nový dokument a zkopírujte tam následující:

```
<?
echo 'jsem nejlepší na světě';
?>
```

a pojmenujte jej jako index.php. Po odeslání na server a zadání vaší adresy se ukáže, kdo vlastně jste. Krátký rozbor: příkaz je vložen mezi <? a ?>, čímž se server dozvěděl, co má zpracovat, příkaz echo slouží k vypsání textu na obrazovku. Text se nachází mezi uvozovkami - a tady poznámka: zásadně pište v php uvozovky jednoduché, složené nechte html, vyhnete se tak spoustě problémů. Příkaz končí středníkem, server se dozví, že příkaz už dále nepokračuje. Zapomenete-li ho, v tomto jednoduchém případě se nic nestane, ale jindy místo hlášky pro zvednutí sebevědomí obdržíte např.: Parse error: parse error, expecting `',' or `';' in

http://vase_domena/index.php on line 3. To se vám stane mockrát, neděste se toho, jedná se obvykle o takhle triviální chyby.
A protože se člověk nejvíce naučí z praxe a ne z manuálu, podíváme se rovnou na jednoduchý příklad: jistě už máte nějaké stránky vytvořené. Předpokládám, že jako správní webmasteři nepoužíváte rámy (frames), ale kvůli pěknému vzhledu vašich stránek pěkně tabulky. Těch stránek je několik a tady přichází problém. co když potřebuji třeba přidat odkaz, který je vidět na každé z nich. Pro každého je takováto editace časově náročná, zvláště, když přichází třeba každý týden. A přesně na to je tu PHP.
Podívejte se na stránku, na které se právě nalézáte - z čeho se skládá? Nad článkem je tabulka, která začíná *Grafika publishing: Grafika...* následuje logo WebTipu a reklama a pod tím proužek *HTML Animace...* Tuto část si nazvěme horní tabulka. Pod ní jsou dva sloupce - samotný text a pravý okraj - *Top články, Aktuality...* a pod tímhle vším ještě dolní tabulka - copyright a pár reklam. Co z těchto částí se mění? Pouze samotný text, a» už se podíváte na kterýkoli jiný článek WebTipu, okolí zůstává neměnné. Co do komplexnosti je asi WebTip poněkud rozsáhlejší než vaše stránky, ale i kdyby rozsah vašeho webu měl jen pět stránek v html, představte si, jak měníte třeba ten copyright - znamená to zásah do každé jednotlivé stránky. Samozřejmě, že na to můžete využít program, který najde konkrétní změny a opraví je, ale mnohem jednodušší je následující řešení: funkce **readfile()**.
Máme tedy třeba pět stránek, kterým se opakuje horní část - odkazy. Chystají se postupné změny a vidíme, že by bylo neúnosné, zdržovat se s každým z těchto souborů zvlášť». Začneme tím, že neustále se opakující část zkopírujeme do jiného souboru - např. hlavicka.php a upravíme odkazy. V tomto souboru máme:

```
<html>

<head>
<title>Název stránek</title>
</head>

<body>

<table width="100%" border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td><a href="diskuze.php <!-- původně diskuze.html-->">DISKUZE</a></td>
    <td><a href="fotky.php">FOTKY</a></td>
    <td><a href="udalosti.php">UDÁLOSTI</a></td>
    <td><a href="bazar.php">BAZAR</a></td>
    <td><a href="kontakty.php">KONTAKTY</a></td>
  </tr>
</table>
```

Normálně tedy následuje text, pak tagy </body> a </html>. Těch pět souborů nyní tedy musíme zeditovat následovně - přejmenovat je na *nazev_souboru.php* a vymazat část textu, která je nyní v *hlavicka.php*. Místo ní dáme na začátek každého z těchto souborů následující příkaz:

```
<?
readfile ('hlavicka.php');
?>
```

Co se teď stane? Po vyžádání souboru index.php server zpracuje příkaz mezi otazníky způsobem - přečti soubor *hlavicka.php* a dosaď ho tam, kde jsem. Výsledek bude vypadat úplně stejně jako před tím, akorát názvy souborů mají jinou příponu. Ve zdroji html není vidět žádný rozdíl. Ale přidání jednoho odkazu nyní znamená pouze jedinou editaci - přidání řádku do souboru hlavicka.php. O zbytek se už postará samo PHP.

Detaily k funkci readfile () - mezi závorkami se nachází cesta k souboru, pokud je pouze naše *hlavicka.php*, pak musí být tento soubor ve stejném adresáři jako ostatní, které k němu přistupují. Pokud je cesta jiná, zadává se *http://nazev_domeny/hlavicka.php*. Nezapomeňte na uvozovky a středník. Vyzkoušejte a uvidíte, za chvíli budete odmítat dělat stránky na serverech bez PHP. Lenost je hlavní příčinou lidského pokroku.

# PHP v praxi - 2. díl

(11.06. 2001)

Dnes se podíváme na jednoduché zaheslování souboru s důvěrnými daty. Toto řešení není dokonalé, ale pro běžné potřeby, např. chcete známým sdělit místo setkání, je toto řešení zcela dostatečné. Představa je taková, že při první návštěvě stránky se nejprve zobrazí jednoduchý formulář požadující heslo. Po jeho zadání se načte stránka s dotyčnou informací. Kvůli bezpečnosti by vše mělo být obsaženo v jediné stránce, pokud povede odkaz někam mimo, cílový soubor zůstane pochopitelně nezaheslovaný. Proto třeba pro fotky je toto řešení nevhodné (tady by bylo nutné pojmenovat obrázek třeba 24sfddf5f.jpg a na zaheslovanou stránku dát odkaz).

Nyní se rovnou podívejte na zdrojový text příkladu. Pro někoho, kdo nikdy neviděl php, bude hodně složitý, ale postupně shledáte, že nezvládnutelné to rozhodně není.

soubor sraz.php:

```
<html>
<head><title>Kdy a kde se tedy sejdeme.</title></head>
<body>
<?
    if ($heslo=='ledoborec'){
      echo 'sejdeme se ve čtvrtek  v kavárně';
      } else {
?>
  <form action="sraz.php" method="get">
      Heslo:<br>
      <input type="password" name="heslo"><br>
      <input type="submit" value="odeslat">
  </form>
<?
      }
?>
</body>
</html>
```

Krása php spočívá také v tom, že lze velmi jednoduše kombinovat prvky skriptovacího jazyka a html. Pro server je tímto znamením právě značka "<?" na začátku a "?>" na konci php příkazu, dál může pokračovat opět klasické html a poté zase vstup do php.
Nyní rozebereme text po částech. Html tagy snad komentář nepotřebují, vrhněme se rovně na php. Hlavní konstrukcí je v našem případě podmínka "if ()". Protože předpokládám pouze minimální znalosti v programování, podíváme se na ni podrobněji. V tom úplně nejzákladnějším tvaru vypadá takto:

if (podmínka)

proveď příkaz

```
if ($prijem > $vydaj)
    echo 'firma je zisková';
```

V tomto jednoduchém příkladu porovnáváme dva údaje, když je podmínka splněna (tedy příjmy převýšily výdaje), na obrazovku se vypíše text mezi uvozovkami. Pokud ovšem chceme použít příkazů více, musíme je uzavřít do složených závorek:

```
if ($prijem > $vydaj) {
    echo 'firma je zisková';
    echo 'ale neradujme se předčasně';
    }
```

Ale tímto způsobem získáme výsledek pouze tehdy, je-li hodnota v závorce za "if" pravdivá, tzn. jestliže příjmy přesáhly výdaje. V případě rovnosti či opačné nerovnosti nezískáme vůbec žádný výsledek, php bude tyto řádky ignorovat. Abychom získali nějakou reakci i v takovýchto případech, využijeme doplňující výrazy "else" (jinak) nebo "elseif" (jinak když).

```
if ($prijem > $vydaj) {
    echo 'firma je zisková';
    }  elseif ($prijem == $vydaj)  {
    echo 'vyrovnaný výsledek';
    } else {
    echo 'firma je proděličná';
    }
```

Jak vidíte, v případě, že příjmy nejsou větší než výdaje, php přeskočí příkaz "echo" s textem o zisku, postupuje dál a narazí na "elseif", což je kombinace jinak a když (to je ale překvapení). V závorce je opět podmínka - zde je to kontrola rovnosti (pozn: dvě rovnítka nejsou překlep). A na závěr je zde "else", tedy v případě nepravdivosti všech předchozích podmínek se provede příkaz obsažený ve složených závorkách.
Vraťme se tedy k našemu srazu, vybereme si nyní jen to důležité:

```
if ($heslo=='ledoborec'){
    echo 'sejdeme se ve čtvrtek  v kavárně';
    } else {
    // tady je jinak  formulář
    }
```

Už vám je jistě jasné, co se zde děje - když proměnná heslo je rovna textu "ledoborec", pak se zobrazí důvěrná informace. Když ne, na stránce se objeví formulář. K funkci "if ()" myslím už není co dodat, pokud přesto máte nějaké dotazy, napište je do diskusního fóra.
Velmi důležitou složkou php jsou proměnné. V manuálu je to vlastně první věc, na kterou narazíte, ale pro běžného tvůrce stránek zbytečně složitě popsaná. Jejím hlavním efektem bývá většinou zastrašení potenciálních php programátorů. Syntaxe proměnné je velmi jednoduchá - začíná dolárkem, po kterém musí následovat písmeno ("$a" je proměnná ale "$4a" nebo "$_a" už nemá smysl). Proměnnou si představte třeba jako zástupce obsahu. Uvedu příklad:

```
$jmeno='honza';

//zde došlo k přiřazení řetězce textu honza proměnné $jmeno,
jinak: pokud si vyžádám proměnnou $jmeno, dostanu honza

 echo $jmeno;

//na obrazovce se objeví honza, všimněte si,
že proměnná zde není v uvozovkách echo '$jmeno';
```

```
//vypíše na obrazovku $jmeno - proměnná se
kvůli uvozovkám nezpracuje.
```
S proměnnými se dá dělat spousta kouzel, ale to teď nechme stranou. Podstatné bylo pochopit, o co vlastně jde. Zdůraznil bych zde problém s rovnítky, v textu jsem na to upozornil - jedno rovnítko znamená přiřazení, teprve dvě rovnost. Kdybychom napsali "if ($heslo='ledoborec')", nedočkáme se ničeho kromě přiřazení textového řetězce "ledoborec" proměnné "$heslo".
Teď je na místě otázka, kde tedy máme dotyčnou proměnnou, kterou chceme porovnávat s textem "ledoborec". Proměnná vznikne automaticky při odeslání dat (resp. jejich přijmutí) z formuláře. V url adrese se objeví "http://neco/sraz.php?heslo=ledoborec". PHP zpracovává všechno za otazníkem jako proměnné, tzn. vytvoří proměnnou "$heslo" a přiřadí jí text "ledoborec". "if ()" tedy vyhodnotí podmínku jako pravdivou a dotyčný už bude vědět, kde se s námi setká.

# PHP v praxi - 3. díl

(16.07. 2001)

Návštěvnost stránky závisí jednoznačně na jejím obsahu - pokud je statický, málokdy dostane surfař chuť znovu stránku navštívit. Internetový magazín těží z průběžně přibývajících článků, které si ale každý samozřejmě dovolit nemůže. Z mého pohledu nejlepším oživením stránek je nějaká forma diskuze mezi čtenáři. Surfař bude zvědav, copak se zde objevilo nového a stránky navštíví, kdykoli bude připojen k Internetu.

V PHP se dá diskuze udělat poměrně snadno (tedy jednodušší varianta). Podíváme-li se na tento problém z technického hlediska, pak potřebujeme: zpracovat data, která uživatel vloží a odešle, uložit je do souboru a načíst. Mou prioritou vždy bylo omezit soubory na minimální počet, k diskuzi proto využijeme pouze dva - diskuze.php (html formulář a zpracování dat), note.txt (uložená data - vzkazy).

Nejprve se tedy podíváme na formulář.

```
<form action="diskuze.php" method="get">
<input type="text" name="jmeno" size="10">
<input type="text" name="zprava" size="60">
<input type="submit" value="Zapsat">
</form>
```
Jak vidíte, odkazuje sám na sebe (neboť se dále bude o data starat), method je po zkušenostech nastavená na get (někteří provideři nedovolují post), jinak je snad vše jasné.
Co se stane po odeslání? Kdo četl pozorně minulý díl, je už jistě v obraze - vytvoří se dvě proměnné, $zprava a $jmeno. S nimi tedy již můžeme pracovat. Pro diskuzi je vhodné formátování do tabulky, kde si můžeme také názorně ukázat, jak snadné je sloučení PHP s HTML.
```
$vzkaz='<tr><td>'.$jmeno.'</td><td>'.$zprava.'</td></tr>';
```
V proměnné $vzkaz máme nyní kompletní řádek tabulky pro zápis do souboru note.txt. Až si to budete zkoušet, nezapomeňte tomuto souboru na serveru změnit atributy -

musí mít nastavena práva na zápis pro celý svět. Tak a teď už zbývá jen dotyčný řetězec do souboru zapsat:

```
$otevrit = fopen ('note.txt','a');
fwrite ($otevrit, $vzkaz);
fclose ($otevrit);
```

Proměnná $otevrit po jejím vyžádání otevře soubor note.txt, atribut 'a' znamená 'otevři k zápsání dat a umísti kurzor na konec souboru'. Funkce fwrite() zapíše obsah proměnné $vzkaz do souboru otevřeného přes proměnnou $otevrit (data zapíše na konec tohoto souboru). Soubor otevřený obsahem proměnné $otevrit zavřeme funkcí fclose(). V tomto okamžiku již soubor note.txt obsahuje tu určitou řádku se vzkazem.
Nyní už nám stačí do souboru diskuze.php dosadit následující:

```
<table>
<tr><td>Jméno</td><td>Zpráva</td></tr>
<? readfile ('note.txt');  ?>
</table>
```

Již popsaná funkce readfile() nám přečte obsah note.txt.
Byl bych rád, kdybyste si sami zkusili diskuzi vytvořit. Příště se podíváme na její rozšíření a hlavně zdokonalení - tato diskuze totiž funguje správně pouze v případě, že nenastane žádná chyba (ať úmyslná či neúmyslná). Např. když někdo nenapíše do jednoho z polí formuláře nic a odešle, vznikne řádek s prázdnou buňkou. Také zde není kontrola obsahu vzkazu - každý tak může vložit do souboru jak formátování HTML, tak i nějakou nebezpečnou funkci PHP. Proto bych byl velmi rád, abyste si tento příklad vyzkoušeli a zjistili, jak opatrní musíte být.
Na závěr tedy obsah souboru diskuze.php (bez základních html tagů):

```
<?
$vzkaz='<tr><td>'.$jmeno.'</td><td>'.$zprava.'</td></tr>';
$otevrit = fopen ('note.txt','a');
fwrite ($otevrit, $vzkaz);
fclose ($otevrit);
?>

<form action="diskuze.php" method="get">
<input type="text" name="jmeno" size="10">
<input type="text" name="zprava" size="60">
<input type="submit" value="Zapsat">
</form>
<table>
<tr><td>Jméno</td><td>Zpráva</td></tr>
<? readfile ('note.txt');  ?>
</table>
```

# PHP v praxi - 4. díl

(30.07. 2001)

Doufám, že jste si poctivě od minula trénovali diskuzi a snad i tušíte, čemu se v tomto díle budeme věnovat. Bude to především kontrola vzkazu, aby nedošlo k chybě či nedovolenému formátování textu ze strany uživatele.

```
 if (!empty($zprava)&!empty($jmeno)){
   $vzkaz='<tr><td>'.$jmeno.'</td><td>'.$zprava.'</td></tr>';
   $otevrit = fopen ('note.txt','a');
   fwrite ($otevrit, $vzkaz);
   fclose ($otevrit);
   }
```

Tučně jsem zvýraznil pasáž, která od minula přibyla. Je to komplikovanější podmínka, kterou si důkladněji rozebereme. Syntax funkce if() doufám už z minula umíte. Tato funkce kontroluje, jestli proměnné $zprava a $jmeno existují a jestli nejsou

prázdné. Proč to? Když otevřete zpracovanou stránku diskuze.php v prohlížeči poprvé, žádné proměnné $zprava a $jmeno ještě neexistují - nebyly vytvořeny odesláním formuláře. Když budou prázdné, dostaneme prázdné buňky tabulky a to dělá neplechu. Kromě toho, nechceme dopustit zaneřádění diskuze tím, že se někdo přehlédne a odešle vzkaz bez jména či zprávy.
Funkce empty() je pravdivá, jestliže proměnná v závorce nebyla vytvořena, má nulovou nebo prázdnou hodnotu. Protože chceme opak (potřebujeme zjistit, jestli již vytvořena byla a nějakou hodnotu má), napíšeme před funkci vykřičník. Ten ji neguje, takže vrací pravdivou hodnotu právě v případě existující proměnné. Protože máme proměnné dvě a potřebujeme je mít funkční obě, vložíme mezi ně & (and), což znamená doslova a zároveň. Ve volném přepisu do češtiny bude tato podmínka znít:
Když proměnné $zprava a $jmeno existují a nemají nenulovou hodnotu, proveď příkaz ve složených závorkách.
Postarali jsme se, aby návštěvník nemohl nechat kolonku prázdnou, teď ještě, aby nemohl zadat jakékoli formátovací příkazy. K tomu slouží funkce strip_tags(), která odfiltruje znaky < a > a vše mezi nimi. Nejde o to, zabraňovat čtenáři zvýraznit si pasáž ve vzkazu, <b> tag nám asi příliš vadit nebude, ale nezapomeňte, že může klidně použít třeba </body></html> a máme konec stránky tam, kde jej rozhodně mít nechceme. A to jsme jen u html, kdyby dal <? ?> už může psát libovolný php příkaz. V této diskuzi to nic nezpůsobí (zkuste si to), protože na čtení vzkazu je zde použita funkce readfile(), která obsah souboru nezpracuje, pouze přečte.
Další celkem důležitou funkcí je stripslashes(), ta odfiltruje zpětná lomítka dodaná samotným php. Když totiž zadáte některé znaky v php užívané (třeba uvozovky), bez této funkce se zapíše \" místo čistého ". Uznáte sami, že to kráse textu příliš neprospívá.
Pro diskuzi je důležité také vědět, kdy dotyčný psal svoji připomínku. Zde využijeme funkci date(), jež má velmi mnoho možných atributů. Uvedu jen ty nejpoužitelnější, více najdete v manuálu.
j - den v měsíci bez nul; tj. od "1" do "31"
n - měsíc bez nul; tj. od "1" do "12"
Y - rok, 4 číslice; např.: "1999"
H - hodina, 24-hodinový formát; tj. od "00" do "23"
i - minuty; tj. od "00" do "59"
Následná proměnná pro získání data a času vypadá tedy následovně:
$datum=date('j.n.Y.H:i');
Jak vidíte, jednotlivé položky jsou od sebe odděleny tečkou nebo dvojtečkou. Záleží na naší potřebě formátování data a času. Tato proměnná zobrazí např.:
1.7.2001.10:56.
Soubor diskuze tedy po dnešním díle vypadá takto (tučně jsou zvýrazněny změny):

```
<?
$datum=date('j.n.Y.H:i');
if (!empty($zprava)&!empty($jmeno)){
  $zprava=strip_tags($zprava);
  $zprava=stripslashes($zprava);
  $jmeno=strip_tags($jmeno);
  $jmeno=stripslashes($jmeno);
  $vzkaz='<tr><td>'.$jmeno.'</td><td>'.$datum.'</td>
      <td>'.$zprava.'</td></tr>';
  $otevrit = fopen ('note.txt','a');
  fwrite ($otevrit, $vzkaz);
  fclose ($otevrit);
}
?>
```

```
<form action="diskuze.php" method="get">
<input type="text" name="jmeno" size="10">
<input type="text" name="zprava" size="60">
<input type="submit" value="Zapsat">
</form>
<table>
<tr><td>Jméno</td><td>Datum</td><td>Zpráva</td></tr>
<? readfile ('note.txt'); ?>
</table>
```

# PHP v praxi - 5. díl

(16.08. 2001)

Určitým nedostatkem naší diskuze je, že vzkaz je přidáván na konec souboru a tak je i zobrazen. Dokonalým řešením by bylo, kdyby se buď vložený vzkaz zapsal na začátek souboru note.txt nebo se tento soubor četl po řádkách odzadu. V prvním případě však narážíme na problém s funkcí fopen(). Připomenu:

```
$otevrit = fopen ('note.txt','a');
```

