Slezská univerzita v Opavě
Obchodně podnikatelská fakulta v Karviné

PORTÁL A JEHO SPRÁVA A

Pro prezenční formu studia

Petr Suchánek
Jan Górecki

Karviná 2013

Projekt OP VK č. CZ.1.07/2.2.00/28.0017
„Inovace studijních programů na Slezské univerzitě,
Obchodně podnikatelské fakultě v Karviné“

Obor: Informatika.

Anotace: Webové portály lze v dnešní době považovat za klíčové rozhraní v prostředí internetu. Jejich využití je možné jak pro pouhou prezentaci obsahu, tak i například pro podnikatelské účely v podobě internetových obchodů, přístup k podnikovým informačním systémům, řízení dislokovaných pracovišť, komunikaci a podporu řízení obsahu v rámci intranetu, apod. Tato studijní opora je zaměřena na problematiku tvorby webových portálů s využitím technologií HTML, PHP, MySQL a lokálního serveru Apache. Primárním cílem je seznámit studenty s tvorbou dynamických webových stránek, metodami práce se skripty a propojením dynamických webových stránek s databázemi. Po studiu dané studijní opory budou studenti schopni vytvořit jednoduchá portálová řešení, která budou funkční jak s využitím lokálního serveru, tak i v rámci webhostingu.

Klíčová slova: Webový portál, skript, databáze, webové formuláře, vytváření dynamických webových stránek, předávání dat.

Autor: **Doc. Mgr. Petr Suchánek, Ph.D.**
Ing. Jan Górecki

Recenzenti: Doc. RNDr. Petr Šaloun, Ph.D.
Ing. Mgr. Roman Šperka, Ph.D.

ISBN Doplní oddělení vědy a výzkumu.

OBSAH

ÚVOD

Jedním z předmětů, který je na Slezské univerzitě v Opavě, obchodně podnikatelské fakultě vyučován v rámci oboru „Manažerská informatika“ studijního programu „Systémové inženýrství a informatika“ je předmět „Portál a jeho správa A“. Tento předmět si jako volně volitelný mohou vybrat i studenti jiných oborů a podle sledovaného zájmu lze konstatovat, že se jedná o oblíbený předmět pro všechny, které zajímá problematika tvorby webových stránek resp. dynamických webových stránek a ještě lépe webových portálů, které jsou dnes primárním komunikačním rozhraním v rámci internetu.

Současné nástroje a technologie se stále více přibližují uživatelům a i lidé s neinformatickým vzděláním jsou schopni vytvořit si kvalitní webové stránky využitelné pro osobní nebo i profesionální účely. A to je právě jeden z primárních cílů předmětu „Portál a jeho správa A“, v rámci kterého si na jedné straně studenti oboru Manažerská informatika prohloubí své dosavadní znalosti z tvorby dynamických webových stránek a portálů a studenti jiných, v našem případě oborů souvisejících s ekonomií, marketingem apod., proniknou do dané problematiky a získají minimálně základní znalosti a dovednosti, které budou moci dle vlastního zájmu a potřeb v budoucnu dále rozvíjet.

Z obsahového hlediska je předmět zaměřen spíše do praktické roviny. Z tohoto hlediska je cílem zvládnout tvorbu struktury webového portálu pomocí HTML (HyperText Markup Language (resp. XHTML (eXtensible HyperText Markup Language)) a kaskádových stylů CSS (Cascading Style Sheets), tvorbu webových formulářů, vkládání dynamických prvků v podobě PHP (Hypertext Preprocessor) skriptů, tvorbu databáze v MySQL a základní metody komunikace webových portálů s databází MySQL (předávání hodnot mezi databází a webovými formuláři).

1 WEBOVÝ & INTERNETOVÝ PORTÁL

Vyjdeme-li z nejobecnějšího pohledu, pak pojem „portál“ pochází z latinského porta (brána) a má celou významů. Lze ho použít například pro označení architektonickými nebo jinými výtvarnými prvky ozdobeného orámování vstupu do významné, obvykle veřejné budovy, např. do katedrály, pohledového orámování divadelního jeviště, v němž bývá zavěšena divadelní opona, vstupu do jeskyně nebo dalších objektů. Ve spojitosti s výukou předmětu Portál a jeho správa se budeme v tomto textu zabývat jiným významem resp. formou portálu, kterou je tzv. webový nebo internetový portál. V této publikaci bude z důvodu zachování jednotnosti a konzistence textu využíván pojem webový portál.

1.1 DEFINICE WEBOVÉHO PORTÁLU A SOUVISEJÍCÍCH POJMŮ

Webový portál můžeme definovat z různých pohledů orientovaných zejména na technologie, oblasti využití, cílové skupiny a funkce (Definice 1 – 3).

DEFINICE 1

Termínem portál v pojetí informačních technologií (Information Technology – IT) označujeme (webovou) aplikaci, která svému uživateli poskytuje jednotným způsobem a centralizovaně informace z různých zdrojů, které uživatele zajímají nebo se ho nějakým způsobem týkají.

Jiná definice říká že:

DEFINICE 2

Technicky je portál internetovou aplikací napojenou na různé datové zdroje a pracuje se s ním prostřednictvím internetového prohlížeče.

Z hlediska funkcí, využitelnosti a cílových skupin může být dále webový portál definován jako:

DEFINICE 3

Webový portál představuje bezpečný a jednotný bod interakce s různorodými informacemi, obchodními procesy a lidmi, přizpůsobený individuálním potřebám konkrétních uživatelů.

Abychom dále upřesnili definice 1 – 3, je nutné nadefinovat ještě pojem webová aplikace (Definice 4) a v návaznosti na tuto definici ještě pojem webový server (Definice 5).

DEFINICE 4

Webová aplikace v softwarovém inženýrství je aplikace poskytovaná uživatelům z webového serveru přes počítačovou síť Internet, nebo její vnitropodnikovou obdobu (intranet).

DEFINICE 5

5 a) Webový server je počítač, který je odpovědný za vyřizování požadavků HTTP od klientů - programů zvaných webový prohlížeč. Vyřízením požadavků se rozumí odeslání webové stránky.

5 b) Webový server je počítačový program, který provádí činnosti popsané v předchozím bodě (démon).

Webové portály jsou primárně založeny na komunikaci klient-server (Definice 6).

DEFINICE 6

Klient-server je síťová architektura, která odděluje klienta (často aplikaci s grafickým uživatelským rozhraním) a server, kteří spolu komunikují přes počítačovou síť.

Klient-server popisuje vztah mezi dvěma počítačovými programy, v nichž první program, klient, žádá o služby jiný program zvaný server. Příkladem je webový prohlížeč, tj. klientský program na uživatelském počítači, který může přistupovat k informacím na libovolném webovém serveru na světě (v rámci Internetu) (samozřejmě s přihlédnutím k autorizaci přístupu).

1.2 TYPY WEBOVÝCH PORTÁLŮ

Webové portály nabízejí širokou škálu služeb a informací s možností jejich přizpůsobení uživateli podle osobních potřeb a zájmů (personalizace). Z hlediska služeb a orientace na cílové skupiny rozdělujeme webové portály na tři typy, kterými jsou horizontální, vertikální a kombinované.

1.2.1 HORIZONTÁLNÍ WEBOVÉ PORTÁLY

Horizontální webové portály obsahují široké tematické zaměření a jsou orientované na co nejširší cílovou skupinu. Do této kategorie řadíme klasické webové vyhledávače, kterými jsou například:

- www.seznam.cz;
- www.google.com;
- www.centrum.cz;
- www.atlas.cz;
- www.volny.cz;
- www.yahoo.com;
- www.msn.com;
- www.aol.com;
- www.altavista.com;
- www.go.com;
- www.dogpile.com;
- a další.

1.2.2 VERTIKÁLNÍ WEBOVÉ PORTÁLY

Tento typ webových portálů je zaměřen na určitou skupinu uživatelů (např. tzv. community portals slouží uživatelům z určitého regionu) nebo jsou orientovány tematicky, opět s ohledem na zájmy určité skupiny uživatelů. Terminologicky jsou označovány i jako oborové portály (vortály). Jako příklady je možné uvést:

- www.diit.cz;
- www.statnisprava.cz;
- www.itnews.sk;
- management.blog.cz;
- www.microsoft.com;
- www.cd.cz;
- www.vodafone.cz;
- www.opf.slu.cz;
- www.idos.cz;
- a další.

Vertikální webové portály můžeme dále konkrétněji dělit na následující typy:

- Osobní portály nabízející možnost personalizace prezentace a výběru informací pro své uživatele (například sociální sítě a podobně);
- Regionální portály;
- Zpravodajské portály;
- Vládní portály;
- Obchodní a byznys portály;
- Podnikové portály (často sloužící i jako intranet).

1.2.3 KOMBINACE HORIZONTÁLNÍHO A VERTIKÁLNÍHO WEBOVÉHO PORTÁLU

Jako speciální případ kombinovaného webového portálu můžeme uvést internetové obchody, které jsou orientované na různé cílové skupiny (buď na širokou cílovou skupinu, nebo úzkou) a nabízejí buď specifické produkty, nebo velmi široký rozsah produktů.

Ve vztahu k vertikálním portálům je nutné uvést, že studijní text se nebude zaměřovat na problematiku podnikových portálů, ale pouze těch, které jsou vytvářené za účelem poskytování a sdílení informací v prostředí Internetu.

Jen pro upřesnění uvedeme, že podnikové portály jsou chápány jako webové stránky (obvykle dynamické webové stránky), které slouží k lepšímu publikování a sdílení informací ve firmě. V podstatě se termínů firemní, podnikový, či vnitrofiremní portál užívá pro označení aplikace s názvem intranet či extranet. Firemní portály jsou většinou vystavěny na redakčních systémech typu ECM (Enterprise Content Management), které dokáží agregovat data z více datových zdrojů.

1.3 TVORBA WEBOVÉHO PORTÁLU

Jak již vyplývá z předchozího textu, webové portály jsou ve svém pojetí „dynamické webové stránky“ představující uživatelské rozhraní poskytující uživateli nejen statický obsah, ale i celou řadu služeb, jako například mailové schránky, vyhledávání, zprávy do mailu, informace o počasí, diskusní fórum, chat, kalendář akcí, katalog zdrojů a další.

Chceme-li vytvářet portál sami nebo ho zadat v rámci externí dodávky, vždy je nutné provést podrobnou analýzu potřeb založenou na jasné vizi a strategii. Předpokladem úspěšnosti projektu tvorby webového portálu jsou:

- Identifikace potřeb potenciálních uživatelů portálu;
- Specifikace použitelnosti a uživatelské přívětivosti;
- Technologické možnosti;
- Bezpečnost;
- Stabilita;
- Redukce technologií potřebných pro fungování webového portálu;
- Maximalizace flexibility portálu potřeby průběžných úprav, vylepšení a doplnění;
- Časový harmonogram tvorby;
- Finanční rozpočet;
- Možnosti personalizace webového portálu;
- Testování použitelnosti a uživatelské přívětivosti.

Z hlediska vlastní tvorby se musí brát do úvahy a zajistit:

- Použitelnost webu - návštěvník musí snadno najít informaci, pro kterou na web přišel. Zejména kontakt, poptávka či objednání zboží musí být po ruce;
- Přehledná struktura webu - vychází z výše uvedené Použitelnosti webu. Jde zejména o logické členění stránek a intuitivní orientaci návštěvníka napříč webem;
- Přístupnost webu - zahrnuje (nejen) technické požadavky, např. výrazné a logické menu pro ovládání stránek, alternativní popisy obrázků, velikost písma, barva odkazů atd.
- Validní kód - vyhledávače mají dnes již velmi sofistikované indexovací algoritmy, které si poradí i s nepřívětivým kódem, ale není jediný důvod jim práci komplikovat.

Základem je orientace na uživatele webu. V rámci seminářů budou vytvářené různé projekty v podobě portálů, resp. dynamických webových stránek. Když je budete vytvářet nebo jejich tvorbu řídit nyní nebo kdykoliv později v praxi, vždy byste měli dodržovat některé důležité zásady, které se staly nebo stávají standardem a jejich aplikací do svých semestrálních projektů si jejich použití můžete nacvičit. Jedná se zejména o následující výčet základních zásad vytahujících se k výšeuvedeným čtyřem oblastem:

- **Obsah webových stránek je dostupný a čitelný:**
  - Každý netextový prvek nesoucí významové sdělení má svou textovou alternativu;
  - Informace sdělované prostřednictvím skriptů, objektů, appletů, kaskádových stylů, obrázků a jiných doplňků na straně uživatele jsou dostupné i bez kteréhokoli z těchto doplňků;
  - Informace sdělované barvou jsou dostupné i bez barevného rozlišení;
  - Barvy popředí a pozadí jsou dostatečně kontrastní. Na pozadí není vzorek, který snižuje čitelnost;
  - Předpisy určující velikost písma nepoužívají absolutní jednotky;
  - Předpisy určující typ písma obsahují obecnou rodinu písem.
- **Práci s webovou stránkou řídí uživatel:**
  - Obsah WWW stránky se mění, jen když uživatel aktivuje nějaký prvek;
  - Webová stránka bez přímého příkazu uživatele nemanipuluje uživatelským prostředím;
  - Nová okna se otevírají jen v odůvodněných případech a uživatel je na to předem upozorněn;
  - Na webové stránce nic nebliká rychleji než jednou za sekundu;
  - Webová stránka nebrání uživateli posouvat obsahem rámů;
  - Obsah ani kód webové stránky nepředpokládá ani nevyžaduje konkrétní způsob použití ani konkrétní výstupní či ovládací zařízení.
- **Informace jsou srozumitelné a přehledné:**
  - Webové stránky sdělují informace jednoduchým jazykem a srozumitelnou formou;
  - Úvodní webová stránka jasně popisuje smysl a účel webu. Název webu či jeho provozovatele je zřetelný;
  - Webová stránka i jednotlivé prvky textového obsahu uvádějí své hlavní sdělení na svém začátku;
  - Rozsáhlé obsahové bloky jsou rozděleny do menších, výstižně nadepsaných celků;
  - Informace zveřejňované na základě zákona jsou dostupné jako textový obsah webové stránky;
  - Na samostatné webové stránce je uveden kontakt na technického správce a prohlášení jasně vymezující míru přístupnosti webu a jeho částí. Na tuto webovou stránku odkazuje každá stránka webu.

- **Ovládání webu je jasné a pochopitelné:**
  - Každá webová stránka má smysluplný název, vystihující její obsah;
  - Navigační a obsahové informace jsou na webové stránce zřetelně odděleny;
  - Navigace je srozumitelná a je konzistentní na všech webových stránkách;
  - Každá webová stránka (kromě úvodní webové stránky) obsahuje odkaz na vyšší úroveň v hierarchii webu a odkaz na úvodní WWW stránku;
  - Všechny webové stránky rozsáhlejšího webu obsahují odkaz na přehlednou mapu webu.
  - Obsah ani kód webové stránky nepředpokládá, že uživatel již navštívil jinou stránku;
  - Každý formulářový prvek má přiřazen výstižný nadpis;
  - Každý rám má vhodné jméno či popis vyjadřující jeho smysl a funkčnost.
- **Odkazy jsou zřetelné a návodné:**
  - Označení každého odkazu výstižně popisuje jeho cíl i bez okolního kontextu;
  - Stejně označené odkazy mají stejný cíl;
  - Odkazy jsou odlišeny od ostatního textu, a to nikoli pouze barvou;
  - Obrázková mapa na straně serveru je použita jen v případě, že nebylo možné pomocí dostupného geometrického tvaru definovat oblasti v obrázkové mapě. V ostatních případech je použita obrázková mapa na straně uživatele. Obrázková mapa na straně serveru je vždy doprovázena alternativními textovými odkazy;
  - Uživatel je předem jasně upozorněn, když odkaz vede na obsah jiného typu, než je webová stránka. Takový odkaz je doplněn sdělením o typu a velikosti cílového souboru.
- **Kód je technicky způsobilý a strukturovaný:**
  - Kód webových stránek odpovídá nějaké zveřejněné finální specifikaci jazyka HTML či XHTML. Neobsahuje syntaktické chyby, které je správce webových stránek schopen odstranit;
  - V metaznačkách je uvedena použitá znaková sada dokumentu;
  - Prvky tvořící nadpisy a seznamy jsou korektně vyznačeny ve zdrojovém kódu. Prvky, které netvoří nadpisy či seznamy, naopak ve zdrojovém kódu takto vyznačeny nejsou;
  - Pro popis vzhledu webové stránky jsou upřednostněny stylové předpisy;
  - Je-li tabulka použita pro rozvržení obsahu webové stránky, neobsahuje záhlaví řádků ani sloupců. Všechny tabulky zobrazující tabulková data naopak záhlaví řádků a/nebo sloupců obsahují;
  - Všechny tabulky dávají smysl čtené po řádcích zleva doprava.

1.4 NÁSTROJE A TECHNOLOGIE PRO TVORBU WEBOVÉHO PORTÁLU

Pro tvorbu webového portálu je možné využít celé řady technologií a nástrojů, pomocí kterých je možné vytvářet webové stránky s dynamickým obsahem. Z hlediska jazyků a skriptovacích jazyků lze pro tvorbu www stránek využít HTML (HyperText Markup Language), XHTML (Extensible HyperText Markup Language), CSS (Cascading Style Sheets), Java Skript, AJAX (Asynchronous JavaScript and XML (Extensible Markup Language)), PHP (Hypertext Preprocessor), JSP (JavaServer Pages), Java, Ruby, Python, Perl, ASP.NET apod. Jedná se dnes o běžné jazyky, které se většinou využívají v nejnovějších verzích, ovšem v některých případech ani pozdější přechod k novější verzi většinou nečiní žádné významnější problémy.

V rámci této studijní opory budeme pro tvorbu webového portálu využívat:

- HTML nebo XHTML (v textu se využívá HTML s tím, že téměř všechny kódy odpovídají i standardu XHTML);
- Lokální server Apache;
- PHP;
- MySQL;

Z hlediska softwarové podpory budou všechny ukázky a příklady realizovány pomocí PSPad a EasyPHP. Aby bylo možné si hned od začátku výkladu doplněného o příklady tyto příklady zkoušet, bude jako první popsán v předchozí větě uvedený software PSPad, ve kterém je možné průběžně, bez jakýchkoliv doplňků a dalších software, zkoušet kódy prezentované v kapitolách HTML a CSS.

1.5 PSPAD

PSPad je celosvětově rozšířený freewarový textový editor a editor zdrojových kódů pro platformu Microsoft Windows vyvíjený v prostředí Delphi. Je navržen jako univerzální editor pro editaci prostých textů a zdrojových kódů mnoha programovacích, skriptovacích a značkovacích jazyků (podporuje přes 30 různých prostředí - PHP, HTML, XML, Java, JavaScript, Perl, C, C++, ASP, SQL, Python, aj.) a další. Základními funkcemi programu jsou:

- práce s projekty;
- práce ve více dokumentech současně s možností uložení rozdělané práce;
- záznam makra s možností uložení a načtení z disku;
- hledání s nahrazením v souborech;
- porovnávání textu s barevným zvýrazněním rozdílů;
- šablony (HTML tagy, skripty, kusy kódu...) včetně uživatelsky definovatelných klávesových zkratek;
- instalace obsahuje šablony pro HTML, PHP, Pascal, JScript, VBScript, MySQL, MS-Dos, Perl, ...;
- uživatelská definice zvýrazňovačů pro exotická prostředí;
- konverze češtiny (LatinII, Kamenických, Windows 1250, ISO 8859-2, UTF-8, UNICODE);
- zvýraznění syntaxe s automatickým nastavením dle typu souboru;
- automatické opravy;
- inteligentní interní HTML náhled pomocí IE i Mozilly;
- plnohodnotný HEXA editor od verze 3.2.5;
- přímé odeslání souboru na disketu, jako obsahu e-mailu nebo příloha e-mailu;
- definice externích programů, ve kterých je možné soubor otevřít;
- tisk zvýrazněné syntaxe s možností náhledu před tiskem;
- export včetně zvýraznění do RTF, HTML, XHTML, TeX souboru nebo do schránky;
- sloupcové a řádkové bloky, záložky v textu, zobrazení čísel řádků;
- přeformátování a komprese HTML kódu, změna velikosti tagů, odstranění tagů;
- integrovaná knihovna TiDy pro formátování a kontrolu HTML, převod do CSS, XML, XHTML;
- integrovaný free editor TopStyle Lite (odkaz ke stažení) pro editaci CSS;
- setřídění řádků s možností třídění dle definovaného sloupce;
- změny velikosti písmen, odstranění diakritiky;
- zobrazení ASCII tabulky s možností tisku;
- práce se dvěma soubory současně v rozděleném okně s možností synchronizace posuvu;
- uživatelská lokalizace do jiných jazyků (doufám, že se podělíte s vlastními lokalizacemi);
- jednoduchá integrace do systému Windows a oblíbených programů.

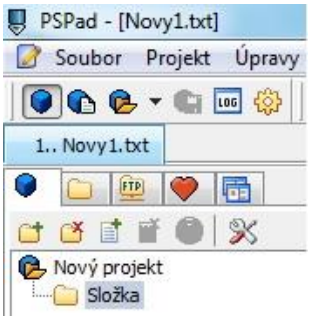
1.5.1 INSTALACE PROGRAMU PSPAD

Software PSPad si lze stáhnout z celé řady odkazů. Základním odkazem je http://www.pspad.com/cz/. Instalace proběhne automaticky po spuštění instalačního souboru.

1.5.2 ZÁKLADNÍ PRÁCE S PROGRAMEM

Po instalaci a spuštění programu se nám zobrazí pracovní plocha programu. Defaultně je při prvním spuštění otevřený soubor označený jako **Novy1.txt**, jehož záložka je zobrazena v levém horním rohu. (Obrázek 1)

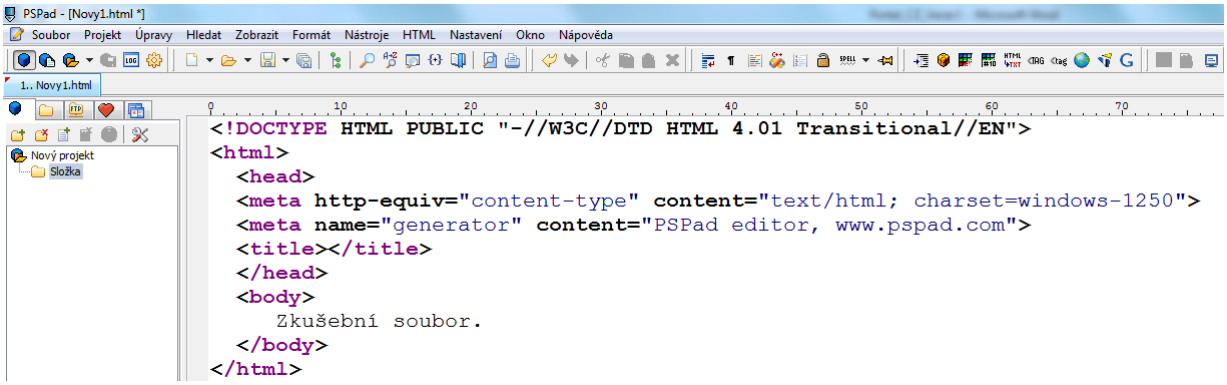
Obrázek 1: Záložka souboru v programu PSPad.



Tento soubor zavřeme způsobem, že klepneme pravým tlačítkem myši na záložku souboru a zadáme příkaz zavřít. Je zřejmé, že se jedná o standardní způsob využitelný v libovolném okamžiku při práci s programem PSPad.

Jak bylo uvedeno v základním popisu programu, tento umožňuje práci s celou řadou různých prostředí (resp. jazyků), přičemž po prvním otevření souboru daného typu automaticky vkládá na pracovní plochu základní strukturu podle výchozí syntaxe. Jako příklad si ukážeme otevření dokumentu html. Zadáme příkaz **Soubor/Nový** a z nabídky vybereme **HTML**. Otevře se nám soubor s názvem **Novy1.html** obsahující základní strukturu html dokumentu (konkrétně bude popsána v následující kapitole). (Obrázek 2)

Obrázek 2: Otevření souboru html v programu PSPad.



Mezi tagy `<body>` a `</body>` vepište podle Obrázku 2 text "**Zkušební soubor.**". Takto vytvořený html dokument nepotřebuje žádnou speciální podporu a proto po kliknutí na funkční tlačítko **F10** se obsah html dokumentu zobrazí v prohlížeči. Obecně tedy je funkční klávesa **F10** je využívána pro zobrazování souborů, které nepotřebují žádnou jinou podporu.

Důležitým krokem je samozřejmě uložení souboru, které se provádí standardním příkazem **Soubor/Uložit jako…**.

2 HTML

2.1 CHARAKTERISTIKA HTML

HTML (HyperText Markup Language) je značkovací jazyk pro hypertext, který umožňuje tvorbu www stránek. Jazyk je aplikací dříve vyvinutého rozsáhlého univerzálního značkovacího jazyka SGML (Standard Generalized Markup Language). Vývoj HTML byl ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka.

2.2 CHARAKTERISTIKA XHTML

XHTML (eXtensible Hypertext Markup Language) je značkovací jazyk pro tvorbu hypertextových dokumentů v prostředí WWW vyvinutý W3C. Původně se předpokládalo, že se stane nástupcem jazyka HTML, jehož vývoj byl verzí 4.01 ukončen. V roce 2007 však došlo k založení pracovní skupiny, která má za cíl vytvořit novou verzi HTML, která ponese označení HTML 5 a její XML variantu XHTML 5.

2.3 ZÁKLADNÍ ROZDÍLY MEZI HTMNL A XHTML

Některé věci platily už v HTML, ovšem XHTML je striktně vyžaduje. Jedná se zejména o:

- Všechny atributy mají hodnoty v uvozovkách;
- Zákaz křížení tagů.

Již konkrétními rozdíly mezi HTML a XHTML jsou:

- Tagy a atributy jsou malými písmeny;
- Nepárové tagy končí lomítkem;
- Párové tagy jsou párové povinně;
- Všechny atributy musejí mít hodnotu;
- Interní javascript a styly se zapisují jiným způsobem;
- Dokument má mít XML prolog;
- Dokument požaduje správný doctype.

Než začneme s následujícím výkladem orientovaným na výklad základních postupů při práci s jazyky a prostředími, pomocí kterých budeme portály vytvářet, je vhodné doporučit, aby všechny příklady a ukázky, které budou níže prezentované, jste si vyzkoušeli prakticky v programech PSPad a EasyPHP.

2.4 TAGY

Jak již bylo uvedeno výše, HTML je značkovací jazyk pro tvorbu www stránek. Jako například Český jazyk má svá slova, tak i HTML obsahuje slova, neboli tagy (značky), které dávají vlastnímu obsahu stránky význam a také dokument formátují. Tagy máme párové a nepárové.

2.4.1 PÁROVÝ TAG

Párový tag – má začátek a konec:

ŘEŠENÝ PŘÍKLAD 1

```
<p>Já jsem krátký odstavec.</p>
```

V příkladu 1 představuje:
** ** - začátek – tag uzavřený do lomených závorek;
** ** - konec – tag uzavřený do hranatých závorek, před tagem je lomítko.

U párových tagů se text (obsah) stránky se vkládá mezi začátek a konec. Konkrétně tag p značí odstavec (z anglického paragraph).

2.4.2 NEPÁROVÝ TAG

Nepárové tagy nemají ukončovací značku. Tyto tagy neobalují text, ale mají svůj vlastní význam (Řešený příklad 2).

