Martin Mikuľák

Programování WW stránek pro úplné začátečníky

Názorná příručka krok za krokem Od úplných základů po hotové aplikace Formuláře, fotogalerie, kniha návštěv Tvorba administrace webu a databáze



Martin Mikuľák

Programujeme WWW stránky pro úplné začátečníky

Computer Press Brno 2013

Programujeme WWW stránky pro úplné začátečníky

Martin Mikuľák

Obálka: Martin Sodomka Odpovědný redaktor: Martin Domes Technický redaktor: Jiří Matoušek

Objednávky knih: http://knihy.cpress.cz www.albatrosmedia.cz eshop@albatrosmedia.cz bezplatná linka 800 555 513 ISBN 978-80-251-3252-4

Vydalo nakladatelství Computer Press v Brně roku 2013 ve společnosti Albatros Media a. s. se sídlem Na Pankráci 30, Praha 4. Číslo publikace 16664.

© Albatros Media a. s. Všechna práva vyhrazena. Žádná část této publikace nesmí být kopírována a rozmnožována za účelem rozšiřování v jakékoli formě či jakýmkoli způsobem bez písemného souhlasu vydavatele.

Dotisk 1. vydání



Obsah

Úvodem	9
Zpětná vazba od čtenářů	10
Zdrojové kódy ke knize	10
Errata	10
Kapitola 1	
Než začneme	11
Dynamické vs. statické stránky	11
Co je a k čemu slouží PHP	12
Instalace potřebného softwarového vybavení	13
Instalace webového serveru	13
Spuštění serveru Xampp	15
Kontrola správné instalace	17
Zkopírování zdrojových kódů ke knize na lokální webový server	18
Adresářová struktura a přístup k souborům	18
Textové editory se zvýrazněním syntaxe	19
Instalace textového editoru pro skripty	19
Jak k práci využít textový editor pro skripty	20
Výchozí podoba ukázkového webu	20
Kapitola 2	
Dynamické zobrazování obsahu	25
Přetypování statických souborů	25
Ahoj světe!	27
Ukládáme údaje do proměnných	28
Volání externích souborů při generování dvnamické stránky	29
Volání externích souborů – princip fungování	30
Části stránky v samostatných souborech a jejich volání	32
Parametry v URL adrese	36
Načítání hodnot parametrů z URL adresy	37

Obrana proti vkládání škodlivého kódu přes parametr URL adresy	38
Volání externích skriptů s texty a ošetření všech možností	39
Rozšiřujeme své znalosti	42
Správné programátorské návyky	43
Podmínky a jejich vyhodnocování	43
Vytváříme cykly	44
Kapitola 3	
Tvoříme kontaktní formulář	49
Fungování formulářů	50
První formulář	51
Metody POST a GET	53
Tlačítka ve formulářích	55
Vstupní pole – různé druhy	55
Klasické jednořádkové textové pole	55
Textová oblast	58
Pole pro heslo	59
Skrytá pole	59
Výběr z možností	60
Roletová nabídka	62
Zaškrtávací políčko	63
Formulář i výkonný skript v jediném souboru	65
Návrh a spuštění kontaktního formuláře	66
Obsah kontaktního formuláře	66
HTML kód formuláře	67
Ověření údajů vložených do formuláře	68
Ověření dat v kontaktním formuláři	76
Odesílání e-mailů	78
Fungování kontaktního formuláře	83
Ochrana formuláře proti spamerům	85
Kapitola 4	
Návrh databází	87
Jak vypadá databáze	87
Datové typy	87
Číselné typy	88
Textové (řetězcové) typy	88
Dvě funkce identifikátoru	89

Práce v prostředí PHPMyAdmin

Vytvoření nové databáze

90

91

Vytvoření nové tabulky v databázi	93
Záložky Struktura a Projít	95
Vkládání záznamu do tabulky	95
Zobrazování záznamů v tabulce	96
Úprava záznamů v tabulce	97
Smazání záznamu z tabulky	97
Úprava nebo smazání více záznamů současně	98
Export a import databází a tabulek	99
Práce s databází v jazyce PHP	103
Připojení k databázi	104
Možné chyby při připojování k databázi a jejich řešení	106
Výběr dat z databáze	106
Vkládání nových záznamů do tabulky	119
Úprava existujících záznamů	120
Mazání záznamů z tabulky	122
Doplňující informace	123

Kapitola 5

Dynamický obsah načítaný z databáze

databáze	125
Vytvoření databáze a tabulky pro texty	126
Skript pro přípravu textů	126
Jak to funguje?	129
Proč vyplňovat značky unikátními texty	131
Co když požadovaná stránka neexistuje?	132

Kapitola 6

Připravujeme administrační zónu	135
Relace – pamatování informací do okamžiku zavření prohlížeče	135
Soubory cookie – "koláčky", které si informace pamatují i mezi relacemi	138
Nastavení času vypršení cookie	140
Zrušení platnosti souboru cookie	142
Zobrazení souborů cookie přes prohlížeče	142
Práce s časem a datem	148
Jednoduchá správa obsahu – administrační zóna	150
Princip fungování	150
Co dělají jednotlivé skripty	151
Administrační zóna v praxi	158
Změna hesla přímo v administrační zóně	163
Tipy na vylepšení administračního rozhraní	175

Kapitola 7 Tvorba fotogalerie a úprava obrázků

Nahrávání souborů na webový server	177
Přesun souborů po nahrání na server	178
Zjištění dalších informací o souboru	182
Vytváření obrázků pomocí PHP	186
Míchání barev	188
Jednoduché tvary	188
Průhlednost	191
Ochrana formuláře proti spamerům	193
Práce s obrázky, které odesílají uživatelé	196
Přetypování obrázků	196
Využíváme dostupné filtry	199
Zmenšování obrázků	205
Vkládáme vodoznaky	208
Mazání souborů	209
Praktická realizace fotogalerie	210
Mazání obrázků přímo v administrační zóně	216
Zobrazení fotogalerie v části pro uživatele	220

Kapitola 8 Kniha návštěv 223 Analýza požadavků 223 Návrh databáze a skriptů 224 Skript pro obsluhu knihy návštěv 224 Další potřebné úpravy 227 Přidání textu/stránky do databáze 227 Testování a ladění 230 Správa knihy návštěv v administrační zóně 233 Zobrazení, úprava a mazání příspěvků 233 Testování a ladění administrační zóny 237 Možná vylepšení 238

Kapitola 9

239
239
239
239

177

Chyby ve funkcích	240
Hlavička odeslána	241
Logické chyby	241
Chyby v proměnných	241
Nenastartované relace	242
Soubory Cookie	242
Cykly a podmínky	242
Nevykonání příkazů uvnitř podmínky	242
Zakomentování částí skriptu	243
Testování a ladění	243
Jak testovat	243
Závěrem	245
Příloha A Časté konstrukce	247
рнр	247
Větvení (rozhodovací struktury)	247
Cykly	250
MvSOI	250
Přinojení k databázi	252
Vkládání nových záznamů do databáze	252
Výběr a výpis dat z databáze	252
Aktualizace záznamu	253
Mazání záznamů	254
Příloha B Přehled funkcí a příkazů	
noužitých v knize	255
	233

Rejstřík	259
Práce s obrázky	257
Odesílaní e-mailů	257
Práce s časem	257
Práce se soubory	256
Práce s poli	256
Práce s textem	255
Všeobecné funkce	255
. ,	

Úvodem

V dnešní době to někdy vypadá, že Internet vládne světu. Dnes už není problém vyřídit mnoho běžných záležitostí přímo od počítače, aniž bychom museli někam chodit, a to během pár minut.

Když potřebujeme něco koupit, na Internetu si vyhledáme informace o dostupných řešeních. Zjistíme si reference lidí na jednotlivá řešení a firmy, které je poskytují. Až pak se rozhodneme, že konkrétní firmu navštívíme osobně a využijeme jejích služeb. Mnohdy však není potřeba ani to.

Internet nabízí mnoho možností: můžeme nakupovat zboží, psát vlastní názory na blogu nebo komunikovat s přáteli, kteří jsou třeba na opačném konci světa. Vše okamžitě a bez problémů.

Pravděpodobně i vás zajímá, jak vytvořit kvalitní dynamické stránky, které budou snadno rozšiřitelné o nové funkce a obsah. To je totiž to, co mají lidé na Internetu rádi – kvalitní obsah podaný v přehledné formě. A zajímavé funkce k tomu.

V této knize si ukážeme, jak přetvořit běžné (X)HTML statické stránky do dynamické podoby. Po přečtení této knihy budete umět psát vlastní skripty a navrhovat a používat databáze. Budete si umět vytvořit vlastní administrační zónu a díky tomu pro vás bude přidávání nového obsahu na stránky hračkou.

Celá kniha je napsaná snadno srozumitelným jazykem a prakticky při všem používá ukázkové příklady. Na začátku jednotlivých kapitol se nejdříve naučíme používat nové funkce a získané znalosti pak použijeme na konkrétním řešení, které implementujeme do bývalých statických stránek.

Původní jednoduché stránky tak rozšíříme o spojení s databází, o fotogalerii s automatickou úpravou fotek, kontaktní formulář s odesíláním zpráv na e-mail, knihu návštěv a také administrační rozhraní pro správu obsahu.

Díky znalostem, které získáte, pro vás nebude problém vytvořit vlastní funkcionalitu nebo modul.

V této knize se budeme věnovat pouze jazykům PHP a MySQL – a to od úplných základů.

Nebudeme se však zabývat vysvětlováním jazyka (X)HTML a jeho značek ani tvorbou stylů pomocí CSS. Předpokládáme, že tyto znalosti již máte.

Celá kniha je členěna do devíti kapitol a obsahuje také dvě přílohy.

V první kapitole si nainstalujeme potřebné softwarové vybavení a podíváme se na statické stránky, které budeme předělávat.

V dalších sedmi kapitolách se budeme postupně učit psát skripty v jazycích PHP a MySQL a budeme předělávat původní statické stránky.

V deváté kapitole naleznete užitečné rady a tipy, jak řešit případné problémy a jak testovat a ladit vaše webové aplikace.

V příloze A pak uvádíme přehled nejčastěji používaných konstrukcí jazyků PHP a MySQL i s krátkými příklady použití.

A nakonec v příloze B najdete seznam funkcí jazyka PHP, které budeme v knize používat při tvorbě našich skriptů.

Dost bylo slov - vrhněme se na tvorbu dynamických stránek!

Zpětná vazba od čtenářů

Nakladatelství a vydavatelství Computer Press stojí o zpětnou vazbu a bude na vaše podněty a dotazy reagovat. Můžete se obrátit na následující adresy:

redakce PC literatury Computer Press Spielberk Office Centre Holandská 3 639 00 Brno nebo sefredaktor.pc@cpress.cz

Computer Press neposkytuje rady a konzultace ohledně programování.

Zdrojové kódy ke knize

Z adresy *http://knihy.cpress.cz/k1868* si po klepnutí na odkaz Soubory ke stažení můžete přímo stáhnout archiv s ukázkovými kódy.

Errata

Přestože jsme udělali maximum pro to, abychom zajistili přesnost a správnost obsahu, chybám se úplně vyhnout nedá. Pokud v některé z našich knih najdete chybu, ať už chybu v textu nebo v kódu, budeme rádi, pokud nám ji nahlásíte. Ostatní uživatele tak můžete ušetřit frustrace a pomoci nám zlepšit následující vydání této knihy.

