26. Tvorba dynamických www stránek s využitím PHP

1 Statické × dynamické webové stránky

1.1 Statické stránky

Představují soubor jednotlivých a pomocí odkazů vzájemně propojených webových stránek. Každá stránka je "hotová" od A do Z a ve stavu v jakém ji zanechal její autor. Nevýhodou těchto stránek je jejich **obtížná udržitelnost** při nárocích na další rozšiřování webových stránek a zejména fakt, že nedokáží zobrazovat data uložená v databázi. Další nevýhodou je nemožnost zapojení jakékoliv přídavné inteligence do statikách www stránek (fulltextové vyhledávání ap).

Statické www stránky se proto hodí zejména pro malé weby a jsou dávno překonány tzv. dynamickými stránkami.

1.2 Dynamické stránky

Základem je programovací technologie, program (software) a data uložená v databázi. Program (**online obchod**, **redakční systém**, **elektronická aukce** ap.) umístěný na internetovém serveru, pak na základě požadavku na zobrazení konkrétní stránky webovou stránku dynamicky sestavuje a odesílá do prohlížeče uživatele. Zjednodušeně se dá říci, že stránka je sestavena na základě "šablony" určující vzhled a dat získaných z databáze (data jsou například seznam zboží, článků nebo konkrétní článek). Sestavení zajišťuje program běžící například na technologii **PHP**. Dynamické stránky tedy **fyzicky neexistují v uloženém stavu**, na rozdíl od stránek statických.

2 Uspořádání klient-server

Klient (browser, prohlížeč) je program, který komunikuje s uživatelem a na základě jeho pokynů se obrací na jednotlivé servery, získává od nich data a zobrazuje je. Nejběžnější klienti: *Microsoft Internet Explorer, Netscape Navigator, Mozilla*. Zdůraznit různorodost klientů.

Server je bezobslužný program, který přijímá a obsluhuje požadavky klientů. Zdůraznit, že WWW server je program, nikoli počítač. Na vlastnostech tohoto programu závisí, co server dovede a jaký má výkon. Nejběžnější servery: *Apache, Microsoft Internet Information Server, Zope*. Protože klienti a servery pocházejí od různých producentů, je velmi důležité dodržování standardů, aby programy byly schopny se navzájem domluvit.

PHP skripty se provádějí na straně serveru a klientu jsou odeslána pouze výsledná data. JavaScript se provádí až na střaně klienta.

3 Umisťování stránek na server

Nejběžnější postup: vytvořit stránky off-line na vlastním počítači a pak je přesunout na cílový WWW server. K přesunu se nejčastěji používá běžně dostupné FTP. MS Windows neobsahují vhodného klienta - lze nainstalovat např. volně šiřitelný *CoffeeCup Free FTP, FTP Commander* či *FileZilla*. Významným rizikem FTP je, že přenáší uživatelské heslo a jméno v otevřeném tvaru - při odposlechu lze zneužít. Vhodnější variantou je použití *scp*, které veškerou komunikaci šifruje. Volně šiřitelnou implementaci pro operační systém MS Windows je např. *Putty*, lepší je však použít grafický program *WinSCP*. V Linuxu bývá *scp* součástí distribuce.

Aby bylo stránky kam umístit, musíme mít vlastní server, nebo využít jednu z mnoha webhostingových služeb a mít zřízenou vlastní doménu (některé freehostingy poskytují zdarma doménu třetího řádu).

4 Web server APACHE

Apache je softwarový webový server s otevřeným kódem vyvíjený společností Apache Foundation. Je multiplatformní (pro Linux, BSD, Microsoft Windows, …), má modulární architekturu, podporu virtual hostů, podporu IPv6 a kompresi pomocí modulu *mod_gzip*. Má velké možnosti konfigurace.

