

```
}else{ { <?
if($b>$c){ echo „Verze OS: „.PHP_OS.“<br>\n“;
$max=$b; vypíše (u mne) WINNT
}else{ echo „Verze PHP? „.PHP_VERSION;
$max=$c; vypíše 4.1.1 if ($a > $b){
?> if ($a > $c){
<? $max=$a;
switch ($znal){else{
case "A": echo "A"; break;
case "B": echo "B";
default: echo "C";
}
$max=$b;
?> }else{ $c;
echo
{
```

# PHP

## nejen pro začátečníky

**Martin Pokorný**

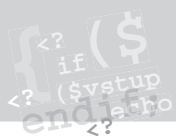
Nakladatelství a vydavatelství

**Computer Media**<sup>®</sup>

Vzdělávání, které baví  
[www.c-media.cz](http://www.c-media.cz)

## Obsah

<b>Vysvětlivky k použitým prvkům v knize .....</b>	<b>8</b>
<b>POZNÁMKA AUTORA .....</b>	<b>8</b>
<b>Slovo autora .....</b>	<b>9</b>
<b>Úvod .....</b>	<b>12</b>
<b>Historie internetu .....</b>	<b>12</b>
<b>Rozvoj WWW .....</b>	<b>13</b>
<b>MOŽNOSTI KOMUNIKACE .....</b>	<b>13</b>
<b>PHP NA SCÉNĚ .....</b>	<b>17</b>
<b>PROČ POUŽÍVAT PHP? .....</b>	<b>18</b>
<b>JAKÝ TYP APLIKACÍ LZE VYTVOŘIT V PHP? .....</b>	<b>19</b>
<b>KOMPILOVANÉ A SKRIPTOVACÍ JAZYKY .....</b>	<b>19</b>
<b>Instalace Apache, PHP a MySQL .....</b>	<b>22</b>
<b>APACHE .....</b>	<b>22</b>
<b>MySQL .....</b>	<b>24</b>
<b>PHP .....</b>	<b>25</b>
<b>PRVNÍ ZKUŠENOSTI .....</b>	<b>26</b>
<b>Začínáme s PHP .....</b>	<b>26</b>
<b>CO POTŘEBUJETE KE SPOUŠTĚNÍ SKRIPTŮ? .....</b>	<b>26</b>
<b>CO JE TO SKRIPT? .....</b>	<b>27</b>
<b>A JEŠTĚ JEDNOU - CO JE TO TEN SKRIPT? .....</b>	<b>28</b>
<b>OHRANIČENÍ SKRIPTU VE STRÁNCE .....</b>	<b>28</b>
<b>MŮJ MILÝ DENÍČKU, DNES JSEM POPRVÉ ZAPSAL SKRIPT .....</b>	<b>29</b>
<b>CO SE DĚJE FUNKČNĚ? .....</b>	<b>30</b>
<b>A DÁLE OBSAHOVĚ? .....</b>	<b>30</b>
<b>PHP INFO .....</b>	<b>31</b>
<b>KOMUNIKACE S UŽIVATELEM .....</b>	<b>32</b>
<b>ZÁKLADNÍ ČÁSTI STRÁNKY .....</b>	<b>33</b>
<b>ZÁKLADY PHP .....</b>	<b>36</b>
<b>Základní skladba jazyka .....</b>	<b>36</b>
<b>PRAVIDLA ZÁPISU .....</b>	<b>36</b>
<b>ZOBRAZENÍ ÚDAJŮ V PROHLÍŽEČI .....</b>	<b>37</b>
<b>Proměnné .....</b>	<b>38</b>
<b>KONVERZE ŘETĚZCŮ .....</b>	<b>40</b>