ŘEŠENÝ PŘÍKLAD 2

```
<hr> - horizontální čára na naší stránce
```

Text můžeme vložit i do více tagů. Tagy se nesmí kžížit! (Řešený příklad 3)

ŘEŠENÝ PŘÍKLAD 3

```
<p><strong>Zvýrazněný odstavec</strong></p> - správný zápis
<p><strong>Zvýrazněný odstavec</p></strong> - špatný zápis
```

2.4.3 STRUKTURA HTML DOKUMENTU

Každý html dokument má svou strukturu. Celý html dokument se skládá ze dvou částí - hlavičky a těla. Hlavička má značku <head>, tělo <body>. HTML dokument má značku <html>. Struktura html dokumentu je tedy následující (Řešený příklad 4):

ŘEŠENÝ PŘÍKLAD 4

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
   <html>
   <head>
   <title>Tady bude titulek stránky</title>
   </head>
        <body>

        </body>
   </html>
```

První řádek nám říká, jaká verze HTML byla použita. Tento údaj není součástí html dokumentu. V hlavičce se umisťují prvky, které mají být interpretovány prohlížečem dříve než tělo - kaskádové styly (tělo se již musí zobrazovat s těmito styly), skripty (obzvláště funkce, které se v těle již volají) a mnohé další. Jedním takovým prvkem je <title>, který

obsahuje titulek v okně prohlížeče. Tento titulek má mimořádnou důležitost, neboť vlastně reprezentuje název stránky - například v historii, v oblíbených položkách a podobně, což je určitě uživatelsky příjemnější než celá adresa včetně dotazovacího řetězce.

V hlavičce se také často vyskytují metainformace (častěji využíván obecnější výraz metadata) - jakési skryté informace pro spolupráci mezi serverem a klientskou stanicí (prohlížečem). Následující příklad dává prohlížeči úkol obnovovat stránku každé dvě sekundy, nastavuje znakovou sadu na windows-1250 a zobrazuje počet sekund v dané minutě (z reálného systémového času) (Řešený příklad 5):

ŘEŠENÝ PŘÍKLAD 5

```
<html>
   <head>
      <title>Meta obnova</title>
      <meta http-equiv="refresh" content=2>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1250">
   </head>
   <body>
      <!-- Obsah těla -->
      <script language="javascript">
         var datum = new Date();
         document.write("Počet sekund v aktuální minutě: "
            + datum.getSeconds());
      </script>
   </body>
</html>
```

Tzv. metainformace jsou důležitou součástí html dokumentů. Pokud použijeme tyto informace v hlavičce stránky, budou je moci vyhledat META vyhledávače. Vedle informací, které byly v rámci metainformací prezentovány v Příkladu 5, lze tímto způsobem (a je to například z hlediska SEO – Search Engine Optimization i doporučeno) uvádět jako součást html dokumentu celou řadu dalších informací, kterými mohou být například (Řešený příklad 6):

ŘEŠENÝ PŘÍKLAD 6

```
<meta name="description" content="popis stránek">
<meta name="keywords" content="klíčová slova">
<meta name="author" content="jméno autora">
<meta http-equiv="refresh" content=" 4 ; url=http://www.jina-
stranka.cz">
<meta http-equiv="content-type" content="text/html;
charset=windows-1250">
<meta http-equiv="expires" content="10 aug 2003 12:00:00">
```

Popis stránek, klíčová slova nebo jméno autora z příkladu 6 jsou zřejmé. Zajímavá metainformace je přesměrování na jinou stránku (4. řádek z Příkladu 6). Pokud vyplníte adresu URL, přesměruje se dokument v tomto případě po 4 sekundách. Pokud se adresa nevyplní, bude se aktuální stránka každé 4 sekundy obnovovat. 5. řádek z Řešeného příkladu

6 charakterizuje znakovou sadu dokumentu. Pro správné zobrazení české diakritiky se používají především tato dvě kódování:

- **windows-1250 nebo utf-8** - preferováno na platformě Windows;
- **ISO 8859-2** - podporované i Unixem a Linuxem, ve Win známé jako Středoevropské jazyky (ISO).

Na posledním řádku v Příkladu 6 jsou prezentovány charakteristiky platnosti html dokumentu. Jedná se o datum a čas kdy informace na stránce přestávají platit. Prohlížeč si stáhne aktuální stránku ze serveru - nikoli uloženou z paměti.

2.5 ODSTAVCE A ZALOMENÍ TEXTU

K oddělení odstavců se používá tag <p> (Řešený příklad 7).

ŘEŠENÝ PŘÍKLAD 7

```
<p>První odstavec <p>Druhý odstavec <p>třetí odstavec </p>
nebo
<p>První odstavec</p>
<p>Druhý odstavec</p>
<p>Třetí odstavec</p>
```

Před zobrazením odstavce, prohlížeč převádí všechny konce řádků na mezery. Pokud je za sebou více mezer, nahradí je jednou mezerou - mezi slovy můžeme psát mezer kolik chceme, ovšem nakonec se zobrazí mezera jediná.

Další věcí je, že se text zarovnává podle aktuální velikosti okna, takže dokument lze zobrazit v různých rozlišeních. Někdy však potřebujeme, aby se text zalomil na určitém místě. To dosáhneme pomocí tagu
 (Řešený příklad 8):

ŘEŠENÝ PŘÍKLAD 8

```
Nějaký text<br>
a další text<br>
a ještě jeden text<br>
```

2.6 NADPISY

K vytváření nadpisů se používá element <hx> (x=1,2,3,4,5,6). Největším a nejvýraznějším písmem je zobrazován text s elementem <h1>. Ostatní elementy h(2-6) se postupně zmenšují (Řešený příklad 9).

ŘEŠENÝ PŘÍKLAD 9

```
<h1>nadpis1</h1>
<h2>podnadpis2</h2>
```

2.7 TYPY PÍSMA

V html dokumentech můžeme používat písma tučná, různě velká, kurzívu, použít jiný font, zvýraznit text nebo dát textu např. jinou barvu. Prostě můžeme s textem dělat to, co umožňují textové editory. Řešený příklad 10 prezentuje zvýraznění textu pomocí tagu `<strong> ... </strong>` a pomocí tagu `<font>...</font>` je nastaven font Arial, barva textu černá a velikost písma 6.

ŘEŠENÝ PŘÍKLAD 10

```
< strong>nadpis</strong>
< font color="black" face="arial" size=6>text</font>
```

2.8 ODKAZY

Odkaz je zvýrazněná část textu, ve které se skrývá adresa dalších stránek (URL). Po kliknutí na tento odkaz prohlížeč otevře okno s touto adresou. Odkaz vložíme do dokumentu následujícím způsobem (Řešený příklad 11):

ŘEŠENÝ PŘÍKLAD 11

```
<a href="http://www.odkaz.cz">text odkazu</a>
```

Kromě textu můžeme odkazovat i obrázkem. Toho jednoduše docílíme vložením obrázku mezi tagy <a>... (Řešený příklad 12).

ŘEŠENÝ PŘÍKLAD 12

```
<a href="http://www.odkaz.cz"><img src="obrazek.gif
border="0">text odkazu</a>
```

Atribut border="0" u obrázku `<img>` použijeme tehdy, pokud nechceme, aby byl okolo odkazujícího obrázku rámeček.

`<a href="url">` slouží k určení adresy, kam odkaz po aktivaci vede. Odkaz nemusí být jen text, ale i obrázek. Někdy potřebujeme, aby odkaz odkazoval do nějaké části dokumentu (třeba doprostřed). K tomu slouží tag `<a name="neco">`. Stačí jen tento tag umístit doprostřed dokumentu a k odkazu připsat (Řešený příklad 13):

ŘEŠENÝ PŘÍKLAD 13

```
<a href="mojestranky.html#001">druhá část</a>
...
...
< a name="001">
druhá část
```

Pokud potřebujeme vytvořit odkaz, který by zajišťoval stažení nějakého souboru, do URL uvedeme přesnou adresu, kde se soubor nachází (Řešený příklad 14).

ŘEŠENÝ PŘÍKLAD 14

```
<a href="http://www.odkaz.cz/soubor.zip">soubor.zip</a>
```

2.9 SEZNAMY

V html dokumentech můžeme používat různé typy seznamů. Členíme je do třech základních typů - číslované, nečíslované a definiční. U číslovaných seznamů je před každou položku automaticky umisťováno číslo. Jednotlivé položky se uvozují tagem <li>. Můžeme definovat styl odrážky `<ol type=a/a/i/i/1>...</ol>`. (*A pro velká písmena, a pro malá písmena, I pro římské číslování, 1 pro arabské číslování*) Ukázka číslovaného seznamu je prezentována v Řešeném příkladu 15):

ŘEŠENÝ PŘÍKLAD 15

```
<ol>
< li>položka č. 1
< li>položka č. 2
< /ol>
```

Stejně se vytváří nečíslovaný seznam. Akorát místo ol napíšeme ul. Stejně jako u číslovaných seznamů, můžeme měnit styl odrážky `<ul type=disc/circle/square>...</ul>`.

Definiční seznamy jsou odlišné od předchozích dvou. Můžeme s ním vytvořit např. slovníček pojmů, který obsahuje termíny a jejich vysvětlení. Uvozuje se tagem `<dl>` a jednotlivé položky pak `<dt>` a `<dd>` (Řešený příklad 16).

ŘEŠENÝ PŘÍKLAD 16

```
<dl>
< dt>název 1
< dd>text
< dt>název 2
< dd>text
< /dl>
```

2.10 OBRÁZKY

Ke vložení obrázku na stránku se používá tag `<img>`, ve kterém specifikujeme cestu v atributu **src**, kde máme obrázek uložen. Dále existuje atribut **alt**, který slouží k jednoduchému popisu obrázku (např. popis fotky). Text se v internetových prohlížečích zobrazí po najetí kurzoru na obrázek. Tento atribut má ale ještě další výhodu a to pro majitele negrafických internetových prohlížečů (*SecureCRT, Lynx*). V těchto prohlížečích se zobrazí místo obrázku jen text v atributu **alt** (Řešený příklad 17).

ŘEŠENÝ PŘÍKLAD 17

```
<img src="obrazek.gif" align="top, middle, bottom, left, right" alt="text" width="x" height="y">
```

Align určuje způsob zarovnání obrázku a okolního textu. Může nabývat pěti hodnot:

- **Top** - zarovnává se s řádkou horní okraj obrázkup;
- **Middle** - obrázek je vzhledem k řádce vertikálně vycentrován;
- **Bottom** - zarovnává se s řádkou spodní okraj obrázku;
- **Left/right** - obrázek je umístěn u levého nebo pravého okraje stránky a obtéká ho text;

Width a **height** určují velikost obrázku v pixelech. Říkají prohlížeči, jak je obrázek velký, takže nedochází k deformaci a reformátování layoutu při načítání. Než se obrázek na stránky vloží, je vhodné si zjistit v libovolném grafickém editoru jeho přesné rozměry. Tyto atributy můžeme také používat ke zvětšování a zmenšování obrázků. Pokud zadáme větší rozměry obrázku, obrázek se sice zvětší, ale jeho kvalita je vůči jeho původní velikosti horší.

2.11 TABULKY

Tabulky je možné je používat jak při prezentaci informací, tak pro přesnou kontrolu nad layoutem stránky. Tabulky se vytvářejí pomocí tagu `<table>`. Jednotlivé řádky se vkládají pomocí tagu `<tr>`. Řádky pak rozdělujeme do sloupců (buněk) pomocí `<td>` (Řešený příklad 18).

ŘEŠENÝ PŘÍKLAD 18

```
<table>
 <tr>
  <td>měsíc</td>
  <td>rok</td>
 </tr>
 <tr>
  <td>leden</td>
  <td>2001</td>
 </tr>
</table>
```

Stejně jako u obrázku můžeme využít atribut **align**, pomocí kterého zarovnáváme texty v buňkách tabulky (Řešený příklad 19).

ŘEŠENÝ PŘÍKLAD 19

```
<table>
 <tr>
  <td align=center>měsíc</td>
  <td align=left>rok</td>
 </tr>
</table>
```

Zcela shodně se používá atribut **valign**, který slouží k vertikálnímu zarovnání. **Valign** může nabývat tří hodnot:

- **Top** (obsah buňky se zarovná s horním okrajem buňky);
- **Bottom** (obsah buňky se zarovná se spodním okrajem buňky);
- **Middle** (obsah buňky se zarovná doprostřed);
- **Valign** i **align** můžeme použít jak u `<td>`, tak u `<tr>`.

Velikost tabulky můžeme určit ručně (natvrdo) a to pomocí atributu **width** a **height**. **Width** je šířka tabulky a **height** je výška tabulky. Zadává se v pixelech. Pokud se ale obsah do tabulky nevejde, prostě se zvětší. Podobně můžeme zadávat velikost buněk <td>. Pokud ovšem zadáme velikost tabulky, součet velikostí buněk se musí rovnat velikosti tabulky (Řešený příklad 20).

ŘEŠENÝ PŘÍKLAD 20

```
<table width=500 height=200>
  <tr>
   <td width=250 height=100>rok</td>
  </tr>
  <tr>
   <td width=250 height=100>2001</td>
  </tr>
 </table>
```

Mezi atributy tagu `<table>` patří **cellpadding**, pomocí kterého určíme vzdálenost obsahu buňky od okrajů buňky. Atribut **cellspacing** nastavuje mezeru mezi buňkami. Dále si můžeme pomocí atributu **border** nastavit velikost rámečku okolo tabulky. Každé tabulce můžeme uvést její nadpis pomocí elementu **caption** (Řešený příklad 21).

ŘEŠENÝ PŘÍKLAD 21

```
<table cellpadding=5 cellspacing=5 border=1>
  <caption>název</caption>
  <tr>
   <td>měsíc</td>
   <td>rok</td>
  </tr>
 </table>
```

Důležitou a často využívanou vlastností u tabulek je tzv. slučování buněk. Atributy, pomocí kterých nastavujeme informaci o tom, kolik "buněk prostoru" zabírá aktuální buňka, jsou **rowspan** a **colspan** (Řešený příklad 22).

ŘEŠENÝ PŘÍKLAD 22

```
<table border="1" cellpadding="10">
<caption>Hlavička</caption>
<tr><td>A</td> <td>B</td> <td rowspan="2">C</td> </tr>
<tr> <td colspan="2">D</td> </tr>
<tr> <td colspan="3">E</td> </tr>
</table>
```

`Rowspan="2"` znamená, že buňka C zasahuje přes dva řádky, `colspan="3"` říká, že buňka zabírá prostor přes tři sloupce.

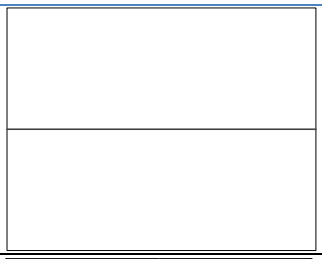
2.12 RÁMY

Rámy se využívají pro definici struktury webové stránky. V současné době se již příliš nevyužívají (a ani se využívat nedoporučují), nicméně pro výuku předmětu Portál a jeho správa jsou využitelné pro ilustraci některých přístupů k vytváření tzv. layoutu web stránek.

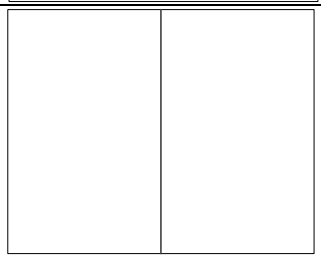
Chceme-li využívat rámy, pak Stránka musí obsahovat definici rozložení rámů `<frameset>`. V tomto párovém tagu používáme atributy **rows** a **cols**, které určují na jaký počet řádků a sloupců se obrazovka rozdělí. Velikost se zadává v pixelech nebo v procentech z celkové výšky (šířky) (Řešený příklad 23).

ŘEŠENÝ PŘÍKLAD 23

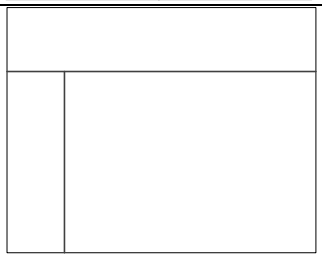
```
<frameset rows="50%,50%">
   obsah rámů
</frameset>
```



```
<frameset cols="50%,50%">
   obsah rámů
</frameset>
```



```
<frameset rows="20%,80%">
  obsah 1. rámu
 <frameset cols="30%,70%">
   obsah 2. a 3. rámu
 </frameset>
</frameset>
```



Jestliže máme hotové rozložení rámů pomocí `<frameset>`, musíme pomocí elementu `<frame>` určit, které html dokumenty se mají v rámech zobrazovat, a to pomocí atributu **src** (Řešený příklad 24).

ŘEŠENÝ PŘÍKLAD 24

```
<html>
 <head>
 </head>
 <frameset rows="50%,50%">
  <frame src="prvni.html">
  <frame src="druha.html">
 </frameset>
 </html>
```

U tagu `<frame>` lze použít spoustu atributů. Atribut **noresize**, který se postará o to, aby nešla velikost rámu měnit. Atribut **scrolling=no/yes/auto** - pokud nastavíme na hodnotu **no**, nepůjde se po obsahu rámu pohybovat. Rámy jsou od sebe odděleny rámečkem a jeho velikost (nebo úplné zrušení) měníme atributem **frameborder=0**. Vzdálenost obsahu rámu od pravého a levého okraje se provádí atributem **marginwidth=x** (za x počet pixelů). Atributem **marginheight=x** zase vzdálenost obsahu od horního a dolního okraje. Velice důležitým atributem je **name**, pomocí něhož zadáváme jméno rámu. To se hodí v případech, kdy chceme, aby se stránky schované za odkazem v jednom rámu zobrazovaly v jiném rámu.

U odkazů `<a>` můžeme použít atribut **target**, který určuje jméno rámu, ve kterém se má dokument zobrazit. Pokud toho chceme dosáhnout, musíme mít definovány jména rámů

v tagu `<frame>` Jako příklad si můžeme uvést situaci, kdy máme v souboru *prvni.html* odkazy a chceme, aby se stránky otevíraly v druhém rámu (Řešený příklad 25).

ŘEŠENÝ PŘÍKLAD 25

```
<html>
 <head>
 </head>
 <frameset rows="50%,50%">
  <frame src="prvni.html">
  <frame src="druha.html" name="hlavni">
 </frameset>
 </html>

Odkazy v souboru prvni.html budou vypadat následovně:
< a href="neco.html" target="hlavni">odkaz</a>
```

2.13 FORMULÁŘE

Formuláře lze nalézt téměř na každém webu. Slouží například pro vyhledávání, pro vkládání uživatelských komentářů, pro komunikaci mezi tvůrci webu a uživateli, pro přihlášení uživatele atd.

Při tvoření formulářů se používá pouze několik tagů. Všechny vstupní pole sjednocuje v jeden formulář párový tag `<form>`. Samotná vstupní pole se tvoří pomocí tagů <input>, `<textarea>` a `<select>`.

V tuto chvíli je nutné podotknout, že k vytvoření funkčního formuláře, který bude umožňovat uživatelům například vložit komentář ke článku, samotné HTML nestačí. To se stará pouze o vizuální stránku formuláře. K chodu formuláře je třeba ještě skript (například PHP nebo JavaScript). Problematika práce s PHP skripty je jednou z hlavních obsahových částí této studijní opory a v dalších částech jí bude věnována velká pozornost.

2.13.1 TAG FORM

HTML tag `<form>` je párový tag, kterým se vyznačuje formulář. Uvádí se vždy na začátku a na konci formuláře (Řešený příklad 26).

ŘEŠENÝ PŘÍKLAD 26

```
<form>
 <input type="text">
 <input type="submit">
</form>
```

Odkazuje se jím také na skript, který má zpracovat data z formuláře a metodu jak mají být data odeslána. Například (Řešený příklad 27):

ŘEŠENÝ PŘÍKLAD 27

```
<form action="nejaky_skript" method="get">
```

2.13.2 TAG INPUT

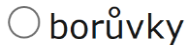
Nepárový HTML tag `<input>` je nejčastěji požívaným tagem u formulářů. Pomocí atributu **type** a jeho příslušných hodnot jej lze použít například pro textové pole, přepínač, zaškrtávací políčko atd. Nejjednodušším příklad může být (Řešený příklad 28):

ŘEŠENÝ PŘÍKLAD 28

```
<input type="text">
```

Možné hodnoty atributu **type** jsou prezentované v Tabulce 1.

Tabulka 1: Možné hodnoty atributu type.

Hodnota	Popis	Ukázka
button	Tlačítko bez jakékoliv předdefinované funkce. `<input type="button" value="tlačítko">`	
checkbox	Zaškrtávací tlačítko. `<input type="checkbox" value="neco">fotbal`	
file	Prvek pro upload souboru na web `<input type="file">`	
hidden	Skryté pole. `<input type="hidden">`	
image	Obrázkové odesílací tlačítko. `<input type="image" src="odesilaci-tlacitko.jpg">`	
password	Textové pole pro heslo. `<input type="password" size="15" value="heslo">`	
radio	Přepínač. `<input type="radio" value="ovoce">borůvky`	
reset	Tlačítko, které resetuje formulář. `<input type="reset">`	
submit	Tlačítko pro odeslání formuláře. `<input type="submit">`	
text	Textové pole. `<input type="text">`	

2.13.3 TAG TEXTAREA

Tag `<textarea>` se používá pro textové pole, které je určeno pro větší množství textu. Pomocí atributu **cols** lze definovat počet znaků na řádek a atributem **rows** počet viditelných řad (Řešený příklad 29).

ŘEŠENÝ PŘÍKLAD 29

```
<textarea cols="30" rows="3"></textarea>
```

2.13.4 TAG SELECT

Tag `<select>` se používá pro vytvoření rolovací nabídky. Na výběr je jedna z několika možností - ty jsou tvořeny pomocí tagu `<option>` (Řešený příklad 30). Ovšem pomocí atributu **multiple** lze vybírat i více možností najednou.

ŘEŠENÝ PŘÍKLAD 30

```
<select>
 <option value="fotbal">Fotbal</option>
 <option value="hokej">Hokej</option>
 <option value="tennis">Tenis</option>
 <option value="golf">Golf</option>
 </select>
```

2.13.5 TAG FIELDSET

Pomocí tagu `<fieldset>` vytváříme tzv. skupiny voleb. Jednoduchá ukázka je prezentována v rámci Řešeného příkladu 31.

ŘEŠENÝ PŘÍKLAD 31

```
<fieldset>
  <legend>Pohlaví:</legend>
  <input type="radio" name="pohlavi" value="muž"> muž <br>
  <input type="radio" name="pohlavi" value="žena"> žena
</fieldset>
```

Využijeme-li některé z výše charakterizovaných formulářových prvků, můžeme si jejich konkrétní využité prezentovat na konkrétním příkladu prezentujícím možnost realizace ankety (Případová studie 1).

PŘÍPADOVÁ STUDIE 1

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
  <head>
  <meta http-equiv="content-type" content="text/html;
charset=windows-1250">
  <meta name="generator" content="PSPad editor,
www.pspad.com">
  <title></title>
  </head>
  <body>
      <h1>Demonstrace formulářů</h1>
 <h2>Anketa</h2>
 <form action="skript.php" method="post">
 Jaký používáte typ procesor má Váš PC?<BR>
 <input type="radio" name="procesor">AMD<BR>
 <input type="radio" name="procesor">Intel<BR>
 <input type="radio" name="procesor" checked>Jiný<BR>
```

```
 Jaká je frekvence Vašeho procesoru? (v MHz) <input
type="text" name="frekvence" size="4" maxlength="4"><BR>
 Jaký používáte OS?<BR>
 <select name="OS" size="4">
 <option value="dos">MS-DOS
 <option value="win95">Windows 95/95
 <option value="winme">Windows ME
 <option value="win2k">Windows NT/2000/XP
 <option value="linux">Linux
 <option value="mac">MAC OS
 <option value="other">Jiný
 </select><BR>
 Jaká je rychlost vašeho připojení k Internetu?<BR>
 <select name="net" size="1">
 <option value="no">bez připojení
 <option value="modem">0-56kb/s
 <option value="radio">56-128kb/s
 <option value="wifi">128-512kb/s
 <option value="university">512kb/s - 1Mb/s
 <option value="god">více než 1Mb/s
 </select><BR>
 Poznámky pro redakci:<BR>
 <textarea>
Zde můžete přidat své názory
 </textarea><BR>
 <b>registrační údaje</b><BR>
 E-mail:<input      type="text"      name="mail"      size="20"
value="@"><BR>
 Heslo: <input type="password" name="heslo" size="20"><BR>
 <input  type="checkbox"  name="novinky"  checked>Zasílejte  mi
novinky e-mailem
 <input type="submit" value="Odešli">
 <input type="reset" value="Vymaž">
 </form>
  </body>
</html>
```

2.14 STRUKTUROVANÝ PŘEHLED VYBRANÝCH TAGŮ

Na konci kapitoly HTML bude nyní uveden strukturovaný přehled vybraných HTML tagů, z nichž některé představují doplnění k tagům a jejich variantám uvedeným v řešených příkladech.

2.14.1 ZÁKLADNÍ ZNAČKY

Tabulka 2: Základní značky.

Tag	Popis
`<html></html>`	značka html dokumentu
`<head></head>`	záhlaví html dokumentu
`<meta>`	meta informace (je uvnitř <head></head>)
`<meta name="keywords" content="x">`	klíčová slova stránky (pro vyhledávající nástroje)
`<meta name="description" content="x">`	popis stránky (pro vyhledávací nástroje)
`<title></title>`	titulek dokumentu (je uvnitř <head></head>)
`<body></body>`	tělo html dokumentu
`<body background="url">`	obrázek na pozadí
`<body bgcolor="#xxxxxx">`	barva pozadí stránky (#000000 - #ffffff)
`<body text="#xxxxxx">`	barva textu
`<body link="#xxxxxx">`	barva odkazu
`<body alink="#xxxxxx">`	barva aktivovaného odkazu
`<body vlink="#xxxxxx">`	barva navštíveného odkazu

2.14.2 TYPOGRAFICKÉ ZNAČKY

Tabulka 3: Typografické značky.

Tag	Popis
`<basefont size="x">`	základní velikost písma (1 až 7) pro celou stránku
`<h></h>`	hlavička (1 až 6) - atd.)
`<b></b>`	tučný text
`<strong></strong>`	zvýrazněný text
`<i></i>`	kurzíva
`<u></u>`	podtržený text
`<strike></strike>`	přeškrtnutý text
`<sub></sub>`	dolní index
`<sup></sup>`	horní index
`<tt></tt>`	pevná šířka písma
`<small></small>`	menší písmo než normální velikost
`<big></big>`	větší písmo než normální velikost
`<font size="x">`	velikost písma (1 až 7)
`<font face="font">`	druh písma (verdana, arial, courier atd.)
`<font color="#xxxxxx">`	barva písma v hexadecimální hodnotě

2.14.3 FORMÁTOVACÍ ZNAČKY

Tabulka 4: Formátovací značky.

Tag	Popis
<blockquote></blockquote>	odsadí blok textu
 	zalomení řádku
	centruje
<hr>	vodorovná dělící čáru
<hr align=left/center/all>	zarovná dělící čáru
<hr size="x">	tloušťka dělící čáry
<hr width="x">	délka dělící čáry
<hr noshade	plně černá dělící čára
<nobr>	zakáže zalamování na hraně stránky
	nový odstavec
	zarovnává text odstavce
<pre></pre>	předformátovaný text

2.14.4 ODKAZY

Tabulka 5: Značky pro práci s odkazy.

Tag	Popis
	odkaz na url
	odkaz na url do nového okna prohlížeče
	odkaz na url do rámu
	odkaz uvnitř rámu na url rámu, z něhož byl odkaz aktivován
	odkaz na url do FRAMESET
	uvnitř rámu vylučuje FRAMESET a odkazuje na url do vlastního okna prohlížeče

2.14.5 OBRÁZKY

Tabulka 6: Značky pro práci s obrázky.

Tag	Popis
<img src="url">	zobrazí obrázek
<img src="url" valign=top/middle/bottom>	zarovnává obrázek svisle
<img src="url" align=left/right>	zarovnává obrázek vodorovně
<img src="url" alt="text">	popisný text
<img src="url" usemap="#název mapy">	obrázková mapa a název mapy
<img src="url" width="x" height="x">	rozměry obrázku v pixelech
<img src="url" border="x">	okraj obrázku
<img src="url" hspace="x" vspace="x">	horizontální a vertikální mezera okolo obrázku v pixelech
<img lowsrc="url" src="url">	počáteční obrázek s nízkým rozlišením, než bude natažen s plným rozlišením

2.14.6 SEZNAMY

Tabulka 7: Značky pro práci se seznamy.