Veškerá existující errata zobrazíte na adrese *http://knihy.cpress.cz/k1868* po klepnutí na odkaz Soubory ke stažení.

Kapitola 1 **Než začneme**

Předtím, než naše statické stránky předěláme do dynamické podoby, bychom si měli říct, co to vlastně dynamické stránky jsou, co všechno umožňují nám jako tvůrcům a co návštěvníkům našich stránek. V této kapitole si také připravíme a nainstalujeme všechno, co budeme později při psaní a ladění dynamických stránek potřebovat.

Dynamické vs. statické stránky

Základní rozdíl mezi statickými a dynamickými stránkami je, že statické stránky jsou neměnné. Celý web je "vyskládán" z mnoha souborů, přičemž každý z nich obsahuje HTML soubor, ve kterém je kompletně celý zdrojový kód dané podstránky.

Dynamické weby však mají v jednotlivých souborech pouze části celkové stránky (jeden soubor obsahuje zdrojový kód hlavičky, další patičky, další soubor třeba kontaktní formulář a tak dále). Výsledná stránka, kterou pak uživatel vidí na svém monitoru, je vyskládaná z těchto častí.



Obrázek 1.1 Statický web má celé stránky v jediném souboru, a pokud máme stránek více, je obtížné změnit třeba menu na každé z nich. Dynamické stránky jsou však vyskládány jako mozaika z několika souborů, a tak se změny v jediném souboru projeví na všech stránkách celého webu.

Tento přístup má výhodu v tom, že pokud máme menu v samostatném souboru a chceme ho změnit, stačí, když tak učiníme pouze v jediném souboru. Změněné menu se pak jako část skládačky použije na celém webu a na každé podstránce, kterou vidí uživatel ve výsledku na svém monitoru. Tento přístup použijeme i v našem ukázkovém projektu, hned v příští kapitole.

Dynamické stránky však umožňují daleko více než jen snadné úpravy a údržbu. Jejich další vlastností je, že jsou většinou propojené s databází a mohou měnit svůj obsah podle toho, co konkrétní uživatel vyžaduje.

Dobrým příkladem je vyhledávaní. Každý uživatel bude na našich stránkách hledat něco jiného, ale nám stačí napsat pouze jeden skript, který projde databázi a ukáže uživateli to, co chtěl najít na stránce s vyhovujícími články. A to vše dynamicky.

Jak přesně dynamické stránky fungují, si vysvětlíme dále.

Co je a k čemu slouží PHP

PHP je open source skriptovací jazyk, který se používá k vytvoření dynamických stránek. Dá se snadno naučit, a přitom má široké možností využití: můžeme ho použít pro malé a jednoduché osobní stránky nebo blogy, ale také pro elektronické obchody či prezentaci velké společnosti.

PHP jazyk se používá pro programování aplikací na bázi klient-server (obrázek 1.2).



Obrázek 1.2 Schéma fungování aplikací na bázi klient-server

- 1. Uživatel někde na Internetu klepne na odkaz vedoucí na naši stránku nebo její adresu napíše do řádku pro URL adresu a stiskne klávesu Enter.
- **2.** Uživatelův prohlížeč odešle požadavek serveru, na kterém jsou naše stránky hostovány. Požadavek obsahuje informaci o tom, kterou stránku si chce uživatel zobrazit.
- **3.** Server spustí náš skript, který dynamicky vygeneruje požadovanou stránku. Vybere z databáze požadované informace a spojí části stránky (hlavička, menu, hlavní text, patička) do jediného (X)HTML zdrojového kódu výsledné stránky. Tento kód už vypadá jako kód běžné statické stránky jen byl vytvořen dynamicky.
- **4.** Tento (X)HTML kód je poslán prohlížeči uživatele, který jej zobrazí a vykreslí, tak jako běžnou statickou stránku.

Jak je vidět, podstata je v tom, že všechno "dynamické" se děje na straně serveru, který na základě požadavku od uživatele vygeneruje statickou stránku a tu pošle prohlížeči uživatele.

Výhody tohoto klient-server přístupu jsou v tom, že všechny výpočty, dotazy do databáze a generování kódu (skládaní mozaiky) probíhají na serveru (což je vlastně velmi výkonný počítač) a uživateli se posílá statický zdrojový kód, který prohlížeč pouze zobrazí – tak jako každou jinou statickou stránku.

Využití skriptů napsaných v jazyce JavaScript

Součástí dynamických stránek jsou často i skripty v jazyce JavaScript. Tyto skripty se však posílají spolu s (X)HTML kódem prohlížeči uživatele a tento prohlížeč teprve vykonává instrukce obsažené v uvedených skriptech. JavaScript neběží na straně serveru jako PHP, ale až na straně uživatele.

V praxi se obvykle využívá kombinace těchto dvou jazyků. PHP skript na serveru vygeneruje statickou stránku třeba s fotogalerií i s vloženým javascriptovým kódem. Výsledek se pošle prohlížeči uživatele. JavaScript pak umožní, že fotogalerie už není jen nudný, statický "výčet" fotek.

Taková fotogalerie pak vypadá velmi efektně díky tomu, že jednotlivé fotky mohou dynamicky vystoupit do popředí, různě rolovat a podobně. Právě díky tomu, že JavaScript běží na straně uživatele, se nezatěžuje server, na kterém jsou naše stránky hostovány.

JavaScript však umožňuje mnohem více než jen vytvoření efektních fotogalerií. Díky němu je možné měnit a aktualizovat některé části (bloky) stránky, aniž by bylo nutné znovu načítat celou stránku. To však není obsahem naší knížky, proto se tomuto tématu dál věnovat nebudeme.

Instalace potřebného softwarového vybavení

Abychom se mohli naučit programovat dynamické stránky v jazyce PHP, musíme si na náš počítač nainstalovat webový server. Slouží k tomu, abychom přímo na našem vlastním počítači mohli spouštět PHP skripty a vytvářet tak naše dynamické stránky. To je nezbytné, abychom nemuseli naše stránky testovat a vyvíjet na reálném, vzdáleném serveru, kde už běží naše statické stránky. Bylo by totiž hodně nepraktické při každé změně kopírovat upravené soubory na vzdálený server. Nemluvě o tom, že pokud by byl skript špatný (třeba se zacyklil do nekonečné smyčky), zbytečně by server zatěžoval. I v praxi se webové aplikace vyvíjí na vlastním počítači a na server se nasazuje už otestovaná aplikace.

Instalace webového serveru

Jako u téměř každého softwaru, i zde existuje více rozdílných možností. My si nainstalujeme Xampp server, protože se instaluje velice snadno, obsahuje webový Apache i databázový MySQL server (využijeme ho později) a má pohodlné ovládání. Je to open source freeware řešení, takže si nemusíme nic kupovat.

Poznámka: Pro instalaci budeme potřebovat administrátorská práva, takže se musíme přihlásit jako administrátor.

Existuje několik možností, jak si tento server nainstalovat. Záleží také na tom, jaký operační systém používáme. Všeobecně platí, že si nejdříve musíme otevřít stránku *http://www.apachefriends.org/en/xampp.html*, na které najdeme vše potřebné.

Instalace ve Windows pomocí instalátoru (doporučujeme):

- 1. Xampp server je podporován prakticky všemi dnes používanými operačními systémy od společnosti Microsoft: Windows 2000, Windows XP, Windows Vista i Windows 7.
- 2. Klepneme na možnost Xampp for Windows a dostaneme se na stránku http://www. apachefriends.org/en/xampp-windows.html.
- **3.** Rolujme stránku níže, až dojdeme k bloku **Download**. V době psaní této knihy je aktuální verze **XAMPP Windows 1.7.4**.
- 4. Klepneme na možnost Installer a stáhneme si jej. Měl by mít velikost kolem 66 MB.
- 5. Po stáhnutí instalátoru na něj poklepeme a spustíme ho.
- **6.** Vybereme si jazyk **English**.
- 7. V dalším kroku pouze klepneme na Next.
- **8.** Vybereme si cílový adresář na disku, do kterého chceme server nainstalovat. Doporučujeme si zvolit adresář **C:\xampp**, protože dále v knize budeme používat právě toto umístnění. Instalace serveru vyžaduje asi 460 MB volného místa na disku. Klepneme na **Next**.
- **9.** V dalším kroku můžeme označit, jaké ikony chceme vytvořit. Možnosti Create a XAMPP desktop icon a Create an apache friends xampp folder in the start menu doporučujeme nechat zaškrtnuté.

Poznámka: Možnosti Install apache as service a Install mysql as service můžeme zaškrtnout také. Znamenají, že chceme, aby se webový server a server pro databáze nainstalovaly jako služby operačního systému. Pokud je tak nainstalujeme, budou se automaticky spouštět již při startu počítače. Tato možnost je vhodná, pokud se programování stránek budete věnovat často. Nebudete tak muset startovat tyto servery ručně pokaždé, když je budete potřebovat.

10. Klepněme na Install.

- 11. Pokud instalace proběhla správně, zobrazí se okno s tlačítkem Finish. Klepneme na něj.
- 12. Pokud vyskočilo okno *Firewallu*, klepneme na možnost Unblock (neblokovat).
- **13.** Instalace je dokončená.

Instalace ve Windows bez instalátoru:

Pokud jste zkušenější uživatel, můžete si Xampp server nainstalovat i bez instalátoru. Ze stránky *http://www.apachefriends.org/en/xampp-windows.html* si v sekci **Download** stáhněte ZIP archiv. Rozbalte ho do adresáře podle svého výběru. Poté spusťte soubor *setup_xampp. bat* a server si nakonfigurujte podle instrukcí na obrazovce.

Důležité: Pokud si vyberete jako cílový adresář C:\, nemůžete spustit soubor *setup_xampp.bat*. Způsobovalo by to problémy.

Instalace v Linuxu:

- 1. Stáhneme si nejnovější verzi ze stránky http://www.apachefriends.org/en/xampp-linux.html.
- 2. V terminálovém okně zadáme příkaz su a přihlásíme se jako root.
- 3. Spustíme následující příkaz: tar xvfz xampp-linux-1.7.4.tar.gz -C /opt.

Poznámka: Verze 1.7.4. je aktuální v době psaní této knihy. Pokud jste si stáhli novější verzi, pozměňte příkaz tak, aby vyhovoval názvu staženého archivu.

Důležité: Používejte pro instalaci pouze tento příkaz. Na rozbalení archivu nepoužívejte žádné jiné nástroje, jinak se instalace nezdaří.

Důležité: Po spuštění tohoto příkazu se přepíše původní (pokud byla) instalace/verze serveru Xampp.

Pro nastartování serveru spustíme příkaz /opt/lampp/lampp start.

Pokud je na obrazovce následující text, server se úspěšně nastartoval:

Starting XAMPP 1.7.4... LAMPP: Starting Apache... LAMPP: Starting MySQL... LAMPP started.

Teď už zbývá jen zkontrolovat správnost instalace.

Spuštění serveru Xampp

Pokud jsme server Xampp nenainstalovali jako službu, a není tudíž ještě spuštěný, klepneme na ikonu Xampp Control Panel. Pokud ji nemáme na ploše, vyhledejme ji v nabídce Start → XAMPP for Windows → Xampp Control Panel. Otevře se ovládací panel serveru (jako na obrázku 1.3).