Problémem je chybějící atribut pro tento případ - umísti kurzor na začátek, vlož a zároveň odsuň text o vkládaný počet bytů. V manuálu se dozvíme:

```
'r' - Otevři jen ke čtení; kurzor umísti na začátek souboru.
'r+' - Otevři pro čtení i zápis; kurzor umísti na začátek souboru.
'w' - Otevři jen pro zápis; kurzor umísti na začátek souboru a
      změn velikost souboru na nulu. Pokud soubor neexistuje,
      pokus se ho vytvořit.
'w+' - Otevři pro čtení i zápis; kurzor umísti na začátek souboru a
      změn velikost souboru na nulu. Pokud soubor neexistuje,
      pokus se ho vytvořit.
'a' - Otevři jen pro zápis; kurzor umísti na konec souboru.
      Pokud soubor neexistuje, pokus se ho vytvořit.
'a+' - Otevři pro čtení i zápis;  kurzor umísti na konec souboru.
      Pokud soubor neexistuje, pokus se ho vytvořit.
```

Potíž je v tom, že když se kurzor umístí na začátek souboru a zapíše se náš vzkaz, přepíše se původní obsah souboru v délce vkládaného vzkazu. Druhá zmiňovaná možnost (čtení po řádkách odzadu) je pro nás zatím zbytečně komplikovaná. Jednodušší řešení je načíst obsah souboru do nějaké proměnné, vynulovat tento soubor a následně do něj vložit obsah s přidaným řádkem. Server to příliš nezatíží (naše diskuze také nebude mít megabajty) a výsledek bude dostačující. Podívejme se tedy na novou konstrukci:

```
$otevrit_ke_cteni = fopen ('note.txt', 'r');
$obsah = fread ($otevrit_ke_cteni, '1000000("note.txt")');

$vzkaz='<tr><td>'.$jmeno.'</td><td>'.$datum.'</td>
      <td>'.$zprava.'</td></tr>'.$obsah;

$otevrit_pro_zapis = fopen ('note.txt', 'w');
fwrite ($otevrit_pro_zapis, $vzkaz);
fclose ($otevrit_pro_zapis);
```

Předpokládám, že už tomuto zápisu z větší části rozumíte, přesto vysvětlím. Začněme od prvního příkazu, který se spustí - fwrite(jak, co). Jak má zapsat určí proměnná zde pro názornost pojmenovaná $otevrit_pro_zapis, jejíž obsah zní: otevři soubor note.txt jen pro zápis, umísti kurzor na začátek a tento soubor vynuluj. Co má zapsat určí zase proměnná $vzkaz, jejímž obsahem je již v minulých dílech zmíněná řádka tabulky, nyní však ještě doplněná o proměnnou $obsah. Ta v sobě ukrývá celý obsah souboru note.txt. K tomu se využila funkce fread(jak, velikost(souboru)) aneb přečti obsah souboru v určité délce. Jak ho má přečíst určí proměnná $otevrit_ke_cteni. Velikost je

udávána v bajtech, můžeme ji nastavit třeba na megabajt, naše diskuze tak velká stejně nebude. Upozornil bych na uvozovky v tomto příkazu - protože název souboru stejně jako jiné názvy musí být v uvozovkách, při použití jednoho druhu uvozovek by došlo ke křížení. Toho se vyvarujeme využití uvozovek dvojitých. Nakonec soubor zavřeme.
Abych řekl pravdu, tato konstrukce vypadá z programátorského hlediska velmi neohrabaně, otevíráme jeden soubor prakticky na jednom řádku dvěma způsoby, přečteme ho a zároveň anulujeme, nakonec zavřeme pouze jeden způsob otevření. PHP je ale velmi tolerantní, a tak nemusíme hodiny vymýšlet mnohem složitější konstrukci.
Jak se již stalo pravidlem, uzavřeme tento díl kompletním obsahem souboru diskuze.php s vyznačenými změnami:

```
<?
$datum=date('j.n.Y.H:i');
if (!empty($zprava)&!empty($jmeno)){
   $zprava=strip_tags($zprava);
   $zprava=stripslashes($zprava);
   $jmeno=strip_tags($jmeno);
   $jmeno=stripslashes($jmeno);
   $otevrit_ke_cteni = fopen ('note.txt', 'r');
   $obsah = fread ($otevrit_ke_cteni, '1000000("note.txt")');
   $vzkaz='<tr><td>'.$jmeno.'</td><td>'.$datum.'</td>
   <td>'.$zprava.'</td></tr>'.$obsah;
   $otevrit_pro_zapis = fopen ('note.txt', 'w');
   fwrite ($otevrit_pro_zapis, $vzkaz);
   fclose ($otevrit_pro_zapis);
}
?>

<form action="diskuze.php" method="get">
<input type="text" name="jmeno" size="10">
<input type="text" name="zprava" size="60">
<input type="submit" value="Zapsat">
</form>
<table>
<tr><td>Jméno</td><td>Datum</td><td>Zpráva</td></tr>
<? readfile ('note.txt'); ?>
</table>
```

# PHP v praxi, 6. díl

(07.09. 2001)

PHP v praxi, 6. díl

Nedávno jsem se opět pustil do přestavby vlastních stránek (jsou víceméně účelové pro moje webové snažení :). Snažil jsem se maximálně zjednodušit jakoukoliv změnu obsahu webu (např. přidávání nových stránek). Abych nechodil dlouho kolem horké kaše - stránka novinky.php vypadá následovně:

soubor novinky.php**:**

```
<?
include ('template_jave.php');
page_prolog ('Novinky');
ramecek ('<div class="nadpis"> NOVINKY </div>
```

```
        <ul>
        <li>18/09/01 ::?:: pár drobných změn.
        <li>09/08/01 ::?:: betaverze spuštěna.
        </ul>
        ','text','',1);
page_epilog ();
?>
```

To je opravdu celý zápis první stránky. Celkem slušně krátký zápis, který se pokusím co nejsrozumitelněji vysvětlit. Skládá se ze čtyř funkcí - include(), page_prolog(), ramecek(), a page_epilog(). Jedinou implicitně definovanou funkcí je funkce include(), ostatní jsou obsaženy v souboru template_jave.php. V manuálu se o include() dozvíme spousty věcí, ale zatím pouze v angličtině a je jich zbytečně moc. Jednoduše: na rozdíl od funkce readfile(), o které byla řeč již v prvním díle seriálu, funkce include() nejen přečte obsah souboru, ona ho i zpracuje, tzn. jsou-li zde další funkce, podmínky, proměnné atd. všechno se řeší a, je-li soubor dobře napsaný, i vyřeší. Teď asi čekáte i obsah souboru template_jave.php, ale je docela velký a pravděpodobně byste se v něm ztratili. Začneme rámečkem, jedná se o ta okénka na mých stránkách ve kterých je text, menu nebo reklama. Tady máte zdroják:
soubor template_jave.php (část):

```
...
function ramecek($obsah=' ',$class='text') {
?>
<tr>
 <td>
  <table width="100%" border="0" cellspacing="0" cellpadding="0">
   <tr>
    <td width="9" height="9"><img src="pics/ram_1_1.gif" width="9" height="9"></td>
    <td background="pics/ram_1_2.gif">
       <img src="pics/_transparent.gif" width="125" height="1">
    </td>
    <td width="9" height="9"><img src="pics/ram_1_3.gif" width="9" height="9"></td>
   </tr>
   <tr>
    <td background="pics/ram_2_1.gif"> </td>
    <td width="100%" bgcolor="#7F7F7F" class="<? echo $class; ?>">
       <?
        if ($obsah=='banner'){
            banner_125 ();
         }else{
            echo $obsah;
         }
        ?>
    </td>
    <td background="pics/ram_2_3.gif"> </td>
   </tr>
   <tr>
    <td height="9"><img src="pics/ram_3_1.gif" width="9" height="9"></td>
    <td background="pics/ram_3_2.gif">
       <img src="pics/_transparent.gif" height="9">
    </td>
    <td><img src="pics/ram_3_3.gif" width="9" height="9"></td>
   </tr>
  </table>
 </td>
</tr>
<?
 }
```

...
Jak se tedy funkce ramecek() zpracuje? Všimněte si, že obsahuje dva atributy - proměnnou $obsah, která, není-li zadána, obsahuje pevnou mezeru a proměnnou $class s implicitní hodnotou text. Přiřazené hodnoty jsou pro případ, když potřebujeme, aby proměnná vždy obsahovala aspoň něco, "přebije" je samozřejmě to, co do nich vložíme jejich voláním. Podívejte se na novinky.php - volá se ramecek() a obsah mezi závorkami se převede následovně: " ... </ul>" se vsadí do proměnné $obsah, "text" do $class.
V template_jave.php se toto zpracuje - nejprve se nastaví css styl "text", to snad bližší komentář nepotřebuje - template_jave.php má prostě na svém začátku definovány různé styly; funkci if() jsme si rozebírali v minulých dílech. Když proměnná $obsah obsahuje slovní řetězec "banner" (vzpomínate na dvě rovnítka?), obsah <td> bude banner (volaný funkcí banner_125() ). Jinak se vypíše proměnná $obsah (<div class=nad... ...ěna.</ul>).
Funkci si můžete představit asi takto - pošlete do ní jakási data a ona vám je zpracuje. Ta data ovšem mají svá pravidla. Musíte dodržet pořadí proměnných, nic nevynechat a text psát mezi uvozovky. Nadefinujete-li funkci moje_funkce($prvni,$druha,$treti), pak ji musíte i stejným způsobem zavolat: moje_funkce(1,2,'tři a kousek').
Příště se podíváme na funkce page_prolog() a page_epilog() a možná už budete moci udělat vlastní stránky v mém designu ;).
Doufám, že to není nepochopitelné, kdyby bylo, stačí se zeptat - k tomu slouží to diskuzní forum pod články ;) Pokračuji celkem rychle, protože jsem přesvědčen, že kdo se o tuto problematiku zajímá, nestuduje jenom webtip... Snažím se vám maximálně ulehčit používání php jako doplněk k html - ke zjednodušení práce a ne k jejímu zkomplikování.

# PHP v praxi, 7. díl - funkce

(14.09. 2001)

Ze začátku si trochu nasypu popel na hlavu. Když jsem nyní četl minulý díl PHP v praxi, bylo mi jasné, že vám asi moc nepomohl. Začátečníci těžko porozuměli, o co vlastně šlo. Proto se nyní pokusím o nápravu a pojedu i trochu podle manuálu. Tento díl bych rád věnoval funkcím, konkrétně těm, které si sami nadefinujete. Už jsem psal o readfile(), include() apod., ty však už mají přesně určeno, co mají dělat - prostě zpracovat to, co mají v závorkách. Uživatelem definované funkce mají podobný zápis, např. minule zmíněná funkce ramecek().

Rád bych se na to s vámi nyní podíval trochu obecněji. K tomu využiji i manuálu, který zatím bohužel do češtiny přeložen nebyl. Obecný zápis si tam najdete sami, já už ukážu něco krátkého, funkčního.

```
function predstav_se ($jmeno)
  {
  echo "Moje jméno je $jmeno";
  }
```

Funkci definujete tedy slovem function, následuje jméno funkce (tímto jménem ji spustíte), v jednoduchých závorkách jsou argumenty té dotyčné funkce a mezi složenými závorkami samotný obsah. Jak ji tedy použijete?

```
predstav_se(Honza);
```

Na obrazovce se objeví: Moje jméno je Honza. Vidíte, že je to velice logické - někde definované funkci predstav_se() se pošle argument Honza, ta ho zpracuje a vrátí výsledek. Líbí se mi tento manuálový příklad:

```
function square ($num)
  {
  return $num * $num;
  }

echo square (4); // výsledek: '16'.
```

Nepovažuji za nutné jej vysvětlovat, snad jen nové slovo - return - jednoduše vrátí to, co následuje za ním. Bez něj to nic nevrátí (zkuste si sami).
Další zajímavou vlastností je přiřazení určité implicitní hodnoty.

```
function predstav_se ($jmeno='Petr')
  {
  echo "Moje jméno je $jmeno";
  }
```

Kdyby nyní při volání funkce chyběl argument, nahradí se slovem Petr.

```
predstav_se(Honza);      //Moje jméno je Honza
predstav_se(Petr);       //Moje jméno je Petr
predstav_se();       //Moje jméno je Petr
```

Snad se vám již podaří definovat vlastní funkce bez problémů a příště se podíváme opět na stránky, jak jsem sliboval.

# PHP v praxi, 8. díl - šablona

(21.09. 2001)

V minulých dvou dílech jste si mohli udělat představu o funkcích a k čemu se vlastně chceme dopracovat. V tomto díle se budeme věnovat vytvoření základního souboru - tzv. template nebo česky šablony - který obsahuje funkce a v nich samotný html kód.

Jak takovou šablonu vytvořit? Popíšu postup pro moje stránky, protože k dispozici máte i výsledek. Jak na stránkách vidíte, skládají se z loga, menu na levé straně, hlavního rámečku s textem, reklamy a zápatí. Vytvořil jsem si tedy jednu stránku čistě v HTML, která vypadala jako nynější index.php (resp. novinky.php). A začal hledat stejné nebo podobné části. Logo je bez debat originální a nic s ním dělat nelze ;) Ale pak už je vše velice podobné - tabulka se stejným grafickým ohraničením. Krokem číslo jedna, zmíněným už v šestém díle, bylo vytvoření funkce ramecek(), jejímž obsahem je tabulka s definovanými grafickými prvky (okraje) a místo pro obsah rámečku. Tak už bylo možno začít stránku přepisovat do PHP. Na konec stránky přišel celý kód pro funkci ramecek() a namísto opakujících se tabulek stačilo napsat ramecek('obsah rámečku'); a tím celý text podstatně zkrátit.

Dalším krokem bylo definovat oblast, která bude neměnná pro všechny budoucí dokumenty. Tato oblast je např. na novinky.php od začátku dokumentu až po konec levého menu a od rámečku s bannerem ke konci. Tyto oblasti bylo tedy vhodné rozdělit na dvě a každou definovat jako funkci - page_prolog() a page_epilog(). Text mezi nimi (hlavní rámeček) jsem přesunul do souboru novinky.php. Výsledkem už byla skutečná šablona, kde jsou vlastně pouze definované funkce a nelze ji použít samostatně jako stránku.

Teoreticky jsme tedy připraveni, podíváme se, jak to vše vypadá ve skutečnosti. Soubor novinky.php se nepatrně od minula změnil, takže se na něj podíváme ještě jednou. Hlavně jej ale potřebujeme, abychom si ukázali, jak se která funkce volá.

```
<?
include ('template_jave.php');
page_prolog ('Novinky');
ramecek
('
  <h1> NOVINKY </h1>
  <ul>
  <li>20/09/01 ::?:: zjednodušení kódu.
  <li>18/08/01 ::?:: pár drobných změn.
  <li>09/08/01 ::?:: betaverze spuštěna.
  </ul>
');
page_epilog ();
?>
```

Šablonu jsem pojmenoval template_jave.php, na začátku každé stránky se tedy musí použít funkce include(), abychom k šabloně mohli přistupovat jako by byla součástí stránky. První volanou funkcí je page_prolog.php(), ta je tedy obsažena v template_jave.php a její obsah je následující:

```
<?
function page_prolog ($title='Welcome')
{
?>
<html>
<head>
<title><? echo $title; ?></title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
<style type="text/css">
<!--
body {  background-color: #CCCCCC  }
.text {  font-weight: 700; color: #33CCCC; font-family: Arial, Helvetica, sans-serif  }
.tlacitko { font-size: 16px; text-align: center }
h1 { font-size: 20px; text-align: center }
a:active, a:visited, a:link {  text-decoration: none; color: #33CCAA  }
a:hover { text-decoration: none; color: #33FFCC  }
-->
</style>
</head>
<body>
<table width="100%" border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td>
      <div align="center"><img src="pics/logo_big_sup.gif" width="338" height="75"></div>
    </td>
  </tr>
</table><br>
<table width="100%" border="0" cellspacing="5" cellpadding="0">
  <tr valign="top">
    <td width="143" align="left">
      <table width="143" border="0" cellspacing="5">
<?
ramecek ('<div class="tlacitko"><a href="novinky.php">novinky</a></div>');
```

```
ramecek ('<div class="tlacitko"><a href="projekty.php">projekty</a></div>');
ramecek ('<div class="tlacitko"><a href="tvorba.php">tvorba</a></div>');
ramecek ('<div class="tlacitko"><a href="download.php">download</a></div>');
ramecek ('<div class="tlacitko"><a href="odkazy.php">odkazy</a></div>');
?>
     </table>
    </td>
    <td width="100%" align="center">
     <table width="100%" border="0" cellspacing="5" cellpadding="0">
<?
}
```

Tuto funkci jsme volali s atributem "Novinky", takže proměnná $title obsahuje právě tento text. Ten se vypíše, jak vidíte, mezi párový tag <title>. Kdyby nebyl žádný argument použit, proměnná $title by obsahovala text "Welcome", ale to už víte. Pak následuje definice css, tabulka s logem a pak opět něco zajímavého - hlavní tabulka (1 řádek, 3 buňky) a v ní tabulka obsahující menu. Od šestého dílu jsem ještě funkci ramecek() zjednodušil, hned se na ni také podíváme. Funkce page_prolog() končí tak, aby mohla navázat funkce ramecek():

```
function ramecek($obsah=' ')
{
?>
    <tr>
     <td>
      <table width="100%" border="0" cellspacing="0" cellpadding="0">
       <tr>
        <td width="9" height="9"><img src="pics/ram_1_1.gif" width="9" height="9"></td>
        <td background="pics/ram_1_2.gif">
          <img src="pics/_transparent.gif" width="125" height="1"></td>
        <td width="9" height="9"><img src="pics/ram_1_3.gif" width="9" height="9"></td>
       </tr>
       <tr>
        <td background="pics/ram_2_1.gif"> </td>
        <td width="100%" bgcolor="#7F7F7F" class="text">
         <?
          if ($obsah=='banner')
          {
               banner_125 ();
          }else{
               echo $obsah;
          }
          ?>
        </td>
        <td background="pics/ram_2_3.gif"> </td>
       </tr>
       <tr>
        <td height="9"><img src="pics/ram_3_1.gif" width="9" height="9"></td>
        <td background="pics/ram_3_2.gif">
           <img src="pics/_transparent.gif" height="9"></td>
        <td><img src="pics/ram_3_3.gif" width="9" height="9"></td>
       </tr>
      </table>
     </td>
    </tr>
<?
}
```

Zjednodušení spočívá v tom, že místo mnoha argumentů je použit jen ten základní - $obsah. Pokud je obsahem text "banner", spustí se funkce banner_125(), pokud ne, vypíše se ten příslušný obsah do rámečku. Následuje funkce page_epilog():

```
function page_epilog ()
{
?>
      </table>
    </td>
    <td align="right" width="143">
      <table border="0" cellspacing="5" cellpadding="0" width="143">
<? ramecek ('banner'); ?>
    </table>
    </td>
  </tr>
</table>
<table cellpadding="10" width="100%" align="center">
<? ramecek ('<div align="right"> e-mail:
 <a href="mailto:jave@yo.cz">jave@yo.cz</a></div>'); ?>
</table>
</body>
</html>
<?
}
```

Do té nevstupují žádná data. V page_epilog() jsou dva rámečky - banner a zápatí stránky. Tato funkce představuje konec stránky. Ale v template_jave.php se vyskytuje ještě jedna funkce - banner_125():

```
function banner_125 ()
{
?>
<!--- Banner.YO verze 2.3.1  START --->
...
<!--- Banner.YO verze 2.3 END --->
<?
}
?>
```

Ale ta je jen pro ilustraci, její kompletní obsah zveřejňovat nehodlám. Na to si sami založte účet u nějakého bannerového systému ;)

Jak jste viděli, tvorba šablony není nejjednodušší, ale rozhodně se vyplatí. Značně se zrychlí postup při globálních změnách stránek ale i přidávání dalších. Například přidání stránky sekce Download znamenalo přidat jeden rámeček do menu v template_jave.php a vytvoření stránky download.php už bylo hračkou:

```
<?
include ('template_jave.php');
page_prolog ('Download');
ramecek
('
<h1> DOWNLOAD </h1>
<br><div align="center">?:: <a href="#maturita">maturita</a> ::?</div>
```

```
');
ramecek
('
<a name="maturita"></a> <h1> Maturita </h1>
<ul>
<li> <a href="download/cestina.zip">čeština</a></li>
<li> <a href="download/fyzika.zip">fyzika</a></li>
<li> <a href="download/nemcina.zip">němčina</a></li>
</ul>
');
page_epilog ();
?>
```

Tento postup je výhodný při častých změnách nebo velkém množství stránek. To, uznávám, není případ stránek mnou prezentovaných, ale alespoň je to na nich jednoduše pochopitelné (alespoň v to věřím). Tak to by bylo k šablonám vše, příště se podíváme opět na něco jiného.