Tag	Popis
<dl></dl>	seznam definic
<dd>	definice pojmu
<dt>	definici termů
	uspořádaný seznam
<ol compact>	kompaktní uspořádaný seznam
<ol type=a/a/i/i/1>	uspořádaný seznam různých typů
	položka seznamu
<li type=a/a/i/i/1>	formát položky seznamu
	neuspořádaný seznam
<ul compact>	kompaktní neuspořádaný seznam
<ul type=disc/circle/square>	styl odrážky

2.14.7 FORMULÁŘE

Tabulka 8: Značky pro práci s formuláři.

Tag	Popis
<form action="url" method=get/post></form>	definuje parametry formuláře
<input type="text/password/checkbox/radio/submit/reset/image">	vstupní pole formuláře
<input type=hidden>	skryté pole
<input name="název_pole">	jméno pole formuláře
<input checked>	zaškrtávací políčko
<input size="x">	velikost pole v počtu znaků
<option>	volby menu
<option selected="implicitní hodnota">	implicitně vybraná volba
<option value="hodnota">	hodnota pro menu volby
<select></select>	menu ve formuláři
<select name="jméno"></select>	data náležící menu
<select multiple></select>	určuje více než jednu volbu v menu
<select size="x"></select>	počet viditelných položek v menu
<textarea rows=x cols=y></textarea>	vstupní box
<textarea name="jméno"></textarea>	název vstupního boxu
<textarea wrap></textarea>	automatické zalamování textu uvnitř boxu

2.14.8 RÁMY

Tabulka 9: Značky pro práci s rámy.

Tag	Popis
<frameset></frameset>	v záhlaví HTML stránky a definuje prvky rámu
<frameset cols="x,x">	šířka sloupcových rámů v pixelech nebo procentech
<frameset rows="x,x">	výška řádkových rámů v pixelech nebo procentech
<frameset border="x">	tloušťka okraje rámu
<frameset framespacing="x">	mezery mezi rámy
<frameset frameborder="x">	okraje rámu
<frame src="url">	url k natažení do rámu
<frame align=left/center/right>	zarovnání položek uvnitř rámu
<frame frameborder="x">	okraje rámu
<frame bordercolor="#xxxxxx">	barva okraje pro rám
<frame framespacing="x">	mezera přidaná mezi rámy
<frame name="jméno">	jméno rámu
<frame noresize>	znemožňuje měnit velikost rámu

Tag	Popis
<frame marginwidth="x" marginheight="x">	šířka a výška okraje uvnitř rámů
<frame scrolling="0/1/auto">	určuje zda rám bude mít rolovací proužek

2.14.9 TABULKY

Tabulka 10: Značky pro práci s tabulkami.

Tag	Popis
<table></table>	tabulka
<table border="x">	okraj tabulky
<table cellspacing="x">	volný prostor mezi buňkami
<table cellpadding="x">	tloušťka vnitřního okraje buněk
<table width="x">	šířka tabulky
<table width="%">	šířka tabulky v procentech
<tr></tr>	řádek tabulky
<tr align=left/center/right valign=top/middle/bottom>	horizontální a vertikální zarovnání řádku tabulky
<td></td>	datová buňka tabulky (je uvnitř tabulkových řádků)
<td align=left/center/right valign=top/middle/bottom>	horizontální a vertikální zarovnání obsahu buňky
<td nowrap>	zakazuje zalamování řádek
<td colspan="x">	počet sloupců pro datovou buňku tabulky, přes něž se má buňka roztáhnout ve vodorovném směru
<td rowspan="x">	počet sloupců pro datovou buňku tabulky, přes něž se má buňka roztáhnout ve svislém směru
<td width="x">	šířka datové buňky tabulky
<td width="%">	šířka datové buňky tabulky v procentech
<th></th>	záhlaví tabulky
<th align=left/center/right valign=top/middle/bottom>	horizontální a vertikální zarovnání obsahu datové buňky záhlaví tabulky
<th nowrap>	zákaz zalamování řádek uvnitř datové buňky záhlaví tabulky
<th colspan="x">	počet sloupců pro datovou buňku záhlaví tabulky, přes něž se má buňka roztáhnout ve vodorovném směru
<th rowspan="x">	počet sloupců pro datovou buňku záhlaví tabulky, přes něž se má buňka roztáhnout ve svislém směru
<th width="x">	šířka datové buňky záhlaví tabulky
<th width="%">	šířka datové buňky záhlaví tabulky v procentech
<caption align="top/bottom"></caption>	nadpis tabulky

3 CSS (CASCADING STYLE SHEETS)

Jazyk byl navržen standardizační organizací W3C (World Wide Web Consortium - mezinárodní konsorcium, jehož členové společně s veřejností vyvíjejí webové standardy pro World Wide Web). Hlavním smyslem vývoje tohoto jazyka je umožnit návrhářům oddělit vzhled dokumentu od jeho struktury a obsahu.

CSS se využívá k formátování obsahu HTML, XHTML a XML dokumentů. Ve srovnání s formátováním pomocí atributů v HTML formátovací schopnosti rozšiřuje. Styly umožňují přesně určit, jak bude který element vypadat. Na rozdíl od atributů stylem můžeme definovat jednotný vzhled elementu pro celý dokument (např. že všechny nadpisy úrovně 1 budou červené) a to jediným zápisem pro příslušný element (nikoli v každém tagu příslušného elementu). Stejně tak můžeme pomocí stylu určit odlišné formátování pro třeba jen jediný výskyt určitého elementu. Tím se jednak zbavíme velkého množství kódu, jednak se tento kód stane mnohem přehlednější. Navíc pokud se jednou rozhodneme změnit například barvu písma všech odstavců, bude to pro nás otázka několika málo vteřin, měnit každý atribut u každého elementu v HTML by bylo mnohem delší. Jeden styl můžeme snadno použít pro libovolné množství stránek.

3.1 STYL

Styl se skládá z pravidel pro jednotlivé elementy, které mají být formátovány. Každé takové pravidlo má dvě části, **selektor** (název elementu, pro který má toto pravidlo platit) a **deklaraci** (co pro něj má platit). V deklaraci určujeme vlastnost a její hodnotu, deklarace je uzavřena do složených závorek. Celé to můžeme zapsat jako: `selektor {vlastnost: hodnota_vlastnosti}`. Konkrétně lze danou situaci prezentovat například v podobě určení stylu pro nadpisy první úrovně (Řešený příklad 32).

ŘEŠENÝ PŘÍKLAD 32

```
h1 {color: blue}
```

Selektorem, tedy elementem, který formátujeme je zde **h1** (nadpis 1. úrovně). Deklarací je `{color: blue}`. Ta určuje, že vlastnost **color** bude mít hodnotu **blue**. Celé dohromady to tedy znamená, že všechny nadpisy 1. úrovně v dokumentu budou mít modrou barvu.

Pokud budeme chtít určit elementu více než jednu vlastnost, jednotlivé vlastnosti od sebe oddělíme středníkem. Takto můžeme definovat libovolné množství vlastností (`selektor {vlastnost1: hodnota_vlastnosti1; vlastnost2: hodnota_vlastnosti2;}`).

Pokud budeme chtít určit dvěma elementům jejich společnou vlastnost, oddělíme od sebe jednotlivé selektory čárkou (`selektor1, selektor2 {vlastnost: hodnota_vlastnosti;}`).

3.2 DĚDIČNOST

Většina vlastností se dědí. To znamená, že element, který nemá vlastnost definovanou, jí dědí po nadřazeném elementu. Týká se to především vlastností písma - barvy, velikosti, stylu atd. Pokud tedy chceme definovat nějakou vlastnost, kterou budou mít všechny elementy společnou (a později případně je vytvářet výjimky) definujeme ji pro element resp. tag `<body>`.

3.3 PŘIPOJENÍ STYLU K SOUBORU

Styl můžeme k dokumentu připojit několika způsoby, můžeme definovat přímo v dokumentu nebo v externím souboru, způsoby můžeme i kombinovat.

3.3.1 *EXTERNÍ SOUBOR*

Pokud chceme mít styl uložený v externím souboru (což je velmi výhodné při používání jednoho stylu pro více dokumentů), v nějakém textovém editoru uložíme námi definovaný styl do souboru s příponou **css**. Ten pak připojíme k dokumentu zápisem v hlavičce (tj. mezi tagy `<head>` a `</head>`) buď v tagu `<link>` nebo v tagu `<style>` (Řešený příklad 33).

ŘEŠENÝ PŘÍKLAD 33

```
<link rel="stylesheet" type="text/css" href="styl.css" />
nebo
<style type="text/css">@import "styl.css";</style>
```

Zápisem `@import "styl.css";` můžeme také vložit jeden styl do druhého stylu.

3.3.2 *DEFINOVÁNÍ STYLU UVNITŘ DOKUMENTU*

To můžeme provést opět v tagu `<style>` kam tentokrát místo odkazu na externí styl umístíme přímo definici stylu nebo můžeme definovat styl přímo nějakému elementu, což se hodí zvláště v případě, kdy máme definovaný jednotný styl, ale pro například jedno konkrétní slovo chceme použít jiné pravidlo. Potom použijeme v příslušném tagu atribut **style (**Řešený příklad 34).

ŘEŠENÝ PŘÍKLAD 34

```
<style type="text/css">body {color: blue}</style>
nebo
<h1 "style=color: green">nadpis</h1>
```

3.4 BARVY

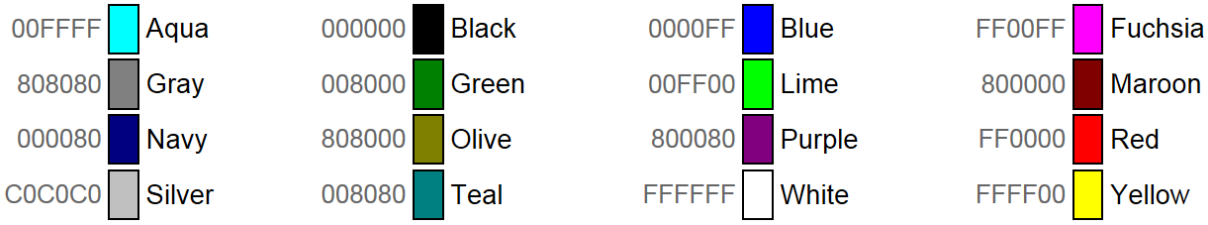
Na stránce se barví text, odkazy, pozadí a ohraničení. Barvit můžeme buď celý dokument selektorem **body,** nebo jednotlivé elementy.

Barvu pozadí volíme tak, aby text byl dobře čitelný, tedy kontrastní textu, barva by neměla být příliš jasná, vysloveně nevhodné je použití jasně červené, opatrně i s modrou. Signální barvy jsou dobré pro zvýraznění textu nebo odkazů. Pokud předpokládáme, že uživatelé si budou stránku tisknout, je třeba dát pozor na podobnou sytost barev (aby se barvy „neslily").

Pro zápis barvy můžeme použít:

- **Název barvy (v angličtině)** – u předdefinovaných (pojmenovaných) barev například `body {color: blue}`
  Pozn.: *Při použití zápisu jménem barvy je validních pouze základních 16 VGA barev.*
  (Obrázek 3)

Obrázek 3: 16 základních VGA barev.



- **RGB zápis:**
  - ***Procentuálně*** - Intenzitu každé barvy určujeme procenty 0% (minimální intenzita) až 100% (maximální intenzita), například: `body {color: rgb(100%,0%,0%)}`
  - ***Desítkově*** - Čísly 0 až 255, například: `body {color: rgb(255,0,0)}`
  - ***Šestnáctkově*** - 00 až FF (tj. 0-9, a-f), zápis začíná hash-markem **#,** například: `body {color: #FF0000}`
    Pokud jsou obě číslice odpovídající jedné barevné složce stejné, můžeme použít zkrácený zápis: `body {color: #F00}`

U většiny elementů můžeme barvit pozadí pomocí vlastnosti **background-color** nebo text pomocí vlastností **color**. (Řešený příklad 35).

ŘEŠENÝ PŘÍKLAD 35

```
body {background-color: #00FF00}
```
nebo
```
h1 {color: #FFFFFF}
```

Pokud nedefinujeme žádné barvy, budou použity defaultní barvy prohlížeče, což je obvykle transparentní pro pozadí, černá pro text, modrá pro odkazy, fialová pro navštívené odkazy a červená pro aktivní odkazy.

Jako doplnění k problematice barev ještě uveďme, že tabulky barev s příslušným šestnáctkovým vyjádřením můžeme nalézt například na:

- http://www.cs.vsb.cz/benes/vyuka/html/barvy.htm;
- http://www.feudal.cz/image/html/tabulka_barev.htm;
- http://www.klikzone.cz/sekce-html/html-barvy.php.

3.5 PÍSMO

3.5.1 RODINA PÍSMA

K určení rodiny písma slouží vlastnost **font-family**. V té definujeme názvem písma, jaké konkrétní písmo bude použito (např. Arial). Dále definujeme písmo, které bude použito v případě, že první definované písmo nemá prohlížeč k dispozici. Takto můžeme určit i několik alternativních písem. Pro případ, že by prohlížeč neměl k dispozici žádné z určených písem můžeme ještě použít obecnou rodinu (tu bychom měli použít vždy a to na konci výpisu písem):

- serif - patkové písmo (např. Times New Roman)
- sans-serif - bezpatkové písmo (např. Arial, Arial CE, Arial Narrow, Verdana, Helvetica)
- `monospace` – neproporciální písmo (např. `Courier`, `Courier New`)

- cursive - ozdobná kurzíva (např. Comic Sans,)
- FANTASY - ozdobné písmo (např. WESTERN)

Celý zápis pak může vypadat např. jako: (Řešený příklad 36)

ŘEŠENÝ PŘÍKLAD 36

```
body {font-family: Arial, Helvetica, sans-serif}
```

3.5.2 STYL PÍSMA

Vlastnost **font-style** určí, zda se bude jednat o písmo normální, kurzívu, nebo skloněné písmo (Řešený příklad 37).

ŘEŠENÝ PŘÍKLAD 37

```
body {font-style: italic}
```

3.5.3 VARIANTA PÍSMA

Vedle typu písma a stylu písma můžeme dále nastavovat například variantu písma, která může být buď normální nebo kapitálky (Řešený příklad 38).

ŘEŠENÝ PŘÍKLAD 38

```
h1 {font-variant: small-caps}
```

3.5.4 VELIKOST PÍSMA

Velikost písma určujeme pomocí vlastnosti **font-size**, u které můžeme nastavovat hodnoty několika způsoby. Prvním z nich je nastavení hodnoty ve tvaru (Řešený příklad 39):

- xx-small
- x-small
- small
- medium
- large
- x-large
- xx-large

ŘEŠENÝ PŘÍKLAD 39

```
h1 {font-size: xx-small}
nebo
…
h1 {font-size: xx-large}
```

Druhou možností je nastavování velikosti písma relativně vzhledem k aktuální velikosti písma (Řešený příklad 40).

ŘEŠENÝ PŘÍKLAD 40

```
h2 {font-size: larger}
nebo
h2 {font-size: smaller}
```

Dále můžeme velikost určit např. v pixelech (px) nebo jiných jednotkách (pt, em, ex, cm, mm, in) nebo relativně vzhledem k aktuální velikosti písma v procentech. (Řešený příklad 41).

ŘEŠENÝ PŘÍKLAD 41

```
p {font-size: 16px}
nebo
h1 {font-size: 150%}
```

Další možnou vlastností písma je tzv. duktus, který představuje poměr tloušťky písmových tahů k výšce písma, tedy to, jak je písmo silné. Určuje se vlastností **font-weight** (Řešený příklad 42).

ŘEŠENÝ PŘÍKLAD 42

```
h1 {font-weight: bold}
```

Tato vlastnost může mít hodnotu ještě **bolder** (o něco silnější písmo než normální) a **lighter** (o něco slabší než normální).

Provedeme-li opět malé shrnutí problematiky definování vlastností písma, můžeme si prezentovat případovou studii (Případová studie 2) obsahující využití CSS v html kódu s cílem formátovat nadpisy 2. úrovně na modrou barvu a kurzívu.

PŘÍPADOVÁ STUDIE 2

```
<html>
 <head>
 <title>První příklad se stylopisem</title>
<style type="text/css">
 h2 {color: blue; font-style: italic;}
 </style>
 </head>
 <body>
 <h1>Hlavní nadpis</h1>
 <h2>Nadpis druhé úrovně</h2>
 <p>Odstavec s normálním textem</p>
 <h2>Další nadpis druhé úrovně</h2>
 <p>Odstavec s dalším textem, který by se měl formátovat
normálně. </p>
 </body>
 </html>
```

3.6 ODKAZY

3.6.1 BARVA ODKAZU

Defaultně se odkazy v prohlížeči zobrazují modré, podtržené. Navštívené odkazy jsou fialové a aktivní odkazy červené. Při přejetí myší přes odkaz se jeho vzhled nijak nemění.

Barvu odkazu změníme stejně jako barvu jakéhokoliv jiného elementu, tedy pomoci vlastnosti **color** (Řešený příklad 43).

ŘEŠENÝ PŘÍKLAD 43

```
a {color: green}
```

Bývá dobrým zvykem odkazy podtrhávat. Ne pro každého uživatele je barevné odlišení odkazů od okolního textu dostačující. Stejně tak dobrým zvykem je nepodtrhávat žádný jiný text. K definování stylu ozdobení slouží vlastnost **text-decoration**. Má vlastnosti **none** (bez ozdobení), **underline** (podtržení), **overline** (nadtržení), **line-through** (přeškrtnutí). Použít můžeme i všechny hodnoty najednou (Řešený příklad 44).

ŘEŠENÝ PŘÍKLAD 44

```
a {text-decoration:underline overline}
```

3.6.2 PSEUDOTŘÍDY

Psudotřídy slouží k odlišnému definování vlastností odkazů (**link**), navštívených odkazů (**visited**) a aktivních odkazů (**active**). Zapisují se s dvojtečkou za tag a (Řešený příklad 45).

ŘEŠENÝ PŘÍKLAD 45

```
a:visited {color: yellow}
```

Pokud chceme, aby odkazy svůj vzhled změnili při přejetí myší, použijeme **hover** (Řešený příklad 46).

ŘEŠENÝ PŘÍKLAD 46

```
a:hover {text-decoration: none}
```

3.7 SEZNAMY

Seznamy můžeme v CSS formátovat jako jakékoliv jiné elementy, navíc však můžeme určit druh odrážky číslovaného i nečíslovaného seznamu. S použitím obrázku můžeme vytvořit i odrážky vlastní.

Druh odrážky i číslování určujeme vlastností **list-style-type**. Tuto vlastnost můžeme přiřadit jak selektoru seznamu **ul** nebo **ol**, tak i položce seznamu **li** (Řešený příklad 47).

ŘEŠENÝ PŘÍKLAD 47

```
ul {list-style-type:square}
```

Seznamy mohou být:

- **Nečíslované seznamy**
  - *disc* – vyplněné kolečko v IE a Opeře; vyplněný kosočtverec v Mozille a Netscape;
  - *circle* – prázdné kolečko v IE a Opeře; prázdný kosočtverec v Mozille a Netscape;
  - *square* – čtvereček vyplněný.
- **Číslované seznamy**

1. *decimal* – arabské číslice (1.);
ii. *lower-roman* – malé římské číslice (i.);
III. *upper-roman* – velké římské číslice (I.);

d. *lower-alpha* – malá písmena (a.);
E. *upper-alpha* – velká písmena (A.);
none – bez odrážky.

- **Další**
  - *decimal-leading-zero* – arabské číslice, u jednomístných čísel začínající nulou (01.); nefunguje v IE a Opeře;
  - *lower-greek* – malá písmena řecké abecedy (α.); nefunguje v IE;
  - *lower-latin* – malá písmena (a.); nefunguje v IE;
  - *upper-latin* – velká písmena (A.); nefunguje v IE;
  - *armenian* – arménština, nefunguje v IE, v Opeře jen pokud je dostupný font.

Odrážku umisťujeme pomocí vlastnosti **list-style-position**. Může být buď uvnitř textu, tomu odpovídá hodnota **inside**, nebo vně textu s hodnotou **outside**.

Pokud chceme použít jako odrážku nějaký obrázek, využijeme vlastnost **list-style-image**, která má jako hodnotu URL obrázku (Řešený příklad 48).

ŘEŠENÝ PŘÍKLAD 48

```
ul {list-style-image:url("adresar/obrazek.gif");}
```

3.8 OHRANIČENÍ

Způsob ohraničení elementů určuje vlastnost **border** – pro ohraničení elementu „ze všech stran" a vlastnosti **border-top**, **border-right**, **border-bottom** a **border-left** pro ohraničení zeshora, zprava, zdola a zleva. Tyto vlastnosti mají tři hodnoty (Řešený příklad 49):

- tloušťka ohraničení - např. 1px
- styl ohraničení
  - *solid* – jednoduchá čára
  - *double* – dvojitá čára
  - *dotted* – tečkovaná čára
- barva ohraničení (viz barvy)

ŘEŠENÝ PŘÍKLAD 49

```
border-top:2px solid green;
```

Hodnotou vlastnosti **border** může být také *none*. Element tedy nebude mít žádné ohraničení (což je standardní nastavení). Tuto vlastnost využijeme například v situaci, kdy máme obrázkový odkaz a kolem obrázku vytvoří zpravidla modrý rámeček (Řešený příklad 50).

ŘEŠENÝ PŘÍKLAD 50

```
a img {border:none}
```

Jednotlivé hodnoty vlastnosti **border** můžeme zapisovat také zvlášť pomocí vlastností **border-width** (tloušťka ohraničení), **border-style** (styl ohraničení) a **border-color** (barva ohraničení) (Řešený příklad 51).

ŘEŠENÝ PŘÍKLAD 51

```
border-width:2px;
border-style:solid;
border-color:green;
```

Rozepsání po jednotlivých vlastnostech je vhodné např. v případě, že chceme třeba všechny nadpisy ohraničit tenkou čárou, ale každý jinou barvou. Pak definujeme tloušťku a styl ohraničení pro všechny nadpisy najednou a barvu definujeme zvlášť pro každou úroveň nadpisu.

3.9 POZADÍ

Pozadí elementu i celé stránky můžeme vytvořit buď prostým určením barvy pozadí nebo použitím obrázku.

Pro vytvoření barevného pozadí použijeme CSS vlastnost **background-color**. Její hodnotou bude požadovaná barva (Řešený příklad 52).

ŘEŠENÝ PŘÍKLAD 52

```
body {background-color:#aaa}
```

Chceme-li na pozadí vložit obrázek, použijeme vlastnost **background-image**, jejíž hodnotou je cesta k obrázku zapsaná v závorce za parametrem **url** (Řešený příklad 53).

ŘEŠENÝ PŘÍKLAD 53

```
body {background-image:url(obrazky/pozadi.gif)}
```

Obrázek na pozadí se může opakovat (např. jako nějaký vzorek) nebo může být umístěn jen jednou. Standardně se obrázek opakuje v obou směrech (x i y) až do zaplnění celé plochy elementu.

Pro určení způsobu opakování se používá vlastnost **background-repeat**. Pokud chceme, aby se obrázek opakoval jen ve směru *x* (tj. horizontálně), použijeme hodnotu **repeat-x**, ve směru *y* (tj. vertikálně) **repeat-y**. Pokud nechceme, aby se obrázek opakoval (např. jde-li o logo či jiný grafický počin, který nemá vytvářet vzorek), použijeme hodnotu **no-repeat** (Řešený příklad 54).

ŘEŠENÝ PŘÍKLAD 54

```
body {background-repeat:repeat-y}
```

Při opakování obrázku je vhodné použít obrázek, který „sám na sebe navazuje". V opačném případě bychom vytvořili kromě námi zamýšleného vzorku ještě nějaké to čtverečkování navíc. Opakovat bychom také neměli jeden velký obrázek, který neobsahuje vzor, ale třeba jen jeden objekt.

Pokud chceme obrázek umístit jinak, než s výchozí pozicí (tj. levý horní roh elementu), použijeme vlastnost **background-position**. Umístění určuje horizontální a vertikální hodnota. Hodnoty horizontálního umístění mohou být **left** (vlevo), **center** (na střed) a **right** (vpravo). Hodnoty vertikálního umístění jsou **top** (nahoru), **center** (na střed) a **bottom** (dolů) (Řešený příklad 55).

ŘEŠENÝ PŘÍKLAD 55

```
h1 {background-position:left center}
```

3.10 TABULKY

Nastavení šířky a výšky buněk tabulky se pomocí CSS provádí pomocí parametrů **width** a **height** (Řešený příklad 56).

ŘEŠENÝ PŘÍKLAD 56

```
td {width: 20ex;
    height: 120px;}
```

Individuální nastavení jednotlivých buněk tabulky lze zařídit pomocí mechanismu identifikátorů jazyka HTML, přičemž meze individualismu vytyčuje princip tabulkového zobrazení tím, že všechny buňky v jednom řádku budou stejně vysoké a analogicky všechny buňky v jednom sloupci stejně široké. Budeme-li tedy nastavovat různou šířku více buněk v jednom sloupci, prohlížeč si vybere tu největší hodnotu pro celý sloupec buněk.

Budeme-li přesnější, pak CSS nabízí pět parametrů pro nastavování dimenzí objektů, a to **min-width** a **max-width** pro vymezení minimální a maximální šířky, analogicky **min-height** a **max-height** pro vymezení minimální a maximální výšky a **table-layout**. Výchozí hodnotou **table-layout** je **auto**, kdy si prohlížeč rozměry buněk přepočítává tak, aby se celá tabulka co nejlépe vešla do okna. Použijeme-li druhou přípustnou hodnotu **fixed**, prohlížeči je toto přepočítávání zakázáno (Řešený příklad 57).

ŘEŠENÝ PŘÍKLAD 57

```
table {table-layout: fixed;}
 td {width: 260px;
     height: 2ex;}
```

Tento příklad prohlížeči sděluje, že bez ohledu na velikost okna prohlížeče i obsah jednotlivých buněk tabulky mají být sloupce široké 260 pixelů a řádky vysoké jako dvě písmena x. Tento parametr v praxi funguje s tou výhradou, že šířky sloupců musí být definované v buňkách prvního řádku, nebo lépe prostřednictvím značek `<colgroup>` a `<col>`, jako například (Řešený příklad 58) a (Řešený příklad 59):

ŘEŠENÝ PŘÍKLAD 58

```
table {table-layout: fixed;}
 #prvniDvaSloupce {width: 160px;}
 #tretiSloupec {width: 60ex;}
```

ŘEŠENÝ PŘÍKLAD 59

```
<table>
   <colgroup id="prvniDvaSloupce" span="2" />
   <colgroup id="tretiSloupec" />
   <tr>
   .
   .
   </tr>
 </table>
```

Nyní přejdeme od šířky sloupců k šířce celé tabulky. Jako výchozí si prezentujme příklad, ve kterém nastavíme šířku tabulky na 0 (Řešený příklad 60).

ŘEŠENÝ PŘÍKLAD 60

```
table {table-layout: fixed;
       width: 0px;}
```

Bude-li šířka nastavena menší (třeba právě na nulu, jak je uvedeno v příkladě), než kolik činí součet šířek jednotlivých sloupců, bude tabulka vykreslena fixně se zadanými šířkami sloupců. Bude-li šířka tabulky nastavena větší než zmiňovaný součet, bude tabulka vykreslena rovněž fixně, ovšem s šířkami sloupců úměrně zvětšenými tak, aby celková šířka tabulky odpovídala nastavení. Tohoto pravidla se dá vhodně využít kombinací absolutních a relativních měřítek (Řešený příklad 61).

ŘEŠENÝ PŘÍKLAD 61

```
table {table-layout: fixed;
       width: 100%;}
 #prvniSloupec {width: 100px;}
 #druhySloupec {width: 500px;}
 #tretiSloupec {width: 200px;}
```

Podle uvedeného příkladu bude šířka tabulky vždy minimálně 100+500+200 = 800 pixelů, bude-li však okno prohlížeče širší, tabulka se roztáhne přes celou šířku. Použijeme-li parametr **width** u tabulky s parametrem table-layout nastaveným na hodnotu auto (což odpovídá případu, kdy prohlížeč tomuto parametru nerozumí), pak výsledná šířka tabulky bude vždy odpovídat právě této hodnotě. Jsou-li tedy uvedeny šířky jednotlivých sloupců, budou poměrově přepočítány tak, aby v součtu korespondovaly s požadovanou šířkou tabulky (Řešený příklad 62).