PROMĚNNÁ CHAMELEON .....	41
POZOR NA TYP DOUBLE .....	42
PRÁCE S ŘETĚZCI .....	42
DALŠÍ FUNKCE PRO PRÁCI S PROMĚNNÝMI .....	44
<b>Konstanty .....</b>	<b>44</b>
ZAVEDENÍ KONSTANT .....	44
<b>Výrazy .....</b>	<b>45</b>
ARITMETICKÉ OPERÁTORY .....	46
POROVNÁVACÍ OPERÁTORY .....	47
LOGICKÉ OPERÁTORY .....	49
UNÁRNÍ A TERNÁRNÍ OPERÁTOR .....	50
DALŠÍ OPERÁTORY .....	50
PRIORITA OPERACÍ .....	51
<b>Řídící struktury .....</b>	<b>54</b>
VĚTVENÍ PROGRAMU .....	54
PŘÍKAZY CYKLU .....	63
<b>Práce s poli .....</b>	<b>73</b>
INICIALIZACE POLE .....	73
PRÁCE S POLEM POMOCÍ FUNKCÍ .....	75
VÍCEROZMĚRNÁ POLE .....	83
<b>Formuláře .....</b>	<b>85</b>
ODESÍLÁNÍ A TVAR POŽADAVKU NA WEBOVÝ SERVER .....	85
DEFINICE FORMULÁŘE .....	87
ELEMENT INPUT .....	87
DALŠÍ PRVKY FORMULÁŘŮ .....	93
ZDOKONALENÍ VZHLEDU POUŽITÍM CSS .....	96
FORMULÁŘE NEBO „FORMULÁŘE“?.....	97
KONTROLA DAT PŘI VSTUPU NA SERVER .....	99
FUNKCE PŘI ZPRACOVÁNÍ DAT FORMULÁŘE .....	101
<b>Funkce v PHP .....</b>	<b>102</b>
OPAKOVANÉ VYUŽITÍ KÓDU .....	102
PŘÍKAZY REQUIRE A INCLUDE .....	103
UŽIVATELSKÉ FUNKCE .....	106
VESTAVĚNÉ FUNKCE PHP .....	117
<b>Práce se soubory .....</b>	<b>129</b>
OTEVÍRÁNÍ A ZAVÍRÁNÍ SOUBORŮ .....	129
ČTENÍ ZE SOUBORU .....	131

ZÁPIS DO SOUBORU .....	133
ZMĚNA AKTUÁLNÍ POZICE V SOUBORU .....	134
MANIPULACE SE SOUBORY .....	135
<b>Všemocná databáze? .....</b>	<b>138</b>
DATABÁZE .....	138
KOMUNIKACE S DATABÁZÍ .....	145
DATOVÉ TYPY A VYTVÁŘENÍ TABULEK .....	150
ÚPRAVY EXISTUJÍCÍCH TABULEK .....	157
TESTOVÁNÍ KÓDU SQL .....	160
JAZYK DML – PRÁCE S DATY .....	164
ADMINISTRACE SYSTÉMU .....	175
UŽIVATELSKÉ ROZHRAŇÍ DATABÁZE .....	177
<b>PHP s MySQL .....</b>	<b>181</b>
FUNKCE PRO KOMUNIKACI PHP S DATABÁZÍ MYSQL .....	181
PŘÍKLAD DATABÁZE OTÁZEK .....	183
<b>Ladění a ošetřování chyb .....</b>	<b>207</b>
DRUHY CHYB .....	207
NASTAVENÍ ÚROVNĚ CHYBOVÝCH ZPRÁV .....	209
OŠETŘOVÁNÍ CHYB .....	213
<b>Objektově orientovaný návrh programu .....</b>	<b>216</b>
TŘÍDA A OBJEKT .....	217
DĚDIČNOST .....	221
<b>Jak se vlastně program tvoří? .....</b>	<b>224</b>
PROBLÉMY PROGRAMOVÁNÍ .....	224
TESTOVÁNÍ .....	226
BEZPEČNOST .....	227

<p><b>PHP na scéně</b></p> <ul style="list-style-type: none"> <li>• historie internetu</li> <li>• rozvoj dynamických WWW stránek a možnosti komunikace</li> <li>• skriptovací jazyky</li> <li>• proč používat PHP a jaké aplikace v něm lze vytvořit</li> </ul>	<p><b>1</b></p>	<p>PHP na scéně Skriptovací jazyky</p>
<p><b>Prostředí pro práci s PHP</b></p> <ul style="list-style-type: none"> <li>• instalace Apache, PHP a MySQL</li> <li>• začínáme s PHP</li> <li>• co je to skript</li> <li>• první fungující skript - PHP info</li> </ul>	<p><b>2</b></p>	<p>Prostředí pro práci s PHP První skripty</p>
<p><b>Základy PHP</b></p> <ul style="list-style-type: none"> <li>• skladba jazyka PHP, konstanty, výrazy, řídicí struktury, funkce</li> <li>• práce s poli</li> <li>• formuláře za pomoci PHP</li> <li>• práce se soubory</li> </ul>	<p><b>3</b></p>	<p>Základy PHP a praktické kroky při tvorbě skriptů</p>
<p><b>PHP a databáze MySQL</b></p> <ul style="list-style-type: none"> <li>• datové typy, tvorba a úprava tabulek v databázi</li> <li>• testování kódu SQL, zabezpečení MySQL a jeho administrace</li> <li>• uživatelské rozhraní databáze, komunikace PHP s MySQL</li> <li>• ladění a ošetřování chyb skriptů</li> </ul>	<p><b>4</b></p>	<p>PHP a databáze MySQL</p>
<p><b>Objektově orientovaný návrh programu Problémy a úskalí programování skriptů</b></p>	<p><b>5</b></p>	<p>Objektově orientovaný návrh programu</p>