# PHP v praxi, 9. díl - menu

(27.09. 2001)

Menu je velice podstatná část každého webu. Stránkám velice prospívá, pokud je interaktivní - na to je tu JavaScript s funkcemi jako onMouseOver apod. Jak už jsem vysvětloval na začátku seriálu, takovouto interaktivitu PHP nenabízí. Často ale potřebujeme, aby se odkaz na stránku, která je právě na obrazovce, nějakým způsobem změnil. Já jsem zvolil jeho deaktivaci - odkaz na aktuální stránku již není odkazem ale pouhým textem. Zvolil jsem k tomu trochu komplikovanější strukturu, kterou se vám pokusím co nejvíce přiblížit.

Nejprve si musíme uvědomit, co vlastně chceme. Každý odkaz v menu musí zkontrolovat, jestli zobrazená stránka není ta, na kterou směřuje, a podle toho se zachovat. Vzhledem k tomu, že celé menu je pouze jednou v šabloně a nikoli na každé stránce zvlášť, vyvstává zde problém: šablona musí vědět, která stránka je zrovna otevřená. K tomu se dá velice jednoduše využít informace, kterou předáváme šabloně funkcí page_prolog(). Ta se pak už jako proměnná $title dosazuje do tagu <title> ale dá se samozřejmě využít i jinak.

V minulém díle jsme si ukazovali menu vypadající následovně:

```
<?
ramecek ('<div class="tlacitko"><a href="novinky.php">novinky</a></div>');
ramecek ('<div class="tlacitko"><a href="projekty.php">projekty</a></div>');
ramecek ('<div class="tlacitko"><a href="tvorba.php">tvorba</a></div>');
ramecek ('<div class="tlacitko"><a href="download.php">download</a></div>');
ramecek ('<div class="tlacitko"><a href="odkazy.php">odkazy</a></div>');
?>
```

Tedy vždy se zobrazí všechny odkazy beze změny. Mohli bychom sem dosadit podmínku, která by kontrolovala otevřenou stránku podle $title. Podmínka pro první odkaz:

```
if($title=='novinky')
{
  ramecek ('<div class="tlacitko">novinky</div>');
}
else
{
  ramecek ('<div class="tlacitko"><a href="novinky.php">novinky</a></div>');
}
...
```

To je velice jednoduché, ale dělat pro každý odkaz zvlášť podmínku, která se skoro neliší (nemění se nic než "novinky" za "projekty", "tvorba" atd.), je nejen zbytečné plýtvání prostorem, ale i časem při každé další editaci.

Rovnou vám teď ukážu moje řešení. Je komplikovanější, ale mnohem kratší:

```
<?
$titles=array("1"=>"novinky","projekty","tvorba","download","odkazy");

for($i=1;$i<6;$i++)
{
  if($title==$titles[$i])
  {
    ramecek ('<div class="tlacitko">'.$titles[$i].'</div>');
  }
  else
  {
    ramecek ('<div class="tlacitko"><a href="'.$titles[$i].'.php">'.$titles[$i].'</a></div>');
  }
}
?>
```

Hned na začátku je něco nového: array(). V manuálu na nás opět vychrlí něco, co začátečník opravdu těžko pochopí. Pokusím se vysvětlit srozumitelněji. Normální proměnná, kterou jsme dosud používali, obsahovala jeden údaj. Ale když je proměnná array, pak obsahuje těch údajů třeba milión a ke všem se můžeme dostat, protože jsou indexované. Třeba proměnná $titles se stává array a obsahuje tato slova: novinky, projekty, tvorba, download, odkazy. Když napíšete příkaz "echo $titles;" ,pak se vám však nevypíše nic - leda text "array". K přístupu k array se používají indexy - "echo $titles[3];" vypíše na obrazovku "tvorba". A k čemu je tam ta jednička následovaná šipkou? Array() počítá od nuly, čili bez té jedničky by "echo $titles[3];" vypsalo "download". A protože je mi bližší počítat od jedné, přinutím od ní počítat i array(). Stejně tak se dá místo čísla použít třeba písmeno nebo slovo, ale to už by pro nás ztratilo smysl.

Funkce for() je naproti tomu v manuálu popsána velmi dobře. Jendá se vlastně o cyklus nebo smyčku (loop). Její syntaxe je:

```
for (výraz1; výraz2; výraz3) příkaz
```

První výraz (výraz1) je zpracován jen jednou na začátku celého cyklu. Výraz2 je podmínka, která se ověřuje při každém cyklu. Když je splněna, cyklus pokračuje. Teď

je však zpracován příkaz, tedy samotný obsah funkce for(), výraz3 je zpracován až po jeho dokončení. Podívejme se tedy na upravenou konstrukci v menu:

```
for($i=1;$i<6;$i++)
{
  print $i;
}
```

Jakmile PHP začne zpracovávat tento příkaz, nejdříve se za proměnnou $i dosadí "1". Potom se zkontroluje, jestli obsah proměnné $i je menší než šest. To souhlasí, takže můžeme postoupit dál - vypíše se obsah proměnné $i. A nakonec se k $i přičte jedna ("$i++" - takto se v PHP i jiných programovacích jazycích zjednodušují triviální matematické operace). A cyklus jede opět od první řádky, avšak výraz1 už ignoruje. Proměnná $i tedy nyní obsahuje "2", což také vyhovuje podmínce výraz2, takže můžeme pokračovat až do té doby, než $i je rovno šesti. Tento krátký cyklus si můžete zkusit - na obrazovce se objeví "12345".

V menu se však $i využívá praktičtěji. Ukážeme si už jen obsah cyklu (víme, že postupně bude $i číslo mezi jedna a pět):

```
if($title==$titles[$i])
{
  ramecek ('<div class="tlacitko">'.$titles[$i].'</div>');
}
else
{
  ramecek ('<div class="tlacitko"><a href="'.$titles[$i].'.php">'.$titles[$i].'</a></div>');
}
```

Jde o podmínku if(), která kontroluje, jestli obsah proměnné $title, předané každou ze stránek přes page_prolog(), je stejný jako ten určitý obsah array $titles.

Konkrétní příklad: Jsme na stránce "projekty". Proměnná $title tedy obsahuje text "projekty". První cyklus - $i rovno jedné. Zkontroluje se, jestli proměnná $title je shodná s array $titles[1]. To však zastupuje "novinky", proto je podmínka nepravdivá a pokračuje se až obsahem v else{}. Využívám toho, že se soubor jmenuje stejně a odkaz na něj také. Všude se proto dosazuje "novinky". Výsledkem je rámeček s odkazem, na který se dá kliknout. Tento cyklus končí přičtením jedné k $i.

$i je nyní 2. $titles[2] zastupuje projekty. Proto je podmínka splněna a výsledkem je rámeček s odkazem, na který se kliknout nedá (chybí tag <a>). K $i se opět přičte 1 a tak se to opakuje až do $i rovno pěti.

Pokud byste chtěli využít cyklu for(), ale neměli zapotřebí zvýrazňovat odkaz na aktivní dokument, pak vlastně stačí následujících pár řádek:

```
<?
$titles=array("1"=>"novinky","projekty","tvorba","download","odkazy");

for($i=1;$i<6;$i++)
{
```

```
  ramecek ('<div class="tlacitko"><a href="'.$titles[$i].'.php">'.$titles[$i].'</a></div>');
}
?>
```

Přidání jednoho odkazu do menu znamená přidat název toho odkazu (a zároveň i souboru) do array() a změnu podmínky $i<6 na $i<7.

Velice často se však liší název odkazu a stránky - problémem mohou být třeba i velká písmena na začátku. To se vyřeší velice snadno přidáním druhé array():

```
<?
$titles=array("1"=>"Novinky","Projekty","Tvorba","Download","Odkazy");
$files=array("1"=>"novinky","projekty","tvorba","download","odkazy");

for($i=1;$i<6;$i++)
{
  ramecek ('<div class="tlacitko"><a href="'.$files[$i].'.php">'.$titles[$i].'</a></div>');
}
?>
```

Věřím, že této konstrukce využijete, není zase tak složitá, i když to na začátku mohlo mnohým z vás tak připadat.

# PHP v praxi, 10. díl - Server

(12.10. 2001)

Zatím nejlepší řešení pro tvorbu stránek je mít jeden stroj s MS Windows a druhý s Linuxem. Pod Windows jsou nejčastěji používané programy (Macromedia Dreamweaver, Adobe Photoshop etc.), Linux je zase skvělý na server (Apache, PHP, MySql). Ale většina lidí má doma jen jeden počítač a nemá důvod, proč si kupovat druhý (tedy toto by důvod byl, ale spíše se nechce učit pracovat v Linuxu ;) Proto vám ukáži, jak nainstalovat **Apache s podporou PHP pod Windows98** (pravděpodobně bude spolehlivě fungovat i pod 95, NT, 2000, ale s těmito zkušenosti nemám). Je pravda, že toto řešení není ideální, ale funguje. Možná se zeptáte, proč instalovat Apache místo využití nějakého serveru od Microsoftu. Důvody jsou jeho rozšíření, kvalita a bezpečnost.

Abych vám co nejvíce ulehčil práci, sám to budu instalovat a popisovat svoji činnost krok za krokem.

Nejprve si potřebujete stáhnout Apache pro Windows. Na oficiálních stránkách Apache Software Foundation se můžete proklikat až na stránky věnované přímo Apache pro Windows, ale já vám nabídnu přímý odkaz na instalační balíček (verze 1.3.20, velikost 1,8 MB), je bez zdrojových textů a podpory instalace - pokud byste neměli Windows Installer v 1.10, odkazy na něj jsou také na již zmiňovaných stránkách Apache pro Windows.

Instalační balíček (apache_1.3.20-win32-no_src-r2.msi) máme doma, začneme tedy instalovat. První problém se objeví v sekci Server Information - zde zadáte informace

o vašem počítači a zvolíte, jestli spuštěný Apache bude přístupný pro všechny uživatele nebo jen pro vás. Vybral jsem pro všechny a pokračuji. Setup type - zvolte klidně Complete (nainstaluje se vám jak server, tak dokumentace k němu). Destination Folder - volím C:\, abych pak měl k souborům serveru jednoduchý přístup. Pak už proběhne samotná instalace, objeví se dosové okno a zase zmizí a pak už na nás čeká jen odklepnutí Finish.

V nabídce Start -> Programy se nám objevila položka Apache httpd Server. Po kliknutí na Start Apache in Console by se vám mělo otevřít dosové okno s textem "Apache/1.3.20 (Win32) running..." Tak a teď spusťte Internetový prohlížeč a do adresy napište "localhost". Otevře se vám stránka s textem Not Acceptable. Zvolte index.html.cz a radujte se z úspěchu ;)

Nyní před námi leží úkol těžší - nakonfigurovat Apache. Ke konfigurace není k dispozici žádná klikací utilitka, je to čistý text. Soubor se jmenuje httpd.conf a nachází se v adresáři C:\Apache\conf\. Otevřete si ho v textovém editoru (nejlépe v Notepadu, ne ve Wordu) a nelekejte se počtu řádků. Většina textu je dokumentace - znak # znamená, že se dotyčný řádek nebude zpracovávat (jako // v PHP). Odskrolujte na řádek 211 - zde začíná sekce 2, která obsahuje pro nás nejdůležitější nastavení. Změny:
1.) Řádek 255 - **ServerName localhost** (nebo 127.0.0.1, to je ekvivalentní)
2.) Na řádku 263 je informace **DocumentRoot "C:/Apache/htdocs"** - cestu přepište na adresář, kde máte uložené své stránky (např. já mám DocumentRoot "C:/home/httpd/u"). To samé proveďte na řádku 288 - **<Directory "C:/Apache/htdocs">**.
3.) Řádek 347 obsahuje informace o jménu souboru, který bude použit jako první - "DirectoryIndex index.html". Vzhledem k pozdějšímu použití doplníme podle potřeby - **DirectoryIndex index.html index.php index.htm default.html** apod.

Úplně na konci konfigračního souboru je věc nejdůležitější - sekce **VirtualHost**. Malá ukázka mého nastavení:

```
NameVirtualHost 127.0.0.1

  <VirtualHost 127.0.0.1>
  DocumentRoot c:/home/httpd/u/jave
  ServerName jave
  </VirtualHost>

  <VirtualHost 127.0.0.1>
  DocumentRoot c:/home/httpd/u/tovarna
  ServerName tovarna
  </VirtualHost>
```

Pozn: 127.0.0.1 je ip adresa pro lokální počítač. DocumentRoot směřuje na adresář se stránkami, ServerName se používá v prohlížeči.
Ale toto nestačí, ještě je tu soubor "C:\Windows\HOSTS" - tam musíte zapsat záznam o překladu jména na dotyčnou ip adresu.

```
127.0.0.1       jave
127.0.0.1       tovarna
```

Restartujte Apache (httpd.conf se načte jen při spuštění) - na ukončení programu zavřete dosové okno (pokud dělá problémy normální zavírání okna, použijte

kombinaci kláves CTRL+C) a znovu spusťte odkazem. Po zadání "jave" v prohlížeči už mi naskočí moje stránky - ale jen text/html.
Čeká nás stažení a instalace PHP modulu. Na stránkách PHP je ke stažení momentálně verze 4.0.6. A pozor: je ve dvou verzích - package [4,859Kb] a installer [755Kb]. Proč stahovat větší soubor? A navíc když se ten menší dokáže i sám nainstalovat? Funguje totiž jako CGI, tzn. aplikace, které se pošlou data a ona vrátí výsledek. Mám s tím velmi špatné zkušenosti - např. při zobrazování některých obrázků ve formátu .jpg je to zpracovalo tak, že nebyly vidět ;) A také se změní adresa z např. http://jave/index.php na http://jave/php/php.exe/index.php (to ale nevidíte, to se dozvíte, až když potřebujete s adresou pracovat...).
Takže si stáhneme ten package. Rozbalíme do adresáře C:\PHP (doporučeno) a pročteme si readme.txt ;) To je dobrý start, ale zdlouhavý, takže ve zkratce:
1.) zkopírujeme **php.ini-dist** do C:\Windows a přejmenujeme na php.ini
2.) do C:\Windows\System zkopírujeme knihovnu **php4ts.dll**
3.) do httpd.conf přidáme tyto řádky:
LoadModule php4_module c:/php/sapi/php4apache.dll
AddType application/x-httpd-php .php
4.) pomodlíme se ;)
5.) spustíme Apache.
Ten by nyní měl zobrazovat text: "Apache/1.3.20 (Win32) PHP/4.0.6 running...". Pokud se vám tak neděje, je něco špatně a znovu si projděte celý postup, jestli jste náhodou něco neopomněli. Nyní zadávám do adresy prohlížeče "jave" a ... vidím své stránky! Doufám, že i vám se to povede.

# PHP v praxi, 11. díl - obrázky

(29.10. 2001)

Tento díl bych rád věnoval celkem málo zmiňované schopnosti PHP - kreslení. Možná se vám zdá směšné, proč kreslit programovacím jazykem, ale představte si tuto situaci: máte textový web, grafikou jsou pouze jednoduché rámečky, každá stránka je ale v jiné barvě těchto rámečků. Normálně to pro nás znamená nařezat rámečky pro každou barvu zvlášť. Řešení v PHP není snadné, strávíte na tom dost času. Vyplatí se však tehdy, chcete-li v budoucnu přidat další sekce (to totiž pak znamená dodefinovat pouze tu jednu barvu, o ostatní se už postárá námi naprogramovaný PHP skript).

Ukázku můžete vidět na serveru SINISTER.cz, konkrétně na adrese www.sinister.cz/jave. Ani nevíte, kolik problémů pro naši redakci představovalo najít takový server, jenž by obrázky podporoval - nakonec se ale přece podařilo (a já tímto Sinisterovi velice děkuji). Každopádně na freeserverech jejich podporu pravděpodobně nenajdete, což je velká škoda.

Důležité upozornění - než začnete dělat cokoliv s obrázky vy sami, musíte mít zapnutou jejich podporu v PHP (implicitně je vypnutá - zřejmě i u těch serverů). Pod Windows to uděláte editací php.ini - jedná se o odstranění středníku před řádkem s obsahem "extension=php_gd.dll".

Nejprve si ukážeme vytvoření jednoduchého barevného čtverečku:

```
<?
header ("Content-type: image/png");
$obrazek = ImageCreate(25,25);
$modra = ImageColorAllocate($obrazek,0,0,255);
ImageFill($obrazek,0,0,$modra);
ImagePNG($obrazek);
ImageDestroy($obrazek);
?>
```

Vysvětlení: první řádek (header) předává informaci, co vlastně je obsahem tohoto souboru. My tvoříme png, ale může to být třeba gif. Proměnné $obrazek jsme pak přiřadili funkci ImageCreate(). Ta vytvoří prázdný obrázek s určenou velikostí (25 x 25 pixelů). Obrázek chceme modrou barvou. K tomu slouží ImageColorAllocate() - první atribut značí, čemu barvu přiřazujeme, další jsou hodnoty RGB. A teď je na nás tyto dva úkony provést - ImageFill(). Prvním atributem je zástupce vytvořeného obrázku, poslední barvy. Ty dva mezitím jsou počáteční body x a y, odkud chceme vyplnění provést. Funkce ImagePNG() konečně obrázek vytvoří a ImageDestroy() po nás vyčistí paměť.

Pokud jste si již zaexperimentovali s jednoduchým obrázkem, přejdeme k vykreslení samotných rámečků. Takový rámeček se skládá z osmi obrázků:

| | | |
|---|---|---|
| a1 | a2 | a3 |
| b1 | b2 | b3 |
| c1 | c2 | c3 |

Pole b2 je samozřejmě text. Soubor image.php musí znát: jaký obrázek tvoří a jakou barvou. To mu předáme stadardní cestou, třeba "image.php?color=black&typ=a1". Teď už je ale na něm, aby podle těchto informací vytvořil příslušný obrázek. Jak to tedy provedeme?

```
<?
//in: $color, $typ
if ($typ==a1|$typ==a3|$typ==c1|$typ==c3)
  {$vel_x=5;$vel_y=5;}

if ($typ==a2|$typ==c2)
  {$vel_x=1;$vel_y=5;}

if ($typ==b1|$typ==b3)
  {$vel_x=5;$vel_y=1;}
```

Jako první se zjistí podle typu velikost budoucího obrázku. To je zde vyjádřeno třemi jednoduchými podmínkami. Opakuji, že svislá čárka "|" značí "nebo". Typy a1, a3, c1 a c3 jsou rohy, ty budou mít velikost 5x5 pixelů. Horní a dolní okraje (a2 a c2) 5 pixelů na výšku a 1 na šířku (v tabulce v html jsou roztaženy - funkce background v tagu <td>); u b1 a b3 je tomu naopak. Výsledkem těchto podmínek jsou dvě proměnné: $vel_x a $vel_y.

```
switch ($color)
	{
	case black:
				$r=0;
				$g=0;
				$b=0;
	break;
	case red:
		$r=255;
		$g=0;
		$b=0;
	break;
	case green:
		$r=0;
		$g=255;
		$b=0;
	break;
	case blue:
		$r=0;
		$g=0;
		$b=255;
	break;
	default:
		$r=255;
		$g=255;
		$b=255;
	break;
	}
```

K získání barvy použijeme novou funkci switch(). To aby se i ti, kteří se o obrázky nezajímají, naučili něco nového ;) Tato funkce je velice často používaná pro svoji jednoduchost a přehlednost. Mezi závorky napíšeme proměnnou, jejíž obsah budeme kontrolovat. O to se starají příkazy case (v překladu výstižně "případ"). Jednoduše: v případě, že proměnná $color obsahuje text black, pak přiřaď proměnné $r hodnotu nula, $g nula a $b také. Pak přestaň - break; - bez toho by se pokračovalo i v dalších případech až do konce funkce switch(). V případě, že $color neobsahuje ani jednu z nabízených možností, funkce switch() použije data udaná za default: (tady nezapomínejte na dvojtečku). Výstupem jsou tedy tři proměnné: $r, $g a $b.

```
header ("Content-type: image/png");
$obrazek = ImageCreate($vel_x,$vel_y);
$podklad=ImageColorAllocate($obrazek,255,255,255);
$trans=ImageColorTransparent($obrazek,$podklad);
ImageFill($obrazek,0,0,$trans);
$barva=ImageColorAllocate($obrazek,$r,$g,$b);
```

Prvním dvěma řádkům jistě rozumíte - informace o obrázku a vytvoření zástupce s funkcí vytvořit obrázek ve velikosti podle atributů v závorkách. Další tři řádky by tam podle mého názoru být nemusely - vytváří se tím vlastně průhledné pozadí. Pokud tam však tyto řádky nedám, výsledkem místo pěkných čárek budou zcela zaplněné obrázky příslušnou barvou. Pokud se dozvím, proč tomu tak je, samozřejmě se o tom s vámi podělím. Poslední řádek je již známé přiřazení barvy obrázku.

```
switch ($typ)
	{
	case a1:
		ImageLine($obrazek,2,4,4,4,$barva);
		ImageLine($obrazek,4,2,4,4,$barva);
	break;
	case a2:
		ImageLine($obrazek,0,4,0,4,$barva);
	break;
	case a3:
		ImageLine($obrazek,0,2,0,4,$barva);
		ImageLine($obrazek,0,4,2,4,$barva);
	break;
	case b1:
		ImageLine($obrazek,4,0,4,0,$barva);
	break;
	case b3:
		ImageLine($obrazek,0,0,0,0,$barva);
	break;
	case c1:
		ImageLine($obrazek,2,0,4,0,$barva);
		ImageLine($obrazek,4,2,4,0,$barva);
	break;
	case c2:
		ImageLine($obrazek,0,0,0,0,$barva);
	break;
	case c3:
		ImageLine($obrazek,0,0,2,0,$barva);
		ImageLine($obrazek,0,0,0,2,$barva);
	break;
}
```

Switch() už známe. Novinkou je ImageLine() - to není nic jiného než vykreslení jednoduché čáry z bodu x1,y1 do bodu x2,y2 - syntaxe: ImageLine(cílový obrázek, x1, y1, x2, y2, barva čáry). To bylo na celé této ukázce asi nejsložitější - uvědomit si, kterým pixelem začínám a kterým končím. Upozorňuji, že se počítají pixely z levého horního bodu s počátkem v 0! Tímto se dá udělat klidně i bod (pro který má samozřejmě PHP také funkci - ImageSetPixel(), ale je dobré vědět o různých způsobech ;)

```
ImagePNG($obrazek);
ImageDestroy($obrazek);
?>
```

Obrázek teď vytvoříme a uklidíme po sobě.