ŘEŠENÝ PŘÍKLAD 62

```
table {width: 400px;}
 #prvniSloupec {width: 100px;}
 #druhySloupec {width: 500px;}
 #tretiSloupec {width: 200px;}
```

Podle tohoto příkladu tedy ve výsledku budou sloupce široké 50 px, 250 px a 100 px, a to ještě za předpokladu, že si obsah některé z buněk nevynutí zvětšení na úkor ostatních sloupců. Bude-li pro naše účely dostačovat pevná šířka tabulky, je tento způsob uspokojivý, chceme-li však míti plnou kontrolu nad šířkami sloupců a přitom musíme zapomenout na parametr table-layout, lze si ještě vypomoci trikem popsaným dále.

Výchozí algoritmus vykreslování tabulek v prohlížeči se snaží zobrazit celý obsah buněk při co nejmenší výšce tabulky. Obsahuje-li tedy nějaká buňka dlouhý nezalamovaný text, bude sloupec s touto buňkou iniciovat roztáhnutí tabulky přes celou šířku okna prohlížeče. Při zužování okna prohlížeče bude nejmenší šířka sloupce odpovídat největšímu nedělitelnému objektu, v případě textu nejdelšímu slovu. Co z toho vyplývá?

Maximální šířku sloupce nastavíme pomocí parametru **width**, přičemž zejména pro Internet Explorer je důležité, aby tato šířka byla definována pro buňku s nejdelším obsahem – bude-li nastavena pouze pro buňku ze stejného sloupce, ale některého z předchozích řádků, prohlížeč ji nebude brát v potaz. Zřejmě nejlepší je nastavovat šířku sloupce skrze značky

t definované v buňkách prvního řádku, nebo lépe prostřednictvím značek `<colgroup>` a `<col>`.

Výška řádků se přizpůsobí obsahu buněk tak, aby nic nebylo skryto. Pomocí parametru **height** je možné přikázat výšku větší, definování menší výšky však budou prohlížeče ignorovat.

Minimální šířka sloupců bude odpovídat nejširšímu objektu, potažmo slovu v daném sloupci, nuže zde může tvůrce stránky kalkulovat s obsahem tabulky, co se týče například nedělitelných nadpisů či obrázků. Podstatné je, že do nestlačitelného obsahu buněk se započítává i vycpávka, téměř univerzálním trikem tedy může být vpašování prázdného řádku do tabulky s definovanou vycpávkou (Řešený příklad 63).

ŘEŠENÝ PŘÍKLAD 63

```
#prvniSloupec {width: 100px;}
 #druhySloupec {width: 500px;}
 #prvniRadek {visibility: hidden;}
 #prvniBunka {padding-left: 100px;}
 #druhaBunka {padding-left: 500px;}
```

Kde popis tabulky bude:

```
<table>
   <colgroup id="prvniSloupec" />
   <colgroup id="druhySloupec" />
   <tr id="prvniRadek">
     <td id="prvniBunka" />
     <td id="prvniBunka" />
   </tr>
   <tr>
   .
   .
   </tr>
 </table>
```

V návaznosti na uvedené příklady se doporučuje šířku sloupců zadávat pomocí parametru `<colgroup>`, u kterého lze pomocí atributu **span** zadat, na kolik sloupců se má vztahovat, případně může reprezentanty těchto sloupců obsahovat jako značku `<col>` (Řešený příklad 64) .

ŘEŠENÝ PŘÍKLAD 64

```
<colgroup span="2" id="prvniDvaSloupce" />
 <colgroup id="druheDvaSloupce">
   <col id="tretiSloupec" />
   <col id="ctvrtySloupec" />
 </colgroup>
```

Vnořené značky `<col>` použijeme zejména v případě, že chceme pro sousední sloupce definovat stejnou šířku, ale jiné zarovnání a naopak.

3.11 TŘÍDY A IDENTIFIKÁTORY

3.11.1 TŘÍDY V CSS

Třídy a identifikátory v CSS slouží k tomu, abychom mohli různé elementy formátovat různě. Například odkazy na stránce. Každý z nás asi chce mít na stránce různé druhy odkazů, ne jen jeden. Jinak se obvykle dělají odkazy v menu, jinak odkazy v textu.

Třídy vytvoříme snadno tak, že k elementu v HTML přidáme atribut class. Jeho hodnotou bude nějaký řetězec písmen, stejný pak budeme používat v CSS stylu jako selektor (Řešený příklad 65).

ŘEŠENÝ PŘÍKLAD 65

```
<p class="poznamka">Nějaký text</p>
```

Tímto říkáme, že tento odstavec bude formátován podle pravidel třídy **poznamka**, na formátování ostatních odstavců se tato pravidla neprojeví. Teď musíme ještě ta pravidla určit v CSS stylu (Řešený příklad 66).

ŘEŠENÝ PŘÍKLAD 66

```
.poznamka {font-size: x-small; color: black}
```

Teď tedy budeme mít všechny odstavce stejné, jen odstavec s třídou **poznamka** bude vypadat jinak (malým černým písmem). Resp. jinak budou vypadat všechny odstavce s třídou **poznamka**, protože stejnou třídu můžeme použít u libovolného množství elementů. Dokonce i u různých elementů (Řešený příklad 67).

ŘEŠENÝ PŘÍKLAD 67

```
<p class="poznamka">Nějaký odstavec</p>
<li class="poznamka">Položka seznamu</li>

.poznamka {color: black}    - styl se aplikuje na všechny
elementy s třídou poznamka
li.poznamka {color: blue}   - styl se aplikuje jen na elementy
li s třídou poznamka
```

3.11.2 DĚDIČNOST

Třída každého elementu bude dědit všechny vlastnosti daného elementu a navíc bude mít své vlastní, tedy např. (Řešený příklad 68):

ŘEŠENÝ PŘÍKLAD 68

```
p {text-align: center; color: blue}
.poznamka {font-size:x-small; color: black}
```

Tímto zápisem je řečeno, že všechny odstavce budou zarovnány na střed a jejich text bude mít modrou barvu (první deklarace). Toto obecně platí pro všechny odstavce stránky. Odstavec s třídou **poznamka** bude mít navíc ještě menší písmo (druhá deklarace). Barva je určená v obou deklaracích, v takovém případě ma vyšší prioritu deklarace zadaná později,

tedy odstavec s třídou **poznamka** bude mít černý text. Všechny odstavce tedy budou modré, zarovnané na střed, kromě odstavce s třídou **poznamka**, ten bude černý, s malým písmem, zarovnaný bude také na střed.

3.11.3 KONTEXTOVÁ DEKLARACE

V CSS lze využít i tzv. kontextovou deklaraci. Možnost kontextové deklarace platí i pro třídu (Řešený příklad 69)

ŘEŠENÝ PŘÍKLAD 69

```
<p class="poznamka"><a href=" … ">Odkaz</a> Nějaký text</p>
.poznamka a {color: red}
```

Zápis v Řešeném příkladu XY říká, že odkazy v odstavci třídy **poznamka** budou červené.

3.12 IDENTIFIKÁTOR ID V CSS

Identifikátor se od třídy liší tím, že se jedná vždy o **jednoznačný identifikátor**. To znamená, že ho na každé stránce můžeme použít jen jednou. Tím je myšleno na každé stránce, tedy v každém *souboru.html* jen jednou. V rámci celého webu je možné stejný identifikátor použít libovolněkrát. Oproti tomu třídu je možné použít libovolněkrát na každé stránce webu.

Identifikátory se tedy používají právě tam, kde je jisté, že se daný element objeví ve stránce jen jednou. Ideálně se tedy hodí pro věci jako je box celé stránky, menu, záhlaví nebo zápatí. Identifikátory se označují dvojkřížkem (#). Jinak je jejich zápis stejný jako zápis třídy (Řešený příklad 70).

ŘEŠENÝ PŘÍKLAD 70

```
<div id="menu"> … </div>

#menu {width:14em; background-color:black}
#menu a {color: white}
```

Poznámka: *v Řešeném příkladu XY je využit zatím v tomto textu nevysvětlený tag . V tomto případě to nevadilo, protože věcná náplň daného příkladu je orientovaná na jinou problematiku. Charakteristika tagu je ovšem vysvětlena hned v následující podkapitole.*

3.13 CSS POZICOVÁNÍ

Jakýkoliv objekt (obrázek, tabulka, text, cokoliv) lze umístit kamkoliv na stránku, s objekty se může posouvat a mohou se překrývat.

Existují dva naprosto odlišné druhy pozicování:

- **Absolutní pozice** - umístí objekt do stránky na udané souřadnice bez ohledu na okolní text.
- **Relativní pozice** - určuje o kolik se má objekt posunout oproti své normální poloze. Vlastnosti, které určují směr a míru posunutí, jsou **top** a **left**. **Top** určuje posunutí objektu směrem dolů a **left** doprava. Pro posouvání nahoru a doleva se nepoužívá **bottom** a **right**,

ale **top** a **left** se zápornými hodnotami. Velikost posunu je možné zadat ve všech jednotkách, které CSS podporuje (nejčastěji px, pt nebo cm).

Pro pozicování se využívají dva tzv. neutrální tagy a . Tyto tagy nedávají obsahu v html dokumentu význam a sami o sobě nijak html dokument neformátují. Jedná se o párové tagy. Pomocí CSS vlastností se tyto tagy dají všemožně formátovat (můžeme jim přidělovat rozměry, umísťovat je na stránce, kam se nám zlíbí – pozicovat je, barvit je atd.). Především pomocí divů se pak tvoří layouty stránek (rozmístění jednotlivých prvků). Jeden div určuje hlavičku, další třeba sloupec s menu, jiný zas patičku atp.

Základním rozdílem mezi tahy a je, že div je blokový a span řádkový. Blokový prvek před sebou i za sebou zalamuje řádek a roztáhne se přes celou dostupnou šířku. Pro porovnání uveďme, že mezi blokové prvky patří také odstavec, nadpisy a další. Vedle tohoto řádkový prvek nezalamuje řádek, tudíž jich v jednom řádku může být více.

Při striktním dodržování Doctype, což by mělo být primárním požadavkem, by pak měl být (nejen) tag umístěn v nějakém blokovém tagu. Obecně by pak blokový tag neměl být v tagu řádkovém. Navíc by neměl být ani v jiném blokovém tagu než zase v tagu (Řešený příklad 71).

ŘEŠENÝ PŘÍKLAD 71

```
<span><p>Řádkové a blokové tagy</p></span> – špatný zápis
<p><span>Řádkové a blokové tagy</span></p> – správný zápis

<p><div>Řádkové a blokové tagy</div></p> – špatný zápis
<div><p>Řádkové a blokové tagy</p></div> – správný zápis
```

Vrátíme-li se konkrétně zpět k pozicování, pak můžeme jako výchozí uvést Řešený příklad 72, na kterém bude daná problematika názorněji vysvětlena:

ŘEŠENÝ PŘÍKLAD 72

```
<body>
 Normální text,
 <span style="position: relative; top: 20px"> relativně
umístěný text</span> a
 <span style="position: absolute; top: 100px; left:
150px">absolutně umístěný text</span>.
 </body>
```

V Řešeném příkladu 72 se relativně umístěný text posouvá o 20 pixelů. Absolutní poloha umisťuje levý horní roh objektu (span) na souřadnice 150, 100 (vlastnost left: 150px zde v tuto chvíli posouvá po ose x o 150 pixelů).

Obecnou syntaxi zápisu pozicování lze vyjádřit ve tvaru:
<*tag* style="position:(absolute|relative); [top: *délka*]; [left: *délka*]; [z-index: *číslo*]">*Pozicovaný element*</*tag*>

Budeme-li konkrétnější, pak **position:relative** je v každém smyslu pouze posunutí. Okolní text se formátuje bez ohledu na jakoukoliv pozici objektu, to znamená, že tam, kde by posouvaný objekt měl být, zůstává prázdné místo. Jinými slovy při relativní poloze element není vyjmut z toku dokumentu (Řešený příklad 73).

ŘEŠENÝ PŘÍKLAD 73

```
<b style="position: relative; top: 12px">relativní</b>
```

Objekt s vlastností **position: absolute** je možné umístit kamkoliv. Okolní text se chová, jako by takový objekt vůbec neexistoval, jinými slovy je vyjmut z toku dokumentu. Při práci s vlastnostmi **top** a **left** lze docílit i některých dalších možností, kterými mohou být:

- ve stylu se neuvede **left** ani **top** => objekt se umístí na svojí normální pozici a následující text bude plynout přes něj (nebo se uvede **left**: auto; **top**: auto);
- při style="**position: absolute**; **left**: 0px; **top**: 0px" se levý horní roh objektu umístí do počátku souřadného systému, což je většinou horní levý roh stránky (pokud nevyužíváme tzv. vnořené pozice);
- uvedení nenulových hodnot **top** a **left** samozřejmě posunuje objekt po stránce (z levé horní polohy) - tento se přitom překrývá s normálním obsahem;
- pokud se uvede pouze jedna ze souřadnic (vlastností **left** nebo **top**), zůstává ta neuvedená taková, jakou by měl objekt v toku dokumentu. Například style="**position: absolute**; **left**: 0px" umístí objekt na levý okraj stránky, ale nezmění jeho pozici vertikální, protože není uvedeno **top**. Stejně jako neuvedení funguje nastavení hodnoty na "auto".

V případě, že pozicovaný prvek (absolutní nebo relativní) obsahuje druhý absolutně pozicovaný prvek, bude ten druhý brát počátek souřadnic podle horního levého rohu prvního prvku. Jinými slovy každý absolutně či relativně pozicovaný prvek definuje souřadnicový systém pro všechny absolutně pozicované prvky, které obsahuje.

Pro lepší pochopení dané problematiky si můžeme uvést dvě konkrétní případové studie (Případová studie 3 a Případová studie 4), ve kterých bude prezentováno použití CSS a HTML.

PŘÍPADOVÁ STUDIE 3

CSS:

```
.barva-1 {color: green; font-size: 120%;}
.barva-2 {color: magenta; font-size: 120%;}
```

Aplikace v HTML:

```
<p>Mezi moje oblíbené barvy patří <span class="barva-
1">zelená</span> a <span class="barva-2">purpurová</span>.</p>
```

V prohlížeči bude výsledek zobrazen jako:

Mezi moje oblíbené barvy patří **zelená** a **purpurová**.

PŘÍPADOVÁ STUDIE 4

CSS:

```
.oramovany {border: 1px solid violet;}
.box {width: 200px; background-color: #0066CC; color: white;
border: 3px solid yellow;}
```

HTML:

```
<div class="oramovany">Ukázka orámovaného divu</span>
<div class="box">
<p>Další ukázka divu formátovaného pomocí CSS.</p>
```

```
<p>Druhý odstavec v divu&hellip;</p>
</div>
```

V prohlížeči bude výsledek zobrazen jako:

Jak je vidět na prvním divu, pokud mu šířku nenastavíme, roztáhne se přes celou dostupnou šířku. Rámeček mu byl přidán z důvodu názornějšího zobrazení.

Pokud bychom měli jen jeden odstavec a chtěli ho vybarvit a orámovat jako div ve druhém případě, stačí nastavit stejné css vlastnosti přímo tomu odstavci a nemusíme ho balit do divu.

Poměrně častým nešvarem je tzv. podivný kód. Tak můžeme nazvat HTML kód, který obsahuje divy i na místech, kde nemají co hledat nebo jich (divů) je v kódu víc, než je třeba, případně obojí. Jako příklad můžeme uvést (Řešený příklad 74):

ŘEŠENÝ PŘÍKLAD 74

```
<div class="nazev">Název mého webu </div>
<h1>Název mého webu </h1>
```

Jelikož pomocí CSS můžeme div i h1 naformátovat zcela stejně, jejich vzhled v prohlížeči bude shodný.

Avšak nejen, že stránky s podivným kódem budou pro vyhledávače hůře viditelné, ale takový kód je méně přehledný a tím jsou složitější jeho úpravy.

3.14 SÉMANTIKA

V souvislosti s propojováním HTML kódu a CSS je nutné uvést, že základním požadavkem je psát sémanticky správné kódy. Při psaní kódu by každá HTML značka měla být používána na to, k čemu byla určena. Ve vazbě na CSS platí pravidlo, nesémantické tagy (nenesou význam) a by pak měly být používány pouze k definici vzhledu stránky. Pro názornou prezentaci si můžeme ukázat dva zcela jednoduché příklady, kdy první představuje nesprávně napsaný kód a druhý správně (Řešený příklad 75).

ŘEŠENÝ PŘÍKLAD 75

Špatně:

```
<div id="hlavni-nadpis">Nadpis (název) webu</div>
<p>
  Střevlík<br>
  Tesařík<br>
  Páteříček<br>
  Krasec<br>
  Potápník
</p>
```

ŘEŠENÝ PŘÍKLAD 76

Správně:

```
<h1>Nadpis (název) webu</h1>
<ul>
  <li>Střevlík</li>
  <li>Tesařík</li>
  <li>Páteříček</li>
  <li>Krasec</li>
  <li>Potápník</li>
</ul>
```

Ačkoliv se v mnohých případech stává, že špatně i správně zapsaný kód zobrazuje v prohlížeči stejný výsledek, je tedy otázkou, proč je důležité klást na správnou sémantiku velký důraz. Důvodů je hned několik:

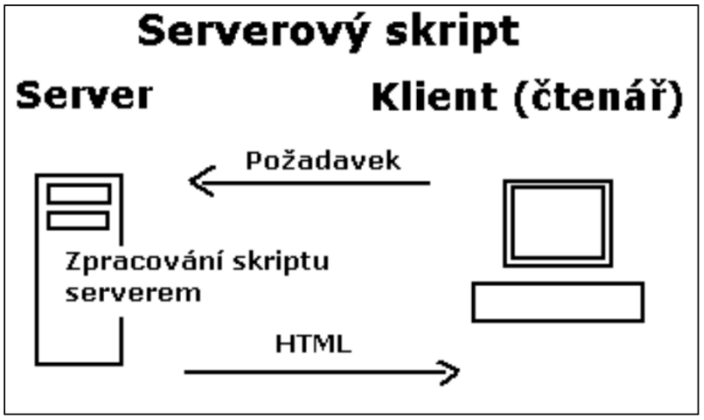
- Stránky se sémantickým kódem jsou pro vyhledávače (Google, Seznam, Bing, Yahoo, …) více viditelné. Ve výsledcích vyhledávání se tak budou naše stránky umísťovat na přednějších pozicích a tím pádem je navštíví více lidí. Na web s nesémantickým kódem, který je ve výsledcích hledání někde na páté stránce či dál, moc lidí nezavítá.
- Sémantický a správně strukturovaný kód se lépe spravuje.
- Pokud se vypnou kaskádové styly, web je i tak poměrně přehledný.
- V neposlední řadě, správně strukturovaný a sémantický web je důležitý pro zrakově postižené uživatele, kterým obsah stránek předčítá hlasová čtečka, jež se řídí právě HTML kódem.

4 PHP

PHP (rekurzivní zkratka *PHP: Hypertext Preprocessor*, česky „PHP: Hypertextový preprocesor“, původně *Personal Home Page*) je skriptovací programovací jazyk. Je určený především pro programování dynamických webových stránek a webových aplikací například ve formátu HTML, XHTML nebo WML (Wireless Markup Language). PHP lze použít i k tvorbě konzolových a desktopových aplikací.

PHP skripty lze tvořit v samostatných souborech nebo je vkládat přímo do html kódu. V obou případech musí mít soubory příponu ***.php.** Při požadavku na php stránku server prochází soubor a php skripty programově vyhodnocuje. Zpracování proběhne na straně serveru a klientovi se zpět odesílá „čisté“ HTML. (Obrázek 4)

Obrázek 4: Zpracování php skriptu na straně serveru.



4.1 INSTALACE LOKÁLNÍHO SERVERU

Pro vývoj webových portálů je možné využít celou řadu prostředí. V tomto studijním materiálu zaměřeného zejména na prezentaci praktických příkladů budeme využívat moduly softwarového balíku EasyPHP. EasyPHP obsahuje Apache (lokální server), PHP, MySQL (databáze) a PHPMyAdmin.

Poznámka: *vedle EasyPHP lze samozřejmě využít některé jiné nástroje, kdy jako příklad můžeme uvést PHPTriad, PHPHome, WAMP nebo jiné.*

Prvotním předpokladem instalace je stažení instalačního souboru, který je možné stáhnout z celé řady webových stránek. EasyPHP se neustále vyvíjí a existuje celá řada vývojových verzí, které je nutné vybrat tak, aby danou verzi podporoval operační systém. Princip činnosti je ve všech verzích stejný. Pro účely vysvětlení základních principů bude využita verze EasyPHP1.8.0.0, který lze snadno stáhnout například z http://www.slunecnice.cz/sw/easyphp/.

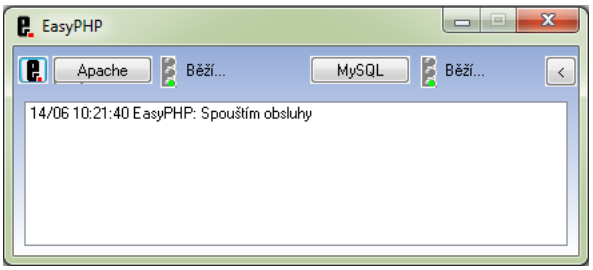
Po stažení instalačního souboru spustíme instalaci (od verze operačního systému Windows Vista je nutné spustit instalaci jako správce). Uvedená verze obsahuje češtinu, přičemž jazyk lze nastavit v prvním okně po spuštění instalace. Následně stačí kladně potvrdit všechny kroky instalace a instalaci dokončit.

Po správné instalaci a spuštění programu EasyPHP se v hlavním panelu Windows zobrazí ikona s blikajícím červeným čtverečkem (Obrázek 5):

Obrázek 5: Ikona informace o spuštěném programu EasyPHP.

Na ploše se dále zobrazí buď automaticky, nebo po spuštění příkazu **Konfigurace/EasyPHP** (lze vybrat po kliknutí pravým tlačítkem myši na ikonu z Obrázku 5) okno informující o běhu lokálního serveru Apache a databáze MySQL. Pokud jsou v daném okně (Obrázek 6) zobrazeny semafory se svítící zelenou a popiskem běží, je vše v pořádku. Pokud ne, lze provést konfiguraci nebo byla zvolena nevhodná verze software.

Obrázek 6: Okno s informací o správně běžícím lokálním serveru a databázi.



V některých případech se po instalaci zobrazí informace, že "**Server Apache nemůže být spuštěn: port už používá jiný Web server**". Toto lze velmi snadno vyřešit tím, že v okně **Konfigurace/EasyPHP** vypneme kontrolu portů pro obsluhy před startem.

4.2 FUNKČNOST LOKÁLNÍHO SERVERU

Po instalaci a spuštění lokálního serveru lze jeho funkčnost ověřit zadáním adresy **http://localhost** do prohlížeče. Měla by se objevit stránka z rootu lokálního serveru (slovo localhost znamená vždy něco jako "server na tomto počítači"). Pokud se neobjeví, můžeme vyzkoušet odkaz **http://localhost/home**. Rovněž tak lze vyzkoušet odkaz **http://127.0.0.1**.

Cesta ke složce, která představuje kořen dokumentů, je **C:\Program files (x86)\EasyPHP1_8\www**. Po umístění jakéhokoliv souboru *.html nebo *.php (a samozřejmě dalších) do této složky a zadání do prohlížeče adresy **http://localhost/*.html** se tento soubor v prohlížeči zobrazí.

4.3 ZÁKLADY PHP

PHP není nijak těžké pochopit a už se základy si lze vystačit. Umí ukládat, měnit a mazat data. Vše se odehrává na webovém serveru (kde jsou uloženy zdrojové kódy webových stránek). PHP skript se nejprve provede na serveru a potom odešle prohlížeči pouze výsledek (znamená to, že nejprve spočítá kolik je 300/30 a pak prohlížeči odešle jen číslo 10). Proto ve zdrojovém kódu najdete jen "10" (to je rozdíl oproti JavaScriptu, který počítá přímo v prohlížeči). Zdrojový kód PHP narozdíl od JavaScriptu a HTML nezobrazíte.

Pomocí PHP je možné na portálu vytvořit diskuzní fórum, knihu návštěv, počítadlo, anketu nebo graf. Navíc máte možnost propojit vaše stránky s databázemi, např. MySQL.

Je rozhodně alespoň jedna funkce PHP, která se hodí snad do každého webu. Na webových stránkách se obvykle opakují některé části, hlavička s odkazy, menu, patička. S PHP si můžete snadno vytvořit šablonu pro web, do které se budou vkládat soubory s menu, patičkou atd. Můžete tedy mít menu jen jednou zapsané a do dalších stránek ho pouze kopírovat. Až budete chtít menu změnit, bude to nesmírně jednoduché. Více v kapitole PHP menu.

Webová stránka s prvky PHP má nejčastěji koncovku .php. Avšak je možné použít i .php3, php4, php5 a phtml. Stávající verze PHP je 5. Nejlépe je používat koncovky .php. Použijete-li .php5, až vznikne nová verze, web bude působit zastarale.

Každý PHP skript musí být ohraničen značkami k tomu určenými. První možností je všechny příkazy skriptu zapsat mezi znaky '<?' a '?>'. Volba právě těchto znaků není náhodná. Elementy, které můžeme používat v jazyce HTML, jsou definovány pomocí jazyka SGML. Jazyk SGML zároveň definuje, že příkazy pro různé preprocesory, které dokument zpracovávají, mají být uzavřeny právě v této dvojici znaků. Následující skript vypíše text v uvozovkách (Řešený příklad 77).

ŘEŠENÝ PŘÍKLAD 77

```
<? echo „Můj první PHP skript.“; ?>
```

Druhá možnost je již trošku upovídanější. Pokud chceme PHP použít pro generování XML-dokumentů, musíme použít zahajovací značku '<?php'. XML je zjednodušená verze SGML, která v některých rysech reaguje na nově vzniklé požadavky. Jedním z nich je možnost zařazení příkazů pro více různých preprocesorů najednou. Proto je vždy nutné určit, pro jaký systém jsou vkládané příkazy určeny. K označení systému PHP slouží právě identifikátor `php` (Řešený příklad 78):

ŘEŠENÝ PŘÍKLAD 78

```
<?php echo „Můj druhý PHP skript.“; ?>
```

Vůbec nejdůležitější částí skriptu PHP je středník (;). Každou funkci, řádek, deklaraci musíte oddělit středníkem. Ne jako v JavaScriptu, který na něm tolik nelpí. Dá se to přirovnat k CSS (když vlastnosti neoddělíte středníkem, prohlížeč jim neporozumí). Vždy, když je skript nefunkční, zkuste zkontrolovat, kde jste zapomněli středník.

Užitečné věci ve skriptu tvoří komentáře. Tj. text, který je vidět jen ve zdrojovém kódu stránky, je určen autorovi, aby se v něm vyznal (Řešený příklad 79):

ŘEŠENÝ PŘÍKLAD 79

```
<?php
/*
víceřádkový
komentář
*/

// jednořádkový komentář
?>
```

Chcete-li na stránku vložit již hotový skript, užijete k tomu příkaz `include` (Řešený příklad 80):

ŘEŠENÝ PŘÍKLAD 80

```
<?php include("funkce.php"); ?>
```

Nyní jsme na stránku vložili obsah souboru `funkce.php`. Obsah tohoto souboru musí být ohraničen tagy značkami `<?php` a `?>`. Může vypadat takto (Řešený příklad 81):

ŘEŠENÝ PŘÍKLAD 81

```
<?php .... obsah v php .... ?>
```

Stejným způsobem je možné do stránek vkládat HTML, např. menu a mít tak celý web v šabloně, kterou lze snadno upravit.