## JAKÝ TYP APLIKACÍ LZE VYTVOŘIT V PHP?

Běžně jsou v PHP vytvářeny různé podnikové informační systémy, diskusní fóra, redakční systémy, internetové obchody, poštovní a databázoví klienti na webu, nejčastějším způsobem použití budou asi stále různé dynamické firemní nebo i soukromé prezentace, počítačla, ankety aj. Důležité je také podotknout, že PHP roste nejen svou číselnou řadou verze a historií, ale především proto, že vývojáři jsou stále odvážnější v mohutnosti aplikací založených na PHP. PHP má tu výhodu, že se dokáže prakticky každému požadavku velice rychle a účinně přizpůsobovat.

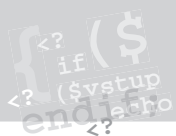
## KOMPILOVANÉ A SKRIPTOVACÍ JAZYKY

Podívejme se nyní krátce na způsob zpracování programu. Napíšeme-li zdrojový kód programu například za pomoci programovacího jazyka C, C++, Pascal, pak jej pro uživatele musíme zkompileovat (přeložit) do spustitelného tvaru. Distribuujeme tedy výsledný spustitelný kód - nejspíše .EXE tvar. Bude-li si uživatel přát v programu cokoli změnit, použijeme zazálohované zdrojové kódy programu, do kterých dopíšeme příslušné úpravy, a výsledný kód opět přeložíme do spustitelného tvaru. Tento druh programovacích jazyků, kdy se k uživateli dostává spustitelná podoba programu, nazýváme **kompileované**.

Nyní vezměme v úvahu druhý případ, kdy budeme chtít pro uživatele zpřístupnit program zapsaný pomocí PHP. Opět se jedná o zápis ve zdrojovém tvaru, který musí být zpracován kompilátorem (překladačem) příslušného programovacího jazyka. Jenže v tomto případě je program uchovávan přímo ve tvaru zdrojového kódu a teprve v případě vyžádání tohoto skriptu dojde ke spuštění PHP modulu, který daný zdrojový kód projde a zpracuje. Vyvoláme-li stejný skript několikrát po sobě, vždy bude zdrojový kód skriptu znovu zpracováván. Takovéto programovací jazyky nazýváme **skriptovací**.

U obou druhů jazyka bychom mohli najít výhody i nevýhody. U skriptovacích jazyků v první řadě asi namítnete, že se požadovaným překladem ze zdrojového kódu prodlužuje čas odezvy programu. V praxi se však jedná většinou o jazyky více či méně související s internetovým světem, tzn. přijde v potaz i doba potřebná ke stahování příslušného spustitelného kódu, který by měl určitě větší velikost než pouhý text zdrojového kódu. Pokud si dále připomenete dnes stále častější a agresivnější virové útoky, pak byste jistotu spíše viděli ve zdrojovém kódu než ve zkompileovaném spustitelném tvaru, který může obsahovat téměř cokoli.

Hlavní rozdíl byste ale měli hledat ve způsobu nasazení jednotlivých druhů programu. Princip činnosti skriptu ve zdrojovém kódu spočívá v získání požadovaných informací, které jsou pak použity ve vygenerované www stránce. Skript tedy vykoná svou činnost a poté se ukončí. Potřebujete-li zadat vstup či pouze aktualizovat výsledky, musí proběhnout celá procedura kompilace znovu, přičemž změna vstupních podmínek skriptu by mohla nastat v hodnotách zadaných vstupních parametrů volaného skriptu.



## Instalace Apache, PHP a MySQL

Popisované instalace budou představeny na u nás nejrozšířenějším operačním systému Windows. Neznamená to však, že Windows jsou původní podhoubí, ze kterého se PHP šíří dále. Právě naopak. Hlavní větev pochází ze světa unixového a Windows je jedna z odboček. Zarytí linuxáři teď jistě dodali „...a slepých“. Pokud ale využíváte jako operační systém třeba některé z distribucí Linuxu, pak vám Apache, PHP a MySQL nebudou cizí a víte jak je spustit. Kromě toho lze využít řady webových stránek, které vás navedou správným směrem.