Nevím, na kolik využijete tvorbu obrázků přes PHP (zvláště pro nepříznivou situaci na straně serverů), spíše jsem chtěl naznačit cestu k dalším a poměrně rozsáhlým možnostem tohoto jazyka. Zájemcům vřele doporučuji manuál, protože tady se tímto problémem nebudu pro výše uvedené důvody tak moc zabývat (pokud mi samozřejmě nepřijde x desítek e-mailů se žádostí o pokračování tohoto tématu ;). S těmito funkcemi totiž lze generovat i grafické odkazy s textem, dokonce se dají dělat grafy apod.

# PHP v praxi, 12. díl - počítadlo

(09.11. 2001)

V ukázce k minulému dílu vás místo rámečku zaujal counter! ;) Vyhovím vám tedy (díky Sinisterovi, který mi skript poskytl) a postupně vyložím, jak si takové jednoduché počítadlo můžete udělat i vy.

```
<?
$soubor = "data.txt";
$fp = fopen("$soubor", "r+");
$count = fgets($fp, 4096);
$count += 1;
fseek($fp,0);
fputs($fp, $count);
fclose($fp);
echo $count;
?>
```

Začneme "kostrou" celého skriptu - funkční částí. Nejprve nadefinujeme cestu k souboru, kam budeme počty zobrazení ukládat. V našem případě tedy soubor data.txt v adresáři totožném se skriptem. Nezapomeňte nastavit práva tomuto souboru i pro zápis.

Proměnné $fp (fp je zkratka pro **f**ile **p**ointer - ukazatel na soubor, to je často používaný termín) se přiřazuje akce fopen(), o které jsem toho napsal již dost v pátém díle. Tedy ve zkratce - otevíráme soubor, na který odkazuje proměnná $soubor, a to v módu "r+" - Otevři pro čtení i zápis a kurzor umísti na začátek souboru.

Pak následuje přečtení obsahu dotyčného souboru do proměnné $count. Děje se tak pomocí funkce fgets() - ta přečte řádek ze souboru, na který ukazuje $fp, a to v délce druhého argumentu zmenšeného o jedna. Jinými slovy to čte až do 4095. znaku v souboru data.txt. Můžeme samozřejmě zvolit i libovolné nižší číslo - kromě nuly ;). V dalším řádku zvyšujeme získané číslo o jedna. Můžeme to napsat i jako $count=$count+1; ale protože jsme líní, píšeme tyto jednoduché matematické operace tak, jak vidíte.

Funkcí fseek() pak nastavíme pozici "kurzoru" (tedy místa, kam budeme zapisovat) na 0 - tedy úplný začátek. Počet zobrazení nyní zapíšeme pomocí fputs() - první argument je opět náš soubor, ten druhý je dotyčný počet zobrazení.

Nakonec otevřený soubor zavřeme a vypíšeme počet, ke kterému jsme došli.

Vcelku triviální, že? Tak přidáme pár řádek (zvýrazněny tučně):

```
<?
$soubor = "data.txt";
$pocet_cislic = 5;

if (! file_exists($soubor))
{
```

```
    $txt = "Chyba";
}
else
{
    $fp = fopen("$soubor", "r+");
    flock($fp, 1);
    $count = fgets($fp, 4096);
    $count += 1;
    fseek($fp,0);
    fputs($fp, $count);
    flock($fp, 3);
    fclose($fp);

     chop($count);
    $pocet_cislic = max(strlen($count), $pocet_cislic);
    $txt = substr("0000000000".$count, -$pocet_cislic);
}

echo $txt;
?>
```

Podmínka je doufám jasná - pokud neexistuje soubor v proměnné $soubor, dosadí se do proměnné $txt textový řetězec "Chyba". Jinak se skript zpracuje.

Nově je zde funkce flock(). Ta se stará o to, aby nemohl se souborem v téže chvíli manipulovat nějaký jiný proces. Hodnota 1 soubor otevřený $fp "zamkne", 3 zase "odemkne".

Potíže by se mohly vyskytnout i kdyby řádek obsahoval nějaké netisknutelné znaky, proto jej proženeme funkcí chop(), jež to zkontroluje a tyto znaky odstraní.

Další řádek kódu obsahuje funkce hned dvě: strlen() a max(). Obě jsou ale velice jednoduché - strlen je vlastně zkratka pro "string length" tedy česky: délka řetězce. Na začátku jsme si nadefinovali číslo pět jako $pocet_cislic, nyní kontrolujeme, jestli náhodou počet zobrazení nemá více jak pět řádů. Funkcí max() právě vyberem tu větší a tu zpátky přiřadíme proměnné $pocet_cislic.

Chceme-li (a my chceme ;), aby se počet číslic doplnil nulami od prvního - tedy aby místo "21" se zobrazilo "00021", potřebujeme funkci substr(). Ta vrací část řetězce - první argument je dotyčný řetězec, ze kterého část vezmeme, druhý argument je počáteční pozice, odkud budeme číst. V našem případě je řetězcem 00000000021 (pro případ dvacátého prvního zobrazení stránky). Protože potřebujeme pouze posledních pět číslic, zadáme -pocet_cislic (minus značí "odzadu") coz se nyní rovná -5. Tato funkce je používaná velmi často - proto uvedu i třetí argument: délku řetězce. Např: echo substr(nazdarek, 3, 2); vypíše "da" (dva znaky od pozice 3).

A konečně provedeme místo vypsání textu vygenerování obrázku:

```
...
     $pocet_cislic = max(strlen($count), $pocet_cislic);
    $txt = substr("0000000000".$count, -$pocet_cislic);
}
```

```
header("content-Type: image/png");
$obrazek = ImageCreate(9 * $pocet_cislic + 1, 17);
$bg_color   = ImageColorAllocate($obrazek, 0 ,0, 0);    // black
$text_color = ImageColorAllocate($obrazek, 255, 255, 255); // white
ImageString($obrazek,5,1,0,$txt,$text_color);
ImagePng($obrazek);
ImageDestroy($obrazek);

?>
```

*Pozn: "echo $txt;" musíte smazat, jakýkoliv jiný výpis na stránku má za následek nezobrazení obrázku.*

Většinu věcí znáte již z minula, proto jen pár detailů - velikost obrázku je vypočtena podle počtu číslic * 9 pixelů +1 (jinak bude pravá číslice příliš "namáčknuta" na stranu.

Novinkou je funkce ImageString(), její argumenty v závorce jsou: (obrázek, font, pozice x, pozice y, text, barva). PHP má na výběr 5 fontů, liší se velikostí. Jiné fonty kdyžtak můžete použít pomocí funkce ImageLoadFont().

Tak doufám, že jsem uspokojil vaše potřeby a budete se moci pyšnit milióny pageviews! ;)

# PHP v praxi, 13. díl - Ohlédnutí

(16.11. 2001)

Protože slavíme takové malé výročí, tento díl vyhradíme vašim problémům s minulými články. Na e-mail mi přišlo velké množství ohlasů, většinou pozitivních, čehož si velice vážím a mám z toho radost. Dále bych rád upozornil na adresu webtip.oceany.cz, kde naleznete slibované ukázky pěkně pohromadě. Průběžně k nim budu samozřejmě přidávat další a další, takže vždycky budete mít přehled...

Asi nejvíce problémů jste měli s diskuzí. Dílem okamžiku bylo vyřešit některé z nich - např. záměnu s a z (soubor diskuse.php odkazoval na diskuze.php), další však již byly složitější. Skript může být sebedokonalejší, ale když server neumožňuje ukládat do souboru, máte smůlu. Pokud vám to nejde na jednom místě, zkuste to provizorně dát na jiný hosting a posléze to tam třeba i nechat a přesměrovávat se na skript tam. Někdy jsem však nepochopil, kde vlastně ten zakopaný pes leží. Zdálo se mi, že někteří si dělají rovnítko mezi chat a diskuze. To je ale zcela něco jiného. Odpověď nepřichází ke všem, ta přichází pouze na server a vy musíte ručně obnovit stránku, aby se nový příspěvek zobrazil (pokud tedy sleduje více lidí diskuzi najednou a zadá ho tam někdo jiný než vy ;).

Instalace Apache. Uf, já věděl, že když ten návod napíši, že si koleduji... Některým lidem to prostě nechodí a problém přitom nikde není vidět (tedy alespoň já jsem ho neobjevil). Dobře byl vidět v pár případech - když jste zapomněli odkomentovat řádky

před virtualhost: # (křížek ;) znamená, že řádky za křížkem se nečtou, takže Apache pak tyto řádky nevidí a vitualhosty prostě nejsou.

Pan Jan Hora popsal zajímavý problém a s jeho svolením zde jeho dva maily zveřejňuji:

#####
#1
podle 10. dílu Vašeho seriálu na Webtipu jsem si předělal instalaci PHP aby běžel jako modul Apache a ne jako CGI jako dosud. Zadařilo
se, všechno fungovalo. Až asi po týdnu jsem zjistil, že přestaly fungovat skripty, které načítaly JPG obrázky a ukládaly je do MySQL. Laborováním jsem zjistil, že prostě nelze načíst binární soubor do proměnné a zas ho uložit pod jiným jménem.

```
$a=fread($in,filesize($infile))
$b=fputs($out,$a)
```

Pro textové soubory to fungovalo normálně, výsledný soubor byl přesná kopie. Pro obrázky, ZIPy a další binární soubory měl výsledný soubor délku jen stovky bytů. Se skřípěním zubů jsem se vrátil k původní CGI konfiguraci a skripty fungují jako víno. Takže docela opačná zkušenost, než jste uváděl Vy. Kdyby jste tušil čím to může být dejte mi prosím vědět, jinak to berte jen jako zajímavost.

#2
záhada je vyřešená, poptal jsem se na PHP fóru na www.builder.cz a hned 1. rada zafungovala. Při otevírání souborů pro vstup a výstup je potřeba použít "rb" a "wb", pak to i pod "modulem" funguje správně. Pokud se použije jen "r" a "w" tak to zlobí. V dokumentaci k PHP
tenhle parametr je jen jako malá zmínka bokem. Navíc jsem se tam nekoukal, když předtím skript běžel, tak jsem hledal chybu jinde. Údajně je takhle potřeba napovídat jen Windozům, jinde to prý funguje bez problémů.

Zdraví Jan Hora
#####

Pokud vám něco nebude fungovat, ptejte se - píšu články a tudíž jsem odpovědný i za jejich dopad. Při dotazu uvádějte ale co nejvíce konkrétních informací, ať netápu a neradím vám o něčem úplně jiném. A také více využívejte diskuzního fóra pod články, jistě se najdou čtenáři, kteří vám dokáží poradit (víc hlav...). Když nebudu dlouho odpovídat, nezoufejte, jsem značně zaneprázdněný člověk, denně mi chodí desítky e-mailů a občas prostě nestíhám. Pokud chcete odpověď rychle a kvalitně - je tu PHP fórum na Builderu, kde vám jistě někdo pomůže.

Co se týče budoucích článků, jakási vize: pomalu budeme přecházet k věcem týkajícím se databází (konkrétně MySQL), budu se věnovat i těm "vyžádaným" článkům, kde je bohužel značná prodleva mezi vašim "zadáním" a mým "výsledkem". Diskuzi budeme vylepšovat (s databázemi je to opravdu o něčem jiném), budeme zjišťovat různé informace od našich návštěvníků apod. Věřím, že se vám tento seriál bude i nadále zamlouvat alespoň tak jako dosud.

Přeji vám všem hodně úspěchů a děkuji za vaši přízeň.

# PHP v praxi, 14. díl - MySQL

(23.11. 2001)

Diskuze, jejíž údaje se ukládají do textového souboru, není právě nejideálnější řešení. V následujících dílech PHP v praxi se budeme věnovat vytvoření diskuze pomocí databáze. Abychom ale mohli s databází pracovat, musíme si vysvětlit některé pojmy a databázi nainstalovat.

Co je to databáze? Stručně řečeno: souhrn tabulek. Pro ty, kteří vůbec netuší, jak taková databáze funguje:

| Seznam | | | |
|---|---|---|---|
| Interpret | Album | Rok | Cena |
| Pink Floyd | Echoes - The Best of Pink Floyd | 2000 | - |
| Ozzy Osbourne | Down To Earth | 2000 | 578.00 |
| Garbage | Beautifulgarbage | 2000 | 484.00 |
| Republica | Republica | - | - |

Vidíte docela obyčejnou tabulku s názvem Seznam. A přesně o tom je to celé, co se vznešeně nazývá databází. Máme zde sloupce a řádky. Každý sloupec má jméno (Interpret, Album...) a na dotaz nám odpoví řádkem. Například hledáme všechna alba vydaná v roce 2000, řekneme tedy databázi:*najdi všechny řádky v tabulce seznam, jejichž hodnota pole "Rok " je "2000" a z toho vypiš pouze pole "Interpret"* , výsledkem je:*Pink Floyd, Ozzy Osbourne, Garbage*. Uvidíte, že obsluhovat databázi není takový problém.

Asi nejpoužívanější webová databáze je MySQL, proto i my se naučíme pracovat právě s ní. Musíme si ji samozřejmě nejprve nainstalovat a o tom bude i dnešní článek. Opět se bude jednat o návod pro Windows (jsem přesvědčen, že linuxář si MySQL zvládne nainstalovat bez problémů). Jistě vás potěší to, že MySQL nijak nekoliduje s PHP a hlavně že nebude potřeba PHP ani Apache nijak složitě upravovat.

Vybereme si verzi 3.23 (verze 4 zatím není zcela dokončená) na stránce http://www.mysql.com/downloads/mysql-3.23.html a zvolíme příslušný soubor (přímý odkaz: MySQL 3.23.44 Windows 95/98/NT/2000/XP 12MB). Instalace je triviální - jen na jeden problém upozorním: instalujte do c:\mysql, jinak musíte vytvořit soubor s cestami k mysql (to je vše v readme). Spustitelné soubory se nacházejí v adresáři c:\mysql\bin a tím nejzajímavějším pro nás je pak "winmysqladmin.exe".

Spusťte jej - výsledkem by mělo být vyskočení okna a jeho minimalizace do traye (ikonka semaforu). Klikněte na tuto ikonku a zvolte "show me" - tady máte takovou rádoby-administraci MySQL. V sekci Environment se dozvíte detaily o vašem

počítači, dále nás vlastně nic nemusí zajímat až po Databases, kde máte přehled o vytvořených databázích a tabulkách (zatím téměř prázdné). Tam kde je modře napsáno Right Click for Menu options pravým tlačítkem myši klikněte a zvolte "Hide me", to jediné opět zminimalizuje aplikaci do traye. K ovládání se vyjadřovat nebudu, prostě to berme tak, že s tím stejně nic nenaděláme ;) V adresáři bin máte pak ještě spoustu souborů, kterými můžete MySQL obsluhovat, ale proč bychom tak činili?

Aby se nám databáze dobře ovládala a nemuseli jsme na to chodit přes příkazový řádek, stáhneme si z phpWizard.net skvělý phpMyAdmin (verze 2.2.1). Jedná se o ovládání databáze přes PHP s příjemným uživatelským rozhraním. Zde není žádná automatická instalace a ani být nemůže.

1. Soubor phpMyAdmin-2.2.1-php.zip rozbalte do adresáře phpmyadmin (např. c:\home\httpd\u\phpmyadmin). Tento adresář ale logicky musí už obsahovat soubory a adresáře, ne aby se vám tam objevil adresář phpMyAdmin-2.2.1 ;)
2. Do hosts nebo jak se tento soubor ve vašem systému jmenuje doplňte řádek s textem "127.0.0.1 phpmyadmin"
3. Do httpd.conf v Apachi připište tento analogický Virtualhost podle cesty k phpmyadmin:
   <VirtualHost 127.0.0.1>
   DocumentRoot c:/home/httpd/u/phpmyadmin
   ServerName phpmyadmin
   </VirtualHost>
4. Spusťte nebo restartujte Apache, spusťte (pokud není) winmysqladmin.exe a internetový prohlížeč. Do adresy vepište phpmyadmin a obdržíte tento výsledek: Vítej v phpMyAdmin 2.2.1; MySQL 3.23.44 běžící na localhost jako root@localhost. Tedy moc si přeji, aby se vám to takhle bez problémů povedlo...

Upřímně řečeno, čeština se mi v těchto aplikacích moc nelíbí. Jestli ani vám: na editaci nastavení je tu soubor config.inc.php a tam na řádku 144 odkomentujeme $cfgLang = 'en'; (tento příkaz přinutí phpmyadmin používat právě zadaný jazyk místo toho, který zjistí z našeho prohlížeče).
V levém framu vidíte vytvořené databáze. Pravděpodobně jediná, kterou tam máte je mysql. Klikněte na ní. Rozbalí se strom s jednotlivými tabulkami. Jedna databáze totiž má v sobě x tabulek, z nichž se data čerpají. Přehled o tom a bližší pochopení získáte při praktických ukázkách. Teď je důležité, že můžeme začít - tedy v přístím díle ;).
Ale abych vás nenechal jen tak: ukážeme si vytvoření jednoduché CD databáze podle tabulky nahoře.
Kliknutím na Home (nebo Úvod - upozorňuji, že budu používat častěji anglických termínů) se přesuneme na úvodní stránků phpmyadminu. Tam najdete jediné pole formuláře - vytvořit novou databázi. Zadáme třeba "cd" a klikneme na create (vytvořit). Tím už jsme vytvořili databázi. Nyní tabulku, která ta data bude v sobě držet. Na stránce, kam vás to přesunulo (nebo když v levém framu kliknete na cd) najděte "Create new table on database cd :", hodnotu "name" zadejte třeba "seznam" a "Fields" "4". Tím vytvoříme tabulku "seznam" o čtyřech sloupcích. Teď ovšem musíme zadat jejich parametry - to už je komplikovanější. Objevila se nám stránka s textem "Database cd - table seznam" a vypadá to tuze komplikovaně. Tak složité to ale není - nás zajímá jen několik parametrů. "Field" samozřejmě ano - zde zadáme "Interpret" na další řádek "Album" atd. "Type" nastavíme u všeho na "tinytext" (sice u

ceny to není zrovna nejlepší, ale teď nejde o detaily). No a zadáme "save". Pokud se vám to povedlo, gratuluji. Vytvořili jste tabulku ;) A jak do ní dostat data? K tomu slouží příkaz insert - objeví se vám stránka s nabídkou čtyř řádků a vy k nim do pole "value" zadáte příslušná data. Příkazem browse si je můžete prohlédnout. A tím bychom měli úvod do MySQL za sebou.

# PHP v praxi, 15. díl - diskuze v.2

(07.12. 2001)

Jak jsem sliboval, v tomto dílu si naprogramujeme jednoduchou MySQL diskuzi. Bude podobná té, kterou jste si měli možnost vytvořit na začátku seriálu. Ovšem pouze vzhledem. Zadávání i vybírání údajů bude už na zcela jiné úrovni. Doufám, že se vám podle minulého dílu podařila instalace MySQL a phpMyAdmin, je to nutný výchozí předpoklad pro zvládnutí dílu dnešního.