Řešení problému: Pokud systém při pokusu o zobrazení kontrolního panelu vypisuje hlášku jako na obrázku 1.4 s textem ERROR: XAMPP control panel is already running (XAMPP control panel už běží), zkuste si tento kontrolní panel zobrazit z nabídky běžících programů, kterou najdete v pravém dolním rohu obrazovky (jako na obrázku 1.5). Xampp Control Panel má oranžovou ikonku s velkým písmenem X obtaženým bílou barvou.

Pokud v levé části okna vedle popisku **Apache** nesvítí zelený obdélník s textem **Running**, klepneme na tlačítko **Start** ve stejném řádku, jako je popisek **Apache**. Totéž můžeme udělat i s druhým řádkem pro **MySQL**. Pokud jsou oba servery spuštěny, svítí vedle jejich názvů dva zelené obdélníky s textem **Running** (viz obrázek 1.6).

XAMPP	Control Panel App	lication		
R	XAMPP Cont	rol Panel	Service	SCM
Modules Svc Svc Svc Svc	Apache MySql FileZilla Mercury	Start Start Start Start	Admin Admin Admin	Status Refresh Explore Help Exit
•	111			▲ ▼

Obrázek 1.3 Xampp Control Panel



SK < ⊠ 17:59

Obrázek 1.4 Chybová hláška: kontrolní panel serveru Xampp už běží

XAMPP Control Panel Application					
R	XAMP	P Control F	Panel	Service	SCM
Modules	Apache MySql FileZilla Mercury	Running Running	Stop Stop Start Start	Admin Admin Admin	Status Refresh Explore Help Exit
•		m			▲ ▼ ■

Obrázek 1.5 Zobrazení kontrolního panelu serveru Xampp z nabídky běžících programů

Obrázek 1.6 Xampp Control Panel ve stavu, kdy běží webový (Apache) i databázový (MySQL) server

Kontrola správné instalace

Pokud už máme nainstalovaný a spuštěný webový server, můžeme zkontrolovat jeho funkčnost. Otevřeme si svůj oblíbený internetový prohlížeč a v něm adresu *http://localhost/*.

Pokud se zobrazila stránka s velkým logem Xampp uprostřed a možností volby jazyka, máme server nainstalován správně a také správně běží. Klepneme na možnost **English** a dostaneme se na další stránku. Měli bychom vidět hlášku: **Congratulations: You have successfully installed XAMPP on this system!**, která potvrzuje správnou instalaci. Obrazovka by teď měla vypadat jako na obrázku 1.7.

នេ	XAMPP for Windows	English / Deutsch / Francais / Nederlands / Polski / Italiano / Norwegian / Español / 中文 / Português (Brasil) /日本語
XAMPP 1.7.4 [PHP: 52.3:3] Welcome Status Security Documentation Components PhIP Phpinfo() CD Collection Biofhythm Instant Art Phone Book Peri Perinfo() Guest Book D2EE Status Tomcat examples Pobla/Zer Mercury Mail FileZilla FTP	Welcome to XAMPP for Windows! Congratulations: You have successfully installed XAMPP on this system! Now you can start using Apache and Co. You should first try For OpenSSL support please use the test certificate with htt Good luck, Kay Vogelgesang + Kai 'Oswald' Seidler	English / Deutsch / Francais / Nederlands / Polski / Italiano / Norwegian / Español / 中文 / Portugués (Brasil) / 日本語 *Status« on the left navigation to make sure everything works fine. <u>ps://127.0.0.1</u> or <u>https://localhost</u>
©2002-2010 APACHE FRIENDS		

Obrázek 1.7 Stav obrazovky po úspěšné instalaci a nastartování webového serveru

Můžeme si zobrazit také stránku *http://localhost/phpmyadmin/*. Na této adrese se zobrazí uživatelské rozhraní pro práci s databázemi. Těm se budeme věnovat později.



Řešení problému: Pokud se na vašem počítači nezobrazila hláška o úspěšném zprovoznění webového serveru, může to být způsobeno hlavně těmito skutečnostmi:

- Pokud na obrazovce máte hlášku prohlížeče "Stránka nenalezena", zkontrolujte, jestli máte nastartován webový server, podle odstavce Spuštění Xampp serveru výše v této kapitole.
- Pokud server nejde nastartovat nebo se ani nezobrazí Xampp Control Panel, zkuste celý server přeinstalovat.
- Pokud Xampp Control Panel vypadá jako na obrázku 1.6 vedle řádku s textem Apache je zelený obdélník s textem Running zkontrolujte, jestli se pokoušíte zobrazit správnou stránku: http://localhost/ nebo http://127.0.0.1/.
- Pokud nic nepomáhá, zkuste server znovu přeinstalovat.

Zkopírování zdrojových kódů ke knize na lokální webový server

Teď si nakopírujeme zdrojové kódy ke knize na náš právě vytvořený lokální webový server, abychom mohli sledovat všechny postupy a řešení, tak jak budou popsány v knize.

Stáhneme si zdrojové kódy dostupné ke knize z webové stránky *http://knihy.cpress.cz/* k1868 a rozbalíme je. Otevřeme si adresář C:\xampp\htdocs\. Vytvoříme si nový adresář kniha a všechny soubory, které jsme si před chvíli stáhli, nakopírujeme do tohoto adresáře. Kódy k jednotlivým kapitolám najdeme v příslušném podadresáři – např. pro tuto první kapitolu jsou v adresáři C:\xampp\htdocs\kniha\01\.

Adresářová struktura a přístup k souborům

Ze začátku bude možná trochu matoucí, jak najít požadovaný soubor na disku (třeba pro editaci) a jak zapsat jeho adresu do prohlížeče tak, aby webový server vykonal instrukce skriptu zapsané v daném souboru.

Pri práci se soubory a skripty budeme postupovat následovně:

- Pokud budeme chtít najít soubor na disku (třeba pro editaci), hledejme ho v adresáři např. takto: C:\xampp\htdocs\kniha\01\index.html.
- Pokud si budeme chtít tento soubor zobrazit v prohlížeči (tak aby webový server vykonal skript, který soubor obsahuje), zapišme do něj adresu http://localhost/kniha/01/index.html.

Jediný rozdíl v přístupu k souboru pak spočívá v začátku cesty k němu: buď *C*:*xampp*\ *htdocs*\..., nebo *http://localhost/...* . Záleží na tom, co s tímto souborem chceme dělat.

Poznámka: Heslo pro uživatele **root** pro přístup k databázi můžete nastavit na stránce *http://localhost/ security/xamppsecurity.php*. Podrobnější informace o bezpečnostních nastaveních pak naleznete na stránce*http://localhost/security/*. Pro naše výukové účely a později i pro vývoj vlastních stránek a aplikací na lokálním počítači tato nastavení nemusíte měnit.

Omezená bezpečnost Xampp serveru

Xampp server je určen téměř výhradně pro vývojáře webových aplikací na jejich lokálních počítačích a tomu odpovídají i jeho nastavení. Je konfigurován jako co nejvíce otevřený a s minimem bezpečnostních omezení, proto je nevhodný pro produkční (ostrý) server. Je navržen tak, aby umožňoval vše, co návrhář potřebuje.

Za všechny bezpečnostní hrozby uveďme alespoň jednu. Pro přístup k databázi se používá standardní uživatel (root), který nemá nastavené žádné heslo. Heslo pro tohoto uživatele se sice dá nastavit, ale pořád zbývají ostatní hrozby.

Proto Xampp server využívejte pouze na svém lokálním počítači a jen pro vývoj webových stánek a aplikací. Pro hotové stránky, které pak budete chtít "zavěsit na Internet", využijte služby poskytovatelů webhostingu, kteří se poskytování těchto služeb věnují. Pokud byste chtěli provozovat vlastní webový sever, rozhodně budete potřebovat nastudovat více informací.

Textové editory se zvýrazněním syntaxe

HTML stránky i skripty jazyka PHP lze psát prakticky v libovolném textovém editoru (třeba i v Poznámkovém bloku nebo editoru Vi), ale existuje obrovské množství editorů, jež umožňují zvýrazňování a některé i doplňování značek kódu konkrétního jazyka. Využít můžete také velké množství placených, ale rovněž freeware editorů, které se dost liší kvalitou i uživatelskou přívětivostí.

Pro jakéhokoli vývojáře nebo programátora je velmi vhodné pracovat s editorem, který dokáže zvýraznit (a třeba i doplnit) značky daného jazyka.

Pro nás je výhodné, když jsou značky HTML kódu zvýrazněné jinou barvou jako proměnné jazyka PHP a když jsou navíc ještě odlišené jinou barvou třeba od funkcí tohoto jazyka. Velmi příjemnou vlastností je také zvýrazňování začátku a konce bloků mezi závorkami / kulatými () i složenými {}/, třeba při cyklech nebo podmínkách jazyka PHP. Častá je také funkce automatického odsazování a zpřehlednění zdrojového kódu.

Všechny tyto funkce editoru výrazně usnadňují práci programátorům a vývojářům, kteří rychleji najdou chybu ve svém skriptu nebo se v něm výrazně rychleji zorientují.

Instalace textového editoru pro skripty

Pro naše účely je velmi vhodný PSPad editor, který je freeware.

- 1. Otevřeme si stránku *http://www.pspad.com/cz/*.
- 2. Klepneme na možnost Stažení PSPadu v levém menu.
- **3.** Stáhneme si *Instalátor*. Je to malý soubor (asi 3,5 MB), v době psaní knihy je aktuální verze 4.5.4.
- 4. Poté na *Instalátor* poklepeme a zahajíme proces instalace.
- 5. Na první kartě pouze klepneme na Další.
- 6. Na druhé kartě musíme souhlasit s licenčními podmínkami, poté klepneme na Další.
- V třetím kroku si vybereme, kam chceme editor naistalovat. Cílový adresář nemusíme (ale můžeme) změnit. Klepneme na Další.
- V dalším kroku si vybereme typ instalace. Klidně můžeme ponechat možnost Plná instalace, poněvadž zabere pouze necelých 12 MB prostoru na disku. Klepneme na Další.
- **9.** V tomto kroku vytvoříme zástupce v nabídce **Start**. Opět můžeme ponechat beze změn. Klepneme na **Další**.
- 10. Na následující kartě si zvolíme další požadované úlohy podle toho, co budeme od editoru požadovat. Můžeme je nechat nezměněné, některé možnosti lze ubrat nebo také zaškrtnout položku Otevírat soubory TXT programem PSPad. Klepneme na Další.
- 11. Klepneme na Instalovat.
- 12. Instalace je ukončená, klepneme na Dokončit.

Jak k práci využít textový editor pro skripty

Spustíme si teď nově nainstalovaný editor – buď klepnutím na ikonku PSPad na ploše, nebo po vyhledání v nabídce **Start** \rightarrow **PSPad**.

Klepneme na příkaz **Soubor** \rightarrow **Nový**. Otevře se okno, ve kterém je poměrně dlouhý seznam podporovaných typů souborů. Vyberme si třeba HTML multihighlighter. Otevře se nový soubor a v něm připravená základní šablona. Napišme mezi značky
 body > a </body > následující kód:

```
<div id="test">
    testovací text
</div>
```

Teď klepneme do značky </div>. Jak vidíme na obrázku 1.8, značka <div id="test"> a k ní patřící uzavírací značka </div> se podbarví modře. Zvýrazňování syntaxe je velmi užitečné, a když si na něj zvyknete, pravděpodobně už nebudete chtít používat editor bez této funkce.