Nyní jste se seznámili se základy PHP a nic vám nebrání začít opravdu programovat. Nejčastěji užívaným příkazem, je příkaz pro vypsaní textu `echo()`, nebo, pro úplnost, její méně užívaná obdoba `print()` (Řešený příklad 82):

ŘEŠENÝ PŘÍKLAD 82

```
<?php
echo ("toto je text" .
 "přes dvě řádky<br>");
echo ("vypíše text <br />");
echo "i bez závorek se to vypíše<br>";
print ("to samé </br>");
echo    ("<strong>též    můžete    používat    HTML    značky
</strong></br>");
echo 'můžete používat jednoduché uvozovky';
?>
```

Víceřádkový text je nutné rozdělit a spojit tečkou. Když vypisujeme text, můžeme použít libovolné značky jazyka (X)HTML. Výsledek předchozího skriptu v prohlížeči nabude tvaru:

```
toto je text
přes dvě řádky
vypíše text
i bez závorek se to vypíše
to samé
též můžete používat tagy

můžete používat jednoduché uvozovky
```

Je dobré se hned od začátku rozhodnout, jaké budete používat uvozovky - " nebo '. Lze je zaměnit a týká se to celého PHP.

Zrádné znaky jsou znaky, které ohrožují chod skriptu především `"  '  /  \`. Před tyto znaky je nutné vložit zpětné lomítko `\` a je po problémech. Pokud chcete zobrazit špičaté závorky < a >, musíte zapsat `<` a `>` (obdobně jako v HTML). Pokud chcete vypsat `<`, musíte do zdroje zapsat `&lt;`. Takže, když chci zapsat `<body bgcolor="red">` zapíši to takto (Řešený příklad 83) a (Řešený příklad 84):

ŘEŠENÝ PŘÍKLAD 83

```
echo ("&lt;body bgcolor=\"red\"&gt;");
```

Další příklady:

ŘEŠENÝ PŘÍKLAD 84

```
echo "Toto \"  je uvozovka";
echo "Toto \'  je také uvozovka";
```

Uvozovky lze požívat vnořeně. Proto je vhodnější používat primárně jednoduché uvozovky, kterými si usnadníte vypisování HTML. Vysvětlíme příkladem (Řešený příklad 85):

ŘEŠENÝ PŘÍKLAD 85

```
    echo '<ahref="http://www.tvorba-webu.cz/">Tvorba-
webu.cz>/a>'
    echo "<a href='http://www.tvorba-webu.cz/'>Tvorba-
webu.cz>/a>"
```

4.4 PROMĚNNÉ

Bez proměnných se neobejde žádný skript. Ani PHP není výjimkou. Proměnné deklarujeme pomocí znaku dolaru `$` (zapíšete ho buď pomocí [PRAVÝ ALT]+[ů] nebo [SHIFT]+[4/Č]) (Řešený příklad 86):

ŘEŠENÝ PŘÍKLAD 86

```
$dolar = "1$";
$den="středa";
```

Název proměnné se uvádí hned za značku $ (bez mezery). Zde jsme tedy zavedli proměnné: dolar, den a datum. Jejich hodnoty jsou uvedené za rovnítkem. Před rovnítkem a za ním nemusí, ale může být mezera (i více). Pokud proměnnou tvoří číslo, se kterým se bude později počítat (sčítat, dělit atd.), musí být zapsáno bez uvozovek, jinak by jej server pokládal za text. Název proměnné nesmí začínat číslem (jinak může čísla obsahovat), toto je chybné `$1promenna = "hodnota"`. Při zavádění a vyvolávání proměnných také musíte respektovat velikost písmen. Toto nefunguje:

```
$prom = "abc";
echo $Prom;
```

Proměnnou vypisujeme pomocí příkazu `echo()` (Řešený příklad 87):

ŘEŠENÝ PŘÍKLAD 87

```
$dolar="1$";
echo($dolar);
echo("$dolar");
echo("mám jen" . $dolar . "<br />");
echo("nemám ani $dolar");
```

Možností je spousta, ale nejlepší je první a třetí - vždy oddělit proměnnou od normálního textu, tím se vyvarujete případných komplikací. Obyčejný text se píše do uvozovek a od proměnné odděluje tečkou.

Pomocí operátorů můžeme proměnné měnit, násobit, porovnávat atd. (Tabulka 11).

Tabulka 11: Operátory v PHP

aritmetické operátory	
`$a+$b`	přičte k A B
`$a-$b`	odečte B od A
`$a/$b`	vydělí A proměnnou B
`$a*$b`	vynasobí A proměnnou B
přirovnávací operátory (zkracující zápisy)	
`$a++`	`$a = $a + 1`
`$a--`	`$a = $a - 1`
`$a += $b`	`$a = $a + $b`
`$a -= $b`	`$a = $a - $b`
`$a *= $b`	`$a = $a * $b`
`$a /= $b`	`$a = $a / $b`
porovnávací operátory	
`$a == $b`	`$a` má stejnou hodnotu jako `$b`
`$a != $b`	`$a` má jinou hodnotu než `$b`
`$a > $b`	`$a` je větší než `$b`
`$a < $b`	`$a` je menší `$b`
logické operátory	
`$a || $b`	vrátí 1 pokud alespoň jedna proměnná je pravdivá
`$a && $b`	vrátí 1 pokud jsou obě proměnné vyhodnoceny jako pravda
`!$a`	negace (opak) $a

Při logickém porovnávání má pravda `true` hodnotu `1`, nepravda `false` má hodnotu `0`.

Proměnné se v PHP vyskytují v několika typech (Tabulka 12):

Tabulka 12: Typy proměnných

Typ	Význam
String	Text, řetězec (sada znaků), příklad: `$retezec = "obyčejný text";`
Integer	Celé číslo, se kterými je možné pracovat, počítat, příklad: `$cislo = 2;`
Float, real nebo double	Desetinné číslo
Boolean	Logická proměnná, hodnota PRAVDA, NEPRAVDA (1, 0), zapisuje se TRUE nebo FALSE

Odlišné typy proměnných se odlišně zavádějí (Řešený příklad 88):

ŘEŠENÝ PŘÍKLAD 88

```
$retezec = "obyčejný text";
$celecislo = 2;
$desetinnecislo = 0.2;
$logickapromenna = TRUE;
```

4.5 VĚTVENÍ

Větvení je skupina příkazů, které rozhodují (porovnávají proměnné) a určují, jak se bude skript dál vyvíjet. Jsou to příkazy typu if, else, elseif a switch.

4.5.1 IF

Příkaz if zjišťuje, zda je daná podmínka pravdivá (Řešený příklad 89):

ŘEŠENÝ PŘÍKLAD 89

```
$a=1;
$b=2;
if($a == $b){
echo ("Jsou si rovny.");
echo ("A i b mají stejnou hodnotu");
}
```

Pokud by proměnná `$a` byla rovna `$b`, skript by vypsal: Jsou si rovny. `$a` i `$b` mají stejnou hodnotu. To se však nestane, jelikož daná podmínka: `$a==$b` je rovna 0 (není pravdivá). Více viz operátory v přechozí části.

4.5.2 IF ELSE A ELSEIF

Podmínka if - else opět rozhoduje, zda pro proměnnou platí zadané podmínky.

```
if (podmínka) proces1;
else proces2;
```

Platí-li podmínka provede se proces1, jinak se provede proces2. Příkazů může být i více.

```
if (podmínka)
 proces1;
elseif(podmínka2)
 proces2;
elseif(podmínka3)
 proces3;
else
 poslední proces;
```

Příklad (Řešený příklad 90):

ŘEŠENÝ PŘÍKLAD 90

```
$a = 5;
if ($a==4)
 echo("4");
 elseif ($a<4)
      echo("menší než 4");
 else{
```

```
if($a>4 && $a<7){
     echo("číslo je větší než 4 a menší než 6");
}
else echo($a);
}
```

Rozhodujeme zda `$a` je rovna 4. Pokud není, nejprve rozhodne, zda je menší než 4, jinak rozhoduje, jestli je větší než 4 a zároveň menší než 7. Pokud ani tato podmínka neplatí, skript vypíše hodnotu proměnné.

Nyní se podíváme, jak lze pomocí PHP vytvářet celé stránky (Řešený příklad 91):

ŘEŠENÝ PŘÍKLAD 91

```
<html>
<head>
<title>generování HTML</title>
<head>
<body>
<?php $action = $_GET["action"]; ?>

<?php if($action == "a1"):?>
<strong>Funkce PHP</strong>
<?php elseif($action == "a2"): ?>
<em>Proměnné</em>
<?php elseif($action == "a3"): ?>
<?php echo ("<strong>Cykly</strong>
<strong>větvení</strong>"); ?>
<?php endif; ?>

</body>
</html>
```

A k čemu je to dobré? Můžete v jedné stránce zobrazovat desítky stránek, které se budou lišit v URL jen předávanou proměnnou. Skript se například jmenuje `skript.php`, když zavoláme URL `skript.php?action=a1`, zobrazí se obsah u první podmínky, když zavoláme `skript.php?action=a2`, zobrazí se obsah u druhé podmínky atd.

4.6 CYKLY

Cykly se používají, chceme-li opakovat určitou činnost několikráte za sebou (opakovaně provádět stejnou proceduru, postupně ověřovat data nebo generovat informace, apod.)

4.6.1 DO WHILE

Chceme zapsat čísla od od 1 do 100. Je zbytečné zapsat 1,2,3... Poslouží nám jednoduchý skript (Řešený příklad 92):

ŘEŠENÝ PŘÍKLAD 92

```
$c = 1;
do {
echo($c . " ");
$c++;
}
while($c<101);
```

`$c` je proměnná. Cyklus `do {proces} while(podmínka)` vždy vypíše hodnotu proměnné `$c`. Zároveň k ní vždy přičte 1 pomocí `$c++`. Nakonec jsme příkazem `while`(podmínka) cyklus omezili, cyklus se bude provádět, pouze když je proměnná menší než 101. V momentě, kdy není menší, se skript přeruší.

Stejným způsobem můžete vytvářet i jiné skripty, například pracovat s polem nebo databází. Projíždět postupně jednotlivé položky a vypisovat jejich obsah.

4.6.2 WHILE

Příkaz `while`(podmínka) je možné použít samostatně (Řešený příklad 93).

ŘEŠENÝ PŘÍKLAD 93

```
$c = 1;
while($c<101){
echo($c . " ");
$c++;
}
```

Výsledek je stejný, pouze pokud `$c` přiřadíte hodnotu 101, příkaz do `while` vypíše 101, protože podmínka je definována až po procesu (skript nezná podmínku a tak napíše 101, teprve poté kontroluje podmínky). Příkaz `while` 101 nezapíše, protože podmínka předchází proces. Příkaz do `while` jednou provede proces, když už podmínka neplatí. Na tento rozdíl je třeba si vždy dát dobrý pozor a ověřit skript, když se dostane k podmínce.

4.6.3 FOR

Posledním příkazem je příkaz `for`. Syntaxe: `for`(inicializace proměnné; podmínka; operace) {skript} (Řešený příklad 94).

ŘEŠENÝ PŘÍKLAD 94

```
for($c=1; $c<101; $c++){
echo($c." ");
}
```

Příkaz `for` se užívá především, když víme, kolikrát chceme proces uskutečnit. Počet je konečný. U `do while` a `while` ověřujeme proměnnou, která se v procesu mění. U `for` zavádíme přímo proměnnou určenou k tomu, aby se určil počet procesů.

4.6.4 BREAK

Příkaz `break` slouží k zlomení, přerušení cyklu ještě v době, kdy platí podmínky (Řešený příklad 95):

ŘEŠENÝ PŘÍKLAD 95

```
$a = 0;
while ($a<10) {
if($a == 5) break;
$a++;
echo($a . "<br>");
}
```

Cyklus umožňuje zapsat 10 čísel. Bude-li se však `$a` rovnat 5, cyklus se ukončí.

4.6.5 CONTINUE

Příkaz `continue` slouží k tomu, aby se cyklus vrátil zpět na začátek, pokud dojde k porušení některé podmínky (Řešený příklad 96):

ŘEŠENÝ PŘÍKLAD 96

```
$a = 0;
while ($a<10) {
if($a == 5) break;
if($a == -1) continue;
$a++;
echo($a . "<br>");
}
```

Pokud se nyní `$a` bude rovnat 5 dojde k přerušení cyklu, pokud se `$a` bude rovnat -1 cyklus se zopakuje.

Obecně lze říct, že cykly a větvení jsou základními strukturami, které v PHP budete používat určitě často. Tyto struktury existují i v jiných programovacích jazycích.

4.7 FUNKCE

Funkce je jakousi zkratkou pro větší počet procesů, pokud je nechceme vypisovat celé. Kromě předem zavedených a vyhrazených funkcí PHP, si můžete deklarovat své vlastní (Řešený příklad 97):

ŘEŠENÝ PŘÍKLAD 97

```
function napis(){
echo("ahoj");
}
```

`function` deklaruje funkci, `napis` je název funkce, (názvy pište bez diakritiky). U názvu funkce je třeba si dát pozor, aby se její název nekryl s názvem již zavedené funkce PHP (nemůžete použít např. echo, return...). V závorkách jsou uvedené argumenty funkce. Funkce může být i bez argumentu. Složené závorky`{ }` vymezují procesy funkce (složené závorky zapíšete pomocí SHIFT + [ú] a SHIFT + [)] nebo pomocí PRAVÝ ALT + [F] a PRAVÝ ALT + [G]).

Kdykoli zavoláme tuto funkci `skript` vypíše `ahoj`. Taková funkce se nám může hodit, pokud třeba chceme opakovaně vypisovat nějaký delší text, nebo část zdrojového kódu.

4.7.1 VYVOLÁNÍ FUNKCE

Nyní jsme deklarovali funkci, ale teď ji chceme vyvolat (Řešený příklad 98):

ŘEŠENÝ PŘÍKLAD 98

```
napis();
```

Zapíšeme pouze název funkce a závorky. Skript vypíše `'ahoj'`.

4.7.2 FUNKCE BEZ ARGUMENTU

Je již zmíněná funkce `napis()`, která nemá uvnitř závorek žádný argument(proměnnou).

4.7.3 FUNKCE S ARGUMENTEM

Kdybychom chtěli někdy napsat `'ahoj'`, jindy `'dobrý den'`, nebo něco úplně jiného, použijeme funkci s argumentem (Řešený příklad 99).

ŘEŠENÝ PŘÍKLAD 99

```
function napis($jmeno){
echo($jmeno);
}
napis("dobrý večer"); //napíše dobrý večer
napis("ahoj"); //napíše ahoj
napis("dobré ráno"); //napíše dobré ráno
```

Deklarace je úplně stejná až na argument `$jmeno`. Ten zavádí proměnnou `$jmeno`. Když jsme vyvolali funkci a jako její argument jsme uvedli `"dobrý večer"` a `"ahoj"`, byly tyto hodnoty přiřazeny k proměnné `$jmeno`, následně je funkce zapíše.

4.7.4 FUNKCE VRACEJÍCÍ HODNOTU, PŘÍKAZ RETURN

Můžete také zavést funkci, která vrací hodnotu (Řešený příklad 100):

ŘEŠENÝ PŘÍKLAD 100

```
function vrat($cislo){
return $cislo*2;
}
echo(vrat(20));
echo(vrat(100)/vrat(10));
```

Funkce `vrat();` příkazem `return` vrací `$cislo*2` (tedy vrací dvojnásobek zadané hodnoty). S danou funkcí lze potom nakládat jako s proměnnou, počítat s ní, vypisovat ji atd.

4.7.5 GLOBÁLNÍ PROMĚNNÉ

Pracujeme-li uvnitř funkce s proměnnými, které jsou definované mimo tělo funkce, je třeba k nim přistupovat trochu odlišně (Řešený příklad 101):

ŘEŠENÝ PŘÍKLAD 101

```
<?php
$name = "Simon";
function jm(){
echo $GLOBALS['name'];
}
jm();
?>
```

Pomocí `$GLOBALS` přistupujeme k proměnné deklarované vně funkce.

4.8 POLE

Pole (Array), to je skupina prvků, které spolu souvisí. Pole je třeba abeceda. Pole jsou základním prvkem PHP, ve složitějších aplikacích a skriptech se bez nich neobejdete. Pole nesmírně usnadní práci s databází (MySQL), ale lze ho využít i jinde.

4.8.1 VYTVOŘENÍ POLE

Pole se tvoří příkazem `array("prvek0", "prvek1", "prvek2")`, každý prvek má své číslo - ten první (`prvek0`) má číslo nula (Řešený příklad 102).

ŘEŠENÝ PŘÍKLAD 102

```
$pole = array ("mrkev", "celer", "brambory");
echo($pole[0]);  //vypíše 'mrkev';

$zahrada = array (3 => "jablon", "hrusen");
echo($zaharda[4]);  //vypíše 'hrusen';

$sklenik[1] = "redkvicky"; //postupné přiřazování prvků i s
indexem(1)
$sklenik[3] = "fazole";
```

Pole se ukládají pod proměnné(`$pole`, `$zahrada` a `$sklenik`). V prvním případě jsme vytvořili `$pole`, jehož položkami jsou mrkev, celer a brambory. Následně `echo($pole[0]);` jsme vypsali nultou položku pole. Pak jsme zavedli pole `$zahrada`, do kterého jsme zařadili prvek jablon pod číslo 3 a hrusen s číslem následujícím, tedy 4. Do posledního pole `$sklenik` jsme postupně zařadili prvky redkvicky a fazole pod čísla 1 a 3.

4.8.2 *VÝPIS PRVKŮ POLE*

Syntax: `echo($pole[cislo prvku])`. To je nejjednodušší způsob, jak vypsat prvek. Druhá možnost je prvky pojmenovat a pak odkazovat pomocí jmen (Řešený příklad 103):

ŘEŠENÝ PŘÍKLAD 103

```
    $pole["br"] = "brambor";
    $pole["kv"] = "kvetak";
    echo("nemám rád ".$pole["br"]." ani ".$pole["kv"]); //nemám
rád bramobory ani květák

    $knihy = array("js" => "JavaScript v příkladech", "php" =>
"PHP pro profesionály");
    echo($knihy["js"]);
```

4.8.3 *PROCHÁZENÍ POLEM*

Procházení polem znamená postupné vybrání všech prvků. Metoda `count($pole)` vrací počet prvků pole (Řešený příklad 104).

ŘEŠENÝ PŘÍKLAD 104

```
    $dodelat = array("PHP", "CSS", "SQL", "XML");
    for ($p = 0; $p < count($dodelat); ++$p){
    echo ("je třeba dodělat ".$dodelat[$p]."<br>");
    }
```

Tento skript postupně vypíše každý prvek pole `$dodelat`. Metodu `count($pole)` lze použít, pouze pokud jsou prvky číslované, pokud jsou pojmenované, je třeba použít `each()`, `list()` a `reset()`(Řešený příklad 105):

ŘEŠENÝ PŘÍKLAD 105

```
    $pole[PHP]="Hotové";
    $pole[CSS]="Rozepsané";
    $pole[XML]="Hotové";
    $pole[SQL]="Nezačaté";
    reset($pole);

    while(list($index, $stav) = each($pole)){
    echo ($index." - ".$stav."<br>");
    }
```

Metoda `reset()` nastavuje ukazatel na první prvek pole. Metoda `list()` bere prvky z pole a jeho číslo uloží jako `$index` a jeho hodnotu jako proměnnou `$stav`. Metoda `each()` značí, že se tak má provést u každého prvku.

4.9 PHP MENU

Může nastat situace, kdy do každé stránky vkládáte stejný text (objekt), například menu, patičku. Pomocí PHP lze snadno vkládat do každého souboru jiný soubor.Veškeré opakující se prvky je vhodné vkládat pomocí PHP. Vytvořte stránku - třeba `menu.html` (ve které budou jen odkazy a prvky společné pro všechna menu, žádné `<body>`,`<head>` ani `<html>`). Tuto stránku vložíte pomocí PHP do dokumentu, např. (Řešený příklad 106):

ŘEŠENÝ PŘÍKLAD 106

```
<?php include_once("menu.html") ?>
```

Nyní se obsah souboru `menu.html` vloží do každého dokumentu, ve kterém bude tento zdrojový kód. Když budete chtít menu změnit, změníte pouze soubor `menu.html` a změna se projeví ve všech dokumentech. Samozřejmě všechny dokumenty musejí mít koncovku `.php` a na serveru musí být podpora PHP.

Obdobným způsobem vkládejte menu, patičky, reklamy, hlavičky a další. Ušetří vám to práci a web bude nesmírně jednoduchý. Ukázkový příklad použití `include` bude v kapitole 8.

4.10 ÚLOHY K ZAMYŠLENÍ

- Máme tři čísla *a*, *b*, *c*. Tato čísla odpovídají délkám stran nějakého trojúhelníku. Jak zkonstruujeme v PHP podmínku, která bude pravdivá pouze v případě, že trojúhelník je pravoúhlý?
- Využijte cyklů v PHP k vypsání prvních *n* sudých přirozených čísel.
- Vykreslete čtvercovou tabulku o libovolném rozměru pomocí spojení HTML a PHP jazyků. Vypište do každé buňky této tabulky její pozici.

5 VYUŽITÍ PHP PRO ZPRACOVÁNÍ FORMULÁŘŮ

Abychom vytvořili prakticky použitelný portál, potřebujeme do hry aktivně zapojit uživatele. Portál musí od uživatele získat vstupní informace, aby mu mohl poskytnout požadované údaje. Jazyk HTML proto obsahuje podporu formulářů. Součástí webové stránky se tak stanou různá vstupní pole a tlačítka — vznikne nám něco velice podobné dialogovým oknům. Uživatel do formuláře zadá data, která jsou odeslána skriptu na serveru ke zpracování. Skript získaná data nějakým způsobem zpracuje a poskytne uživateli odpověď. Vidíme, že bez formulářů by uživatel neměl příliš šanci aktivně vstoupit do dění. V jedné z předchozích kapitol jsme se již s tvorbou formulářů v jazyce HTML seznámili. Nyní se budeme věnovat zpracováním dat z formulářů právě pomocí skriptů v PHP.

Pro vložení formuláře do stránky slouží HTML element `FORM`. Pro správnou funkci celého formuláře jsou nezbytné dva atributy — `ACTION` a `METHOD`. Atribut `ACTION` určuje URL skriptu, který se použije pro zpracování dat z formuláře. V PHP je obvykle stránka s formulářem i skript v jednom adresáři a s výhodou použijeme samotné jméno skriptu, které v tomto případě odpovídá relativnímu URL. Pomocí atributu `METHOD` určujeme způsob, jakým budou data z formuláře předána zpět serveru. K dispozici jsou dvě metody — `GET` a `POST`. Metoda `GET` je vhodná pro přenášení kratších dat z malých formulářů, protože data formuláře se připojí na konec URL ukazujícího na obslužný skript. Metoda `POST` se hodí pro odesílání větších formulářů, které obsahují mnoho dat. Údaje z formuláře jsou v tomto případě přenášeny v těle HTTP požadavku. Z hlediska psaní skriptů v PHP není mezi oběma metodami žádný rozdíl - data jsou vždy přístupná stejným způsobem. Výše zmíněné poznatky shrneme a ukážeme si, jak má vypadat základ každého formuláře (Řešený příklad 107):

ŘEŠENÝ PŘÍKLAD 107

```
<FORM ACTION="URL skriptu zpracovávajícího skriptu" METHOD="metoda"
Definice formuláře
</FORM>
```

PHP bylo navrženo speciálně jako jazyk pro tvorbu dynamicky generovaných webových stránek a v mnoha ohledech se tomuto požadavku přizpůsobuje. Říkali jsme si, že data mohou být skriptu odeslána metodami GET a POST. PHP pak automaticky nastaví odpovídající proměnné. Kromě toho máme k dispozici dvě asociativní pole `$HTTP_GET_VARS` a `$HTTP_POST_VARS`, která obsahují všechny proměnné odeslané metodami GET a POST. Tímto způsobem můžeme zpracovávat data i z formulářů, u kterých neznáme jména vstupních polí.

Nyní si využití PHP pro zpracování formulářů ilustrujeme na jednoduchém příkladu kalkulačky, která umí jen jednu operaci, a to sčítání. Do formuláře vložíme dvě vstupní textová pole s názvy `Scitanec1` a `Scitanec2` . Dále přidáme tlačítko pro odeslání údajů. Parametr action nastavíme na skript `vypocet.php`, který součet zadaných hodnot provede a vypíše. Údaje pošleme na server metodou `get` (Řešený příklad 108).

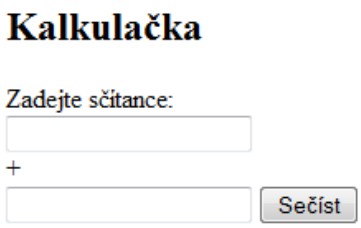
ŘEŠENÝ PŘÍKLAD 108

```
<html>
<head>
<title>Příklad zadávání údajů</title>
</head>
<body>
```

```
<h2>Kalkulačka</h2>
  Zadejte sčítance:
<form name="Kalkulator" action="vypocet.php" method="get">
<input type="text" name="Scitanec1" value="" />
<br />+<br />
<input type="text" name="Scitanec2" value="" />
<input type="submit" value="Sečíst" />
</form>
</body>
</html>
```

Výše uvedený kód vytvoří takovýto formulář (Obrázek 7):

Obrázek 7:Formulář jednoduché kalkulačky



Pokud zadáme do vstupních polí hodnoty např. 10 a 50, v URL se objeví `http://localhost/vypocet.php?Scitanec1=10&Scitanec2=50`. Následující skript, který uložíme jako `vypocet.php`, poslaná data zpracuje (sečte) a vypíše výsledek (Řešený příklad 109).

ŘEŠENÝ PŘÍKLAD 109

```
<?php
      $Scitanec1=$_GET["Scitanec1"];
      $Scitanec2=$_GET["Scitanec2"];
    ?>
<h2>Kalkulačka</h2>
    První sčítanec: <?php echo ($Scitanec1); ?><br />
    Druhý sčítanec: <?php echo ($Scitanec2); ?><br />
    ============================== <br />

Součet: <?php echo ($Scitanec1 + $Scitanec2); ?><br />
<p><a href="kalkulacka.html">Zpět k zadání</a></p>
```

V případě, že bychom parametr method formuláře nastavili na post, v URL se objeví pouze http://localhost/vypocet.php. Odeslaná formulářová data bychom potom přistupovali pomocí příkazu `$_POST`, jako v následujícím příkladu (Řešený příklad 110):

ŘEŠENÝ PŘÍKLAD 110

```
<?php
      $Scitanec1=$_POST["Scitanec1"];
      $Scitanec2=$_POST["Scitanec2"];
    ?>
<h2>Kalkulačka</h2>
    První sčítanec: <?php echo ($Scitanec1); ?><br />
    Druhý sčítanec: <?php echo ($Scitanec2); ?><br />
    ============================ <br />
    Součet: <?php echo ($Scitanec1 + $Scitanec2); ?><br />
<p><a href="kalkulacka.html">Zpět k zadání</a></p>
```

5.1 ÚLOHY K ZAMYŠLENÍ

Zamyslete se nad tím, jak by se kalkulačka dala rozšířit na více aritmetických operací, např. děleno, krát a mínus. Zkuste do stávajícího kódu přidat další prvek, který umožní volbu aritmetické operace. Do zpracovávajícího skriptu pak zakomponujte odpovídající reakci na volbu aritmetické operace.