Nejprve si vytvoříme databázi a tabulku, kam budeme naše data ukládat. Spusťte phpMyAdmin a vytvořte si databázi (např. "webtip";). PhpMyAdmin se vám automaticky přepne na vytvořenou databázi. Do místa pro příkaz (Run SQL query/queries on database webtip...) vložte následující:

```
CREATE TABLE diskuze (
  ip text NOT NULL,
  time datetime NOT NULL default '0000-00-00 00:00:00',
  name tinytext NOT NULL,
  text mediumtext NOT NULL
)
```

Klikněte na "go" a jestli je všechno v pořádku, napíše vám: "Your SQL-query has been executed successfully". Abyste byli v obraze, co se vlastně stalo: příkaz CREATE TABLE je jasný, ale jeho argumenty už asi méně. "ip" je název jednoho pole (jakoby název sloupce), "text" je typ tohoto pole (dalo by se říci formát buňky), k čemu je "NOT NULL" není lehké vysvětlit a také to není v tomto okamžiku potřeba. Analogicky další řádek: "time" je název pole, formát je "datetime" (tzn. do buňek nebudete moci vkládat text ale jenom časový údaj), "default" je hodnota buňky, pokud nebyla žádná hodnota zadána - je zde krásně vidět právě ten formát buňky (rok-měsíc-den hodina:minuta:vteřina).

Po úspěšném vytvoření tabulky se již můžeme pustit do samotného PHP skriptu.

```
<?
$prispevku=5;
$host = 'localhost';
$user = 'root';
$pass = '';
$dbase = 'webtip';
mysql_connect ($host, $user, $pass);
mysql_select_db ($dbase);
```

Nejprve si vytvoříme několik proměnných. $prispevku - maximální počet příspěvků, které později budeme vidět na stránce. Ty další se týkají připojení k databázi.

Protože to zkoušíme na vlastním počítači, zadáváme jako host "localhost" a sebe jako "root", heslo tam pravděpodobně nastavené nemáte, takže to necháme jako prázdný řetězec. Databáze se jmenuje "webtip". Výše zmíněné proměnné také hned využijeme: funkce mysql_connect () slouží k připojení k mysql (syntaxe a pořadí argumentů je, myslím, natolik jasné, že jej není třeba dále rozebírat). Následně vybereme databázi, se kterou hodláme pracovat.

```
if (isset($sent))
{
 switch ($sent)
 {
  case "Zapsat":
   $gethost=gethostbyaddr($REMOTE_ADDR);
   $time=date("Y-m-d H:i:s", time());
   $query = "INSERT INTO diskuze (ip, time, name, text)
      VALUES ('$gethost', '$time', '$name', '$text')";
   mysql_query($query);
   break;
  case ">>":
   $od-=$prispevku;
   break;
  case "<<":
   $od+=$prispevku;
   break;
 }
}
```

Nepropadejte panice ;) Popořadě si rozebereme tento úsek: vše je ve funkci "if (isset($sent))" - proměnná sent se vytvoří, pokud něco odešleme (míní se příspěvek do diskuze nebo zobrazování starších/novějších příspěvků). Jsme-li na stránce poprvé a nic neuděláme, nic v této podmínce se neprovede. "switch ($sent)" má úlohu přepínače mezi jednotlivými akcemi, které se provedou podle hodnoty proměnné $sent (o té bude ještě řeč později). V případě, že skript má za úkol zapsat příspěvek, provede se následující: zjistí se ip adresa počítače, který na stránku přistupuje a pomocí funkce gethostbyaddr() dostaneme místo číselné adresy jméno (195.119.180.3 -> www.seznam.cz). Do proměnné $time ukládáme aktuální čas zápisu vytvořený pomocí funkce date(). A pak už opět MySQL - do proměnné $query uložíme příkaz databázi. Jeho syntaxi vidíte v ukázce: INSERT INTO jméno_tabulky (říkáme, do jaké tabulky se ta data mají uložit), (název_sloupce1, název_sloupce2...) - to vyjadřuje, kam budou patřit hodnoty (VALUES) uvedené v příslušném pořadí v další závorce. Následně se tento příkaz provede funkcí mysql_query(). Dva další případy (case) jsou důležité pro čtení starších (resp. novějších) příspěvků. Uvidíte jak.

```
if(!isset($od)||$od<0) $od=0;
$query = "SELECT * FROM diskuze";
$vysledek=mysql_query($query);
$pocet=mysql_num_rows($vysledek);
?>
```

Právě kvůli zobrazování starších příspěvků (skrolování diskuze nazpět v čase) musíme zjistit, kolik je vlastně celkem příspěvků a od kterého máme zrovna zobrazovat. Jestliže jsme si na začátku nastavili počet příspěvků na stránce na číslo

pět, při žádosti o cestu zpět musíme do proměnné $od dostat právě číslo 5. K tomu slouží $od+=$prispevku; Při zobrazování novějších by však mohlo dojít k chybě, jestliže by $od dosáhla hodnoty menší než 0. K této kontrole použijeme podmínku "if($od<0) $od=0;". Ale my ještě vůbec nemáme proměnnou $od nastavenou (jsme na stránce poprvé), proto ještě přidáme kontrolu funkcí "isset()". S počtem všech příspěvků budeme pracovat později. Hlavně ale názorně vidíte, jak se z tabulky vybírají data a dále zpracovávají. Přeložím-li příkaz do češtiny: VYBER vše Z tabulky "diskuze". Funkce mysql_num_rows() provede přesně to, co potřebujeme: zjistí počet řádků.

```
<html>
<head>
<title>Diskuze v.2</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
</head>
<body bgcolor="#FFFFFF" text="#000000">
<table width="80%" align="center">
  <tr>
    <td width="20%" align="right" valign="top">
      <form action="<? echo $PHP_SELF; ?>" method="get">
         <input type="hidden" name="od" value="<?= $od ?>">
         <input type="submit" name="sent"
             <?= ($od >=($pocet-$prispevku)) ? 'disabled' : '' ?> value="<<">
      </form>
    </td>
    <td width="60%" align="center">
      <form action="<? echo $PHP_SELF; ?>" method="get">
         <input type="text" name="text" size="40">
         <input type="text" name="name" size="10"
             value="<? if (isset($jmeno)){ echo $jmeno; } ?>">
         <input type="submit" name="sent" value="Zapsat">
      </form>
    </td>
    <td width="20%" align="left" valign="top">
      <form action="<? echo $PHP_SELF; ?>" method="get">
         <input type="hidden" name="od" value="<?= $od ?>">
         <input type="submit" name="sent"
           <?= ($od==0) ? 'disabled' : '' ?> value=">>">
      </form>
    </td>
  </tr>
</table>
```

Tady už jsme na chvíli zpět v html. Jedná se o tři formuláře - první obsahuje button na posun zpět v diskuzi, do prostředního se zapisuje text a jméno, poslední je posun k novějším příspěvkům. Apachovská proměnná $PHP_SELF; je věc k nezaplacení. Už se nemusíte starat, jak se soubor diskuze jmenuje, v $PHP_SELF; je jeho název uložen (k vyzkoušení funkčnosti stačí příkaz "echo $PHP_SELF;"). První formulář (a analogicky i třetí) předává informace v proměnných $od a $sent. Upozorňuji na příkaz "<?= ($od >=($pocet-$prispevku)) ? 'disabled' : '' ?>" - výsledkem je, že pokud $od je větší nebo rovno rozdílu všech příspěvků a jejich maximálního množství na stránku (celkem jsou tři, povoleno pět = výsledek -2), button se nastaví na disabled (to funguje bez problémů jen v Internet Exploreru, bohužel). O to ale ani tak nejde: důležitá je ta podmínka z hlediska syntaxe "<?= podmínka ? pravda : nepravda ?>". Jedná se o zkrácenou verzi if() { pravda } else { nepravda }. Právě v html zápisu jsou

tyto zkratky k nezaplacení. K druhému formuláři snad není co dodat. Pokud jste se prokousali návodem k diskuzi verze 1, nepředpokládám výraznější obtíže při zvládnutí tohoto zdrojáku.

```
<?
$query = "SELECT * FROM diskuze ORDER BY -time LIMIT ".$od.",".$prispevku;
$vysledek=mysql_query($query);
echo '<table width="80%" align="center"><tr><td>';
while ($data=mysql_fetch_array($vysledek))
{
        echo "<hr><table><tr><td>Autor: ".$data['name']." (".$data['ip'].")<br>Datum: ";
        echo $data['time']."<br><br>".$data['text']."</td></tr></table>\n";
}
echo "<hr>";
echo "</td></tr></table>";
?>
</body>
</html>
```

Tato část je zodpovědná za zobrazení požadovaných dat z databáze. Opět přeložím do češtiny, co vlastně chceme: VYBER vše Z tabulky "diskuze" a SEŘAĎ v opačném směru toku času OHRANIČENÉ zdola $od, shora maximem příspěvků na stránku. Provedení tohoto příkazu necháme v proměnné $vysledek, neboť tu budeme potřebovat pro cyklus while(). Ten také přeložím do srozumitelna (ověřil jsem si, že je to nejsnadnější a nejrychlejší cesta k pochopení problému): "dokud(je tato podmínka pravdivá) { dělej tohle; }" - podmínka je v tomto případě pravdivost přiřazení dat do pole (array) $data pomocí funkce mysql_fetch_array(). Ta bere data z tabulky pěkně po řádcích a přiřazuje je do pole. Když už data nejsou, cyklus while se zastaví, protože není pravda, že se nějaká data přiřazují. Z pole se hodnota konkrétní buňky získá jak už víme: $array['jmeno_sloupce'].

Na závěr shrnu, co se tedy všechno děje: Navštívíte takovouto diskuzi poprvé. Vytvoří se proměnné a připojíme se k databázi (první ukázka). Přeskočí se přepínač ($sent ještě neexistuje). Nastaví se proměnná $od na 0 a zjistí se počet všech příspěvků v tabulce databáze. Dojde k samotnému zobrazení stránky (tedy začátku), zjistí se, jestli je v databázi více než maximální počet příspěvků, v tom případě podmínka umožní zobrazení starších údajů (resp. tlačítka "<<"), zobrazí se i dvě okénka pro zadání textu a jména, pak ještě tlačítko pro zobrazení novějších příspěvků ">>". Nakonec se vygeneruje html kód za použití cyklu while() s výběrem z databáze. Pokud kliknete na tlačítko "<<" tedy "starší", předá se hodnota proměnné $od a $sent se nastaví na "<<". V přepínači (switch) proběhne jenom zvýšení hodnoty $od o číslo v proměnné $prispevku. To má význam při generování dat z databáze. LIMIT je zvýšen, takže se berou příspěvky od určitého počtu a to od posledního nazpět (-time). V případě vložení nového příspěvku a jeho odeslání se nastaví hodnota $sent na "Zapsat". Přepínač podle toho pozná, že má zjistit ip adresu, datum a čas a pak všechny hodnoty včetně jména a textu vložit do databáze.

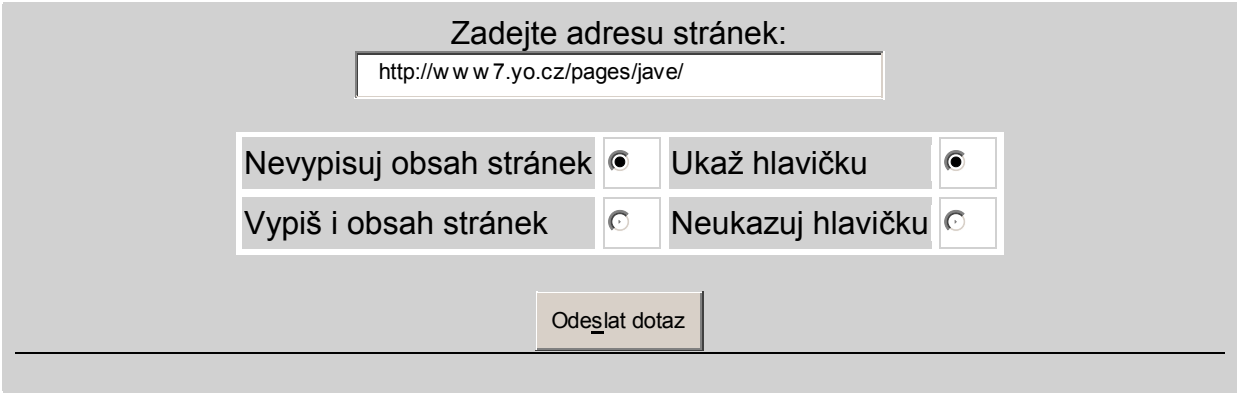
Ukázku máte jako obvykle k dispozici: webtip.oceany.cz/diskuze_v2.

# PHP v praxi, 16. díl - SiteMapper

(14.12. 2001)

"Site map" je dobře známé zaklínadlo. Občas je potřeba vidět, jak vypadá "kostra" celého webu - tzn. z jakých stránek se skládá a kam směřují jednotlivé odkazy. Velice užitečné je také zjistit, které odkazy nefungují ("broken links"). Ručně kontrolovat nějaký rozsáhlejší web je nadlidský úkon. Abyste si mohli práci ulehčit, můžete použít skript od Earla C. Terwilligera.

Skript si můžete stáhnout buď přímo ze stránek projektu nebo mnou upravenou verzi (jedná se pouze o přidání html rozhraní kvůli jednoduššímu nastavování). Užitečnost tohoto skriptu si můžete ověřit ihned:




Výsledek si samozřejmě můžete uložit a dále s ním pracovat.

Skript má i svá omezení - odkazy musí být buď přímo na konkrétní soubor nebo i na adresář, ale v tom případě odkaz musí končit lomítkem. Jinak se takový odkaz bere jako odkaz na neexistující soubor. Zbytečně nemapujte velké weby, trvá to příliš dlouho.

Tento skript je ale pro normální používání (na vlastní projekty) využitelný více než dobře.

# PHP v praxi, 17. díl - Výpis adresáře

(04.01. 2002)

Často potřebujeme automaticky vypsat obsah adresáře. Využitelné je to například tehdy, nabízíme-li soubory ke stažení, jejichž seznam se stále mění. Předělávat pokaždé html kód měněním odkazů je zdlouhavé a také - zbytečné. Ukážeme si tedy, jak řešit výpis souborů pomocí PHP.

V manuálu máte k dispozici tento relativně jasný a krátký skriptík:

```
<?
$handle=opendir('.');
while (false!==($file = readdir($handle)))
{
    if ($file != "." && $file != "..")
    {
        echo "$file<br>\n";
    }
}
closedir($handle);
```

```
?>
```

Proměnné $handle přiřazujeme funkci otevření adresáře opendir(). V závorce je potom cesta k tomuto adresáři - tečka znamená "současný adresář", dvě tečky "o jeden adresář výše", lomítko "root" atp. Pak následuje cyklus while() - ten vždy do proměnné $file zapíše jeden soubor, který přečte pomocí funkce readdir(). Ta postupně vrací soubory a adresáře. Sami si můžete zkusit dát několik "echo readdir($handle);" za sebe. Cyklus while() se zastaví v případě neúspěšného přiřazení - jinými slovy: když "dojdou" soubory. Aby se nezobrazovaly standardní adresářové cesty ("." a ".."), je v cyklu ještě podmínka na jejich ověření. Pak se už konečně vypíše soubor na obrazovku. Adresář po cyklu zase zavřeme, ať nezůstává zbytečně v paměti.

Z tohoto skriptu ovšem dostaneme něco na způsob:

```
ahhh.txt
ahoj
ble.jpg
index.php
index.php.bak
mama.miaaa
nazdar
nnnn.mpeg
```

Toto bych ohodnotil jako silně nedostatečné. Minimálně by skript ještě měl zjistit, co je soubor a co adresář, velikost souborů a datum jejich poslední změny.

```
<?
$nas_sou="index.php";
$handle=opendir('.');
$i=0;
$j=0;
while (false!==($file = readdir($handle)))
{
   if ($file!="."&&$file!=".."&&!is_dir($file)&&$file!=$nas_sou)
   {
      $soubor[$i]="$file";
      $velikost[$i]=filesize ($file);
      $zmena[$i]=date("H:i:s d.m.Y ",filemtime($file));
      $i++;
   }
   if ($file != "." && is_dir($file))
   {
     $adresar[$j]="$file";
      $j++;
   }
}
closedir($handle);
function vypis($s,$v,$z)
{
      echo "<table>\n";
      for($i=0;$i<count($s);$i++)
      {
       echo "<tr><td>";
```

```
        echo '<a href="'.$s[$i].'">'.$s[$i].'</a>';
        echo "</td><td>$v[$i]";
        echo "</td><td>$z[$i]";
        echo "</td></tr>\n";
       }
       echo "</table>\n";
}
echo "<hr>Adresáře:<hr>\n";
vypis($adresar,"","");
echo "<hr>Soubory:<hr>\n";
vypis($soubor,$velikost,$zmena);
?>
```

Takto vypadá již použitelnější skript. V cyklu přibyla kontrola adresáře pomocí funkce is_dir() a také jsme zamezili zobrazení samotného souboru ($nas_sou). Do polí $soubor[], $velikost[] a $zmena[] se předávájí informace o názvu souboru (přímé přiřazení názvu z proměnné $file), jeho velikosti (funkce filesize()) a datu poslední změny (pomocí funkce filemtime() získáme čas v sekundách od 1.1.1970 a funkcí date() s příslušnými parametry převedeme na srozumitelnější výsledek). U adresáře nás zajímá jen jeho jméno.

Zcela nová je funkce vypis (název souboru, velikost, změna), která zpracuje vložená pole na tabulku s odkazy. V ní je cyklus for(), díky němuž se dostaneme ke všem hodnotám uloženým v těchto polích.

Věřím, že vám tento skript dobře poslouží a ulehčí práci. Formátování do "pěkného" html už nechám na vás, s tím si jistě hravě poradíte. Ukázku prohlížení adesářů můžete vidět na adrese webtip.oceany.cz.

# PHP v praxi, 18. díl - WebCalendar

(23.01. 2002)

Možná jste si všimli služby Diář na serveru www.redbox.cz. Je to pěkná aplikace, ale pokud máte někde vlastní stránky s podporou PHP a MySQL (popř. jiné databáze), můžete mít podobný diář také. Osobně preferuji přístup "vše na svém", proto raději budu používat aplikaci, kterou budu mít plně pod svou kontrolou než službu poskytovanou někým cizím.

Pátral jsem tedy po něčem podobném a našel - WebCalendar. Tento program je distribuován opět pod gpl licencí, takže jej můžete využívat plně podle svých potřeb.

Upřímně mne překvapil svoji komplexností, funkčností a přitom jednoduchostí. Kromě standardních možností kalendáře (denní/týdenní/měsíční/roční rozpisy, přidávání záznamů apod.) také plně využívá svoji internetovou podstatu - např. upozorňování na e-mail nebo velmi dobře fungující podpora více uživatelů. Jeden má práva administrátora - ten může přidávat nové uživatele (a také je mazat a měnit jim hesla, proto je nutné, aby šlo o důvěryhodného člověka). Ostatní jsou obyčejní uživatelé, kteří ovšem kromě výše zmíněné možnosti mohou bez omezení využívat všech schopností WebCalendáře.

Velkou předností je možnost přidat záznam i ostatním uživatelům. Např. si potřebujete domluvit poradu - přidáte nový záznam a zvolíte i ostatní uživatele.

Potvrdíte a těm zvoleným se zašle e-mail s textem, obsahujícím informaci o potřebě potvrzení tohoto záznamu. Vy zatím máte u těchto uživatelů otazníček. Jakmile však poradu ve svém kalendáři potvrdí, také u vás dojde ke změně - budou označeni za účastníky. Když tento záznam smaže jeho tvůrce, smaže se i ostatním uživatelům a pošle se jim e-mail o této změně. Pokud jej smaže pouhý účastník, pouze ho to vyškrtne ze seznamu.

Dalšími příjemnými vlastnostmi je čeština (ne sice úplná, ale použitelná) nebo ukládání hesla do cookies (lze samozřejmě kvůli bezpečnosti zakázat).

Doufám, že jsem vás dostatečně zlákal, tak se pustíme do instatalace.