5 Tvorba formuláře, metody POST a GET

5.1 Tvorba formuláře

Základní strukturu adresáře tvoří HTML párový tag <form>. Nejpoužívanější parametry jsou *action*, kterým se udává cíl, kam se má formulář odeslat a *method*, který specifikuje jakou metodou se mají formulářová data odeslat (POST a GET). Mezi tagy <form> a </form> se vkládají prvky formuláře (tagy <input>, <textarea>, <select>). Ke zpracování formuláře skriptem PHP je duležitý jejich atribut *name*, kde je název formulářové položky.

5.2 Metody POST a GET

Při použití metody GET se veškerá formulářová data předají jako součást URL za otazníkem, při použití metody POST se předají v těle dotazu, takže v URL nejsou vidět. Z toho plyne i vhodnost jejich použití – pokud se předaná data dají chápat jako parametry stránky (tedy např. předání ID článku nebo vyhledávaného řetězce), je vhodné je poslat metodou GET, v ostatních případech je lepší použít metodu POST (obzvláště pokud je dat hodně).

6 Postup zpracování webového formuláře skriptem

Odeslaná data z formuláře, jsou ve skriptu PHP přístupná přes globální proměnné $_POST, $_GET a $_FILES. Každá tato proměnná je datového typu **pole** a identifikátory jednotlivých položek jsou názvy prvků odesílaného formuláře.

Odesláná data od uživatele by se měla zkontrolovat, jestli jsou povinné položky vyplněné a zda mají správnou syntaxi. Správná syntaxe se kontroluje zpravidla pomocí regulárních výrazů.

Funkce *trim()* odstraní prázdné znaky (mezery, entery, tabulátory, …) ze začátku a konce řetězce. Funkce *ereg()* zkontroluje řetězec pomocí zadaného regulárního výrazu, při schodě vrátí **true**, jinak vrátí **false**.

```
if(Trim($_POST['jmeno']) != '')
{
  echo 'Jméno bylo vyplněno';
}
if(ereg('^[_a-zA-Z0-9\.\-]+@[_a-zA-Z0-9\.\-]+\.[a-zA-Z]{2,4}$',
        $_POST['email']))
{
  echo 'E-mail je ve správném formátu';
}
```

7 Funkce PHP pro práci s databází MySQL

Následující text popisuje práci s jedním aktuálním spojením s databázovým serverem. Pokud by jste v nějaké aplikaci potřebovali mít otevřeno více spojení, je třeba využít u mysql funkcí pracující s databází nepovinný parametr link, do kterého se vkládá výsledek funkce mysql_connect().

7.1 Vytvoření spojení

Funkce *mysql_connect()* vytvoří spojení s databází. Po ukončení práce s databází se spojení uzavře pomocí funkce *mysql_close()*.

```
mysql_connect('server', 'uživatelské jméno', 'heslo');
...
//práce s databází
...
mysql_close();
```

7.2 Výběr databáze

Jelikož na databázovém serveru je obvykle vytvořeno více databází, je potřeba vybrat tu, se kterou budeme pracovat. K tomu slouží funkce *mysql_select_db()*.

```
mysql_select_db('jmeno databaze');
```

7.3 Vytvoření dotazu

Dotazy na MySQL server mají klasickou SQL syntaxi. Předávájí se pomocí funkce *mysql_query()*, která při úšěšném dotazy vrátí sadů výsledků a proměnná **$result** bude na ní odkazovat. Při selhání dotazu vrátí logickou hodnotu **false**.

```
$query = 'SELECT * FROM knihy';
$result = mysql_query($query);
```

7.4 Zpracování výsledku

K uložení řádku do pole (asociativní i číselné indexy), na který aktuálně ukazuje ukazatel, se využije funkce *mysql_fetch_array()*. Každé volání této funkce přesouvá ukazatel řádku na nový řádek v sadě výsledků. Tato funkce vrací logickou hodnotu **false**, pokud už není žádný další řádek. Funkce *mysql_fetch_row* nám vrátí počet řádků v sadě výsledků.

```
if(mysql_fetch_row($result) == 0)
{
  echo 'V tabulce není žádný záznam.';
}
else
{
  while($record = mysql_fetch_array($result))
  {
    echo $record['autor'] . ' - ' . $record['nazev'] . "<br />\n";
  }
}
```