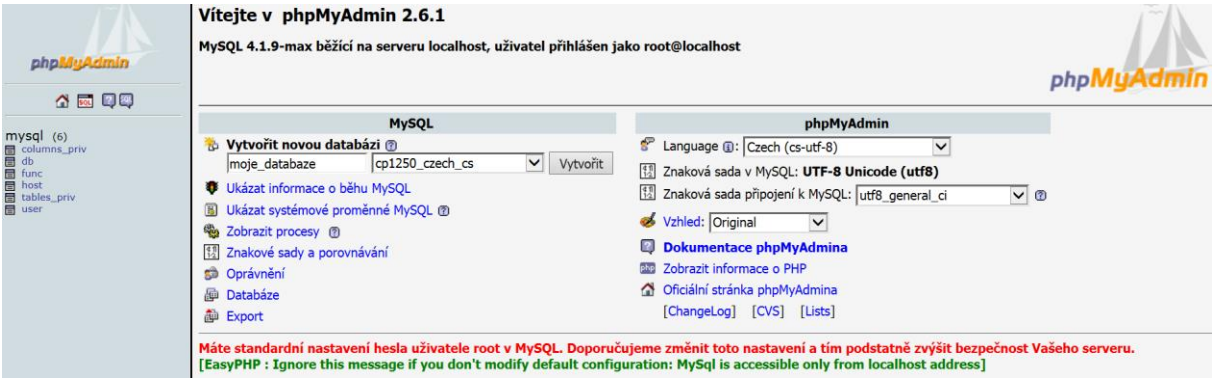
6 MYSQL

MySQL je relační databázový systém typu DBMS (Database Management System), vlastněný společností Oracle. Každá databáze v MySQL je tvořena z jedné nebo více tabulek, které mají řádky a sloupce. V řádcích rozeznáváme jednotlivé záznamy (řádek=>záznam). Sloupce mají jméno a uvozují datový typ jednotlivých polí záznamu (sloupec=>pole). Práce s databázemi, tabulkami a daty se provádí pomocí příkazů, respektive dotazů. Dotazy vycházejí z deklarativního programovacího jazyka SQL (Structured Query Language). Systém MySQL je využitelný v ADO.NET, C, C++, JDBC, ODBC, Perl, PHP, Python Ruby. Systém MySQL je šířen jako Open source. Pro vzdálenou správu MySQL přes WWW je vhodný phpMyAdmin nebo český Adminer. PhpMyAdmin je součástí balíku EasyPHP, jehož instalace byla popsána v kapitole 4. Programový systém phpMyAdmin je nástroj napsaný v jazyce PHP umožňující jednoduchou správu obsahu databáze MySQL prostřednictvím webového rozhraní. V současné době umožňuje vytvářet/rušit databáze, vytvářet/upravovat/rušit tabulky, provádět SQL příkazy a spravovat klíče.

6.1 VYTVOŘENÍ DATABÁZE

Do správy databáze se dostaneme přes program EasyPHP z nabídky **Konfigurace/PhpMyAdmin**. Zobrazí se nám rozhraní, ve kterém můžeme začít s databází pracovat. Prvním krokem je vytvoření nové databáze. Její název vepíšeme do pole **Vytvořit novou databázi**. V tomto příkladu vytvoříme databázi s názvem **moje_databaze** (viz Obrázek 8). U databází a všech databázových objektů je nutné věnovat velkou pozornost nastavení znakových sad. Budeme-li chtít využívat v záznamech českou diakritiku, musíme zvolit správné kódování, které by mělo být v celé databázi jednotné. V našem případě zvolíme **cp1250_czech_cs**.

Obrázek 8: Rozhraní phpMyAdmin.



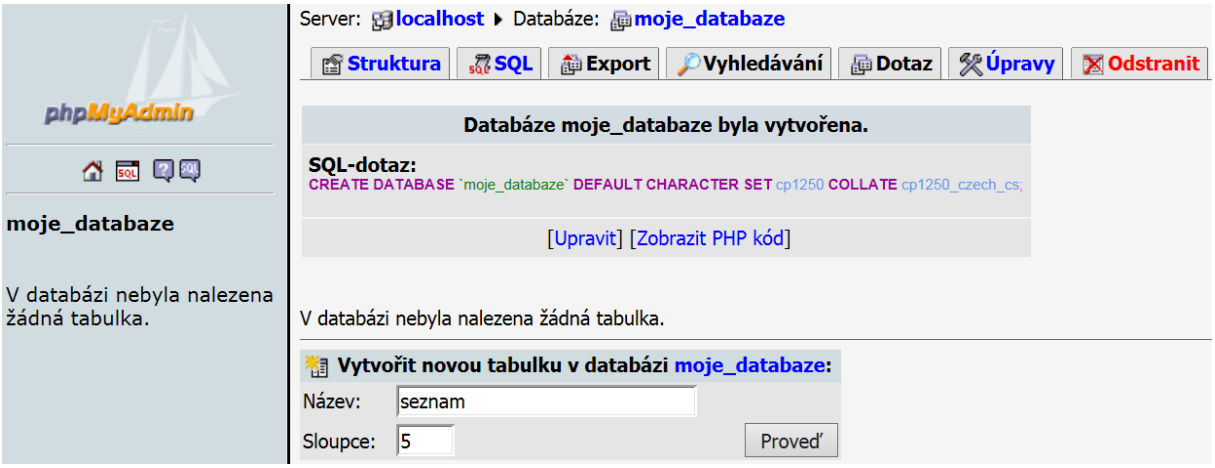
Po zadání všech hodnot klepneme na tlačítko **Vytvořit**. Tím se nám vytvoří nová prázdná databáze. Jak je a bude vidět počínaje Obrázkem 9, paralelně při provádění jednotlivých akcí se nám generuje SQL kód pro jejich provedení.

6.1.1 TVORBA TABULKY

Základním objektem v databázích je tabulka. Tabulku můžeme vytvořit několika způsoby. Prvním z nich je varianta, kdy vytvoříme prázdnou tabulku (bez záznamů) obsahující pouze názvy příslušných polí (sloupců) a to pomocí webového rozhraní phpMyAdmin. Druhým způsobem je vytvoření tabulky pomocí příkazu SQL pro tvorbu tabulky a třetím je import existující tabulky v podobě textového nebo jiného souboru.

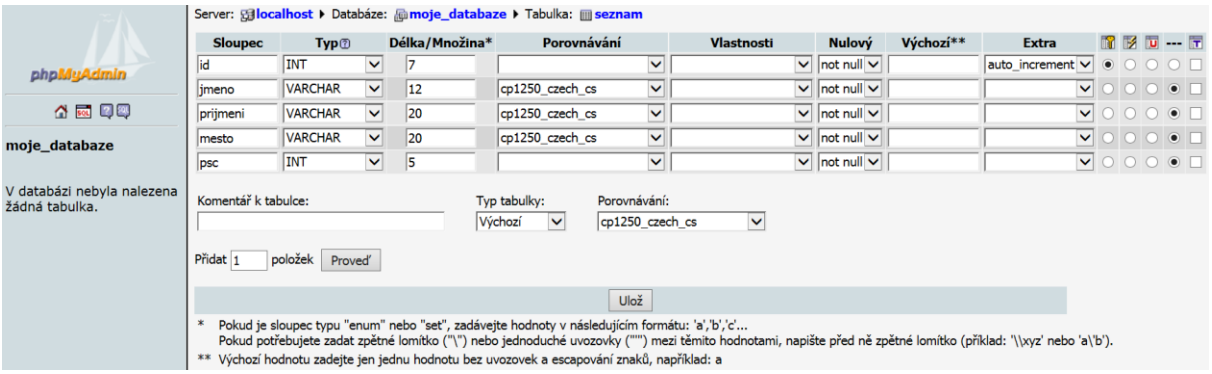
Jako základní si v této první části ukážeme tvorbu tabulky pomocí webového rozhraní phpMyAdmin.

Obrázek 9: Okno databáze s názvem moje_databaze.



Předpokládejme, že vytvoříme tabulku s názvem **seznam**, která bude obsahovat pole **id**, **jmeno**, **prijmeni**, **město** a **psc**. V okně obsahujícím, mimo jiné, pole **Vytvořit novou tabulku v databázi moje_databaze** (Obrázek 9) tedy vyplníme název tabulky (**seznam**) a počet sloupců (**5**). Po klepnutí na tlačítko **Proveď** se dostaneme do okna pro editaci jednotlivých polí tabulky (Obrázek 10).

Obrázek 10: Editace polí tabulky.

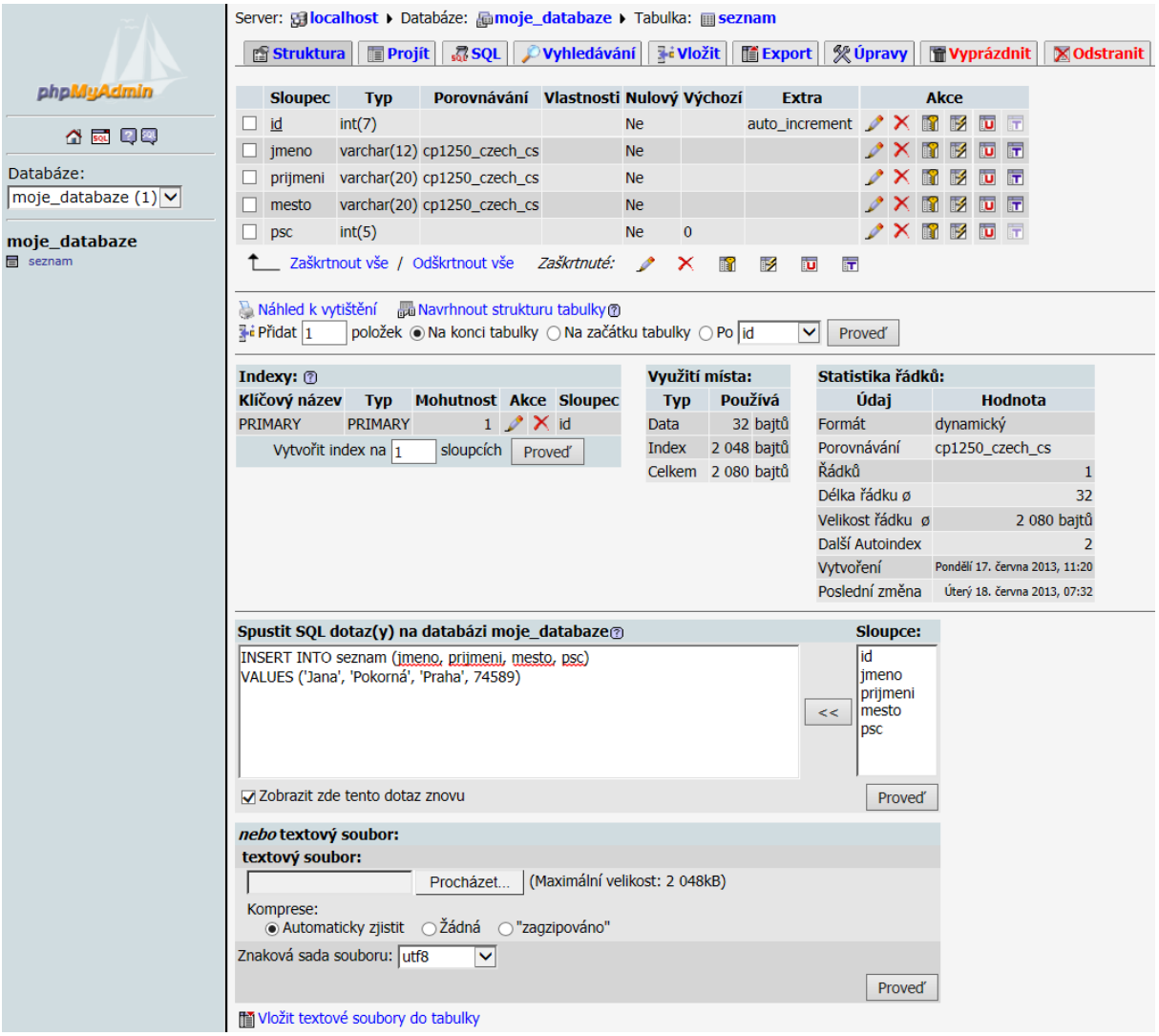


U každého pole tabulky budeme pro naše potřeby jako klíčové editovat:

- Sloupec (název pole);
- Typ (datový typ) – u textových polí to je nejčastěji TEXT nebo VARCHAR, u číselných polí INT (celá čísla) nebo DOUBLE (reálná čísla) apod.;
- Délka/Množina – nastavujeme nejdelší možnou délku, kterou je možné vložit jako hodnotu záznamu v daném poli;
- Porovnání – představuje znakovou sadu;
- Extra – v našem případě pole id představuje primární klíč, kdy jako hodnoty pole bude vkládána posloupnost celých čísel. Toto pole bude nastaveno jako primární klíč, což zajistíme tak, že na konci řádku přepneme přepínač na první pozici.

Po nastavení uvedených hodnot a kliknutí na tlačítko **Ulož** dojde k vytvoření tabulky **seznam** a dostaneme se do základního rozhraní, ve kterém je možné provádět celou řadu úprav vztahujících se k dané tabulce. (Obrýzek 11)

Obrázek 11: Okno editace tabulky.



6.1.2 *VLOŽENÍ ZÁZNAMŮ DO TABULKY*

Do tabulky můžeme vložit záznamy ručně, pomocí SQL příkazu, importem z externího souboru nebo například přebírám hodnot z webových formulářů, což je jedním z primárních obsahových zaměření tohoto studijního textu a této problematice bude věnována velká část výkladu v následujících kapitolách. Prohlédnout záznamy v tabulce lze pomocí karty **Projít**. Ručně lze záznamy do tabulky vkládat pomocí karty **Vložit** (Obrázek XY). V případě, že chceme využít SQL dotaz, ten můžeme vložit do pole **Spustit SQL dotaz(y) na databázi nazev_databaze** (nebo pomocí karty SQL). Jeden z možných příkladů již je zobrazený na Obrázku 11, kde je vepsán SQL dotaz přidávající do tabulky seznam jeden záznam.

ŘEŠENÝ PŘÍKLAD 111

```
INSERT INTO seznam (jmeno, prijmeni, mesto, psc)
VALUES ('Jana', 'Pokorná', 'Praha', 74589)
```

Jak je patrné z Řešeného příkladu 111, za klíčovými slovy INSERT INTO se udává jméno tabulky, do které chceme data vkládat (je-li tabulka v jiném databázovém schématu, použijeme tvar jméno_schématu.jméno_tabulky). Seznam sloupců uvedený v závorkách je

nepovinný. Za klíčovým slovem `VALUES` je uveden seznam hodnot, které vkládáme. Pořadí hodnot musí přesně odpovídat pořadí sloupců v tabulce. Pokud toto pořadí hodnot nechceme dodržet, pak musíme uvést část jména_sloupců za klíčovým slovem INTO, ve které budeme vlastně specifikovat, v jakém pořadí vkládáme jednotlivé hodnoty. Pořadí sloupců tedy musí odpovídat pořadí hodnot.

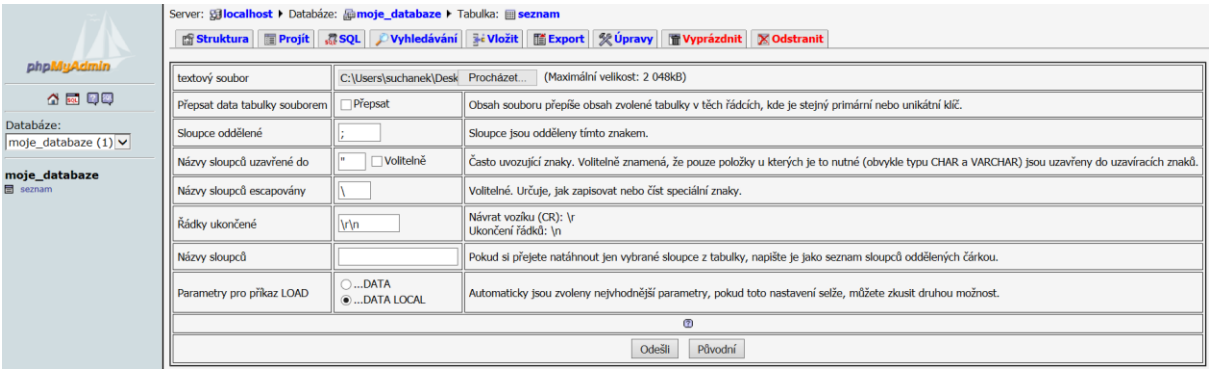
Třetí možností je import dat z externího souboru. K tomuto účelu samozřejmě potřebujeme soubor se správnou strukturou. V tomto příkladu zůstaneme u nejjednodušší varianty, kdy budeme importovat soubor *.txt (konkrétně se bude jednat o soubor data.txt). Soubor data.txt bude obsahovat:

ŘEŠENÝ PŘÍKLAD 112

```
;Jan;Novák;Ostrava;75412
;Hana;Drobná;Karviná;73586
;Jiří;Hanák;Opava;76512
;Lenka;Pokorná;Brno;71245
```

Import dat do dané tabulky lze provést pomocí odkazu **Vložit textové soubory do tabulky**, umístěném zcela ve spodní části okna pro editaci tabulky (Obrázek 11). Po klepnutí na tento odkaz se nám zobrazí okno, ve kterém nastavíme potřebné parametry. Primárním je samozřejmě výběr konkrétního souboru (Obrázek 12).

Obrázek 12: Okno výběru textového souboru pro import dat do tabulky.



Po klepnutí na tlačítko **Odešli** se akce provede a záznamy se do tabulky vloží.

6.2 PŘÍKAZY SQL

V této kapitole si uvedeme přehled některých důležitých příkazů jazyka SQL, které budeme dále potřebovat.

6.2.1 VYTVOŘENÍ TABULKY

Pro vytvoření tabulky se používá příkaz `CREATE TABLE` v následující syntaxi:

```
CREATE TABLE [název_tabulky]
(název_sloupce1 typ
[, název_sloupce2 typ, název_sloupce3 typ, …])
```

Konkrétně je daná struktura využita v Řešeném příkladu 113):

ŘEŠENÝ PŘÍKLAD 113

```
CREATE TABLE telefony (
    id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    jmeno VARCHAR(40),
    telefon VARCHAR(14),
    email VARCHAR(60),
    pracoviste INT)
```

Tento příklad vytvoří prázdnou tabulku s následující strukturou (Tabulka 13).

Tabulka 13: Nová tabulka telefony

id	jmeno	telefon	email	pracoviste

Jak jste si všimli, při vytváření tabulky je za každým názvem sloupce uveden jeho typ. Každý typ sloupce se může skládat z několika vlastností. První vlastností by měla být definice datového typu sloupce. My se pro naše účely omezíme pouze na dva datové typy – `INT` a `VARCHAR`. Pokud chceme do sloupce ukládat pouze celá čísla, použijeme datový typ `INT`. Pokud budeme chtít do daného sloupce ukládat data nečíselného charakteru, např. jméno, adresa, titul, atp., použijeme typ `VARCHAR`, který nám umožní vkládat do sloupce libovolné textové řetězce. V případě použití tohoto typu je potřeba ještě uvést maximální délku řetězce, tzn. víme-li, že do sloupce `jmeno` budeme zapisovat jména o délce maximálně 40 znaků, definujeme datový typ tohoto sloupce jako `jmeno(40)`.

Další vlastnosti sloupce budeme definovat pouze u prvního sloupce, který budeme vždy nazývat `id` a vždy bude sloužit jako primární klíč (viz příklad výše), což definujeme pomocí klíčových slov `PRIMARY KEY`. Vlastnost `NOT NULL` zajišťuje, že u každého nového záznamu (řádku) vkládaného do tabulky bude tento sloupec obsahovat neprázdnou (not null) hodnotu. Vlastnost `AUTO_INCREMENT` nám usnadňuje práci s tímto sloupcem a při vkládání nového záznamu do tabulky se o hodnotu v tomto sloupci nemusíme starat – nastaví se totiž automaticky a to tak, že se zjistí prozatímní nejvyšší hodnota v tomto sloupci (označme jako *n*) a pro nový záznam se hodnota v tomto sloupci nastaví jako *n* + 1, např. je-li nejvyšší hodnota ve sloupci `id` číslo 45, hodnota v `id` pro nový záznam se nastaví jako 46.

6.2.2 VLOŽENÍ ZÁZNAMU DO TABULKY

V předchozí části textu jsme se již vložení záznamu do tabulky v databázi věnovali v souvislosti s prostředím phpMyAdmin. Pro jistotu si jej však ještě jednou zopakujeme na příkladu s tabulkou `telefony`. Příkaz INSERT INTO má obecně následující syntaxi:

```
INSERT INTO [název_tabulky] (sloupec, ...)
VALUES (hodnota, ...)
```

Pro tabulku `telefony` se může použít takto (Řešený příklad 114):

ŘEŠENÝ PŘÍKLAD 114

```
INSERT INTO telefony (jmeno, telefon, email, pracoviste)
VALUES ('Jan Novák', '+420212131415', 'novak@email.cz', '25'
)
```

Tento příklad vloží do tabulky `telefony` jeden nový záznam. Výsledkem je následující tabulka (Tabulka 14).

Tabulka 14: Tabulka telefony s jedním novým záznamem

id	jmeno	telefon	email	pracoviste
1	Jan Novák	+420212131415	novak@email.cz	25

Pozn.: Všimněte si automaticky nastavené hodnoty ve sloupci `id`. Pokud je tabulka prázdná, hodnota n (viz vlastnost `AUTO_INCREMENT`) je rovna 0. Novému záznamu je tedy do sloupce `id` automaticky přiřazena hodnota 1.

6.2.3 ZMĚNA HODNOT V TABULCE

Pro změnu hodnot v tabulce se používá příkaz `UPDATE`, který má následující syntaxi:

```
UPDATE tabulka SET sloupec=hodnota [,jiný sloupec=hodnota...]
WHERE něco='něco'
```

Nyní si přestavme situaci, kdy je potřeba přečíslovat některá pracoviště, např. pracoviště č. 25 je potřeba přečíslovat na pracoviště č. 26. Následující příklad změní u všech pracovníků z pracoviště č. 25, uvedených v tabulce telefony, jejich pracoviště na č. 26 (Řešený příklad 115).

ŘEŠENÝ PŘÍKLAD 115

```
UPDATE telefony SET pracoviste=26  WHERE pracoviste=25
```

Výše uvedená tabulka by se tedy změnila takto (změna je vyznačena kurzívou) (Tabulka 15):

Tabulka 15: Tabulka telefony se změněným záznamem

id	jmeno	telefon	email	pracoviste
1	Jan Novák	+420212131415	novak@email.cz	*26*

6.2.4 SMAZÁNÍ ZÁZNAMŮ V TABULCE

Pro smazání záznamů v tabulce se používá příkaz `DELETE`, který má následující syntaxi:

```
DELETE FROM název_tabulky
WHERE název_sloupce=hodnota
```

Rušíme-li pracoviště č. 26, odstraníme všechny kontakty z tabulky telefony. Toto provede následující příklad (Řešený příklad 116):

ŘEŠENÝ PŘÍKLAD 116

```
DELETE FROM telefony
WHERE pracoviste=26
```

Výsledkem bude prázdná tabulka `telefony` (Tabulka 16).

Tabulka 16: Tabulka telefony po odstranění záznamu

id	jmeno	telefon	email	pracoviste

6.2.5 VÝBĚR ZÁZNAMŮ Z DATABÁZE

Zřejmě nejdůležitější SQL příkaz je příkaz pro výběr dat z databáze. Má jméno `SELECT` a následující syntaxi.

```
SELECT [co vybrat (které sloupce)]
FROM [z jaké tabulky nebo tabulek]
WHERE [podmínka]
GROUP BY [podle čeho seskupit]
ORDER BY [seřazení podle něčeho]
```

Pokud bychom chtěli vypsat celý obsah nějaké tabulky, použijeme příkaz `SELECT` následovně (Řešený příklad 117):

ŘEŠENÝ PŘÍKLAD 117

```
SELECT * FROM telefony
```

Značka `*` je zástupný symbol pro všechny sloupce v dané tabulce a používá se pro zkrácení zápisu.

Abychom si příkaz lépe ilustrovali, vložme do tabulky `telefony` dva záznamy, tak aby vypadala následovně (Obrázek 17).

Tabulka 17: Tabulka telefony se dvěma záznamy

id	jmeno	telefon	email	pracoviste
1	Jan Novák	+420212131415	novak@email.cz	25
2	Jana Nová	+420123456789	nova@gmail.com	28

Představme si, že nás nyní zajímají pouze jména a telefonní čísla zaměstnanců v tabulce. V takovém případě je vhodné použít příkaz `SELECT` takto (Řešený příklad 118):

ŘEŠENÝ PŘÍKLAD 118

```
SELECT jmeno, telefon FROM telefony
```

Výsledkem bude (Tabulka 18):

Tabulka 18: Výsledek příkazu SELECT

jmeno	telefon
Jan Novák	+420212131415
Jana Nová	+420123456789

Nyní nás budou zajímat pouze zaměstnanci z pracoviště č. 25, a to opět pouze jejich jména a telefon. V tomto případě je vhodné použít příkaz `SELECT` takto (Řešený příklad 119):

ŘEŠENÝ PŘÍKLAD 119

```
SELECT jmeno, telefon FROM telefony
WHERE pracoviste = 25
```

Dle očekávání bude takovýto výsledek (Tabulka 19):

Tabulka 19: Výsledek příkazu SELECT

jmeno	telefon
Jan Novák	+420212131415

Pokud nás zajímají pouze zaměstnanci, kteří mají svůj e-mail na určité doméně (např. gmail.com), můžeme použít příkaz `SELECT` takto (Řešený příklad 120):

ŘEŠENÝ PŘÍKLAD 120

```
SELECT jmeno, telefon, email FROM telefony
WHERE email like '%@gmail.com'
```

Znak `%` je použit jako zástupný znak pro jakýkoli řetězec a porovnávací operátor `like` funguje v podstatě stejně jako operátor =. Dotaz tedy zobrazí všechny záznamy, u kterých email končí řetězcem `@gmail.com`. Výsledkem tedy bude (Tabulka 20):

Tabulka 20: Tabulka zaměstnanců s emailem na gmail.com

jmeno	telefon	email
Jana Nová	+420123456789	nova@gmail.com

6.3 DATA VE VÍCE TABULKÁCH

Zatím jsme si představili využití SQL příkazů pouze v souvislosti vždy jen s jednou tabulkou najednou. Data v databázích jsou však velmi často rozdělena do více tabulek, které jsou mezi sebou určitým způsobem svázány. V této kapitole si ukážeme, jak lze v takovém případě získávat data z více tabulek najednou.

Pro ukázku použijeme dvě tabulky. První tabulka `dum` obsahuje záznamy o třech různých domech (Tabulka 21). Druhá tabulka `osoba` obsahuje záznamy o šesti osobách (Tabulka 22).

Tabulka 21: Tabulka `dum`

id	nazev_domu	mesto	pocet_pokoju
1	Penzion Aldo	Karviná	15
2	Hotel Clarion	Ostrava	30
3	Hotel Hilton	Praha	200

Tabulka 22: Tabulka `osoba`

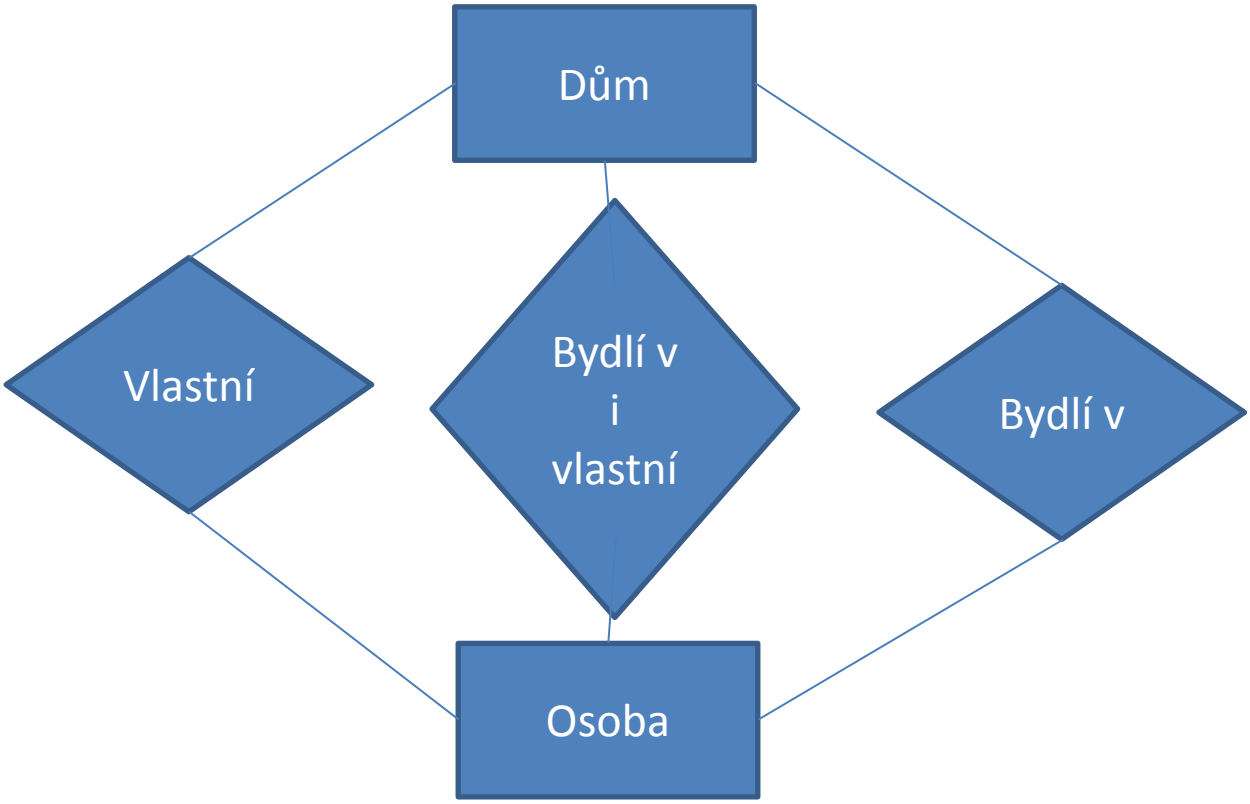
id	jméno	vlastni	bydli_v
1	Jan Górecki	-	-
2	Kamil Rychlý	-	1
3	Emil Sklář	-	3
4	Jan Novák	1	2
5	Petr Koukal	2	3
6	Richard Hilton	3	3

Tabulka `osoba`, kromě informací o dané osobě, také obsahuje informace o tom, jak je daná osoba spojena s nějakým domem z tabulky `dum`. Tyto informace jsou uloženy ve sloupcích `vlastni` a `bydli_v`, které uvádějí identifikační čísla domů (`id` z tabulky `dum`), které daná osoba vlastní, resp. ve kterém z těchto domů daná osoba bydlí. Tyto sloupce se nazývají jako tzv. **vazební sloupce**, jelikož reprezentují vazbu mezi tabulkami.