Obrázek 1.8 Zvýrazňování syntaxe ulehčuje práci vývojářům

Výchozí podoba ukázkového webu

Podívejme se na ukázkové testovací stránky, které postupně předěláme do dynamické podoby. Najdeme je v adresáři /01/ mezi zdrojovými kódy doprovázejícími tuto knihu. Pokud jste si server nainstalovali stejně jako já, je cesta k těmto stránkám *C:\xampp\ htdocs\kniha\01\.* Jak vidíme, v tomto adresáři máme:

- tři statické stránky: index.html, o-nas.html a kontakt.html,
- adresář obrazky, ve kterém jsou obrázky použité na těchto stránkách,
- a adresář css, v němž je jenom jeden soubor style.css, který popisuje vzhled stránek.

Otevřeme si v prohlížeči soubor index.html. Měl by vypadat jako na obrázku 1.9.



Obrázek 1.9 Soubor index.html zobrazený v prohlížeči

Teď si rychle prohlédneme zdrojový kód souboru *index.html*, abychom přesně poznali stránky, které budeme předělávat do dynamické podoby.

Nejdříve definujeme typ souboru, kódování, titulek stránky, připojíme soubor s definicí stylu, ikonku, informace pro roboty procházející stránky na Internetu a prozatím prázdné značky pro metapopis a klíčová slova:

Teď následuje samotné tělo stránky. Uspořádání jednotlivých bloků ukazuje obrázek 1.10.

Testovací st	Programování WWW stránek pro úplné začátečníky …a statické stránky se změní na dynamické blok pro hlavičku
Domů	Programování WWW stránek pro úplné začátečníky
0 nás blok pro menu	Tito stránky slouží jako podpůrný materiál ke knize <u>Programování WWW stránek pro</u> <u>úplné začátečníky</u> . V první kapitole jsou tito stránky statické, ale postupně je předěláme do dynamické podoby. blok pro hlavní obsah
	Přidáme kontaktní formulář, fotogalerii, knihu návštěv a také administrační zónu ve které budeme vědět spravovat obsah stránek.
Uká	zka ke knize: <u>Program blok pro paričku k pro úplné začátečníky</u>

Obrázek 1.10 Uspořádaní bloků ukázkové stránky

Celá stránka je zabalená v bloku obalStranky. Nahoře je hlavička, která obsahuje dva bloky – jeden vlevo pro logo a druhý vpravo pro hlavní nadpis a podnadpis stránek.

```
<body>
</div id="obalStranky">
</div id="hlavicka">
</div id="hlavicka">
</div id="logo">
</a href="index.html" />
</a>
<//div>
</div>
</div id="nadpisy">
</div id="nadpisy">
</div>
</div>
</div>
</div.a statické stránky se změní na
</div>
```

Střední část stránky obsahuje také dva bloky. Vlevo je sloupec pro menu a případně další malé bloky. Vpravo se zobrazuje samotný obsah stránek – hlavně texty.

```
<div class="menulink">0 nás</div>
         \langle a \rangle
         <a href="kontakt.html">
           <div class="menuLink">Kontakt</div>
         \langle /a \rangle
      </div>
   </div>
   <div id="obsah">
      <h3>Programování WWW stránek pro úplné začátečníky</h3>
      Tyto stránky slouží jako podpůrný materiál ke knize
      <a href="http://knihy.cpress.cz/k1868" target="_blank">
      <strong>Programování WWW stránek pro úplné
      začátečníky</strong></a>. V první kapitole jsou tyto
      stránky statické, ale postupně je předěláme do
      dynamické podoby.
      Přidáme kontaktní formulář, fotogalerii, knihu
      návštěv a také administrační zónu. v které budeme
      umět spravovat obsah stránek.
   </div>
 <div style="clear:both:"></div>
</div>
```

Nakonec přidáme patičku, zavřeme párové značky a stránka je hotová.

```
<div id="paticka">
    Ukázka ke knize: <a href="http://knihy.cpress.cz/k1868"
    target="_blank"><strong>Programování WWW stránek
    pro úplné začátečníky</strong></a>
    <//div>
    </div>
    </body>
</html>
```

Tím jsme dokončili první kapitolu a můžeme se pustit do další, v níž se budeme věnovat dynamickému zobrazení obsahu webové stránky.

Kapitola 2 **Dynamické zobrazování obsahu**

Jak jsme si již řekli, mezi základní vlastnosti dynamických stránek patří možnost dynamického generování výsledného kódu, který se odesílá uživateli. Také jsme už zmínili, že jednotlivé části stránky jsou obyčejně uloženy v samostatných souborech. Tento přístup má několik výhodných vlastností:

- Každou část stránky (hlavička, menu, patička) máme uloženu v samostatném souboru, a je tudíž velice snadné udělat jakoukoliv změnu.
- Změnu stačí udělat v jediném souboru a ta se projeví na všech stránkách, kde se tento soubor používá jako část výsledného zdrojového kódu (mozaiky).
- Díky tomuto dělení jsou i samotné soubory kratší a přehlednější.
- Při vývoji se tak snadněji hledají chyby a později provádějí změny a úpravy a doplňují nové funkce.

Přetypování statických souborů

Prvním krokem k dynamickému webu je přetypování souborů z typu *.html na typ, ve kterém jsou skripty jazyka PHP – *.php. Je to nezbytné, aby server věděl, jak má k souboru přistupovat – aby vykonal skripty zapsané v souboru a nebral je pouze jako statickou stránku.

Poznámka: Všimněte si adresáře *C:\xampp\htdocs\kniha\ts*. Ts odpovídá zkratce Testovací stránky. V tomto adresáři jsou stejné soubory, jako byly v adresáři *C:\xampp\htdocs\kniha\01*. Soubory jsou v *.html formátu.

Přetypování souborů je velmi jednoduché. Následující změny budeme vykonávat na souborech v adresáři *C:\xampp\htdocs\kniha\ts*.

- 1. Otevřeme si soubor *index.html*. Můžeme použít editor PSPad, Poznámkový blok nebo jiný textový editor. Je to běžná statická stránka.
- 2. Klepneme na možnost Soubor a Uložit jako.
- 3. Otevře se nové okno, jak je vidět na obrázku 2.1. V nejspodnějším řádku (možnost Uložit ve formátu) zvolíme možnost Všechny soubory *.*. V řádku pro Název souboru napíšeme index.php. Klepneme na Uložit.

Ulož ako						X
Uložiť do:	📗 ts			- 0 🕯	₽ 🖽 -	
Použité miesta	Názov Css Obrazky	Dátum úpr	Тур	Veľkosť	Značky	
Počítač						
2	Názov súboru: Uložiť vo formáte:	index.php PHP files (*.p	hp;*.php3;*.php	4;*.phtml;*.inc)	•	Uložiť Zrušiť
		 HTML Docum PHP files (* p Javascript files VBScript files XML Docum Cascading S Tcl/Tk Files Všetky súbor 	ment (*.html;*.htm hp;*.php3:*.php es (*.js) ; (*.vbs;*.asa) ent (*.xml;*.xsd;*. tylesheets (*.css) (*.tcl) y (*.*)	n;*.shtml;*.hhc) 4;*.phtml;*.inc) xsl;*.xslt;*.vxml;*)	.dtd;*.fo;*.xul;*.rd)

Obrázek 2.1 Okno s možnostmi pro uložení souboru

- **4.** Stejným způsobem jako v krocích 1 až 3 přetypujeme i soubory *o-nas.html* na *o-nas. php* a *kontakt.html* na *kontakt.php*.
- **5.** Všechny soubory s příponou *.html teď smažeme.
- **6.** Pokud nemáme spuštěný webový server, učiníme tak podle postupu v odstavci **Spuštění Xampp serveru** v předešlé kapitole.
- 7. Pokud už máme server spuštěný, zobrazíme si stránku http://localhost/kniha/ts/index.php.
- 8. Měla by se zobrazit úvodní stránka testovacího webu.

Problém však nastane, když klepneme na některý z odkazů v menu nebo na logo testovacích stránek v hlavičce. Tyto odkazy totiž vedou na soubory s příponou *.html, které jsme již smazali. Prohlížeč nám ukáže pouze chybovou stránku s hláškou "Stránka nenalezena".

Odkazy v menu a odkaz z loga v hlavičce budeme muset změnit tak, aby odkazovaly na stránky s příponou *.php.

1. Otevřeme si soubor *index.php* a změníme odkaz v řádku 16 tak, aby parametr href v odkazu směřoval na soubor *index.php*. Výsledný odkaz bude vypadat následovně:

 Podobným způsobem změníme i všechny odkazy v menu. Nacházejí se na řádcích 26 až 28.

```
<a href="index.php"><div class="menuLink">
    Domů</div></a>
<a href="o-nas.php"><div class="menuLink"></a>
```

```
O nás</div></a>
```

```
<a href="kontakt.php"><div class="menuLink">
Kontakt</div></a>
```

3. Stejným způsobem upravíme i odkazy v souborech *o-nas.php* a *kontakt.php*.

Opět si otevřeme stránku *index.php* v prohlížeči (na adrese *http://localhost/kniha/ts/index.php*). Po klepnutí na kterýkoliv odkaz na stránce (v menu nebo na logo) se dostaneme na existující stránku s příponou *.php.

První krok k dynamickému webu tak máme úspěšně za sebou.

Ahoj světe!

Teď nastal čas, abychom si ukázali, jak se vytvářejí skripty v jazyce PHP a jak přesně fungují. Generování výsledné stránky se skládá ze dvou základních částí:

- Veškerý (X)HTML kód, který je mimo značek <?php a ?>, se jako část výsledné statické stránky posílá uživateli beze změn tak jak je.
- Instrukce pro webový server, které jsou mezi značkami <?php a ?>, se vykonají a (X)HTML kód, jenž z nich vznikne, se jako součást výsledné statické stránky pošle prohlížeči uživatele.

Vše si podrobně vysvětlíme a princip fungování lépe pochopíte na ukázkovém skriptu *ahoj.php*, který najdeme v adresáři *C:\xampp\htdocs\kniha\02*. Vypadá následovně:

Na začátku skriptu (prvních pět řádků) máme pouze klasický HTML kód, který bude ve výsledné stránce beze změn.

Dále je kód mezi značkami <?php a ?>. Jak jsme si již řekli, tento kód vlastně představuje instrukce pro webový server. Příkaz echo znamená, že se textový řetězec uvedený za ním mezi apostrofy pouze vypíše. Tento text se pak vloží do výsledné statické stránky jako běžný HTML kód.

Poslední dva řádky jsou běžné HTML uzavírací značky. Vypíšou se stejně jako kód v prvním bodě – tak jak jsou.

Poznámka: Značky <?php a ?> se do výsledného zdrojového kódu stránky (který se posílá uživateli) nevkládají. Slouží pouze k tomu, aby webový server věděl, že kód uvedený mezi nimi není běžný text, ale instrukce, které je třeba vykonat.

Pokud si tento velmi jednoduchý skript spustíme (na adrese *http://localhost/kniha/02/ahoj. php*), uvidíme na obrazovce vypsaný text **Ahoj světe!** Když si zobrazíme zdrojový kód, uvidíme, že je to klasický statický HTML kód, tak jak jej známe (bez značek <?php a ?>).

Poznámka: Možná si říkáte, že jsme mohli stejného výsledku dosáhnout i bez použití jazyka PHP – stačilo by napsat statickou stránku, která by se jednoduše odeslala uživateli. Máte pravdu, ale teď si už ukážeme práci s proměnnými, které jsou dalším krokem k plnohodnotnému dynamickému webu.