1. Stáhněte si poslední verzi ve formátu tar.gz (to zvládá např. WinRAR). Rozbalte ji do adresáře, kde chcete tuto aplikaci provozovat.
2. Najděte si soubor *tables-mysql.sql*. Vytvořte si novou databázi v MySQL (např. "*webcalendar*") a spusťte na ni příkaz obsažený v tomto souboru (*Run SQL query/queries on database webcalendar* -> vložit text do formuláře -> *go*).
3. Zeditujte soubor *config.inc* v adresáři *includes*. Bude nás zajímat řádek 25 - 28:

```
// MySQL example
$db_type = "mysql";
$db_host = "localhost";
$db_login = "root";
$db_password = "";
$db_database = "webcalendar";
```

Zde nastavte podle svých potřeb práva a informace o serveru. Toto by měla být věškerá nutná instalace. Nyní si již můžete v prohlížeči WebCalendar spustit. Objeví se vám okno požadující po vás přístupové jméno a heslo. Poprvé zadejte admin/admin. V nabídce "Uživatelé" jej pak můžete změnit.
Pokud se vám vše povedlo, máte k dispozici rychlý a efektivní multiuživatelský kalendář. Blahopřeji ;)

# PHP v praxi, 19. díl - Výpis všech adresářů a souborů

(07.02. 2002)

V 17. díle jsme si ukázali jednoduché řešení pro výpis jediného adresáře. Jsou ale případy, kdy potřebujeme mít přehled o kompletní adresářové struktuře. Tedy nejen obsah jednoho konkrétního adresáře, ale i všech dalších adresářů v jeho podadresářích až k tomu poslednímu.

Ukážeme si nejdříve cyklus, na kterém skript postavíme.

```
$i=0;
$adr=opendir($pozice);
while ($file = readdir($adr))
{
   if ($file!="."&&$file!="..")
   {
```

```
      $polozka[$i]['name']=$file;
      $polozka[$i]['pozi']=$pozice;
      $polozka[$i]['size']=filesize($pozice.$file);
      $polozka[$i]['time']=date("H:i:s d.m.Y",filemtime($pozice.$file));
      if (is_dir($pozice.$file))
      {
            $polozka[$i]['type']="dir";
            $dir2handle[count($dir2handle)]=$pozice.$polozka[$i]['name']."/";
      }
      else
      {
            $polozka[$i]['type']="file";
      }
      $i++;
   }
}
```

Hned na začátku operujeme s pomocnou proměnnou $i - ta se na konci cyklu vždy zvýší o jedna ($i++) a později se stává identifikujícím prvkem pro pole $polozka. Funkcí opendir() otevřeme pro zpracování adresář, k němuž je cesta uložena v proměnné $pozice. Nyní while() při každém cyklu přiřazuje proměnné $file jednu položku z tohoto adresáře. Podmínka if() je jasná - stejně jako při prohlížení jednoho adresáře v 17. díle i zde odstraňujeme adresářové cesty na současný adresář a adresář o jeden výše.

Následuje přiřazování různých vlastností vícerozměrnému poli (multidimensional array) $polozka[][]. S jednoduchým polem jsme se již setkali. Je to vlastně proměnná obsahující více informací najednou a přistupujeme k ní přes identifikátor mezi hranatými závorkami. Vícerozměrná znamená, že se další informace skrývají pod dalšími identifikátory v dalších hranatých závorkách. Pro názornost: pole $polozka[0][name] je jméno souboru, který se načetl jako první (začínáme jako obvykle od nuly), $polozka[0][size] je velikost tohoto souboru. $polozka[1][pozi] však už patří k souboru (či adresáři) dalšímu. Tímto způsobem získáme všechna data dobře přístupná v jednom poli, protože $i se zvyšuje vždy po ukončení jednoho celého cyklu (jinými slovy: až se dokončí všechny operace s jedním souborem či adresářem).

Do $polozka[$i]['name'] se ukládá jméno dotyčného souboru (či adresáře), $polozka[$i]['pozi'] obsahuje jméno nadřazeného adresáře (to je důležité pro zjištění struktury a jednotlivých vazeb souborů a adresářů mezi sebou), v $polozka[$i]['size'] se pomocí funkce filesize() zjistí velikost souboru (u adresáře to bude vždy 0, cesta k souboru je udána $pozice.$file, což je skutečná a úplná cesta k dotyčnému souboru). Do $polozka[$i]['time'] se zapíše pomocí již známé funkce date() datum poslední změny souboru (funkce filemtime()). Následuje podmínka, jež má za úkol zjistit, zda-li je aktuálně zpracovávaná položka soubor či adresář. Podle výsledku se poté poli $polozka[$i]['type'] přiřadí buď řetězec "file" nebo "dir". Jestliže se jedná o adresář, uloží se kompletní cesta k němu do pole $dir2handle[]. Jeho identifikátor vytvoříme tak, že sečteme počet jeho identifikátorů (0,1,2,3,4 - počet 5) - pokud neobsahuje nic, začneme číslem 0. Tím velice jednoduše získáme identifikátor vždy o 1 větší než je ten poslední.

Tento cyklus je však funkční pozue pro jeden adresář. Abychom jej mohli použít úplně pro všechny adresáře, musí se stále měnit $pozice a to do té doby, než vyčerpáme všechny adresáře určené ke zpracování. Provedeme tedy následující:

```
function loop_dir($pozice)
{
        GLOBAL $dir2handle, $polozka;
        $i=count($polozka);
        $adr=opendir($pozice);
        while ($file = readdir($adr))
        {
           if ($file!="."&&$file!="..")
           {
              $polozka[$i]['name']=$file;
              $polozka[$i]['pozi']=$pozice;
              $polozka[$i]['size']=filesize($pozice.$file);
              $polozka[$i]['time']=date("H:i:s d.m.Y",filemtime($pozice.$file));
              if (is_dir($pozice.$file))
              {
                    $polozka[$i]['type']="dir";
                    $dir2handle[count($dir2handle)]=$pozice.$polozka[$i]['name']."/";
              }
              else
              {
                    $polozka[$i]['type']="file";
              }
              $i++;
           }
        }
}
$dir2handle[0]='./';
for($j=0;$j<(count($dir2handle));$j++)
{
        loop_dir($dir2handle[$j);
}
```

Cyklus jsme včlenili do námi definované funkce loop_dir(). Voláme ji v dalším cyklu, for(). První cyklus probíhá následovně: pole $dir2handle[0] obsahuje './', tedy současný adresář. Cyklus for probíhá od 0 až do čísla menšího než počet položek v poli $dir2handle[]. Tam máme nyní jednu položku, takže cyklus by měl probehnout pouze jednou - pro nulu. Zavolá se funkce loop_dir(), která předá proměnné $pozice obsah pole $dir2handle[0], tedy to './'. Jak už jsme si říkali, z funkcí je omezen výstup proměnných, proto nastavujeme pomocí GLOBAL tzv. globální proměnné, tedy ty, jež budeme moci využívat i mimo naši funkci. Pomocné $i se poprvé skutečně přiřadí 0 (pole $polozka[] je zatím prázdné). Dále se zpracuje současný adresář přesně tak, jak jsme si ukázali výše. Výstupem jsou tedy pole $dir2handle[] a $polozka[][]. $dir2handle[] obsahuje všechny adresáře, jež máme za úkol dále zpracovat a v $polozka[][] jsou úplně všechny dosud získané informace o souborech a adresářích.

Tím jsme ovšem neskončili! Na konci cyklu for() se k pomocné proměnné $j přičetla jednička ($j++). Kdyby se nic nezměnilo, cyklus by se ukončil, jenže pokud námi zvolený adresář obsahoval další adresáře, jsou přeci v poli $dir2handle[], které tedy změnilo počet svých položek a z toho důvodu se o ně navýšilo omezení tohoto cyklu. Takže se funkci loop_dir() předá další adresář ke zpracování a tak to bude pokračovat až do doby, kdy již nebudou žádné další adresáře k dispozici.

Vícerozměrné pole $polozka[][] se mezitím zaplní všemi adresáři a soubory, jež těmito cykli prošly.

Půlka celého problému je za námi, nyní přejdeme k řešení problému, jak ta data dostat v nějaké rozumné podobě na obrazovku.

```
sort ($dir2handle);
for($i=0;$i<(count($dir2handle));$i++)
{
		echo '<b><a name="'.$dir2handle[$i].'">';
		echo $dir2handle[$i]."</a></b><br><blockquote>";
		for($j=0;$j<(count($polozka));$j++)
		{
			if($polozka[$j]['pozi']==$dir2handle[$i]&&$polozka[$j]['type']=="dir")
			{
				echo '<a href="#'.$polozka[$j]['pozi'];
				echo $polozka[$j]['name'].'/">';
				echo $polozka[$j]['name']."</a> (dir)<br>";
			}
		}
		for($j=0;$j<(count($polozka));$j++)
		{
			if($polozka[$j]['pozi']==$dir2handle[$i]&&$polozka[$j]['type']=="file")
			{
				echo '<a href="'.$polozka[$j]['pozi'];
				echo $polozka[$j]['name'].'">';
				echo $polozka[$j]['name']."</a> (";
				echo $polozka[$j]['size']." B, ";
				echo $polozka[$j]['time'].")<br>";
			}
		}
			echo "</blockquote>";
}
```

Nejprve se pomocí funkce sort() abecedně seřadí pole $dir2handle. Dále následují dva do sebe vnořené cykly, jež mají za úkol porovnat nadřazené adresáře všech položek s obsahem $dir2handle[] a podle toho je vypsat či nevypsat. Aby se od sebe logicky oddělily adesáře a soubory, jsou v cyklu cykly dva, první pro adresáře, druhý pro soubory. Vypsání jednotlivých části pole $polozka[][] a html tagy pro vytvoření odkazů je už předpokládám jasné. Toto rozhodně není jediné možné řešení problému, jistě vymyslíte lepší, nám ale zatím stačí.

Nyní si ukážeme celý zdrojový kód včetně zadávání cesty přes formulář a kontroly správnosti zadání:

```
<html>
<body>
<form>
<input type="text" name="cesta" value="<?= $cesta ?>">
<input type="submit">
</form>
<?
if (isset($cesta))
{
		if(is_dir($cesta))
```

```
	{
		if(substr($cesta,-1)!="/")
		{
			$dir2handle[0]=$cesta."/";
		}
		else
		{
			$dir2handle[0]=$cesta;
		}
	}
	else
	{
		echo "Zadaná cesta <b>$cesta</b> není platným adresářem<br>";
		$dir2handle[0]='./';
	}
}
else
{
	$dir2handle[0]='./';
}
function loop_dir($pozice)
{
	GLOBAL $dir2handle, $polozka;
	$i=count($polozka);
	$adr=opendir($pozice);
	while ($file = readdir($adr))
	{
	   if ($file!="."&&$file!="..")
	   {
	      $polozka[$i]['name']=$file;
	      $polozka[$i]['pozi']=$pozice;
	      $polozka[$i]['size']=filesize($pozice.$file);
	      $polozka[$i]['time']=date("H:i:s d.m.Y",filemtime($pozice.$file));
	      if (is_dir($pozice.$file))
	      {
		$polozka[$i]['type']="dir";
		$dir2handle[count($dir2handle)]=$pozice.$polozka[$i]['name']."/";
	      }
	      else
	      {
		$polozka[$i]['type']="file";
	      }
	      $i++;
	   }
	}
}
for($i=0;$i<(count($dir2handle));$i++)
{
	loop_dir($dir2handle[$i]);
}
sort ($dir2handle);
for($i=0;$i<(count($dir2handle));$i++)
{
	echo '<b><a name="'.$dir2handle[$i].'">';
	echo $dir2handle[$i]."</a></b><br><blockquote>";
	for($j=0;$j<(count($polozka));$j++)
	{
		if($polozka[$j]['pozi']==$dir2handle[$i]&&$polozka[$j]['type']=="dir")
		{
			echo '<a href="#'.$polozka[$j]['pozi'];
```

```
                                        echo $polozka[$j]['name'].'/">';
                                        echo $polozka[$j]['name']."</a> (dir)<br>";
                                }
                        }
                        for($j=0;$j<(count($polozka));$j++)
                        {
                                if($polozka[$j]['pozi']==$dir2handle[$i]&&$polozka[$j]['type']=="file")
                                {
                                        echo '<a href="'.$polozka[$j]['pozi'];
                                        echo $polozka[$j]['name'].'">';
                                        echo $polozka[$j]['name']."</a> (";
                                        echo $polozka[$j]['size']." B, ";
                                        echo $polozka[$j]['time'].")<br>";
                                }
                        }
                                echo "</blockquote>";
}
?>
</body>
</html>
```

Do budoucna bych chtěl napojit tento výpis adresářové struktury s nečím podobným SiteMapperu ze 16. dílu. Tím bychom dostali kromě informací o slepých odkazech také zprávu o tom, na které soubory pro změnu žádný odkaz nevede.

# PHP v praxi, 20. díl - Přístupová práva

(15.02. 2002)

Windows 95 a 98 usnadňují v některých případech práci až příliš. Přístupová práva jsou právě jedním z těchto zjednodušení - prakticky nejsou (až na "jen ke čtení"). Díky tomu vzniká problém, chce-li uživatel umístit na serveru stránky, obsahující určité interaktivní prvky (počítadlo, diskuze), jež se neobejdou bez možnosti zápisu do souboru. Ke změně souboru ale dojít nemůže, protože onen soubor nemá správně nastavena práva k zápisu. A tady uživatel Windows končí - o co se vlastně jedná? Jaká práva? Jak je nastavit?

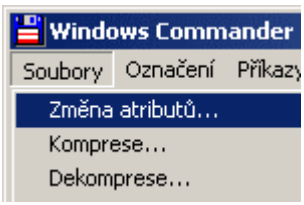
Servery provozující hosting používají mnohem častěji Linux a ten už řeší otázku bezpečnosti mnohem zodpovědněji. Každému souboru (resp. adresáři) jsou přiřazena určitá práva - kdo jej může číst, editovat nebo spouštět. Nás tento problém nezajímá z pohledu uživatele Linuxu ale jako majitele účtu u poskytovatele prostoru pro www stránky. Proto si můžeme dovolit i některá zjednodušení.

Nejprve se podíváme, jak bez dalších podrobnějších znalostí správně nastavit právo k zápisu pro určitý soubor. V základní verzi diskuze, kde jsme používali zápis dat do souboru, jsme potřebovali nastavit souboru note.txt právo k zápisu. Nejjednodušším způsobem může být použití Windows Commandera.

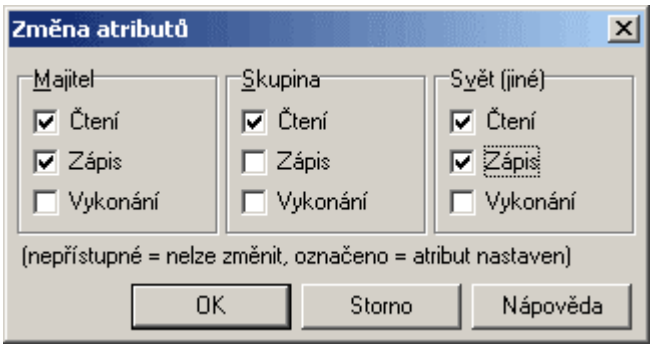
*Pozn: Pokud se vám následující návody nepodaří aplikovat, není chyba s největší pravděpodobností ve vašem konání. Některé servery změnu práv nepodporují.*

Postup:

1. Připojíte se standardním způsobem ke svému ftp účtu.
2. Označíte příslušný soubor.
3. V menu vyberete nabídku "Soubory" a zvolíte "Změna atributů".

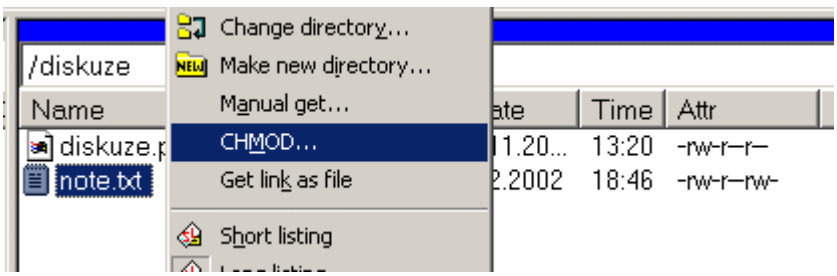



4. V tabulce již snadno nastavíte práva k zápisu.

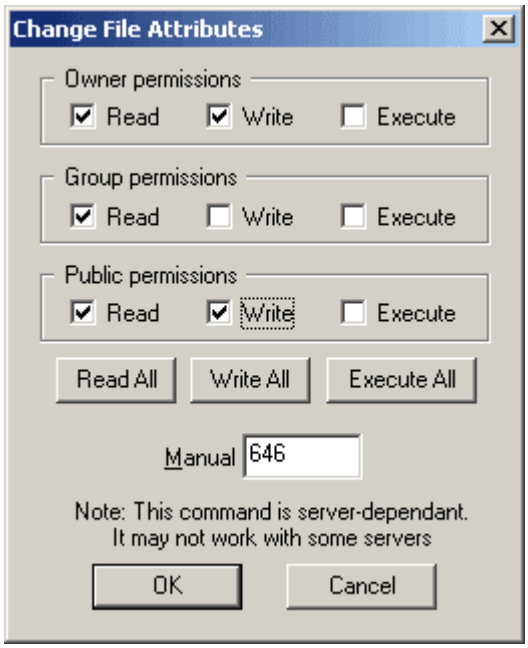



Používáte-li radši Cute-FTP, pak po připojení k serveru:

1. Klikněte na příslušný soubor pravým tlačítkem myši.
2. Z nabídky zvolte příkaz CHMOD.




3. Objeví se vám nabídka podobná té z Windows Commandera.




CuteFTP ovšem nabízí již více informací, jež si probereme podrobněji, abychom pochopili také samotný princip přístupových práv. Jak již bylo řečeno a ukázáno, existují tři druhy vztahů k "vlastnictví" souboru (resp. adresáři). Jedním je samotný vlastník, druhým skupina, do které patří, a třetím jsou všichni ostatní. Nás zajímá především vlastník a potom ostatní, skupinou se zabývat nemusíme.
U první ukázky z CuteFTP si můžete všimnout kolonky Attr (Attributes).
**-rw-r--rw-** znamená následující: minus na začátku určuje, že jde o soubor. První trojice (**rw-**) jsou přístupová práva pro vlastníka souboru, druhá trojice (**r--**) určuje práva pro skupinu a třetí trojice (**rw-**) určuje práva pro ostatní. Ve všech vlastnických skupinách je možno povolit nebo zakázat stejné operace rwx. "**r**" znamená, že soubor je povoleno číst, "**w**" - do souboru je povoleno zapisovat a "**x**" - soubor je povoleno spustit.
Rozdíl mezi soubory a adresáři se promítá i do významu příslušných práv: "**r**" - adrésář je povoleno vypsat, "**w**" do adresáře je povoleno zapisovat, vytvářet a rušit v něm soubory, "**x**" - do adresáře je možno vstoupit. Zápis vypadá např. takto: **drwxr-xr-x**. "**d**" znamená directory, jde o označení adresáře.
Místo značek rwx lze použít také čísla. Proto soubor s právy -rw-r--rw- může být jinak označen jako 646. Každá z těchto číslic představuje kombinaci práva pro jednotlivé vlastníky a to ve standardním pořadí (vlastník, skupina, ostatní). Následující tabulka obsahuje přehled všech číselných kombinací:

| číslo | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| práva | --- | --x | -w- | -wx | r-- | r-x | rw- | rwx |

Ke změně přístupových práv se na serverech používá příkaz chmod a to způsobem "chmod 644 note.txt" nebo "chmod u=rwx,go=r note.txt". Zde se dávají všechna práva vlastníkovi (u jako user) a práva načtení skupině a ostatním (group, others).
Abychom se vůbec věnovali PHP, manuálová stránka zabývající se funkcí chmod() je na adrese http://www.php.net/manual/cs/function.chmod.php ;)

# PHP v praxi, 21. díl - soubor.php?jak=na&t=o

(22.02. 2002)

Pro rozsáhlejší weby jsme zvolili metodu šablony, která se na začátku každého souboru načetla a dále využívala (viz PHP v praxi, 8.díl - šablona). Máte-li však menší web s pár stránkami, může pro vás být jeho správa mnohem jednodušší, uděláte-li jej pomocí PHP jako jeden jediný soubor. Vše se dá vyřešit velice prostě pomocí předávání proměnných přes odkaz. Podíváme se na první ukázku:

```
<h1>muj web</h1>
<a href="?txt=novinky">novinky</a> |
<a href="?txt=kdo_jsem">kdo jsem</a> |
<a href="?txt=reference">reference</a>
<br><br>
<?
if (!isset($txt)) $txt="novinky";
switch ($txt)
{
        case "novinky":
                echo "dnes byl spusten tento web...";
                break;
        case "kdo_jsem":
                echo "webmaster tohoto webu.";
                break;
        case "reference":
                echo "zatim nemam zadne reference.";
                break;
}
?>
```

V odkazech tentokrát není žádný soubor, nýbrž řetězec znaků, který se předá PHP skriptu. Protože cesty jsou relativní, bude výsledná adresa např. "http://www.neco.cz/soubor.php?txt=novinky". Otazníkem se odděl adresa od proměnných. Pokud je tedy tímto způsobem zavolána stránka soubor.php, automaticky se v ní vytvoří proměnná $txt a jejím obsahem bude řetězec "novinky". V našem skriptu je jedna podmínka if(), jež slouží pro první načtení stránky (kdy je volána adresa "http://www.neco.cz/soubor.php" a proměnná $txt se nevytvoří odkazem). Skript pokračuje přepínačem switch(), jehož význam i syntaxi jsme již probírali.