6.3.1 ERD DIAGRAM

Tyto dvě tabulky jsou mezi sebou svázány několika vazbami skrze vazební sloupce. Tyto vazby lze reprezentovat pomocí tzv. **ERD diagramu** (z angl. Entity-Relationship Diagram). Obdélníky v diagramu reprezentují jednotlivé tabulky v databázi, kosočtverce a hrany pak reprezentují vztahy mezi nimi. Jak lze vidět na příkladu níže, mezi dvěma tabulkami může existovat libovolné množství různých vztahů. Nás zajímají tři vztahy. První vztah nazvaný v ERD diagramu **Vlastní** popisuje, který dům je vlastnictví dané osoby. V databázi je tento vztah reprezentován sloupcem `vlastni` v tabulce `osoba`, kde je uvedeno `id` domu, který daná osoba vlastní. Další vztah nazvaný v ERD diagramu **Bydlí v** popisuje, ve kterém domě daná osoba bydlí. V databázi je tento vztah reprezentován sloupcem `bydli_v` v tabulce `osoba`, kde je uvedeno `id` domu, ve kterém daná osoba bydlí. Další vztah nazvaný v ERD diagramu **Bydlí v i vlastní** je zřejmou kombinací obou předchozích vztahů. V databázi je tento vztah reprezentován pomocí obou sloupců `vlastní` a `bydli_v` (Obrázek 13).

Obrázek 13: ERD diagram



6.3.2 PŘÍKAZ SELECT PRO VÍCE NEŽ JEDNU TABULKU

V případě, že chceme vybírat data z více než jedné tabulky najednou, můžeme opět použít známý příkaz `SELECT`. Všechny tabulky, ze kterých chceme výběr provést, napíšeme za klíčové slovo `FROM`. Výběr dat ze dvou tabulek najednou si vyzkoušíme na následujícím příkladu, kde si necháme zobrazit hodnoty ze sloupců `id` jak z tabulky `dum`, tak z tabulky `osoba`. Výběr dat provedeme následujícím příkazem (Řešený příklad 121):

ŘEŠENÝ PŘÍKLAD 121

```
SELECT dum.id id_dum, osoba.id id_osoba
FROM dum, osoba
```

Jak už víme, po klíčovém slově `SELECT` následuje seznam sloupců, které chceme ve výsledku zobrazit. Jelikož vybíráme sloupce z více tabulek najednou, je potřeba u každého sloupce specifikovat, ze které tabulky má být tento sloupce vybrán. Dva sloupce v různých tabulkách se mohou jmenovat stejně, např. právě sloupce `id` z tabulky `dum` a `id` z tabulky `osoba`. Specifikace se provede tak, že se před název sloupce přidá název dané tabulky a tečka (viz `dum.id` a `osoba.id` v příkladu výše). Někdy je také výhodné jednotlivé sloupce ve výsledku přejmenovat. To se dosáhne tak, že se za specifikaci sloupce přidá mezera a nový název sloupce, např. `dum.id` přejmenujeme na `id_dum` a `osoba.id` na `id_osoba`. Záhlaví výsledku tedy budou tvořit názvy sloupců `id_dum` a `id_osoba` (Tabulka 23).

Tabulka 23: Kartézský součin tabulek dum a osoba

id_dum	id_osoba
1	1
1	2
1	3
…	…
1	6
2	1
…	…
3	6

Záznamy výsledné tabulky pak tvoří kartézský součin záznamů z jednotlivých tabulek. V našem případě to znamená, že výsledek mám 3 x 6 záznamů, ve kterém jsou kombinace všech záznamů z tabulky `dum` a všech záznamů z tabulky `osoba`, tzn. „každý s každým“, jak lze vidět v tabulce výše.

Samotný kartézský součin nám však často není příliš k užitku, jelikož se ve výsledku neprojeví vazba mezi tabulkami. Abychom mohli vybírat záznamy z různých tabulek, které jsou v nějaké vazbě, je nutno tuto vazbu do příkazu `SELECT` zahrnout. To zařídíme tak, že přidáme klíčové slovo `WHERE`, za které vložíme tzv. **vazebnou podmínku**. Vazebná podmínka se tvoří tak, že spojíme operátorem rovnosti = odpovídající si sloupce z různých tabulek.

Představme si situaci, kdy nás zajímá, kdo vlastní který dům, přesněji řečeno, chceme si nechat vypsat jméno osoby vlastníka ke každému názvu domu. Pro výběr dat tedy využijeme vazbu Vlastní, která je reprezentována sloupcem `vlastni` v tabulce `osoba` a jemu odpovídajícímu sloupci `id` z tabulky `dum`. Vazební podmínka tedy bude `dum.id = osoba.vlastni` a celý příkaz pro výběr bude vypadat takto (Řešený příklad 122):

ŘEŠENÝ PŘÍKLAD 122

```
SELECT dum.id id_dum, osoba.id id_osoba, dum.nazev_domu,
osoba.jmeno, osoba.vlastni
FROM dum, osoba
WHERE dum.id = osoba.vlastni
```

Výsledkem pak bude tato tabulka (Obrázek 24):

Tabulka 24: Vlastníci domů

id_dum	id_osoba	dum.nazev_domu	osoba.jmeno	osoba.vlastni
1	4	Penzion Aldo	Jan Novák	**1**
2	5	Hotel Clarion	Petr Koukal	**2**
3	6	Hotel Hilton	Richard Hilton	**3**

Z kartézského součinu všech záznamů z obou tabulek jsou vypsány je záznamy, ve kterých se hodnota ve sloupci rovná hodnotě `dum.id` ve sloupci `osoba.vlastni` (vyznačeno tučně).

Zajímalo-li by nás, kde která osoba bydlí, tzn. u každého jména osoby bychom chtěli vypsat název domu, ve kterém daná osoba bydlí, postupovali bychom obdobně. Vazební

podmínku bychom formulovali jako `dum.id = osoba.bydli_v` a celý příkaz by vypadal následovně (Řešený příklad 123):

ŘEŠENÝ PŘÍKLAD 123

```
SELECT dum.id id_dum, osoba.id id_osoba, dum.nazev_domu,
osoba.jmeno, osoba.bydli_v
FROM dum, osoba
WHERE dum.id = osoba.bydli_v
```

Výsledkem by pak pak byla tato tabulka (Obrázek 25):

Tabulka 25: Obyvatelé jednotlivých domů

id_dum	id_osoba	dum.nazev_domu	osoba.jmeno	osoba.bydli_v
1	2	Penzion Aldo	Kamil Rychlý	**1**
3	3	Hotel Hilton	Emil Sklář	**3**
2	4	Hotel Clarion	Jan Novák	**2**
3	5	Hotel Hilton	Petr Koukal	**3**
3	6	Hotel Hilton	Richard Hilton	**3**

Realizace vazby Bydlí v a vlastní by se provedla takto (Řešený příklad 124):

ŘEŠENÝ PŘÍKLAD 124

```
SELECT dum.id id_dum, osoba.id id_osoba, dum.nazev_domu,
osoba.jmeno, osoba.vlastni, osoba.bydli_v
FROM dum, osoba
WHERE dum.id = osoba.bydli_v and dum.id = osoba.vlastni
```

Vazební podmínka je zde tvořená dvěma podmínkami, `dum.id = osoba.bydli_v` a `dum.id = osoba.vlastni`, které jsou spojeny logickou spojkou `and`. Výsledná tabulka, která zobrazuje osoby, které bydlí ve vlastním domě, vypadá takto (Obrázek 26):

Tabulka 26: Obyvatelé domů, kteří jsou zároveň jejich vlastníky

id_dum	id_osoba	dum.nazev_domu	osoba.jmeno	osoba.vlastni	osoba.bydli_v
3	6	Hotel Hilton	Richard Hilton	3	3

tzn., že pouze pan Hilton bydlí ve svém vlastním domě.

7 REGISTRACE A OVĚŘENÍ UŽIVATELE

Téměř každý webový portál obsahuje formulář pro přihlášení uživatele. Registrovaným uživatelům, po zadání uživatelského jména a hesla a jejich úspěšném ověření, jsou pak nabídnuty personalizované služby jako vlastní e-mail, úložiště dat, nástroje pro komunikaci s ostatními uživateli, atd. Proces jak registrace, tak ověření uživatele většinou zahrnuje 3 části. V první části je uživateli portálu nabídnut formulář, pomocí kterého jsou získány informace o uživateli, hlavně tedy uživatelské jméno a heslo, pod kterým se poté bude na portál přihlašovat. Pro zajištění (naprogramování) této části můžeme vystačit jen s HTML. Druhá část procesu zajišťuje převzetí dat z formuláře a jejich zpracování, např. odeslání do databáze nebo ověření správnosti uživatelského hesla. Tuto část můžeme zpracovat pomocí jazyka PHP. Třetí část procesu zahrnuje uložení dat na správná místa databáze, popř. získání správných dat z databáze, např. zjištění uživatelského hesla k zadanému uživatelskému jménu. Tuto část procesu můžeme zajistit pomocí jazyka SQL. Jak je tedy vidět, proces registrace a ověření uživatele zahrnuje veškeré poznatky, kterým jsme se doposud v této práci věnovali, a jeho zpracování bude určitým vrcholem celého našeho snažení.V této kapitole si ukážeme, jak je možno zmiňovaný proces zpracovat s doposud nabytými znalostmi na úrovni současného standardu většiny menších portálových aplikací.

7.1 REGISTRACE UŽIVATELE

Technicky jde při registraci uživatele o to, získat od uživatele určité informace a tyto informace pak uložit do odpovídající předem připravené tabulky (tabulek) v databázi. Nejdříve je tedy potřeba si uvědomit, jaké informace chceme od uživatele získávat. V příkladu, na kterém budeme celý proces registrace ilustrovat, budeme od uživatele získávat tyto informace: jeho jméno a příjmení, město, ve kterém bydlí a přihlašovací údaje uživatelské jméno (z angl. login) a heslo (z angl. password). Předpokládejme, že už máme vytvořenou databázi `registrace` na serveru `localhost` a vytvoříme zde odpovídající tabulku `uzivatele`, kde budeme informace získané od uživatele ukládat (viz předchozí kapitola). Tabulka bude obsahovat sloupec `id` a dalších 5 sloupců pro získávané informace – vyhneme se mezerám a diakritice, tudíž sloupce nazveme: `id`, `jmeno`, `prijmeni`, `město`, `login` a `password`. Tabulku vytvoříme např. pomocí následujícího SQL dotazu (Řešený příklad 125):

ŘEŠENÝ PŘÍKLAD 125

```
CREATE TABLE uzivatele (
id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
jmeno VARCHAR(20),
prijmeni VARCHAR(30),
mesto VARCHAR(20),
login VARCHAR(20),
password VARCHAR(200)
)
```

Pro `password` rezervujeme 200 znaků z důvodu případného kódování hesla, viz níže.

7.1.1 HTML - REGISTRAČNÍ FORMULÁŘ

Nyní můžeme připravit formulář, kterým budeme získávat dané informace od uživatele. V našem příkladu by mohl být formulář vytvořen takto (Řešený příklad 126):

ŘEŠENÝ PŘÍKLAD 126

```
<form name="registrace" action="zapis.php" method="post">
<h2>Registrace</h2>
<table>
<tr>
<td>Jméno:</td>
<td><input type="text" name="jmeno" value="" ></td>
</tr>

<tr>
<td>Příjmení:</td>
<td><input type="text" name="prijmeni" value="" ></td>
</tr>

<tr>
<td>Město:</td>
<td><input type="text" name="mesto" value="" ></td>
</tr>

<tr>
<td>Uživatelské jméno:</td>
<td><input type="text" name="login" value="" ></td>
</tr>

<tr>
<td>Heslo:</td>
<td><input type="password" name="password" value="" ></td>
</tr>

<tr>
<td></td>
<td align=right><input type="submit" value="Odeslat"></td>
</tr>
</table>
</form>
```

Takto vytvořený formulář vypadá v prohlížeči následovně (z estetických důvodů vždy dbáme na to, aby byl úhledně naformátován do tabulky) (Obrázek 14):

Obrázek 14: Registrační formulář

Parametr `action` jsme nastavili na hodnotu `zapis.php`. Poté, co uživatel vyplní požadované informace a klikne na tlačítko "odeslat", odešlou se informace na server a zavolá se (spustí se) tento PHP skript, který bude tyto informace zpracovávat. V následující kapitole si ukážeme, jak si takový skript vytvořit.

7.1.2 PHP –SKRIPT PRO ZPRACOVÁNÍ FORMULÁŘOVÝCH DAT

Druhá část procesu registrace uživatele obecně zajišťuje převzetí dat z formuláře a jejich odeslání do databáze. Toto můžeme implementovat v PHP.

Nejdříve převezmeme informace z formuláře poslané na server a uložíme si je do proměnných. To můžeme v našem příkladu zařídit takto (Řešený příklad 127):

ŘEŠENÝ PŘÍKLAD 127

```
$a=$_POST["jmeno"];
$b=$_POST["prijmeni"];
$c=$_POST["mesto"];
$d=$_POST["login"];
$e=$_POST["password"];
```

Je třeba velmi důsledně dbát na to, aby jména přebíraných polí přesně odpovídala jménům polí ve formuláři. Jména (`jmeno`, `prijmeni`, `město`, `login` a `password`) tedy zásadně neopisujeme, ale vždy pouze kopírujeme (CTRL + C, VTRL + V) z kódu formuláře. Jelikož jsme odeslali informace metodou `post`, pro jejich převzetí a uložení do proměnných `a` až `e` použijeme funkci `$_POST`.

Nyní se připojíme na databázi. Nejprve se připojíme na server, na kterém se databáze nachází. To se provede pomocí příkazu

```
mysql_connect("název serveru db", "jméno", "heslo");
```

Parametry příkazu `mysql_connect`, tedy název serveru, kde je databáze (zkráceně db) uložena, přístupové jméno a heslo na server, poskytuje zprostředkovatel webhostingu. Dále se připojíme na databázi pomocí příkazu

```
mysql_select_db("název db");
```

Parametr příkazu je název databáze, na kterou se chceme připojit, a taktéž ji získáme od zprostředkovatele webhostingu. V našem příkladu, kdy se připojujeme na server `localhost`, můžeme připojení na databázi `registrace` implementovat takto (Řešený příklad 128):

ŘEŠENÝ PŘÍKLAD 128

```
    $link = mysql_connect("localhost", "root")
            or die("Nelze se připojit: " . mysql_error());
    print "Připojeno úspěšně";
    mysql_select_db("registrace")    or    die("Nelze    vybrat
databázi");
```

Nyní můžeme data převzatá z formuláře poslat do databáze. To se provede pomocí příkazu

```
mysql_query("SQL dotaz");
```

Parametr "SQL dotaz" by měl obsahovat nějaký SQL dotaz, který informace z formuláře, nyní uložené v proměnných, uloží do určené tabulky.

7.1.3 SQL – DOTAZ ZAJIŠŤUJÍCÍ ZÁPIS DAT DO DATABÁZE

SQL příkaz, kterým zapisujeme data do tabulky je příkaz `INSERT INTO`. Jelikož v našem příkladu chceme informace z formuláře zapsat do předem připravené tabulky `uzivatele`, příkaz formulujeme takto:

```
    INSERT INTO uzivatele (jmeno, prijmeni, mesto, login,
password) VALUES ('$a','$b','$c','$d', '$e')
```

Vždy pečlivě dbejme na to, zda vkládáme správnou proměnnou do správného sloupce v tabulce, např. proměnná `$e`, která obsahuje údaj z formulářového pole nazvaného `password`, se vkládá do sloupce `password` v tabulce `uzivatele`. Jména jednotlivých formulářových polí (hodnoty parametru `name`) jsme schválně nastavili stejně jako jména sloupců v databázi. Toto však nemusí být pravidlem! Velmi často chybou potom je, že v případech, kdy jsou jména odpovídajících si formulářových polí a sloupců v tabulce různá, snažíme se zapsat obsah proměnné z údajem z formuláře do sloupce, který v tabulce vůbec neexistuje.

Celý PHP skript `zapis.php` zajišťující zpracování dat z formuláře `registrace` by potom vypadal takto (Řešený příklad 129):

ŘEŠENÝ PŘÍKLAD 129

```
    $a=$_POST["jmeno"];
    $b=$_POST["prijmeni"];
    $c=$_POST["mesto"];
    $d=$_POST["login"];
    $e=$_POST["password"];

    $link = mysql_connect("localhost", "root")
            or die("Nelze se připojit: " . mysql_error());
```

```
    print "Připojeno úspěšně";
    mysql_select_db("registrace")    or    die("Nelze    vybrat
databázi");

    $dotaz = "INSERT INTO uzivatele (jmeno, prijmeni, mesto,
login, password) VALUES ('$a','$b','$c','$d', '$e')";
    mysql_query($dotaz);
```

7.1.4 ŠIFROVÁNÍ HESLA

Pokud chceme zvýšit zabezpečení přístupu na naše stránky, je vhodné při registraci uživatele ukládat do databáze uživatelské heslo v zašifrované podobě. V takové případě pak znemožníte komukoli nepovolanému, aby si heslo zjistil a případně ho nějakým způsobem zneužil. Můžeme k tomu použít jednosměrné šifrování (tzv. hashování) hesla. To probíhá tak, že se daný řetězec přetvoří na shluk znaků, přičemž ani my nevíme, na základě jakého klíče. Není tedy možné, aby šel daný hash rozšifrovat – to je rozdíl od šifrování obousměrného, kde k šifře máme k dispozici také klíč, jak šifru rozluštit a získat zpátky to, co jsme si do šifry „uložili".

V PHP existuje několik funkcí pro bezpečné ukládání hesel / hashování. Jsou jimi např. `md5()` nebo `sha1()`, nicméně tyto algoritmy jsou dnes již poněkud zastaralé a jejich bezpečnost byla zpochybněna. My si ukážeme na našem příkladu vhodnější funkci `hash()`, pomocí níž lze zvolit nějaký modernější šifrovací algoritmus, např. "whirpool" nebo "sha512".

Při přebírání informace z formuláře obsahující heslo, tedy v řádku

```
$e = $_POST["password"];
```

upravíme příkaz tak, aby se do proměnné `$e` uložilo heslo v zašifrované podobě. To provedeme následující úpravou řádku na:

```
$e = hash("sha512", $_POST["password"]);
```

Tím zajistíme šifrování hesla pomocí algoritmu "sha512". Heslo v databázi pak máme uloženo v zahashované podobě. Když např. uživatel při registraci vyplnil heslo 123456, pomocí funkce `hash("sha512", "123456")` se převedlo na shluk znaků, např. `ba3253876aed6bc2c6a956df346eab413`, a v této podobě se také uložilo do databáze. Jelikož se heslo tímto způsobem výrazně prodlouží, je důležité dbát na dostatečnou kapacitu tohoto pole v tabulce databáze – v našem příkladu jsme pro jistotu nastavili u pole `password` kapacitu 200 (viz definice tabulky `uzivatele`). Při ověřování uživatele potom budeme heslo porovnávat v nějaké takové zašifrované podobě (viz níže).

7.2 OVĚŘENÍ UŽIVATELE

Při ověřování (autentizaci) uživatele jde o to, porovnat údaje, které uživatel zadává při přihlašování, s údaji, které zadal při registraci. Technicky to znamená, porovnat, pro dané uživatelské jméno, heslo, které uživatel zadává při přihlašování, s heslem, které uživatel zadal při registraci. V případě, že je uživatel již registrován, tzn. jeho uživatelské jméno je již uloženo v databázi, a obě hesla se shodují, je identita uživatele ověřena a umožníme mu vstup na personalizované části webu. V opačném případě pak identita uživatele ověřena není a přístup na personalizované části portálu je mu odepřen.

7.2.1 HTML – PŘIHLAŠOVACÍ FORMULÁŘ

Formulář pro přihlášení uživatele většinou obsahuje pouze dvě vstupní pole, uživatelské jméno a heslo. V našem příkladu bychom vystačili s tímto formulářem (Řešený příklad 130):

ŘEŠENÝ PŘÍKLAD 130

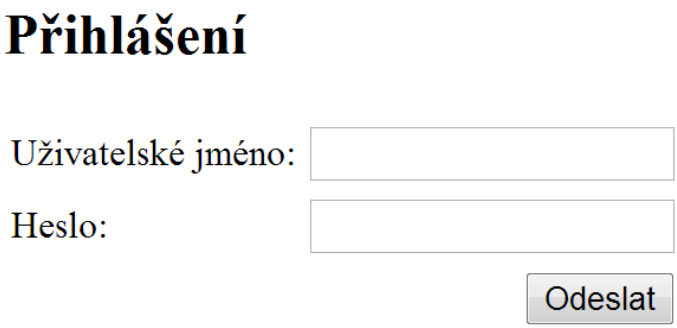
```
<form name="prihlaseni" action="overeni.php" method="post">
    <h2>Přihlášení</h2>
    <table>
    <tr>
      <td>Uživatelské jméno:</td>
      <td><input type="text" name="login" value="" ></td>
    </tr>

    <tr>
      <td>Heslo:</td>
      <td><input   type="password"   name="password"   value=""
></td>
    </tr>

    <tr>
      <td></td>
      <td            align=right><input            type="submit"
value="Odeslat"></td>
    </tr>
    </table>
</form>
```

Uložíme pod názvem `form2.html`. V prohlížeči vypadá takový formulář takto (Obrázek 15):

Obrázek 15: Přihlašovací formulář



Parametr `action` jsme nastavili na hodnotu `overeni.php`. Poté, co uživatel vyplní požadované informace a klikne na tlačítko "Odeslat", odešlou se informace na server a zavolá se (spustí se) tento PHP skript, který bude tyto informace zpracovávat, tzn. který ověří správnost zadaného hesla pro dané uživatelské jméno.

7.2.2 PHP – SKRIPT PRO OVĚŘENÍ UŽIVATELE

Skript pro ověření nejprve převezme údaje z přihlašovacího formuláře a uloží je do nastavených proměnných `$form_uziv_jmeno` a `$form_heslo`. V případě, kdy využíváme šifrování hesla je potřeba ještě heslo převést na šifrovanou podobu, viz řádek `$form_heslo = hash("sha512", $_POST["password"]);`. Následující PHP skript se postará o ověření uživatele (Řešený příklad 131).

ŘEŠENÝ PŘÍKLAD 131

```
    <?php
      // nastavte zde všechny údaje
      // údaje o databázi
      $server = "localhost";
      $uzivatel = "root";
      $heslo = "";
      $databaze = "registrace";

      // údaje o tabulce v db s registračními údaji
      $db_nazev_tabulky = "uzivatele";
      $db_sloupec_uziv_jmeno = "login";
      $db_sloupec_heslo = "password";

      // převzetí polí z formuláře
      $form_uziv_jmeno = $_POST['login'];
      $form_heslo  =  hash("sha512",  $_POST["password"]);  //
heslo je šifrováno

      // chranena sekce - zde bude uživatel přesměrován po
úspěšném přihlášení
      $private_url = "private.php";

      //-------------------------------------------------------
      // zde už není třeba nic měnit

      // SQL dotaz, který vrátí heslo z db pro uživatelské
jméno zadané do formuláře
      $sql =
        "SELECT $db_sloupec_heslo FROM $db_nazev_tabulky WHERE
$db_sloupec_uziv_jmeno='{$form_uziv_jmeno}'";

      echo "Do db posílám dotaz:<br>".$sql;

      $spojení = mysql_connect($server, $uzivatel, $heslo)
        or die("Nelze se připojit: " . mysql_error());;    //
pripojeni na server
      mysql_select_db($databaze);
// pripojeni na databazi

      print "<br>Připojeno úspěšně";
```

```
      $výsledek = mysql_query($sql);
    // dotaz na heslo pro uzivatele

      // ověřovací proces
      if (!$výsledek):
        echo "<br>Ověření nelze použít. Zřejmě máte špatně
nastavené údaje o db. Server nevrátil odpověď na dotaz ".$sql;
        exit;
      endif;
        if(!mysql_num_rows($výsledek)):
        echo "<br>Uživatel není v tabulce.";
        exit;
      else:

         $i=0;
        if (mysql_result($výsledek, $i, $db_sloupec_heslo) !=
$form_heslo ):
          echo "<br>Nesprávné heslo. Uživatel je registrován v
tabulce, ale není zadáno správné heslo.";
          exit;
          else:
        endif;
      endif;

      //pokud odpovídají přihlašovací údaje
         session_start();
         header("Cache-control: private");
         //zaregistruje proměnou user_is_logged a nastaví ji na
1
         $_SESSION["user_is_logged"] = 1;
         //a pošle na úvodní soubor chráněné sekce
         header("Location: ".$private_url);
         exit;

    ?>
```

První část skriptu (část nad znaky -------) je potřeba nastavit dle našich podmínek. První 4 řádky

```
      $server = "localhost";
      $uzivatel = "root";
      $heslo = "";
      $databaze = "registrace";
```

nastavíme dle údajů od poskytovatele webhostingu, tedy ip adresa serveru, uživatelské jméno a heslo na tento server a název databáze, kterou máme k dispozici. V našem příkladu už by bylo nastaveno vše správně.

Další 3 řádky nastavují informace o tabulce s registračními údaji, tzn. název tabulky, jméno sloupce, kde jsou uloženy uživatelská jména a jméno sloupce, kde jsou uloženy uživatelská hesla. V našem příkladu je to takto:

```
$db_nazev_tabulky = "uzivatele";
$db_sloupec_uziv_jmeno = "login";
$db_sloupec_heslo = "password";
```

Další dva řádky určují jména formulářových polí, ze kterých budou převzaty přihlašovací údaje. V našem příkladu:

```
$form_uziv_jmeno = $_POST['login'];
$form_heslo = hash("sha512", $_POST["password"]);
```

Všimněme si, že heslo ukládané do `$form_heslo` je šifrováno, tzn. tento skript bude pracovat správně pouze s šifrovanou verzí registrace.