Ukládáme údaje do proměnných

Nyní si ukážeme práci s proměnnými a začneme objevovat další výhody dynamických stránek. Ve skriptu *ahoj.php* si upravme kód mezi značkami <?php a ?> do této podoby:

```
<?php
$uzivatel='Host';
echo 'Ahoj '.$uzivatel.'!';
?>
```

Opět si zobrazíme stránku *http://localhost/kniha/02/ahoj.php* a měli bychom vidět text **Ahoj Host!**

Přidali jsme novou proměnnou se jménem *suzivatel* a přiřadili jí hodnotu **Host**. Příkaz echo teď vypíše řetězec **Ahoj**, pak hodnotu proměnné *suzivatel* a následně vykřičník (!) – tedy celkově tři textové řetězce. Ty jsou mezi sebou spojené znakem "tečka", abychom nemuseli použít celkově třikrát příkaz echo s vypsáním pouze jediného řetězce. Kód je tak lépe organizovaný, jednodušší a ve výsledku výrazně přehlednější.

Jednotlivé příkazy jazyka PHP jsou od sebe odděleny středníkem (;). V našem příkladě teď máme dva příkazy – první pro přiřazení hodnoty **Host** proměnné *suzivatel* a druhý pro vypsání tří řetězců spojených tečkou.

Tvorba proměnných se řídí několika základními pravidly:

- Před použitím není třeba proměnnou deklarovat (neboli založit) jako v jiných jazycích.
- Proměnné mají tradiční typy jako celé číslo, desetinné číslo, textový řetězec. Ve většině případů se není třeba pozastavovat nad přetypováním (změna z textového řetězce na číselný typ či naopak). Vše se děje automaticky.
- Jméno proměnné vždy začíná znakem dolaru \$ (správně: \$uzivatel).
- Jméno proměnné se může skládat z písmen anglické abecedy, čísel a znaku _ (správně: \$uzivatel, \$uzivatel, \$uzivatel).
- Číslo však nesmí následovat hned po znaku \$ (špatně: \$2uzivate1).
- Diakritika se nepoužívá (špatně: \$žlutá, správně: \$zluta).

- Jméno proměnné nesmí obsahovat mezeru ani interpunkční znaménka ani tečku (špatně: \$zluta!, \$modra.1, \$cervena 2, správně: \$zluta, \$modra1, \$cervena2).
- Když si zvolíme jméno proměnné složené ze dvou slov, je vhodné je oddělit znakem _ nebo druhé slovo začít velkým písmenem je to důležité kvůli přehlednosti kódu (správně: \$zluta_barva, \$modraBarva).
- Vždy se snažme zvolit jméno proměnné co nejvýstižněji, ale tak, aby název nebyl příliš dlouhý.

^A* Tip: Hledat chybu spočívající v překlepu v názvu proměnné je někdy poměrně náročné a název proměnné \$ciselnyIdentifikatorPrihlasenehoUzivatele nám to určitě neusnadní.

Tip: Při vývoji stránek je vhodné jednotlivé části kódu okomentovat. Určitě to oceníte při pozdějším návratu ke kódům třeba při dodělávání nových funkcí nebo dalším vylepšování stránek. Ve skriptech jazyka PHP existují dva druhy komentářů:

- // obyčejný jednořádkový komentář
- /* delší, i několikařádkový komentář */

Komentáře budeme používat i při skriptech v dalších kapitolách této knihy.

Volání externích souborů při generování dynamické stránky

Celý princip fungování volání externích skriptů si ukážeme na velmi jednoduchém příkladu a poté jej použijeme i na testovací stránce.

Pro volání externích souborů a skriptů existují dvě funkce – require a include. Princip jejich fungování ukazuje obrázek 2.2.



Obrázek 2.2 Fungování funkcí require a include

- 1. V hlavním skriptu použijeme funkci require (nebo include) pro zavolání nějakého externího souboru či skriptu.
- Pokud tento soubor existuje, obě funkce se chovají stejně požadovaný soubor se připojí a běh hlavního skriptu běží dál.

- 3. Rozdílně se však chovají v případě, kdy požadovaný soubor neexistuje:
 - funkce require skončí běh hlavního skriptu s chybovou hláškou,
 - funkce include běh hlavního skriptu neukončí.

Rozdílné chování funkcí při neexistujícím externím souboru určuje, kdy je která vhodnější:

- Funkce require je vhodná pro volání životně důležitých souborů. Používá se třeba při volání souboru, v kterém jsou definovány přístupové údaje k databázi. Pokud se je totiž nepodaří získat, nepodaří se připojit k databázi a získat tak požadované informace.
- Funkce include se pak používá u souborů, bez kterých se lze obejít, např. skript pro anketu apod. Anketa se sice nenačítá, ale uživatel bude i přesto rád, že se dostal k požadovaným informacím a že stránka neskončila chybou.

Poznámka: V praxi je však vhodné pohlídat, aby všechny volané soubory existovaly a stránka se načetla bez chyb nebo varování a tak, jak jsme to měli v úmyslu. V praxi by se stránka měla chovat stejně bez ohledu na to, zda používáme funkci require, nebo include.

Volání externích souborů – princip fungování

Otevřeme si hlavní skript, ze kterého budeme volat jiný, externí skript. Najdeme ho mezi zdrojovými kódy k druhé kapitole (*C*:*xampp\htdocs\kniha\02\hlavni.php*). Zdrojový kód hlavního skriptu bude vypadat následovně:



Obrázek 2.3 Chybová hláška při pokusu o volání neexistujícího souboru externi.php pomocí funkce include()

Nejdříve použijeme funkci include a zkusíme zavolat neexistující soubor *externi.php* zobrazením stránky *http://localhost/kniha/02/hlavni.php*.

Jak vidíte, server vypsal varování, že se pokoušíme zavolat neexistující soubor. Běh hlavního skriptu však pokračoval a byla vypsána i věta **Text vypsaný po zavolání externího skriptu.** Vzhled obrazovky ilustruje obrázek 2.3.

Změňme teď funkci volání skriptů z include na require. Kód bude vypadat následovně:

<?php require "externi.php"; ?>

Opět si zobrazíme stránku *http://localhost/kniha/02/hlavni.php*. Server znovu vypíše varování, ale i fatální chybu, a běh skriptu se ukončí. V tomto případě už skript nevypsal větu **Text vypsaný po zavolání externího skriptu.** Vzhled obrazovky ilustruje obrázek 2.4.



Obrázek 2.4 Chybová hláška při pokusu o volání neexistujícího souboru externi.php pomocí funkce require()

Vytvoříme si nový soubor (v adresáři *C:\xampp\htdocs\kniha\02*) a uložíme jej jako *externi. php*. Kód tohoto skriptu bude velmi krátký – bude obsahovat pouze následující větu:

Text, který byl v externím skriptu.

Znovu si zobrazíme stránku *http://localhost/kniha/02/hlavni.php*. Text vypsaný na obrazovce by měl vypadat jako na obrázku 2.5.

Obrázek 2.5 Úspěšně jsme zavolali externí skript, který obsahoval střední větu



Úloha: V souboru *hlavni.php* změňte funkci, která volá externí skript, z require zpátky na include. Znovu si stránku *hlavni.php* zobrazte v prohlížeči. Výsledek by měl vypadal stejně jako na obrázku 2.5, díky tomu, že volaný soubor existuje.

Části stránky v samostatných souborech a jejich volání

Teď se vrhneme na "rozbití" stránek na několik častí, které jsme zmiňovali již dříve. Stránku si rozbijeme na několik častí. Následující úpravy se budou opět týkat souborů v adresáři *C:\xampp\htdocs\kniha\ts*.

Testovaci	stránky hlavicka.php	Programování WWW stránek pro úplné začátečníky …a statické stránky se změní na dynamické
Domů		
O nás		
menu.php		
	Ukázka ke knize: <u>Programová</u> p	aticka.php <u>ánek pro úplné začátečníky</u>

Obrázek 2.6 Kód některých částí statické stránky vložíme do samostatných souborů, abychom jej pak mohli snadno měnit na jediném místě a změna se projevila všude tam, kde kód vkládáme do jiné stránky

Jak je vidět na obrázku 2.6, vytvoříme si samostatné soubory pro:

- hlavičku,
- ∎ menu,
- patičku.

Tyto soubory pak jednoduše zavoláme a ony do výsledného zdrojového kódu přispějí svojí částí, svým obsahem.

- 1. Otevřeme si soubor *index.php*.
- **2.** Označíme a zkopírujeme následující část kódu, ve které se popisuje hlavička stránky (nachází se na řádcích 14 až 22):

```
</div>
</div id="nadpisy">
</div id="nadpisy">
</hi>
Programování WWW stránek pro úplné začátečníky</hi>
</h2>
</div>
</div>
</div>
```

- **3.** Klepneme na nabídku **Soubor** a zde na příkaz **Nový**. V nově otevřeném okně si vybereme možnost **PHP** a klepneme na **OK**.
- **4.** Kód, který nám editor předepsal, prozatím můžeme smazat a nahradit jej zkopírovaným kódem pro hlavičku. Celý soubor pak bude obsahovat pouze zdrojový kód jako ve 2. bodě tohoto postupu.
- 5. Klepneme na Uložit jako a napíšeme jméno souboru: hlavicka.php. V nabídce Typ souboru by měla být vybrána možnost PHP files a v závorce vypsáno několik možných přípon PHP souborů.
- 6. Zkontrolujeme si adresář, do kterého se soubor uloží. Pokud je nastaven na C:\xampp\ htdocs\kniha\ts\, klepneme na Uložit; pokud není, tak ho tímto způsobem nastavíme a až poté ho uložíme.
- 7. To stejné teď musíme udělat i pro menu a patičku.
- **8.** Připravíme si samostatný soubor pro menu. Z původního souboru *index.php* si zkopírujeme následující část kódu (řádky 25 až 29):

```
<a href="kontakt.php"><div class="menuLink">
Kontakt</div></a>
```

```
</div>
```

- 9. Tento kód uložíme jako menu.php.
- **10.** Poslední částí bude kód pro patičku. V souboru *index.php* ho najdeme na řádcích 38 až 40:

</div>

- **11.** Tento kód uložíme jako *paticka.php*.
- **12.** Právě jsme si vytvořili soubory, které budeme jako část mozaiky vkládat do výsledných zdrojových kódů dynamických stránek. Proto původní kód pro hlavičku (krok 2 tohoto postupu) můžeme nahradit příkazem, který zavolá tento externí skript:

```
<?php
// zavolání skriptu s kódem pro hlavičku
require "hlavicka.php";
?>
```

13. Kód z bodu 8 nahraďte příkazem, který zavolá externí skript menu.php:

```
<?php
// zavolání skriptu s kódem pro menu
require "menu.php";
?>
```

14. Kód pro patičku v souboru *index.php* (bod 10 tohoto postupu) změníme na příkaz, kterým zavoláme externí soubor s kódem pro patičku:

```
<?php
// zavolání skriptu s kódem pro patičku
require "paticka.php";
?>
```