Podobným způsobem můžeme předávat i více proměnných - budou se oddělovat znakem &.

```
<?
if (!isset($txt))
{
        $txt="novinky";
        $min="odnikud";
}
?>
<h1>muj web</h1>
<a href="?txt=novinky&min=<?= $txt ?>">novinky</a> |
<a href="?txt=kdo_jsem&min=<?= $txt ?>">kdo jsem</a> |
<a href="?txt=reference&min=<?= $txt ?>">reference</a>
```

```
<br><br>
<?
switch ($txt)
{
	case "novinky":
		echo "dnes byl spusten tento web...";
		break;
	case "kdo_jsem":
		echo "webmaster tohoto webu.";
		break;
	case "reference":
		echo "zatim nemam zadne reference.";
		break;
}
?>
<br><br>
prisel jsi ze stranky: <?= $min ?>
```

Můžeme např. předávat informaci, ze které naší stránky uživatel přišel (k tomu se používá jiný způsob ale tento se výborně hodí k ukázce). Zde už tedy nastavení hodnoty $txt musíme předřadit odkazům, neboť ji v nich budeme potřebovat. Za proměnnou $min tedy dosazujeme aktuální stránku. Ta zvláštní konstrukce <?= $txt ?> je jedním z řady zjednodušení PHP syntaxe. Ve "standardní řeči" znamená <? echo $txt; ?>. Tímto způsobem se vypisuje i proměnná $min na konci skriptu.

U webu tak do pěti stran může být sloučení šablony a obsahu stránek velice výhodné. Na server posíláte vždy jen jeden soubor a přitom jeho velikost není přehnaně veliká. Předávání proměnných odkazy ale může být použito i v rámci rozsáhlejšího webu např. u náhledů s obrázky, jejichž plnou velikost však chcete vidět v rámci stránky, nikoliv jako samostatný soubor. Na stránce obrazky.php máte tedy 20 náhledů fotografií a u každé z nich odkaz ve tvaru "?show=01.jpg". Skript ve stránce pak vypadá např. takto:

```
<h1>fotky</h1>
<?
if (!isset($show)||$show=="nahledy")
{
?>
	<a href="?show=foto1.jpg">
	<img src="foto1_nahled.jpg">
	</a>
	<a href="?show=foto2.jpg">
	<img src="foto2_nahled.jpg">
	</a>
<?
}
else
{
?>
	<img src="<?= $show ?>"><br>
	<a href="?show=nahledy">nahledy</a>
<?
}
?>
```

Přes proměnnou $show se tentokrát předává název konkrétního obrázku. Tady ovšem už je na místě určité ošetření, protože kdokoliv může do prohlížeče zadat adresu ve tvaru "http://www.neco.cz/soubor.php?show=http://www.jinde.cz/jiny_obrazek.jpg". To sice nepředstavuje nějaké nebezpečí, ale kdybychom takovýmto způsobem otevírali něco jiného než jen obrázky, mohlo by. Jeden ze způsobů, jak toto obejít, je použití přepínače:

```
<h1>fotky</h1>
<?
if (!isset($show)||$show=="nahledy")
{
?>
		<a href="?show=foto1">
		<img src="foto1_nahled.jpg">
		</a>
		<a href="?show=foto2">
		<img src="foto2_nahled.jpg">
		</a>
<?
}
else
{
		switch($show)
		{
			case "foto1":
				$ukaz="foto1.jpg";
				break;
			case "foto2":
				$ukaz="foto2.jpg";
				break;
		}
?>
		<img src="<?= $ukaz ?>"><br>
		<a href="?show=nahledy">nahledy</a>
<?
}
?>
```

Zde je názorně vidět, že se do následného odkazu na obrázek nedostane nic nechtěného, pouze předdefinované cesty k souborům. Není to však nikterak krátké. Lepší je využití pole:

```
<h1>fotky</h1>
<?
if (!isset($show)||$show=="nahledy")
{
?>
		<a href="?show=0">
		<img src="foto1_nahled.jpg">
		</a>
		<a href="?show=1">
		<img src="foto2_nahled.jpg">
		</a>
<?
}
```

```
else
{
        $ukaz=array("foto1.jpg","foto2.jpg");
?>
        <img src="<?= $ukaz[$show] ?>"><br>
        <a href="?show=nahledy">nahledy</a>
<?
}
?>
```

Zde využíváme automatického počítání v poli $ukaz. V něm jsou opět zapsané jednotlivé cesty, ke kterým se přistupuje pomocí indexu (proměnná $show).

V jednoduchosti je síla, ale nesmí být na úkor přehlednosti a bezpečnosti. Nezapomeňte, že tímto způsobem se dá nastavit jakákoliv proměnná, proto by na začátku práce s ní měla být nastavena na určitou hodnotu, aby se zabránilo olivnění "z vnějšku".

# PHP v praxi, 22. díl - Vytvoření zmenšených obrázků

(01.03. 2002)

Častým obsahem nejen osobních stránek jsou fotografie. A kolik je s nimi práce! Nepoužijete-li nějaký speciální software, znamená příprava stránky několik časově velmi náročných úkonů. Nejprve všechny obrázky zmenšit pro náhledy a pak vytvořit HTML dokument, kde bude zobrazen každý náhled a na něm odkaz na velkou fotografii. Jistě, existují na to programy, ale jak uvidíte, není třeba žádný z nich instalovat. S pár řádky PHP kódu to dokážete vy sami!

V tomto díle bude naším úkolem vytvořit náhledy (často se setkáte s názvem "thumbnails") fotografií a to - logicky - v celém adresáři najednou. K tomu využijeme znalostí z dílu PHP v praxi, 17. díl - Výpis adresáře. Kromě toho je třeba zapnout podporu práce s obrázky. Pod Windows se to dělá jednoduše tak, že zeditujeme php.ini (ve složce C:\Windows nebo C:\WINNT) a to následujícím způsobem:
1) Nastavíme extension_dir na adresář, kde máme soubor php_gd.dll. Obvykle to bývá "c:\php\extensions\".
2) Odkomentujeme (odstraníme středník před) "extension=php_gd.dll".

Logická struktura PHP skriptu bude vypadat přibližně takto: Nejprve musí být k dispozici HTML formulář, ve kterém zadáme umístění souborů, způsob změny jejich velikosti (pevná velikost všech vzniklých náhledů nebo poměrná), jejich jména a kvalitu výsledného jpegu. Samotný PHP skript pak musí načíst všechny soubory v daném adresáři, vyřadit ty, s nimiž nechceme pracovat, a nakonec u těch správných změnit velikost a uložit je do souborů.

Načtení souborů a jejich selekce:

```
$adresar_name=dirname($cesta)."/";
$adresar=opendir($adresar_name);
$j=0;
```

```
while ($soubor=readdir($adresar))
{
		@$overeni=GetImageSize($adresar_name.$soubor);
		if ($soubor!="."&&$soubor!=".."&&$overeni[2]==2)
		{
			$in_name[$j]=$adresar_name.$soubor;
			$out_name[$j]=$adresar_name.$prefix.$soubor;
			$j++;
		}
		elseif($soubor!="."&&$soubor!=".."&&(!$overeni||$overeni[2]!=2))
		{
			echo "Nelze pracovat se souborem: $soubor  <br>";
		}
}
closedir($adresar);
```

Z HTML formuláře získáme cestu ke konkrétnímu souboru. Z ní pomocí funkce dirname() získáme jméno adresáře určeného ke zpracování. Ten pak již jednou probíraným způsobem otevřeme a cyklem while() z něj získáme všechny soubory. Novinkou je použití funkce GetImageSize(). Předáváme jí celou cestu k dotyčnému souboru a ona zjistí hned několik věcí - šířku, výšku a typ obrázku - a vrátí je jako pole. K jednotlivým informacím se pak přistupuje přes indexy. $overeni[] je tedy pole s těmito informacemi a použitím indexu 2 se dozvíme, zda se jedná o jpeg (1 = GIF, 2 = JPG, 3 = PNG, 4 = SWF, 5 = PSD, 6 = BMP). Pracovat se soubory tedy budeme jedině v případě, že $overeni[2]==2. Proč je před $overeni[] zavináč? Ten zakazuje výpis chybové hlášky. My totiž GetImageSize() využíváme též k tomu, aby nám sdělila, zda se vůbec o obrazový soubor jedná. V případě, že nikoliv, nastane chyba. To ověřujeme v druhé podmínce - (!$overeni||$overeni[2]!=2) - tzn. když $overeni[] vůbec neexistuje nebo soubor není jpeg, vypíše se "Nelze pracovat se souborem...". Tímto jsme vybrali jen a pouze jpeg soubory a jejich jména dosadíme do polí $in_name[] a $out_name[]. Pole $out_name[] se liší v tom, že soubor se jmenuje podle zadání jinak - např. fotka má název "foto.jpg" ($in_name[]) a náhled bude "t_foto.jpg" ($out_name[]).

Získání konkrétních názvů souborů bylo cílem této části skiptu. Samotné jejich zpracování je cílem části následující:

```
for($i=0;$i<count($in_name);$i++)
{
		$in_size=GetImageSize($in_name[$i]);

		if($velikost=="pevna")
		{
			if($prevratit=="ano"&&$in_size[0]<$in_size[1])
			{
				$x=$p_y;
				$y=$p_x;
			}
			else
			{
				$x=$p_x;
				$y=$p_y;
			}
		}
```

```
        elseif($velikost=="pomerna")
        {
                $x=$pomer*$in_size[0] /100;
                $y=$pomer*$in_size[1] /100;
        }
        $out_size=array($x,$y);

        $in=ImageCreateFromJPEG($in_name[$i]);
        $out=ImageCreate($out_size[0],$out_size[1]);
        ImageCopyResized($out,$in,0,0,0,0,$out_size[0],
          $out_size[1],$in_size[0],$in_size[1]);
        ImageJpeg($out,$out_name[$i],$out_q);
        ImageDestroy($in);
        ImageDestroy($out);
}
```

Kromě názvů potřebujeme také znát (vstupní) velikosti fotek a (výstupní) velikosti náhledů. Pole $in_size[] získá funkcí GetImageSize pod indexem 0 a 1 šířku a výšku fotek. Pomocí informací z formuláře a vstupních rozměrů vypočítáme výslednou velikost náhledů. V prvém případě - velikosti pevné - provedeme ještě větvení podmínek podle orientace fotek. Nevypadalo by dobře, kdyby i fotky na výšku byly zmenšené stejně jako fotky na šířku (v případě, že nechcete náhledy jako čtverce, samozřejmě ;) . Proto se zjišťuje, zda-li bylo ve formuláři zaškrtnuto převracení a dále se ověří, jestli je šířka fotky menší než její výška. Poměrná velikost se spočítá jednoduše z původních velikostí (jako procenta). Výsledek se dosadí do pole $out_size[].

Nyní přistupme již k samotné tvorbě obrázků. Funkcí ImageCreateFromJPEG() vytvoříme v proměnné $in obsah obrázku načteného ze souboru určeného polem $in_name[]. Výstupní obrázek - zatím virtuálně - vytvoříme v proměnné $out funkcí ImageCreate() s parametry velikosti v poli $out_size[]. Následující funkcí ImageCopyResized() provedeme tolik očekávanou operaci změny velikosti původního obrázku a jeho kopii do obrázku v $out. Z virtuálního obrázku se stane skutečný soubor funkcí ImageJpeg(), jejímiž parametry jsou kromě obrázku také jméno souboru náhledu a kvalita výsledného jpegu (zadá se také přes formulář). Nakonec po sobě "uklidíme" standardním způsobem pomocí ImageDestroy().

Celá naše aplikace včetně formuláře a pár dalších drobností bude vypadat následovně:

```
<html>
<body>
<h1>Náhledy - Resizer</h1>
<?
if(!isset($submit)||empty($cesta)){
?>
<form action="<?= $PHP_SELF ?>" method="get">
  <table width="400" border="0" cellpadding="5">
    <tr>
      <td>Cesta k adresáři: </td>
      <td colspan="2">
        <input type="file" name="cesta">
      </td>
    </tr>
```

```
    <tr>
      <td>Velikost: </td>
      <td>šířka</td>
      <td>výška</td>
    </tr>
    <tr>
      <td>
        <input type="radio" name="velikost" value="pevna" checked>
        Pevná:</td>
      <td>
        <input type="text" name="p_x" value="150" size="5">
        px</td>
      <td>
        <input type="text" name="p_y" value="100" size="5">
        px</td>
    </tr>
    <tr>
      <td>  </td>
      <td colspan="2">
        <input type="checkbox" name="prevratit" value="ano" checked>
        Převrátit podle orientace fotografie</td>
    </tr>
    <tr>
      <td>
        <input type="radio" name="velikost" value="pomerna">
        Poměrná: </td>
      <td colspan="2">
        <input type="text" name="pomer" value="25" size="5">
        % </td>
    </tr>
    <tr>
      <td>Prefix náhledů:</td>
      <td colspan="2">
        <input type="text" name="prefix" value="t_" size="5">
      </td>
    </tr>
    <tr>
      <td>Kvalita (jpg):</td>
      <td colspan="2">
        <input type="text" name="q" value="75" size="5">
        %</td>
    </tr>
  </table>
  <br>
  <input type="submit" name="submit" value="Start">
  <input type="reset" name="reset" value="Reset">
</form>
<?
}else{

if($q>0||$q<=100)
{
        $out_q=$q;
}
else
{
        $out_q="75";
}

$adresar_name=dirname($cesta)."/";
```

```
$adresar=opendir($adresar_name);
$j=0;
while ($soubor=readdir($adresar))
{
	@$overeni=GetImageSize($adresar_name.$soubor);
	if ($soubor!="."&&$soubor!=".."&&$overeni[2]==2)
	{
		$in_name[$j]=$adresar_name.$soubor;
		$out_name[$j]=$adresar_name.$prefix.$soubor;
		$j++;
	}
	elseif($soubor!="."&&$soubor!=".."&&(!$overeni||$overeni[2]!=2))
	{
		echo "Nelze pracovat se souborem: $soubor <br>";
	}
}
closedir($adresar);

for($i=0;$i<count($in_name);$i++)
{
	$in_size=GetImageSize($in_name[$i]);

	if($velikost=="pevna")
	{
		if($prevratit=="ano"&&$in_size[0]<$in_size[1])
		{
			$x=$p_y;
			$y=$p_x;
		}
		else
		{
			$x=$p_x;
			$y=$p_y;
		}
	}
	elseif($velikost=="pomerna")
	{
		$x=$pomer*$in_size[0] /100;
		$y=$pomer*$in_size[1] /100;
	}
	$out_size=array($x,$y);

	$in=ImageCreateFromJPEG($in_name[$i]);
	$out=ImageCreate($out_size[0],$out_size[1]);
	ImageCopyResized($out,$in,0,0,0,0,$out_size[0],
	  $out_size[1],$in_size[0],$in_size[1]);
	ImageJpeg($out,$out_name[$i],$out_q);
	ImageDestroy($in);
	ImageDestroy($out);
}

echo '<h2>Hotovo</h2><a href="'.$PHP_SELF.'">Návrat</a>';
}
?>
</body>
</html>
```

Ve formuláři používáme proměnnou $PHP_SELF, ta vždy ukrývá název konkrétního souboru (PHP skriptu), takže při jeho přejmenování nemusíte měnit odkazy. Kvalita

výsledného jpegu se kontroluje, aby byla mezi 0 a 100. Jinak je vše předpokládám jasné.

Vidíte - stačí nám soubor menší než tři kilobajty na výsledek srovnatelný s mnohdy drahým softwarem. Pravda, je stále co vylepšovat, o tom není pochyb, ale na tvorbu jednoduchým náhledů to bohatě stačí. Sami můžete prozkoumat další PHP funkce týkající se obrázků, možností je mnoho - třeba přidat přímo do každého obrázku nějaký popisek (oblíbené je např. jméno fotografa či serveru). Příště se podíváme, jak jednoduše vygenerovat stránku plnou náhledů.

# PHP v praxi, 23. díl - Tvorba fotoalba

(08.03. 2002)

V předchozím díle jsme se věnovali vytvoření náhledů fotografií, je tedy na místě vytvořit skript také pro generování náhledových stránek. Úkolem PHP skriptu bude načíst z adresáře všechny fotografie, rozlišit od sebe náhledy od velkých obrázků a nakonec vytvořit nový soubor, jenž bude obsahovat další PHP skript na výpis obrázků, popř. i popisků k nim.

Nejprve si ukážeme načtení cest k fotografiím:

```
$adresar_name=dirname($cesta)."/";
$adresar=opendir($adresar_name);
$j=0;
while ($soubor=readdir($adresar))
{
        @$overeni=GetImageSize($adresar_name.$soubor);
        if ($soubor!="."&&$soubor!=".."&&$overeni[2]==2
                &&substr($soubor, 0, strlen($prefix))!=$prefix)
        {
                $obrazky[$j][t]=$adresar_name.$prefix.$soubor;
                $obrazky[$j][f]=$adresar_name.$soubor;
                $j++;
        }
        elseif($soubor!="."&&$soubor!=".."&&(!$overeni||$overeni[2]!=2))
        {
                echo "Nelze pracovat se souborem: $soubor <br>\n";
        }
}
closedir($adresar);
```

Od příkladu v minulém díle se mnoho nezměnilo. V podmínce se objevilo ověření, zda se nejedná o náhled: Funkce substr() vrací řetězec znaků podle námi zadaných argumentů. Prvním z nich ($soubor) je řetězec, ze kterého budeme znaky vybírat, druhý (0) značí, od kterého znaku budeme výběr provádět a poslední, kolik znaků má mít vrácený řetězec. To spočítáme pomocí funkce strlen(), jež vrací délku řetězce. Jednoduše řečeno - kontrolujeme, jestli obrázek nemá před sebou např. "t_". Pokud nikoliv, jeho jméno se zařadí do vícerozměrného pole $obrazky[][], kde první index znamená pořadí obrázků, druhý pak zda-li se jedná o náhled (t) nebo samotnou fotografii (f). Takže pod $obrazky[0][t] se skrývá např. c:\home\t_foto01.jpg a $obrazky[0][f] zastupuje c:\home\foto01.jpg.

Výsledek nyní potřebujeme zobrazit (kvůli ověření správnosti) a připsat popisky. To bude obsaženo v dalším formuláři (kompletní zdrojový soubor včetně přístupu k formulářům necháme opět na konec dílu):

```
for($i=0;$i<count($obrazky);$i++)
{
		$t_size=GetImageSize($obrazky[$i][t]);
		$f_size=GetImageSize($obrazky[$i][f]);
?>
<table border="1">
 <tr>
  <td width="<?= $t_size[0] ?>">
   <img src="<?= $obrazky[$i][t] ?>" <?= $t_size[3] ?>>
  </td>
  <td valign="top" width="280">
   <b>obrazek:</b><br>
   <?= basename($obrazky[$i][f]) ?><br>
   <?= $f_size[0].'x'.$f_size[1] ?><br>
   popisek:<br>
   <textarea name="f_pop[<?= $i ?>]"></textarea>
  </td>
  <td valign="top" width="280">
   <b>nahled:</b><br>
   <?= basename($obrazky[$i][t]) ?><br>
   <?= $t_size[0].'x'.$t_size[1] ?><br>
   popisek:<br><input type="text" name="t_pop[<?= $i ?>]" size="30">
  </td>
 </tr>
</table>
<?
		echo '<input type="hidden" name="obrazky['.$i.'][f]"
			value="'.basename($obrazky[$i][f]).'">';
		echo '<input type="hidden" name="obrazky['.$i.'][t]"
			value="'.basename($obrazky[$i][t]).'">';
}
```

Pro každý obrázek, cesta k němuž je uložena v poli $obrazky, se vytvoří samostatná tabulka. Pomocí nám již známé funkce GetImageSize() získáme informace o právě zpracovávaném obrázku včetně náhledu. Pod indexem 0 (např. $t_size[0]) je skryta šířka, pod 1 výška. Typ obrázku zjišťovat nepotřebujeme, k tomuto formuláři se dostaly již jen jpegy. Index 3 skrývá šířku a výšku ve formátu vhodném pro HTML dokument (width="x" height="y"). Zatím jsme se věnovali pouze výstupu na obrazovku, který však nijak neovlivní výsledný PHP skript. V této části jsou důležité popisky, jež se budou předávat přes pole $f_pop[] a $t_pop[], kde indexy se dosadí podle aktuálně zpracovávaných obrázků. Také se předá pole $obrazky[][], ale tentokrát již jen názvy souborů, nikoliv celé cesty.