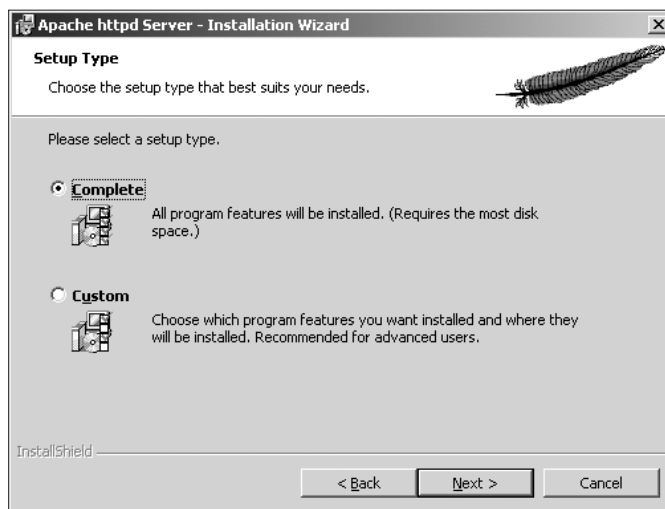
### APACHE

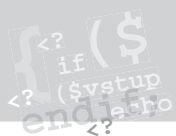
Ze začátku zřejmě nebudete chtít experimentovat s nastavením. Proto doporučuji využít zkompilevanou *exe* verzi Apache pro Windows. Získat ji můžete například na domovském serveru projektu Apache – <http://www.apache.org>. Instalace probíhá standardním způsobem instalace běžných aplikací pro Windows. Výchozím adresářem k instalaci bývá nastavení `C:\Program Files\Apache Group\Apache`. Před samotnou instalací pak dostáváte možnost rozhodnout o nainstalovaných součástech – Apache, zdrojové kódy, manuály. Samotnou konfiguraci serveru provedete po úspěšné instalaci úpravou konfiguračního souboru *httpd.conf*, k čemuž se ještě dostaneme.

Samotné spuštění serveru Apache můžete provést dvěma různými způsoby. Základní možnost startu spočívá ve spuštění přes *konzolové okno*. Tato možnost je nutná pro Windows 95/98 a jedna z možných pro Windows NT a výše. U skupiny vyšších verzí Windows pak lze využít i spuštění Apache jako *služby*.

#### Konzolové okno

Apache se spustí přes položku **Start Apache** z menu **Start**, nebo přímým vyhledáním a spuštěním souboru *Apache.exe* ve vaší instalaci. Okno Apache musí zůstat otevřené – nedokáže běžet na pozadí. Zastavení Apache pak lze logicky provést třeba jen pomocí zavření tohoto okna, popř. využitím kombinace kláves **Ctrl+C**.





```
if($a>0 || $b>0) // výsledek true
    echo "<br>Obě proměnné jsou kladné.";
if(($a>0 && $b>0) xor (!$a)) // výsledek true
    echo "<br>Obě proměnné \$a a \$b jsou kladné nebo \$a je nula.";

$b=0;
if($b!=0 && $a/$b) // výsledek false
    echo "<br>Proběhlo dělení";
else
    echo "<br>Nemusíte se bát, dělení nulou je hlídáno.";
?>
```

## UNÁRNÍ A TERNÁRNÍ OPERÁTOR

Prozatím jsme se zabývali především operátory binárními neboli operátory, které pro své vyhodnocení vyžadují dva operandy. **Unární operátor** už zde však byl také zmíněn. Je jím například operátor negace (**!***\$a*). Operátor **!** vyjadřuje opačnou logickou hodnotu, než má jeho operand. Jiným příkladem může být operátor minus (**-**), který funguje jak pro verzi dvou operandů (binární operátor), tak pro verzi jednoho operandu (unární operátor).

```
$a = -4;
$b = -$a; // b = 4
```

**Ternární operátor**, nebo také podmíněný výraz, existuje v PHP jediný. Ternární operátor provádí jednu operaci se třemi operandy. Používá k tomu operátor složený ze dvou znaků **"?:"**. První částí je logická podmínka zapsaná před znakem **"?"**, podle níž se rozhoduje o výsledku. Když bude výsledkem vstupní podmínky **true**, bude výsledkem operace hodnota zapsaná za otazníkem a před dvojtečkou. Bude-li výsledkem vstupní podmínky **false**, pak bude výsledkem celé operace hodnota zapsaná za dvojtečkou.

Základní skladba výrazu vypadá následovně: **výraz\_podmínka ? výraz\_1 : výraz\_2**

```
<!-- 021.php -->
<?
    $x = 2; $y = 5;
    $vysledek = ($x > $y) ? 10 : 20;
    echo "<br>Výsledkem je: $vysledek"; // Výsledkem je 20
?>
```

Závorky kolem podmínky nejsou nutné, ale zvyšují čitelnost tohoto výrazu. S podmíněným výrazem se dnes setkáte již málo, neboť jej lze velice jednoduše nahradit konstrukcí **if-else**, která je mnohem čitelnější.

## DALŠÍ OPERÁTORY

Zde se seznamte se