Stejným způsobem upravíme i skripty *o-nas.php* a *kontakt.php*. Nahradíme kód pro hlavičku, menu a patičku pouze voláním už vytvořených externích skriptů – zopakujeme kroky 12 až 14 předešlého postupu.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd";</pre>
<html xmlns="http://www.w3.org/1999/xhtml">
  chead>
     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
     <title>Testovací stránkv</title>
      <style type="text/css" media="all">@import "css/style.css";</style>
     <link rel="shortcut icon" href="obrazky/favicon.ico" />
     <meta name="robots" content="index,follow" />
     <meta name="description" content="" />
     <meta name="keywords" content="" />
  </head>
  <body>
     <div id="obalStranky">
        <?php
        require "hlavicka.php";
        2>
        </div>
        <div id="stred">
           <div id="levySloupec">
                                                             zbytečně opakovaná část kódu
               <?php
               require "menu.php";
           </div>
           <div id="obsah">
              <h3>Programování WWW stránek pro úplné začátečníky</h3>
               Tito stránky slouží jako podpůrný materiál ke knize <a href="http://knihy.cpress.cz/k1868"</p>
               target=" blank"><strong>Programování WWW stránek pro
              úplné začátečníky</strong></a>. V první kapitole jsou tito stránky statické, ale postupně je
              předěláme do dynamické podoby.
              Přidáme kontaktní formulář, fotogalerii, knihu návštěv a také administrační zónu ve které
              budeme vědět spravovat obsah stránek.
           </div>
           <div style="clear:both;"></div>
        </div>
        <?php
                                                             zbytečně opakovaná část kódu
        require "paticka.php";
        ?>
     </div>
   </body>
</html>
```

Obrázek 2.7 Každá podstránka ještě stále obsahuje zbytečně mnoho duplikovaných částí zdrojového kódu

Upravené výsledné skripty si můžeme zobrazit i v prohlížeči na adrese *http://localhost/kniha/ts/index.php*. Stránka vypadá úplně stejně jako v první kapitole, kdy to byly pouze statické stránky.

Zkusme však změnit text v hlavičce, třeba spodní podnadpis. Otevřeme si soubor *hlavicka. php* a přepíšeme text mezi značkami <h2> a </h2>. Napíšeme tam např. text **Tuto stránku upravil** a svoje jméno. Soubor uložíme a zobrazíme si stránku *index.php* v prohlížeči – načtením stránky *http://localhost/kniha/ts/index.php*.

Teď klepneme na odkazy na obě další stránky – **O nás** i **Kontakt**. Po jejich načtení si všimněme, že podnadpis v hlavičce je změněný. A změnu jsme přitom provedli pouze v jediném souboru – *hlavicka.php*. V tom spočívá jedna z klíčových výhod dynamických stránek.

Zamysleme se, jak bychom asi postupovali, jestliže bychom chtěli přidat novou podstránku, např. s novým textem. Museli bychom si zkopírovat celý soubor (třeba *o-nas.php*) a pak původní text v hlavní části zaměnit za nový. Když se však zamyslíme, pořád to není ideální, protože v každém souboru s textem jsou zbytečně duplikovány řádky, např. hlavička stránky mezi značkami <head> a </head>.

Zbytečně duplikované části jsou zvýrazněny na obrázku 2.7.

Proto se v praxi obvykle používá pouze jeden soubor *index.php*, do kterého se pak volají externí soubory tak, aby výsledná stránka, jež se posílá uživateli, vypadala tak, jak chceme. Tento princip blíže popisuje obrázek 2.8.



Obrázek 2.8 Struktura webových stránek s jediným souborem index.php, do kterého se následně volají externí skripty, tak aby spolu vytvořily stránku, jež se pak pošle uživateli

Teď si ukážeme, jak takové stránky fungují. Používat budeme soubor *index.php* jako hlavní nebo základní a do něj pak budeme vkládat externí skripty nejen pro hlavičku, menu a patičku, ale také pro všechny texty – hlavní obsahovou část stránky.

1. Otevřeme si soubor *o-nas.php* v adresáři *C:\xampp\htdocs\kniha\ts*. Změníme jeho zdrojový kód pouze na:

```
<h3>0 nás</h3>
Text o nás. Tento text slouží jen na to, aby zde bylo něco
napsáno, abychom lépe pochopili všechno, co v knize probereme.
```

Ponecháme jen obsahovou část – nadpis a samotný text.

2. Soubor *o-nas.php* uložíme a otevřeme si soubor *kontakt.php*. Změníme ho do následující podoby:

```
<h3>Kontakt</h3>
Tady by standardně byly vypsány kontaktní informace.
Později přidáme funkční kontaktní formulář.
```

3. Uložíme ho a otevřeme si soubor *index.php*. Označíme část zdrojového kódu pro samotný text a uložíme ji do nového souboru *titulka.php*:

```
<h3>Programování WWW stránek pro úplné začátečníky</h3>
Tyto stránky slouží jako podpůrný materiál ke knize
<a href="http://knihy.cpress.cz/kl868" target="_blank">
<strong>Programování WWW stránek pro úplné začátečníky
</strong></a>. V první kapitole jsou tyto stránky statické, ale
postupně je předěláme do dynamické podoby.
Přidáme kontaktní formulář, fotogalerii, knihu návštěv
a také administrační zónu, v které budeme umět spravovat
obsah stránek.
```

- **4.** Když máme text pro titulní stránku uložený v samostatném souboru, můžeme ho vymazat ze souboru *index.php*.
- **5.** Vytvoříme si ještě soubor, který se načítá, pokud uživatel bude chtít zobrazit stránku, jež na našem webu neexistuje. Vložíme do něj následující text a uložíme ho jako *error404.php*.

```
<h3>Stránka nenalezena</h3>
Litujeme, ale požadovaná stránka neexistuje.
```

Jak však bude webový server vědět, který z těchto souborů má zavolat a který text zobrazit? To vyřešíme nyní.

Parametry v URL adrese

Každá stránka musí mít vlastní unikátní, tedy jedinečnou URL adresu. Právě podle ní bude webový server vědět, který soubor s textem má zavolat. Vzhledem k tomu, že budeme používat pouze jeden základní soubor (*index.php*), musíme použít takzvaný parametr, který v kombinaci s názvem souboru vytvoří unikátní URL adresu konkrétní podstránky našeho webu.

Jak vypadá URL adresa stránky s parametry?

URL adresa stránky, tak jak ji prozatím známe, vypadá následovně:

- http://localhost/kniha/ts/index.php taková adresa obsahuje prakticky jen jméno souboru, který se má zobrazit.
- *http://localhost/kniha/ts/index.php?text=nazev-textu* tato adresa už obsahuje jméno souboru, ale také parametr text s hodnotou nazev-textu.

Poznámka: URL adresa může obsahovat i více parametrů. Ty jsou pak spojeny znakem & – např. takto: *http://localhost/kniha/ts/index.php?text=nazev-textu&anketa=ano&reklama=ne*. V této kapitole však budeme používat URL adresy pouze s jedním parametrem.

Načítání hodnot parametrů z URL adresy

Aby webový server věděl, který externí soubor s textem má zavolat, je třeba, aby skript hodnotu parametru z URL adresy probral. Jak takové načítání hodnot parametrů funguje, si ukážeme na jednoduchém příkladu.

V adresáři *C:\xampp\htdocs\kniha\02* se nachází PHP skript se jménem *prober.php*, který obsahuje následující kód:

```
<?php
echo 'Hodnota: '.$_GET['par'];
?>
```

Tento skript, jak už víme, vypíše textový řetězec **Hodnota:** a pak hodnotu parametru. Tato hodnota je uložena v proměnné *§_GET['par']*.

Část \$_GET[] znamená, že se hodnota načte z pole (speciální typ proměnné), které se naplní hodnotami parametrů z URL adresy. Řetězec 'par' uvnitř znamená, že se bude načítat hodnota parametru s názvem par.

Zobrazme teď stránku *http://localhost/kniha/02/prober.php*. Na obrazovce je napsáno pouze slovo **Hodnota:**. Je to proto, že parametr par nemá nastavenu žádnou hodnotu.

Firefox 🔻		
http://localhost/kniha/02/prober.php +	-	
★ → E http://localhost/kniha/02/prober.php ☆ ~ C		
Hodnota:		

Obrázek 2.9 Jestliže parametr par v URL adrese není nastaven, stránka vytvořená skriptem prober.php vypadá takto

Zkusme si zobrazit stránku *http://localhost/kniha/02/prober.php?par=hodnota-parametru*. Teď by mělo okno prohlížeče vypadat jako na obrázku 2.10. Skript načetl hodnotu parametru a vypsal ji.



Obrázek 2.10 Skript prober.php úspěšně probral hodnotu parametru par a vypsal ji

Když už víme, jak používat parametry v URL adrese stránek, upravíme i testovací stránky, tak aby zobrazovaly požadovaný text. Budeme také muset ošetřit možné problémové stavy, např. když není parametr určen, když má nesprávnou hodnotu nebo požadovaný soubor neexistuje.

Obrana proti vkládání škodlivého kódu přes parametr URL adresy

Velmi důležité je myslet také na bezpečnost naší aplikace. Musíme zabránit tomu, aby mohl kterýkoliv potenciální uživatel našeho webu narušit jeho bezpečnost. Nejdříve si ukažme, co hrozí. Uživatel se může pokusit dostat na naše stránky škodlivý kód, právě přes nechráněné a nedomyšlené načítání obsahu proměnných. Zobrazme si stránku *http://localhost/kniha/02/prober.php?par=<script>alert('Potencialne%20skodlivy%20kod')</script>.*

Přes obrazovku by se mělo zobrazit nové okno s hláškou JavaScriptu: "Potencialne skodlivy kod". Kód, který jsme vložili jako parametr do URL adresy, se vypsal ve zdrojovém kódu stránky (můžeme si jej zobrazit). Situaci na obrazovce vidíme na obrázku 2.11. Zdrojový kód stránky s nechráněně načteným parametrem zachycuje obrázek 2.12.

Firefox Fir	/script%3E +
+> http://localhost/ki	iha/02/prober.php?par= <script>alert('Potencialne skodlivy kod')</script> 🏫 🛪 💌
Hodnota:	
	Potencialne skodlivy kod
Transferring data from localbost	ОК

Obrázek 2.11 Neošetřené načítání parametrů může být velmi nebezpečné. Tento vložený skript pouze vykreslil nové okno s varováním.

Source of: http://localhost/kniha/02/prober.php?par=%3Cscript%3Ealert(%	
<u>File Edit View H</u> elp	
Hodnota: <script>alert('Potencialne skodlivy kod')</script>	

Obrázek 2.12 Zdrojový kód, který obsahuje parametr načtený bez ošetření. Právě tato část kódu způsobila zobrazení nového okna s varováním.

Tento javascriptový kód vyvolá pouze nové okno s upozorněním. Potenciální útočník by nepoužil takové okno, ale jiný škodlivý kód, kterým by se mohl nabourat do našeho systému.

Upravme kód v souboru prober.php do následující podoby:

echo 'Hodnota: '.htmlspecialchars(strip_tags(\$_GET['par']));

Když si teď zobrazíme stránku *http://localhost/kniha/02/prober.php?par=<script> alert('potencialne%20skodlivy%20kod')</script>*, uvidíme vypsaný pouze varovný text

jako na obrázku 2.13, aniž by se zobrazilo nové okno spuštěné JavaScriptem. Úspěšně jsme zabránili vložení potenciálně škodlivého kódu.



Obrázek 2.13 Při ošetření vstupu jsme zkomplikovali vkládání potenciálně nebezpečných skriptů

Funkce strip_tags() nejdříve odřízne značky <script> a </script>, mezi které se vkládá kód JavaScriptu. Funkce strip_tags() všeobecně ořezává ze zadaného řetězce všechny značky jazyka HTML a vrací řetězec bez nich.

Funkce htmlspecialchars() potenciálnímu útočníkovi dále ztěžuje napadení našeho webu. Není moc podstatné, jak vše přesně funguje, důležitý je výsledek – potenciálně nebezpečný kód se nespustí.

Kombinace těchto dvou funkcí je dobrým základem pro ochranu našeho webu.

Poznámka: Zabezpečení aplikací je velmi široké téma, kterému se v této knize nebudeme podrobně věnovat. Ukážeme si základní techniky a kroky, jak naše webové stránky ochránit proti nejčastějším útokům. Toto zabezpečení bude dostatečné pro běžné osobní stránky, určitě však ne třeba pro internetový platební systém v bance. Takové ambice však v této fázi zvládání jazyka PHP určitě nemáte.

Volání externích skriptů s texty a ošetření všech možností

Otevřeme soubor index.php v adresáři C:\xampp\htdocs\kniha\ts\ a mezi řádky

```
<div id="obsah">
```

a

</div>

napíšeme příkazy, které zavolají text, podle toho, jakou hodnotu bude mít parametr s názvem text. Skript se bude snažit zavolat soubor se stejným názvem, jako má parametr text (název souboru včetně přípony *.php*).

<?php
\$text=htmlspecialchars(strip_tags(\$_GET['text']));</pre>

Nejdříve si do proměnné text uložíme hodnotu parametru *S_GET['text']*. Použijeme při tom i dříve vysvětlené funkce htmlspecialchars() a strip_tags(). V proměnné *stext* teď máme bezpečný řetězec.