Poslední "stránka" skriptu již odpovídá za vytvoření toho správného zdrojového kódu. Protože se jedná o kód v kódu není zrovna snadné se v něm orientovat:

```
function print_multi($pole,$m,$rel)
{
		$x="array(";
		for($i=0;$i<count($pole);$i++)
		{
```

```
                        $x.="'".$rel.$pole[$i][$m]."',";
                }
                $x.=");\n";
                return $x;
}

function print_simple($pole)
{
                $x="array(";
                for($i=0;$i<count($pole);$i++)
                {
                        $x.="'".$pole[$i]."',";
                }
                $x.=");\n";
                return $x;
}

$text="<html>\n<body>\n<h1>fotky</h1>\n <?\n";

$text.='$ukaz='.print_multi($obrazky,'f',$relativni);
$text.='$thumbs='.print_multi($obrazky,'t',$relativni);
$text.='$f_popisek='.print_simple($f_pop,",");
$text.='$t_popisek='.print_simple($t_pop,",");

$text.="\nif (!isset(\$show)||\$show=='nahledy')\n{\n";
$text.='   for($i=0;$i<count($ukaz);$i++)'."\n   {\n ?>\n";
$text.='      <a href="?show=<?= $i ?>"><img src="<?= $thumbs[$i] ?>"></a>'."\n";
$text.='      <br><?= $t_popisek[$i] ?><br><br>'."\n";
$text.=" <?\n   }\n";

$text.="}\nelse\n{\n ?>\n";
$text.='   <img src="<?= $ukaz[$show] ?>"><br>'."\n";
$text.='   <?= $f_popisek[$show] ?><br><br>'."\n";
$text.='   <a href="?show=nahledy">nahledy</a>';
$text.="\n <?\n}\n ?>\n";
$text.="</body>\n</html>";

$fp = fopen ("$soubor.php", "w");
fwrite($fp,$text);
fclose ($fp);

echo "Hotovo.";
```

Na začátek si vytvoříme dvě funkce na výpis pole. Funkce print_multi() je použita pro vícerozměrné pole - podle argumentů zpracuje buď informace o náhledech nebo plných fotografiích. Předává se také relativní URL. Výstupem (returm $x) bude např. array('cesta/foto1.jpg,'cesta/foto2.jpg',);. Print_simple() slouží k výpisu obsahu jednoduchého pole. Do souboru budeme zapisovat obsah proměnné $text. Tu tvoříme po řádcích, proměnné "sčítáme" jako obvykle tečkou. Aby byl výsledný skript přehledný, používáme formátování pomocí \n tedy "new line". Výsledkem bude např.:

```
<html>
<body>
<h1>fotky</h1>
<?
$ukaz=array('01Calais.JPG','02Calais.JPG',);
$thumbs=array('t_01Calais.JPG','t_02Calais.JPG',);
```

```
$f_popisek=array('popisek k obrazku1','popisek k obrazku2',);
$t_popisek=array('popisek k nahledu1','popisek k nahledu2',);

if (!isset($show)||$show=='nahledy')
{
  for($i=0;$i<count($ukaz);$i++)
  {
?>
    <a href="?show=<?= $i ?>"><img src="<?= $thumbs[$i] ?>"></a>
    <br><?= $t_popisek[$i] ?><br><br>
<?
  }
}
else
{
?>
  <img src="<?= $ukaz[$show] ?>"><br>
  <?= $f_popisek[$show] ?><br><br>
  <a href="?show=nahledy">nahledy</a>
<?
}
?>
</body>
</html>
```

Princip takovéto náhledové stránky je vysvětlen na konci článku PHP v praxi, 21. díl - soubor.php?jak=na&t=o.

Slibovaný zdrojový kód bude samozřejmě obsahovat i HTML formuláře a další "detaily", jež však není nutné podrobně rozebírat:

```
<html>
<body>
<h1>fotky</h1>
<?
$ukaz=array('01Calais.JPG','02Calais.JPG',);
$thumbs=array('t_01Calais.JPG','t_02Calais.JPG',);
$f_popisek=array('popisek k obrazku1','popisek k obrazku2',);
$t_popisek=array('popisek k nahledu1','popisek k nahledu2',);

if (!isset($show)||$show=='nahledy')
{
  for($i=0;$i<count($ukaz);$i++)
  {
?>
    <a href="?show=<?= $i ?>"><img src="<?= $thumbs[$i] ?>"></a>
    <br><?= $t_popisek[$i] ?><br><br>
<?
  }
}
else
{
?>
  <img src="<?= $ukaz[$show] ?>"><br>
  <?= $f_popisek[$show] ?><br><br>
  <a href="?show=nahledy">nahledy</a>
<?
```

```
}
?>
</body>
</html>
```

Princip takovéto náhledové stránky je vysvětlen na konci článku PHP v praxi, 21. díl - soubor.php?jak=na&t=o.

Slibovaný zdrojový kód bude samozřejmě obsahovat i HTML formuláře a další "detaily", jež však není nutné podrobně rozebírat:

```
<html>
<body>
<h1>Náhledy - PageMaker</h1>
<?
if(!isset($submit)){
?>
<form action="<?= $PHP_SELF ?>" method="post">
  Cesta k adresáři:
  <input type="file" name="cesta">
  (určeno k načtení obrázků)<br>
  Prefix náhledů:
  <input type="text" name="prefix" value="t_" size="5">
  (slouží k odlišení náhledů od vlastních fotek)<br>
  <br>
  <input type="submit" name="submit" value="Dalsi">
  <input type="reset" name="reset" value="Reset">
</form>
<?
}
elseif($submit=='Dalsi')
{

$adresar_name=dirname($cesta)."/";

$adresar=opendir($adresar_name);
$j=0;
while ($soubor=readdir($adresar))
{
        @$overeni=GetImageSize($adresar_name.$soubor);
        if ($soubor!="."&&$soubor!=".."&&$overeni[2]==2
            &&substr($soubor, 0, strlen($prefix))!=$prefix)
        {
                $obrazky[$j][t]=$adresar_name.$prefix.$soubor;
                $obrazky[$j][f]=$adresar_name.$soubor;
                $j++;
        }
        elseif($soubor!="."&&$soubor!=".."&&(!$overeni||$overeni[2]!=2))
        {
                echo "Nelze pracovat se souborem: $soubor <br>\n";
        }
}
closedir($adresar);
?>
<form action="<?= $PHP_SELF ?>" method="post">
  Relativní cesta k adr.:
  <input type="text" name="relativni" value="./">
```

```
 (např. "fotky/", jestliže bude výsledný PHP skript v adresáři nadřazeném
 adr. fotky)<br><br>
 Název souboru
 <input type="text" name="soubor" value="thumbs">.php
 <br><br>
<?
for($i=0;$i<count($obrazky);$i++)
{
        $t_size=GetImageSize($obrazky[$i][t]);
        $f_size=GetImageSize($obrazky[$i][f]);
?>
<table border="1">
 <tr>
  <td width="<?= $t_size[0] ?>">
   <img src="<?= $obrazky[$i][t] ?>" <?= $t_size[3] ?>>
  </td>
  <td valign="top" width="280">
   <b>obrazek:</b><br>
   <?= basename($obrazky[$i][f]) ?><br>
   <?= $f_size[0].'x'.$f_size[1] ?><br>
   popisek:<br>
   <textarea name="f_pop[<?= $i ?>]"></textarea>
  </td>
  <td valign="top" width="280">
   <b>nahled:</b><br>
   <?= basename($obrazky[$i][t]) ?><br>
   <?= $t_size[0].'x'.$t_size[1] ?><br>
   popisek:<br><input type="text" name="t_pop[<?= $i ?>]" size="30">
  </td>
 </tr>
</table>
<?
        echo '<input type="hidden" name="obrazky['.$i.'][f]"
       value="'.basename($obrazky[$i][f]).'">';
        echo '<input type="hidden" name="obrazky['.$i.'][t]"
       value="'.basename($obrazky[$i][t]).'">';
}
?>
  <br>
  <input type="submit" name="submit" value="Vytvor">
  <input type="reset" name="reset" value="Reset">
</form>
<?
}
elseif($submit=='Vytvor')
{

function print_multi($pole,$m,$rel)
{
        $x="Array(";
        for($i=0;$i<count($pole);$i++)
        {
                $x.="'".$rel.$pole[$i][$m]."',";
        }
        $x.=");\n";
        return $x;
}
function print_simple($pole)
{
        $x="Array(";
```

```
		for($i=0;$i<count($pole);$i++)
		{
			$x.="".$pole[$i]."',";
		}
		$x.=");\n";
		return $x;
}

$text="<html>\n<body>\n<h1>fotky</h1>\n <?\n";

$text.='$ukaz='.print_multi($obrazky,'f',$relativni);
$text.='$thumbs='.print_multi($obrazky,'t',$relativni);
$text.='$f_popisek='.print_simple($f_pop,",");
$text.='$t_popisek='.print_simple($t_pop,",");

$text.="\nif (!isset(\$show)||\$show=='nahledy')\n{\n";
$text.='   for($i=0;$i<count($ukaz);$i++)'."\n   {\n ?>\n";
$text.='      <a href="?show=<?= $i ?>"><img src="<?= $thumbs[$i] ?>"></a>'."\n";
$text.='      <br><?= $t_popisek[$i] ?><br><br>'."\n";
$text.=" <?\n   }\n";

$text.="}\nelse\n{\n ?>\n";
$text.='   <img src="<?= $ukaz[$show] ?>"><br>'."\n";
$text.='   <?= $f_popisek[$show] ?><br><br>'."\n";
$text.='   <a href="?show=nahledy">nahledy</a>';
$text.="\n <?\n}\n ?>\n";
$text.="</body>\n</html>";

$fp = fopen ("$soubor.php", "w");
fwrite($fp,$text);
fclose ($fp);

echo "Hotovo.";
}
?>
</body>
</html>
```

HTML formátování výsledné stránky již nechám na vás, ukázkový skript měl za úkol hlavně vygenerovat seznam fotek, jejich náhledů a popisků k nim.

# PHP v praxi, 24. díl - E-mail

(15.03. 2002)

Naším dnešním úkolem bude seznámit se se základy odesílání pošty přes webový formulář. Není na tom nic složitého, ale najdou se určité rány pod pás, jež nás obvykle stojí mnoho času.

Než začneme pracovat na skriptu, je důležité umožnit odesílání e-mailů přes PHP. Nezbývá nám nic jiného, než zeditovat php.ini:

1. nastavit "extension_dir" na správný adresář , např: extension_dir = "c:\php\extensions\" (358)
2. odkomentovat "extension=php_imap.dll" (438)

3. nastavit server pro odesílání pošty SMTP = smtp.vas_provider.cz (476)
4. napsat svoji e-mailovou adresu za "sendmail_from =" (479)

Osnovou celého skriptu bude následující podmínka:

```
if (!isset($submit)) formular();
else posli($mail);
```

Funkce formular() zobrazí HTML formulář, do kterého napíšeme údaje o odesílateli, příjemci, předmět e-mailu a samotný text. Informace odešleme tlačítkem submit té samé stránce. Proto se vytvoří proměnná $submit a podmínka bude nepravdivá. Zavolá se funkce posli(), obsahující veškeré informace z formuláře v poli $mail[]. Použitím pole si zde výrazně usnadníme práci - při každé změně vkládaných informací bychom jinak stále museli měnit i parametry této funkce.

```
function posli($mail)
{
   $hlavicka = "From: $mail[odesilatel]\r\n";
   $hlavicka.= "Reply-To: $mail[odesilatel]\r\n";
   $hlavicka.= "X-Mailer: php-mailer";
   mail ("$mail[adresa]", "$mail[predmet]", "$mail[telo]", "$hlavicka")
            or die ("error");
   echo "OK";
}
```

Základem je samozřejmě PHP funkce mail(), na tu se podívame podrobněji.
*bool mail ( string to, string subject, string message [, string additional_headers])*
Toto je definice z manuálu, je důležité ji umět přečíst. Před funkcí je slůvko bool - to značí, že funkce udává výsledek ve formě boolean, čili pravda nebo nepravda. Jednoduše: když se e-mail podaří odeslat, rovná se funkce pravdě (1), jinak nepravdě (0). V závorkách jsou uvedené parametry - všechny jsou stringy (řetězce). První je e-mailová adresa příjemce, druhý je předmět zprávy, třetí samotná zpráva. Tyto jsou povinné, v hranatých závorkách jsou parametry volitelné. V tomto případě informace v hlavičce, bez kterých se sice můžeme obejít, ale není to to nejlepší řešení.
Tyto informace jsou pro přehlednost dosazeny do proměnné $hlavicka. Všimněte si, že nejprve je v každém řádku určité slovo (např. From:), pak samotná informace a každý řádek končí \r\n (\r = carriage return, \n = linefeed; jedná se o ukončování řádků). S tímto je asi největší trápení: pokud váš smtp server pracuje pod Windows, je funkční právě tato syntaxe, Linuxový server si vystačí s \n.

```
<?
function formular()
{
 ?>
<form action="<?= $PHP_SELF ?>" method="post">
  <table width="400" border="1" cellpadding="5" align="center">
    <tr>
      <td colspan="2" align="center">php mailer</td>
    </tr>
    <tr>
      <td>adresa příjemce:</td>
      <td>
        <input type="text" name="mail[adresa]">
      </td>
    </tr>
    <tr>
      <td>adresa odesílatele:</td>
```

```
    <td>
      <input type="text" name="mail[odesilatel]">
    </td>
  </tr>
  <tr>
    <td>předmět:</td>
    <td>
      <input type="text" name="mail[predmet]">
    </td>
  </tr>
  <tr>
    <td>text:</td>
    <td>
      <textarea name="mail[telo]" cols="50" rows="10"></textarea>
    </td>
  </tr>
  <tr>
    <td colspan="2" align="center">
      <input type="submit" name="submit" value="odeslat">
    </td>
  </tr>
 </table>
</form>
 <?
}

function posli($mail)
{
   $hlavicka = "From: $mail[odesilatel]\r\n";
   $hlavicka.= "Reply-To: $mail[odesilatel]\r\n";
   $hlavicka.= "X-Mailer: PHP";
   mail ("$mail[adresa]", "$mail[predmet]", "$mail[telo]", "$hlavicka")
            or die ("error");
   echo "OK";
}


if(!isset($submit)) formular();
else posli($mail);

?>
```

Používání funkce mail() je velice časté, proto je nutné ji zprovoznit. Především k tomu měl sloužit dnešní návod, příště se podíváme na odesílání e-mailu s přílohou.

# PHP v praxi, 25. díl - Třídy a e-mail s přílohou

(29.03. 2002)

Minule jsem slíbil, že náš e-mail obdaříme přílohou. Využijeme toho však trošku jinak, než by se dalo předpokládat: na praktické ukázce si totiž ukážeme základy práce se třídami v PHP. Zejména pro začátečníky to nebude příliš snadné, ale uvidíte, že se to vážně pochopit dá. Na Internetu je k sehnání mnoho skriptů pod GPL licencí, není tedy rozumné ztrácet čas s programováním něčeho, co už za nás udělali jiní a lépe. Dnes využijeme právě třídu, jež má za úkol odesílání elektronické pošty a hodí se k vysvětlení objektového programování.

Co tedy budete potřebovat? Ze stránek projektu PHPMailer http://phpmailer.sourceforge.net/ si stáhnout dotyčný skript (zde verze 1.50). Bude nám z toho celého balíčku však stačit jen jeden soubor: "class.phpmailer.php". Dále se podívat na práci se třídami na stránkách PHP manuálu (dokonce v češtině!) http://www.php.cz/manual/cs/ref.classobj.php. Samozřejmostí je správná konfigurace PHP podle předchozího dílu, abyste mohli výsledný e-mail odeslat.

Objektové programování vzbuzuje u neprogramátorů pocit tajemna, přitom je to (alespoň co se týče toho základu, který potřebujeme pro stránky) celkem snadná věc. Třída (nebo objekt) lze dobře vysvětlit na konkrétních předmětech z našeho okolí. Tak třeba auto - tento pojem je třída: předpokládáme, že auto má čtyři kola, má nějakou barvu a jezdí. No a teď přejdeme ke konkrétnímu autu: vytvoříme jeho tzv. instanci, reálný předmět. Čtyři kola má každé, to je předdefinovaná vlastnost samotné třídy. Barvu mu přiřadíme červenou, to je individuální vlastnost právě této nově vytvořené instance. S touto instancí následně pojedeme do jiného města.

Když zůstaneme u zeleninového příkladu z manuálu:

```
class Zelenina {

    var $jedla;
    var $barva;

    function Zelenina( $jedla) {
        $this->jedla = $jedla;
    }

    function je_jedla() {
        return $this->jedla;
    }

    function jaka_barva() {
        return $this->barva;
    }

}

$zel = new Zelenina (true);
$zel->barva="cervena";
echo $zel -> je_jedla();
echo $zel -> jaka_barva();
```

Třída Zelenina pracuje s proměnnými $jedla a $barva. To jsou tzv. členské proměnné, inicializované na začátku pomocí var (zkratka pro variable). Pak následují jednotlivé funkce (zde se nazývají metody) - Zelenina(), je_jedla() a jaka_barva(). Hlavní metoda, Zelenina(), je konstruktor, jenž vytvoří novou instanci tohoto objektu. Všimněte si jak - použije se jako funkce s parametry, jen se před ní napíše "new". Proměnná $zel tedy zastupuje instanci třídy Zelenina s nutným parametrem (zde true).

Parametry se v konstruktoru zpracují poněkud divně vypadajícím způsobem $this->jedla = $jedla. $this zastupuje dotyčnou třídu, ve které se právě pohybujeme, šipka volá metodu nebo proměnnou v této třídě. Zde pozor: voláme tímto členskou

proměnnou, obyčejná proměnná je $barva. O členských proměnných se dá říci to, že se chovají jako globální proměnné v rámci jedné třídy. Proto k nim můžeme přistupovat v metodách je_jedla() nebo jaka_barva(). Opět ale jen přes volání $this->proměnná (bez dolaru). Platí, že v rámci třídy nemůže být nic volně, vše musí být v metodách, jinak se k tomu nedostanete. Volání se provádí přes šipku "->", přistupujeme tím k proměnným a metodám uvnitř konkrétní třídy. Např. barvu přiřadíme dotyčné instanci jednoduchým způsobem $zel->barva="cervena";

V praxi si ukážeme práci se třídou PHPMailer. Zmiňovaný soubor class.phpmailer.php umístěte do jednoho adresáře s prázdným souborem phpmail.php a jedním obrázkem podle vašeho výběru. Do prázdného souboru přijde následující kód:

```
require("class.phpmailer.php");

$mail = new phpmailer();

$mail->From  = "odnas@server.cz";
$mail->FromName = "jmeno";
$mail->AddAddress("cil@servr.cz","Jan Novák");
$mail->AddAddress("cil2@servr2.cz");
$mail->AddReplyTo("cil3@servr3.cz","Information");

$mail->WordWrap = 50;
$mail->AddAttachment("obrazek.jpg");
$mail->AddAttachment("obrazek.jpg", "new.jpg");
$mail->IsHTML(true);

$mail->Subject  =  "Predmet zpravy";
$mail->Body     =  "Ukazka <b>HTML</b>";
$mail->AltBody  =  "Cisty text";

if(!$mail->Send())
{
   echo "Zprava nebyla odeslana";
   exit;
}

echo "Zprava byla odeslana";

?>
```

Z předchozího už byste měli odvodit, že proměnná $mail obsahuje instanci třídy phpmailer. Její vlastnosti vytváří konstruktor phpmailer(). Např. $mail->From = "odnas@server.cz"; přiřadí členské proměnné $From ve třídě phpmailer konstruktoru phpmailer hodnotu mezi závorkami. Jiná věc je volání metody: $mail->AddAddress("cil@servr.cz","Jan Novák"); předá dva parametry funkci uvnitř třídy phpmailer, jejíž název je AddAddress() a má za úkol přidat do určitých polí správné informace o adrese příjemce. Tímto způsobem nastavíme celou instanci této třídy k naší spokojenosti a odešleme $mail->Send(). Její hodnota je boolean, čili buď pravda nebo nepravda. Když neuspěje, je výsledkem nepravda a vypíše se chybová hláška.

Sami si vyskoušejte práci s touto třídou. Je dobře zdokumentovaná (byť anglicky), takže se v ní tak snadno neztratíte. Rozhodně vám ale dá hodně prostoru k

experimentování. Doufám, že jste alespoň z části pochopili, o čem je řeč, když se řekne "objekt". Pro další programování je to velmi důležité, neboť většina PHP skriptů funguje právě na tomto principu.