```
if($text==''){
    $text='titulka';
}
VOLÁNÍ EXTERNÍCH SOUBORŮ PŘI GENEROVÁNÍ DYNAMICKÉ STRÁNKY
```

Pokud parametr text neobsahuje hodnotu, znamená to, že chceme zobrazit úvodní stránku.

```
if (file_exists($text.'.php')) {
   require $text.'.php';
} else {
   require 'error404.php';
}
```

Těmito příkazy zjišťujeme, zda požadovaný soubor existuje. Pokud ne, zavoláme soubor *error404.php*, který uživatele informuje, že se pokouší zobrazit neexistující stránku (text).

?>

Použili jsme dva nové prvky:

Funkci pro ověření, zda požadovaný soubor existuje:

```
file_exists($text.'.php')
```

Tato funkce vrací hodnotu **true** (1), pokud požadovaný soubor existuje, a **false** (0), jestliže takový soubor neexistuje.

Podmínku:

Rozhodovací struktury budeme při programování používat velice často, protože se využívají k ovlivňování chodu skriptu. Konstrukce rozhodovací struktury je následující:

```
if ( podminka_která_se_vyhodnocuje ) {
    plati_větev
    příkazy_které_se_vykonaji_pokud_podminka_plati
} else {
    neplati_větev
    příkazy_které_se_vykonaji_pokud_podminka_neplati
}
```





Podmínka, která se vyhodnocuje, může být jakýkoliv výraz, o němž se dá jednoznačně určit, zda je pravdivý, nebo nikoliv. Podmínky se mohou týkat jak číselných hodnot, tak textových řetězců. Jak tvořit podmínky a jaké operátory je možné použít, názorně ukazuje tabulka 2.1.

typ	operátor	význam	příklad
Číselné hodnoty	==	Je rovný	\$cislo==5
	!=	Není rovný	\$cislo!=5
	<	Je menší	\$cislo<5
	<=	Je menší nebo rovný	\$cislo<=5
	>	Je vetší	\$cislo>5
	>=	Je větší nebo rovný	\$cislo>=5
Textové řetězce	==	Je shodný	\$text=="ano"
	!=	Není shodný	<pre>\$text!="ano"</pre>

Tabulka 2.1 Operátory nejčastěji používané k tvoření podmínek

Jak je vidět na obrázku 2.14, nejdříve se rozhoduje o pravdivosti podmínky. Pokud podmínka platí, vykonají se příkazy v příslušné větvi (mezi následujícími složenými závorkami). Pokud podmínka neplatí, vykonají se příkazy z druhé větve (uzavřené také mezi složené závorkami – viz dále).

Kromě úplné rozhodovací struktury existuje i jednodušší verze. Vypadá následovně:

```
if ( podmínka_která_se_vyhodnocuje ) {
    příkazy_které_se_vykonají_pokud_podmínka_platí
}
```

Příkazy mezi složenými závorkami se vykonají pouze tehdy, když podmínka (uvedená v kulaté závorce) platí. Pokud neplatí, pokračuje běh skriptu za touto strukturou, jak je vidět na obrázku 2.15.

Jak jste již zřejmě odhalili, naše rozhodovací struktura nejdříve zjistí, zda soubor v parametru URL s názvem *text* existuje. Pokud ano, pak struktura soubor zavolá a doplní jeho text do výsledného zdrojového kódu a ten se jako celek pošle uživateli (do jeho prohlížeče). Jestliže takový soubor neexistuje, zavolá struktura soubor *error404.php* a dá tak uživateli vědět, že se pokouší zobrazit stránku, která neexistuje.



Obrázek 2.15 Pokud podmínka platí, vykonají se příkazy uvnitř této struktury. Pokud podmínka neplatí, pokračuje běh skriptu za touto strukturou.

Nyní zbývá pouze upravit soubor *menu.php*, tak aby jednotlivé odkazy odkazovaly vždy pouze stránku *index.php*, ale pokaždé s jiným parametrem:

Teď už zbývá pouze naše řešení otestovat.

V prohlížeči zobrazíme stránku *http://localhost/kniha/ts/index.php*. Měli bychom vidět úvodní stránku, kterou už dobře známe. Klepneme na některý odkaz v navigační nabídce. A měla by se zobrazit požadovaná stránka.

Zkusme zobrazit i stránku, která neexistuje – např. *http://localhost/kniha/ts/index. php?text=neexistujici-obsah*. Měla by se zobrazit stránka s chybovou hláškou jako na obrázku 2.16.



Obrázek 2.16 Vzhled stránek v situaci, kdy se pokoušíme zobrazit neexistující stránku (text)

Gratulujme si, další krok k dynamickému webu máme za sebou. Teď si nabyté znalosti o práci s proměnnými, rozhodovacími strukturami a tvořením podmínek trochu rozšíříme.

Rozšiřujeme své znalosti

Pro další kapitoly budeme potřebovat umět používat časté konstrukce, jako jsou rozhodovací struktury, nebo umět tvořit podmínky a cykly. Na následujících stránkách si vše vysvětlíme přímo na příkladech. Také si řekneme něco víc o správných programátorských návycích.

Správné programátorské návyky

Nejdříve bychom si měli říct něco o správných návycích při programování. Když se je naučíme používat už od začátku, oceníme to v budoucnosti, kdy budeme tvořit větší projekty.

- Pravidla pro tvorbu proměnných jsme si již objasnili dříve v této kapitole.
- Pravidla pro psaní kódu:
 - Vnořené řádky odsazujeme (třeba tabelátorem), aby bylo ihned vizuálně poznat, který řádek patří do dané rozhodovací struktury nebo cyklu.
 - Jednotlivé logické části kódu můžeme odsadit volným (prázdným) řádkem zvýší se tak přehlednost kódu.
 - Kód komentujme když se po pár měsících či letech vrátíme ke svým zdrojovým kódům, uvidíme, že nám velmi v orientaci v kódu pomohou.

Podmínky a jejich vyhodnocování

Jak jsme řekli již dříve, podmínkové výrazy se ve skriptování a programování využívají velice často, protože jsou nezbytně nutné pro směrování běhu skriptů. Teď si ukážeme několik dalších možností použití i s komentářem. Jednotlivé kódy si můžeme tak, jak jsou napsány tady, uložit jako PHP skript a zkusit si jeho chování i "naostro".

```
<?php
$cislo=5;
if ( $cislo>4 && $cislo<6 ) {
echo 'Číslo je větší než 4 a zároveň menší než 6.';
}
?>
```

Do proměnné \$cislo jsme vložili hodnotu 5. Podmínkový výraz se teď skládá ze dvou částí – \$cislo>4 a \$cislo<6. Mezi nimi je logický operátor &&, který znamená anglické and – v překladu a současně. Tento operátor znamená, že musí platit obě podmínky, aby měl celý výraz hodnotu pravda.

Tento skript vypíše větu **Číslo je větší než 4 a zároveň menší než 6.**, protože hodnota **5** je větší než **4** a zároveň menší než **6**. Proto má i celý podmínkový výraz v závorce pravdivostní hodnotu **pravda** (platí).

Tip: Zkuste hodnotu proměnné změnit např. na 7. Skript v tomto případě nevypíše nic, protože sice platí první část podmínky (7>4), ale druhá část již neplatí (7<6). Proto neplatí ani celá podmínka – má nepravdivou hodnotu.</p>

Podívejme se na další příklad:

```
<?php

$cislo=7;

if ( $cislo>4 || $cislo<6 ) {

echo 'Číslo je větší než 4 nebo menší než 6.';

}

?>

ROZŠIŘUJEME SVÉ ZNALOSTI
```

Velice často se používá také logický operátor || – anglicky or – ve významu **anebo**. Znamená, že aby byl celý výraz vyhodnocen jako pravdivý, stačí, aby byla pravdivá alespoň jedna podmínka (nebo také obě).

Tento skript vypíše **Číslo je větší než 4 nebo menší než 6.**, protože platí první podmínka (7>4), a to při použití logického operátoru or postačuje, aby i celý výraz měl pravdivostní hodnotu **pravda** (platí).

```
<?php
$cislo=8;
if ( $cislo>4 && ( $cislo==5 || $cislo<9 ) ) {
    echo 'Číslo je větší než 4 a současně
        rovné 5 nebo menší než 9.';
}
</pre>
```

V praxi se také setkáme se závorkováním částečných výrazů do bloků uzavřených závorkami. Celý výraz se vyhodnotí jako pravdivý, pokud bude platit, že číslo v proměnné \$cislo je větší než 4, a zároveň bude platit i výraz v menší závorce. Ten bude platit, pokud je \$cislo rovné 5 nebo menší než 9.

Tento skript vypíše větu **Číslo je větší než 4 a současně rovné 5 nebo menší než 9.**, protože výraz ve vnitřní závorce platí (8 se sice nerovná 5, ale 8 je menší než 9, takže výraz ve vnitřní závorce je pravdivý) a současně platí, že 8 je větší než 4.



Úloha: Zkuste si sami různé kombinace podobných logických podmínek a závorek, tak abyste tuto problematiku dokonale pochopili – určitě se vám to v příštích kapitolách i v dalším programování bude hodit.

Vytváříme cykly

Cykly jsou v programování velmi často využívané, proto je důležité, abychom je uměli bez problémů používat a rozuměli jim. Cyklus je smyčka příkazů, která se opakuje tak dlouho, dokud není splněná výchozí podmínka (například dokud nějaká proměnná, jíž se při každém proběhnutí cyklu přičte jednička, nedosáhne určité číselné hodnoty).

V principu existují dvě kategorie cyklů:

- Cykly s pevným počtem opakování; používáme je, když víme, kolikrát se má nějaká část kódu (posloupnost příkazů) vykonat.
- Cykly, které vykonávají příkazy, dokud platí řídicí podmínka.

Cyklus s pevným počtem opakování

Tento typ cyklu se používá, když přesně víme, kolikrát se mají nějaké příkazy zopakovat. Řekněme, že chceme vypsat čísla o 1 do 10 a k nim jejich druhé mocniny. Můžeme na to použít následující skript s cyklem:

```
<?php
for($i=1;$i<=10;$i++){
```

```
echo 'Číslo: '.$i.': druhá mocnina = '.$i*$i.'<br />';
}
echo 'Aktuální hodnota i je '.$i;
?>
```

Tento velice jednoduchý skript vypíše deset řádků a použije v nich aktuální hodnoty řídicí proměnné \$ i. Co znamená celý první řádek tohoto skriptu?

- 1. Klíčové slovo for značí, že jde o skript s pevným počtem opakování.
- **2.** Část *i=1* inicializuje na začátku proměnnou *i na hodnotu*.
- Druhá část \$i <=10 je podmínka jaké hodnoty může maximálně proměnná dosáhnout. Určuje, kdy se zastaví provádění příkazů uvnitř cyklu – mezi složenými závorkami.
- 4. Poslední část \$i++ je proměnná s operátorem inkrementace (++). Určuje, co se má udělat s řídicí proměnnou (v tomto případě \$i) vždy po vykonání cyklu v našm případě se vždy přičte jednička a proměnná nabude nové hodnoty (1, pak 2, pak 3 atd.).



V praxi to vypadá jako na obrázku 2.17.

Po spuštění cyklu se stane následující:



- Hodnota řídicí proměnné se nastaví na 1 a vypíše se HTML kód Číslo: 1: druhá mocnina = 1
.
- **2.** Hodnota řídicí proměnné se zvýší o 1 (\$ i++) a opět se vykoná cyklus se všemi příkazy. Tentokrát se vypíše **Číslo: 2: druhá mocnina = 4<br**/>.
- **3.** Hodnota se opět změní a to se opakuje, až je hodnota řídicí proměnné rovna 10 a vypíše se řádek Číslo: 10: druhá mocnina = 100
br />.
- Opět se zvýší hodnota proměnné teď už na 11. Protože však již neplatí podmínka \$i<=10, příkazy uvnitř cyklu se už nevykonají.
- **5.** Běh skriptu pokračuje dále příkazy pod cyklem for. V našem případě to je řádek, který vypíše aktuální hodnotu proměnné \$i.

Zkrácené zvyšování a snižování hodnot

V předchozím příkladu jsme se možná pozastavili nad zápisem inkrementace (což je matematický termín): \$i++. Je to často využívaný zápis, který znamená, že se má hodnota proměnné zvýšit o 1. Je to stejné jako zápis \$i=\$i+1.

Výhodou tohoto zápisu je fakt, že je přehlednější a na zapsání téhož stačí méně znaků. Existuje také operátor \$i--, který hodnotu proměnné snižuje o 1. V tomto případě se tomu říká dekrementace.

Tyto zápisy se využívají velmi často v cyklech a v případě nejrůznějších počítadel.

Cyklus s podmínkou

Existují situace (a není jich málo), kdy potřebujeme vypsat nějaký seznam položek nebo podobných prvků, a přitom dopředu nevíme, kolikrát bude cyklus muset proběhnout.

Například při vyhledávání: Uživatel může zadat dotaz, jemuž nevyhovuje ani jeden článek či produkt. Může též zadat dotaz, kterému vyhovuje 10, 20 nebo také 500 článků či výrobků. V takovém případě se používá cyklus s podmínkou a příkazy uvnitř cyklu se opakují, dokud podmínka platí.

Zjednodušeně řečeno, vypisování informací pomocí tohoto cyklu funguje na principu "dokud máš co vypisovat, tak to vypisuj".

Předešlý skript můžeme upravit tak, aby fungoval jako cyklus s podmínkou:

```
<?php
$i=1;
while($i<=10){
    echo 'Číslo: '.$i.': druhá mocnina = '.$i*$i.'<br />';
    $i++;
}
echo 'Aktuální hodnota i je '.$i;
?>
```

Vypíše to stejné jako cyklus s příkazem for, ale funguje poněkud jinak. Vše názorně ukazuje obrázek 2.18.

Skript s tímto typem cyklu provádí následující kroky:

- Nejdříve jsme si nastavili proměnnou \$i na hodnotu 1.
- Klíčové slovo while znamená v češtině dokud. Příkazy v těle cyklu se tak budou vykonávat, dokud bude podmínka v závorce platit. Teď je



Obrázek 2.18 Fungování cyklů s podmínkou na začátku

hodnota proměnné \$ i rovna 1 a podmínka říká, že se mají příkazy vykonávat, pokud bude menší nebo rovna 10. Příkazy uvnitř se tedy vykonají.

- **3.** V tomto případě musíme sami zvětšovat hodnotu proměnné o 1 (je to vlastně počítadlo), abychom skript někdy i ukončili (aby nevznikl zacyklený tedy nekonečný cyklus).
- **4.** Pokud se hodnota proměnné zvýší na 11, podmínka \$i<=10 už neplatí a cyklus se ukončí. Pokračuje se příkazy za cyklem.

Kromě příkazu s podmínkou na začátku existuje i cyklus s podmínkou na konci. Postačí nám však dobře umět používat cyklus while().



Úloha: Zkuste si vytvořit několik cyklů sami a sledujte, jak se chovají a co vypíší. Zkuste napsat také nějaký složitější skript, který bude obsahovat cyklus i úplnou a neúplnou rozhodovací strukturu.



Řešení problému: Pokud se vám stane, že napíšete a spustíte zacyklený nekonečný cyklus, nepropadejte panice. V prohlížeči klepněte na ikonku **Zastavit načítání stránky**. Toto tlačítko se většinou nachází vedle tlačítka pro znovunačtení stránky nebo u tlačítka **Zpět**. Pokud to nepomůže, zavřete celé okno prohlížeče, skript opravte a pak jej spusťte znovu.



Poznámka: Nekonečného zacyklení se nemusíte bát. Učíte se a téměř jistě se vám občas podaří vytvořit a spustit nekonečný skript. I proto neděláme vývoj a testování webových aplikací na ostrém vzdáleném serveru, ale na lokálním počítači.

V případě, že se vám testovací stránky v adresáři *C:\xampp\htdocs\kniha\ts* nepodařilo dostat do stavu, v jakém by teď podle knihy měly být (nebo se vám nechce přepisovat a upravovat skripty podle postupů v této kapitole), můžete celý obsah adresáře *C:\xampp\ htdocs\kniha\ts* smazat a nahradit jej obsahem adresáře *C:\xampp\htdocs\kniha\02\ts*.

Důležité: Tento adresář obsahuje kompletní testovací stránky v provozuschopném stavu, v jakém mají být po výuce této kapitoly. Je nezbytné, abyste si případnou chybu nepřenášeli do dalších kapitol.



Kapitola 3 **Tvoříme kontaktní formulář**

Další z významných vlastností dynamických webových stránek jsou formuláře, pomocí nichž může návštěvník komunikovat se stránkou a ovlivňovat tak její chování. Pomocí formuláře může návštěvník odeslat e-mail se svým dotazem firmě, objednat si zboží, nebo pouze sdělit své pocity celému světu na sociální síti.

Mezi nejčastěji používané druhy formulářů patří:

- vyhledávání,
- kontaktní formuláře,
- registrační formuláře,
- přihlašovací formuláře,
- rezervační formuláře,
- objednávkové formuláře
- a mnohé jiné.

KONTAKTU	JTE NÁ S
Vaše jméno *	
Váš e-mail *	
Váš telefon	
Oblast	Obecné informace, objednávky, reklamace, fakturace 🔻
Předmět *	
Text:	
	Odeslat

Obrázek 3.1 Kontaktní formulář

Přihlašovací údaje		
Login: Nové heslo:	Zopakujte nové heslo:	
Zákaznické údaje	Dodací adresa (pokud je jiná než fakturační)	
Jméno a přijmení: (max. 40 znaků)	Jméno a příjmení/firma: (max. 40 znaků)	
Ulice a č.p:	Kontaktní osoba:	
Město:	Ulice a č.p:	
PSČ:	Město:	Přihlášení
		Uživatelské jméno:
Souhlasim s <u>obchodními podmínkami</u>		Heslo:
	Zpēt Zaregistrovat	Přihlásit
Obrázek 3.2 Registrační formulář		Obrázek 3.3 Přihlašovací

Obrázek 3.3 Přihlašovac formulář

Fungování formulářů

Princip fungování formulářů je velice jednoduchý. Samotný formulář tvoří pole a popisky či jiný text a tlačítka. Pole (vstup nebo anglicky input) slouží k zadávání údajů od uživatele. Existuje více typů vstupů – podrobněji se na ně podíváme později.

Hodnoty ze vstupu jsou poté odeslány PHP skriptu, který ověří, jestli jsou vyplněna všechna povinná pole a zda obsahují korektní údaje (kupř,

jestli má e-mailová adresa standardní tvar).

Pokud jsou data v pořádku, skript vykoná požadovanou operaci (zapíše je do databáze, odešle e-mail apod.). Jestliže data nejsou v pořádku, skript většinou vypíše chybovou hlášku a nabídne doplnění údajů do chybějících povinných polí nebo opravu špatně zadaných údajů (např. pokud telefonní číslo nebo e-mailová adresa mají nestandardní tvar nebo obsahují nepovolené znaky).





První formulář

Podívejme se na velice jednoduchý formulář a výkonný skript, který k němu patří. Formulář je v souboru *form.php* a v adresáři *C:\xampp\htdocs\kniha\03*. Celý soubor vypadá následovně:

Celý formulář je "zabalen" mezi značky <form> a </form>. V parametru action je uvedeno jméno skriptu, kterému se údaje z formuláře odešlou. Parametr method určuje metodu (způsob), jakou budou údaje odeslány (budeme se jí věnovat dále).

Formulář (jak je vidět na obrázku 3.5) obsahuje pouze jeden vstup – textové pole pro jméno, popisek k tomuto poli (mezi značkami <label> a </label>) a tlačítko pro odeslání celého formuláře.



Obrázek 3.5 První formulář

Formulář se odešle skriptu, který se jmenuje jmeno.php a vypadá takto:

Jak vidíme na obrázku 3.6, tento skript pouze vypíše větu **Vaše jméno je XY.**, kde XY je jméno, které jsme uvedli ve formuláři.

Spusťme si tedy skript s formulářem (v mém případě má adresu *http://localhost/kniha/03/ form.php*).

Měli bychom vidět jednoduchý formulář (jako na obrázku 3.5). Napišme své jméno a klepněme na tlačítko **Odeslat**. Pokud všechno proběhlo správně, měli bychom vidět text **Vaše jméno je XY.**, kde XY je jméno, které jsme zadali ve formuláři.



Obrázek 3.6 Výsledek po odeslání prvního formuláře se jménem

Firefox 🔻	tání stránky +
	ttp://localhost/kniha/03/kontakt.php · C
	Chyba spojení
	Firefox nemůže navázat spojení se serverem localhost.
	Server je dočasně nedostupný. Zkuste to prosím znovu za chvíli.
	 Pokud se vám nezobrazují ani ostatní stránky, zkontrolujte síťové připojení svého počítače.
	 Pokud je váš počítač chráněn firewallem nebo proxy serverem, zkontrolujte, zda má Firefox přístup na Internet.
	Zkusit znovu

Obrázek 3.7 Chybová hláška při nespuštěném serveru Apache

Toto je pouze náhled elektronické knihy. Zakoupení její plné verze je možné v elektronickém obchodě společnosti eReading.