Poslední řádek, který je potřeba nastavit, je úvodní stránka neveřejné sekce, na kterou prohlížeč přejde po úspěšném přihlášení. V našem příkladu jsme ji nastavili na stránku `private.php`, tedy takto:

```
$private_url = "private.php";
```

Tato stránka se tedy zobrazí pouze v případě, kdy uživatel při přihlašování zadal správnou kombinaci jména a hesla. V opačném případě se zobrazí buď hláška `Nesprávné heslo. Uživatel je registrován v tabulce, ale není zadáno správné heslo.`, to v případě, že je zadáno uživ. jméno, které je v databázi, ale nesprávné heslo, nebo hláška `Uživatel není v tabulce.`, to v případě, že uživatelské jméno není v tabulce uživatelů vůbec.

Do zbylé části skriptu `overeni.php` není třeba zasahovat a ani se nebudeme věnovat detailně popisu jeho funkčnosti, jelikož bychom už výrazně překročili rámec tohoto výukového materiálu. Spokojíme se se pouze s vědomím toho, že skript získá pro dané uživatelské jméno (`login` z formuláře) příslušné heslo v zašifrované podobě z tabulky `uzivatele` a porovná jej s zašifrovaným uživatelským heslem z formuláře (pole `password`). Na základě jejich porovnání pak postupuje tak, jak již bylo popsáno v předchozím odstavci.

Nyní je potřeba zajistit, aby každou stránku v neveřejné sekci bylo možno zobrazit pouze přihlášeným uživatelům. K těmto účelů využijeme pomocný skript `protection.php`, který vypadá takto (Řešený příklad 132):

ŘEŠENÝ PŘÍKLAD 132

```
<?php
 session_start();
 header("Cache-control: private");
 //Pokud není uživatel přihlášen, pošleme mu přihlašovací
formulář
 if ($_SESSION["user_is_logged"] != 1){
 header("Location: form2.html");
 exit();
  }
?>
```

Tento skript pak vložíme na začátek každé stránky neveřejné sekce pomocí jednoho řádku v PHP, který vypadá takto (Řešený příklad 133):

ŘEŠENÝ PŘÍKLAD 133

```
<?php include('protection.php'); ?>
```

Stránka neveřejné sekce, např. `private.php`, pak může vypadat třeba takto (Řešený příklad 134):

ŘEŠENÝ PŘÍKLAD 134

```
<!DOCTYPE HTML>
<html>
  <head>
  <title>Private</title>
  </head>
  <body>
       <?php include('protection.php'); ?>

       Nyní se nacházíte v privátní sekci. Tyto řádky se
zobrazí pouze přihlášeným uživatelům.
  </body>
</html>
```

V případě, že URL stránky zadá do prohlížeče nepřihlášený uživatel, skript `protection.php` ho okamžitě přesměruje na přihlašovací formulář `form2.php` a tudíž zbylou část neveřejné stránky neuvidí. Takto uchráníme obsah našich neveřejných stránek před očima nepřihlášených uživatelů.

Pro odhlášení ze stránek lze použít následující skript (Řešený příklad 135):

ŘEŠENÝ PŘÍKLAD 135

```
<?php
 session_start();
 $_SESSION = array();
 session_destroy();
 //Ověř zda došlo ke korektnímu ukončení session
 if ($_SESSION["user_is_logged"]){
  echo "FATAL ERROR: Cannot terminate session!";
  } else {
   //pokud je vše o.k., pošli uživatelovi přihlašovací
dialog
   //(nebo jinou  vhodnou stránku podle vlastního uvážení)
   header("Location: form2.html");
  }
?>
```

Tento skript nás odhlásí a přesměruje na přihlašovací formulář `form2.html`.

Celá procedura přihlašování a kontroly přihlášení využívá tzv. sessions. Díky sessions jsou předávána data ukládána do souboru přímo na serveru a programátor k nim přistupuje prostřednictvím speciálního pole proměnných `$_SESSION`. Každá stránka, která má mít přístup ke sdíleným informacím (v našem případě o stavu přihlášení uživatele), musí ještě před jakýmkoliv výstupem zavolat funkci `session_start()`. Kvůli podivnému chování s diskovou mezipamětí (cache) v MS Internet Exploreru, je vhodné v každém skriptu nastavit vlastní správu cachování, čehož docílíme posláním hlavičky `header("Cache-control: private")`. I tento příkaz je nutno zadat před jakýkoliv výstup do HTML! Jakmile máme session nastartovanou, můžeme do ní uložit nějaká data přiřazením, např.

```
$_SESSION['moje_promenna']=hodnota;
```

Tento příkaz zaregistruje (případně přepíše) session proměnou `$moje_promenna` a uloží do ní hodnotu (řetězec, číslo, ...). Samozřejmě můžeme registrovat větší počet proměnných. K této hodnotě se následně dostaneme opačným přiřazením

```
$hodnota=$_SESSION['moje_promenna'];
```

Pozor, na levé straně MUSÍ být proměnná. Pro zrušení proměnné ji stačí přiřadit prázdnou hodnotu (`''`,případně `false`). Pro zrušení celé session je potřeba na začátek stránky (ale až za funkci `session_start()`) zadat tyto dva příkazy:

```
$_SESSION = array(); //Vyčistí všechy registrované
proměnné
session_destroy(); //uzavře session
```

To jsme provedli ve skriptu `logout.php`. Ke zrušení session dojde automaticky po zavření prohlížeče a také po určité době nečinnosti (standardně kolem 20 minut).

Celé ověřovaní uživatele pak probíhá velmi jednoduše. Nabídneme uživateli přihlašovací formulář `form2.html`. Pokud zde zadá správné přihlašovací údaje, což ověří skript `overeni.php`, je na serveru v sessions nastavena proměnná `user_is_logged` na hodnotu 1. Na všech neveřejných stránkách pak pomocí skriptu `protection.php` ověřujeme, zda je proměnná `user_is_logged` rovná 1. Pokud ano, uživatel je přihlášen a obsah neveřejné sekce mu zobrazíme. V opačném případě, když `$_SESSION["user_is_logged"] != 1`, zobrazíme uživateli pouze přihlašovací formulář.

Jak lze z předchozího vidět, proces registrace a ověření uživatele již není zcela triviální záležitostí a využívá znalosti nabyté ve všech třech programovacích jazycích, se kterými jsme se v tomto materiálu setkali, tedy HTML, PHP a SQL. Na druhou stranu, jen málokterý portál se bez implementace těchto procedur obejde, tudíž i v semestrální práci bude kladen důraz na správnou implementaci právě těchto procedur, bez kterých pak nebude možno práci považovat za obhajitelnou.

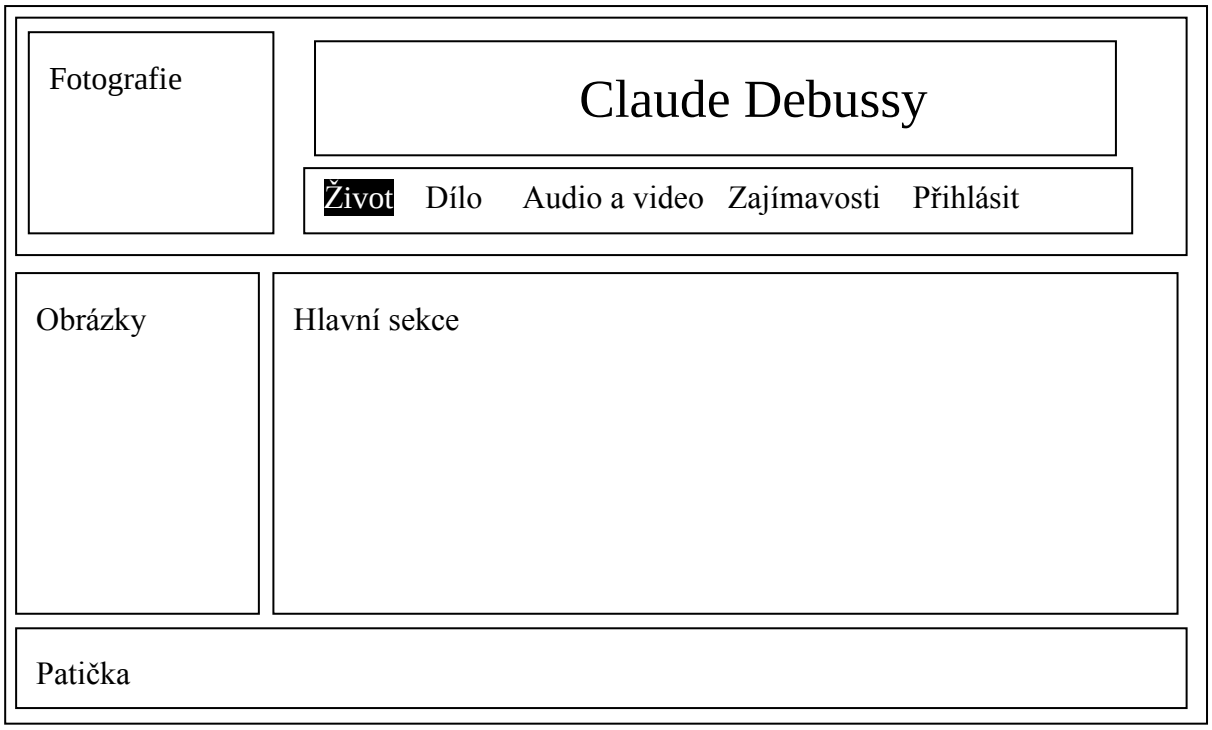
8 PŘÍPADOVÁ STUDIE

V této kapitole si ukážeme praktickou ukázku velmi jednoduchého portálu, na kterém ilustrujeme využití většiny z toho, co jsme se doposud naučili. Téměř každému, kdo s programováním začíná, se stane, že ačkoli již má znalosti o všech prvcích portálu samostatně, má problém všechno dohromady propojit. V této kapitole si proto tvorbu portálu ukážeme v celé své komplexnosti. Jelikož vše, co je v ukázce použito, je již známé, omezíme se v komentářích na minimum a zaměříme se především na jednotlivé kusy kódu, ve kterých je vše obsaženo. Ukázkový portál bude věnovat významnému francouzskému hudebnímu skladateli jménem Claude Debussy a bude obsahovat informace o jeho životě, díle, nějaké multimediální soubory, zajímavosti a přihlašovací formulář. Funkční portál (pro localhost) je umístěn v ZIPu na této URL: http://suzelly.opf.slu.cz/~gorecki/others/Debussy.zip .

Než začneme náš portál implementovat, vždy si nejprve dobře rozmysleme, jak by náš portál měl vypadat, co by měl obsahovat a co by měl umět. Pokud v této první fázi něco zanedbáme, nepříjemně se nám to vrátí v pozdějších fázích, kde budeme muset stále něco znova a znova předělávat. Předělávání se samozřejmě nevyhneme nikdy, ale právě díky detailnímu rozpracování návrhu portálu v první fázi se mu můžeme ze značné části vyhnout. Nejlepší je tedy vzít si obyčejnou tužku a papír a návrh portálu si pořádně rozpracovat. Až poté, co bude vše jasné, můžeme sednout k počítači a pustit se do implementace.

Náš ukázkový portál, resp. jeho úvodní stránku, si rozvrhneme (na papír) takto (Obrázek 16):

Obrázek 16: Návrh rozvržení (layout) portálu (jeho úvodní stránky)



S využitím HTML a CSS pak můžeme implementovat toto rozvržení stránek, tzv. layout, který se bude skládat z hlavičky s fotografií, levého pruhu s obrázky, hlavní sekce a patičky. Implementaci této první stránky, kterou pojmenujeme `index.php`, je dobré věnovat maximální pozornost. Tato stránka bude totiž to první, co uživatel při vstupu na náš portál uvidí. Pokud hned první stránka bude nepřehledná a nevzhledně vyhlížející, nemůžeme

čekat, že se na ní zdrží déle než pár vteřin. Až bude první stránka (sekce) precizně navržena a zpracována, teprve potom se začneme věnovat zpracování dalších sekcí (dílo, audio a video, zajímavosti a přihlásit).

Celý portál budeme formátovat pomocí stylopisu vytvořeného v CSS, který uložíme do souboru `style.css`, který má tento kód (Řešený příklad 136).

ŘEŠENÝ PŘÍKLAD 136

```
    /*CSS3 styl */
    /*Lay-out*/

    html {
      margin: 20px;
     background: white;
    }

    body {
     width: 840px;
     margin: 20px auto;
      padding: 10px;
     font:  12pt/1.5em  Tahoma,  Helvetica,"Helvetica  neue",
Arial, sans-serif;
      background: url(figures/blbkgnd.gif) repeat;

    }

    /*HTML 5 specific*/
    header,section,article,aside,footer, nav{
     display: block;
        /*border: black solid 1px;*/
    }

    /*Headings*/
    h1 {
     font: 32pt Tahoma,Helvetica,"Helvetica neue", Arial, sans-
serif;
    }

    /* nastaveni seznamu odkazu v navignacni oblasti */
    nav {
      margin: 10px 0 10px 0;
      height: 45px;
      text-align: center;
      clear:both !important;
      color: #222;
      background: #333;
      background: -moz-linear-gradient(#444, #222);
      background: -o-linear-gradient(#444, #222);
      background: -webkit-linear-gradient(#444, #222);
      box-shadow: rgba(0, 0, 0, 0.2) 0px 2px 2px;
      border: 1px solid #222;
```

```
}


nav ul{
  margin: 10px;
}

nav li{
  display: inline;
  list-style-type: none;
  padding: 10px;
}

/* nastaveni odkazu */
nav a:link, nav a:visited {
 font:16pt normal Tahoma,"Helvetica neue", Arial, sans-
serif;
  color:white;
 text-decoration:none;
}

/* kdyz na link najedu mysi */
nav a:hover {
  background-color: white;
  padding-top: 12px;
  padding-bottom: 12px;
  color: #222;
}

#bright {
  background-color: white;
  padding-top: 12px;
  padding-bottom: 12px;
      color: #222;
}


nav a {
  padding: 20px;
}

aside{
 float: left;
 width: 150px;
 min-height: 400px;
 margin: 0 0 20px 0;
  padding: 10px;
}

section{
 float: left;
```

```
 border-left: 1px dotted black;
 padding-left: 20px;
 width: 640px;
 margin: 0 0 20px 0px;
}


article {
  margin: 0 0 10px 0;
}

footer{
 clear:both !important;
 height: 70px;
 padding: 2px;
  text-align:center;
}

#flip {
        -moz-transform: scaleX(-1);
        -o-transform: scaleX(-1);
        -webkit-transform: scaleX(-1);
        transform: scaleX(-1);
        filter: FlipH;
        -ms-filter: "FlipH";
}
```

Tento stylopis je výsledkem mnoha hodin dá se říct designerské práce. Ve stylopisu je definován formát základních HTML prvků jako `html`, `body`, `header`, `section`, `aside`, `footer`, `nav`, atd. Veškeré formátování, které do stylopisu šlo přenést, je do něj přeneseno, a v ostatních souborech je formátování využito minimálně. Pokud se někdy rozhodneme vzhled stránek měnit, bude nám pak stačit upravovat tento soubor.

Rozhodli jsme se, že hlavička a patička vždy zůstanou stejné (až na rozsvícení tlačítek v navidařní oblasti, viz níže) a bude se měnit pouze levý pruh s obrázky a hlavní sekce. Každá sekce tedy bude odpovídat jednomu php souboru, do kterého vložíme hlavičku a patičku pomocí PHP příkazu `include` a potom informace relevantní pro danou sekci.

Začněme tedy implementací hlavičky `header.php` (Řešený příklad 137).

ŘEŠENÝ PŘÍKLAD 137

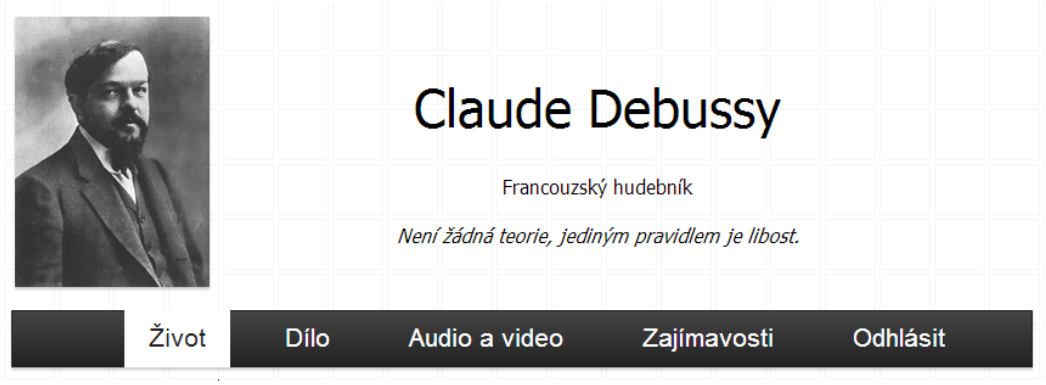
```
 <header>
<table width = 800px>
   <td width = 160><img src=figures/debussy.jpg width=160
style="box-shadow: rgba(0, 0, 0, 0.2) 0px 2px 2px;">  </td>
   <td style="text-align: center">
   <h1>Claude Debussy</h1>
   <p>Francouzský hudebník
   <p><i>Není žádná teorie, jediným pravidlem je
libost.</i>
    </td>
```

```
    </table>

      <nav>
        <ul class="sf-menu" id="nav">
          <li>
          <a <?php  if ($menu == 1) {echo "id=bright";}
           ?> href="index.php">  Život</a>
          </li>
          <li>
          <a <?php if ($menu == 2) {echo "id=bright";}
           ?> href="dilo.php">Dílo</a>
          </li>
          <li>
          <a <?php if ($menu == 3) {echo "id=bright";}
           ?> href="aav.php">Audio a video</a>
          </li>
          <li>
          <a <?php if ($menu == 4) {echo "id=bright";}
           ?> href="zajimavosti.php">Zajímavosti</a>
          </li>
          <li>
          <a <?php if ($menu == 5) {echo "id=bright";} ?>
           href="
           <?php if (isset($_SESSION) == 0) {session_start();}
           if      (isset($_SESSION["user_is_logged"])      and
($_SESSION["user_is_logged"] == 1) ) {echo "logout.php";}
           else {echo "members.php";}
           ?>
           ">
           <?php
           if      (isset($_SESSION["user_is_logged"])      and
($_SESSION["user_is_logged"] == 1) ) {echo "Odhlásit";}
           else  {echo "Přihlásit";}
    ?>
           </a>
          </li>
        </ul>
      </nav>
          </header>
```

Výsledkem nák kódu je takováto hlavička (Obrázek 17).

Obrázek 17: Vzhled hlavičky



Hlavička má v horní části tabulku 1 řádek x 2 sloupce. V prvním sloupci je umístěna fotka a ve druhém jméno skladatele a jeho známý citát. Ve spodní části je navigační sekce `nav`, ve které jsou umístěny v nečíslovaném seznamu `<ul>` odkazy na další sekce. Abychom zajistili, že odkaz pro danou sekci je rozsvícen, využili jsme následující kousek kódu (např. pro první sekci Život) (Řešený příklad 138):

ŘEŠENÝ PŘÍKLAD 138

```
    <a
    <?php  if ($menu == 1) {echo "id=bright";} ?>
href="index.php">Život</a>
```

Pokud je proměnná `$menu` nastavená na hodnotu 1, výsledkem zpracování PHP je HTML kód

```
<a id=bright href="index.php">Život</a>
```

který zajistí rozsvícení odkazu tím, že ho nastyluje podle třídy `bright` ze stylopisu. V opačném případě je výsledkem zpracování PHP tento HTML kód:

```
<a href="index.php">Život</a>
```

Ten odpovídá nerozsvícenému odkazu. Případ, kdy chceme rozsvítit pátou sekci Přihlásit, probereme později.

V každé sekce nesmíme opomenout nastavit proměnnou `$menu` na nějakou hodnotu. Tu nastavíme podle toho, kterou sekci zrovna zobrazujeme. Pokud zobrazuje první sekci Život, vložíme před příkaz `include("header.php");` ještě příkaz `$menu = 1;`. Tím zajistíme, že v hlavičce bude rozsvícen odkaz Život. V další sekci dílo před příkaz `include` vložíme příkaz `$menu = 2;`, atd. Takto lze jednoduše využít propojení PHP a HTML k implementaci této poměrně často používané funkci na většině webů.

Patičku můžeme implementovat takto (Řešený příklad 139):

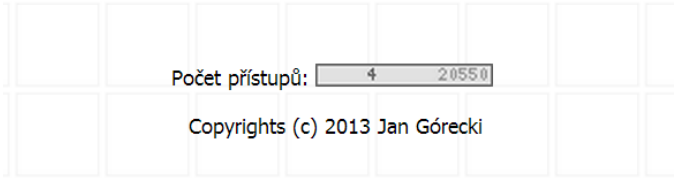
ŘEŠENÝ PŘÍKLAD 139

```
    <footer>   <p style="font-size:10px">   Počet přístupů:
    <a     href="http://pocitadlo.abz.cz/"     title="počítadlo
přístupů:                                  pocitadlo.abz.cz"><img
src="http://pocitadlo.abz.cz/aip.php?tp=di"
alt="počítadlo.abz.cz" border="0" /></a>

     <br>Copyrights (c) 2013 Jan Górecki
     </footer>
```

Výsledek bude vypadat takto (Obrázek 18):

Obrázek 18: Vzhled patičky



Patička obsahuje počítadlo přístupů na stránky, které lze najít na této URL: http://pocitadlo.abz.cz/. Máme-li připravenou hlavičku i patičku, můžeme implementovat celou úvodní stránku `index.php` takto (Řešený příklad 140):

ŘEŠENÝ PŘÍKLAD 140

```
    <!DOCTYPE html>
    <html>
      <head>
        <meta charset="UTF-8">
        <title>Claude Debussy - francouzský hudebník - život
        </title>
        <link rel="stylesheet" href="style.css">
      </head>
      <body>
     <?php
      $menu = 1;
      include("header.php");
     ?>
          <aside>
            <img      id=flip      src=figures/debussy_youth.jpg
title="Debussy zamlada." width=150>
            <img      id=flip      src=figures/claude-debussy.jpg
title="Claude debussy." width=150>
            <img src=figures/sLily.jpg title="Debussy s Lily."
width=150>
                  <img      src=figures/ClaudeDebussy_5517.png
title="Debussy ve chvílích volna." width=150>
```

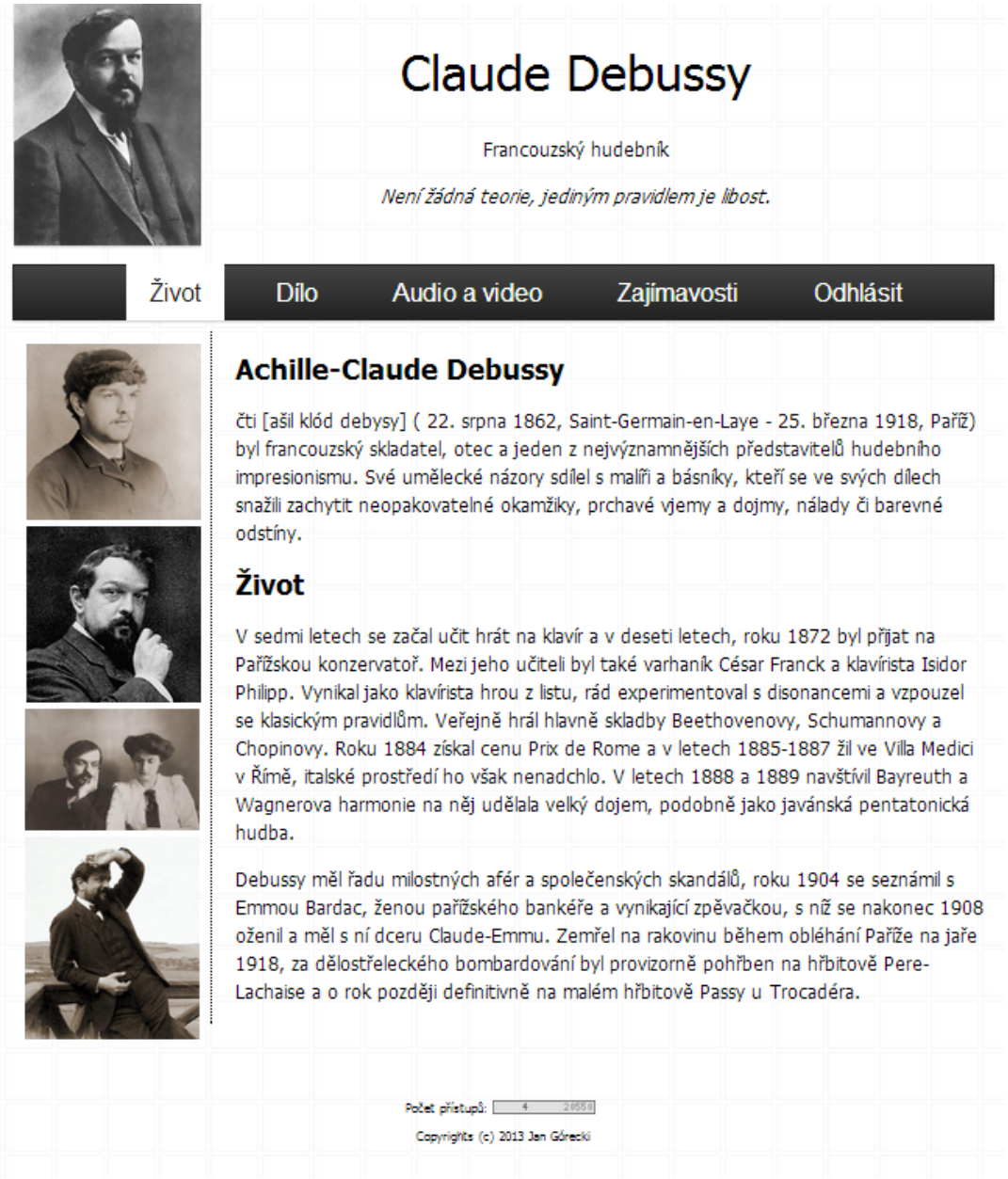
```
          </aside>
          <section>
            <article>
            <h2>Achille-Claude Debussy</h2>
            <p> čti [ašil klód debysy] ( 22. srpna 1862, Saint-
Germain-en-Laye - 25. března 1918, Paříž) byl francouzský
skladatel, otec a jeden z nejvýznamnějších představitelů
hudebního impresionismu. Své umělecké názory sdílel s malíři a
básníky, kteří se ve svých dílech snažili zachytit
neopakovatelné okamžiky, prchavé vjemy a dojmy, nálady či
barevné odstíny.

            <h2>Život</h2>
            <p>V sedmi letech se začal učit hrát na klavír a v
deseti letech, roku 1872 byl přijat na Pařížskou konzervatoř.
Mezi jeho učiteli byl také varhaník César Franck a klavírista
Isidor Philipp. Vynikal jako klavírista hrou z listu, rád
experimentoval s disonancemi a vzpouzel se klasickým
pravidlům. Veřejně hrál hlavně skladby Beethovenovy,
Schumannovy a Chopinovy. Roku 1884 získal cenu Prix de Rome a
v letech 1885-1887 žil ve Villa Medici v Římě, italské
prostředí ho však nenadchlo. V letech 1888 a 1889 navštívil
Bayreuth a Wagnerova harmonie na něj udělala velký dojem,
podobně jako javánská pentatonická hudba.
    <p>Debussy měl řadu milostných afér a společenských
skandálů, roku 1904 se seznámil s Emmou Bardac, ženou
pařížského bankéře a vynikající zpěvačkou, s níž se nakonec
1908 oženil a měl s ní dceru Claude-Emmu. Zemřel na rakovinu
během obléhání Paříže na jaře 1918, za dělostřeleckého
bombardování byl provizorně pohřben na hřbitově Pere-Lachaise
a o rok později definitivně na malém hřbitově Passy u
Trocadéra.

            </article>
          </section>
            <?php
      include("footer.html");
     ?>
      </body>
    </html>
```

Výsledkem kódu je pak tato úvodní stránka (Obrázek 19):

Obrázek 19: Vzhled celé uvodní stránky ukázkového portálu



Všimněme si tučně zvýrazněného kódu, který zajišťuje rozsvícení prvního odkazu v navigační sekci v hlavičce. Do levého pruhu `<aside>` je umístěno několik obrázku a do hlavní sekce `<section>` je pak umístěn jeden `<article>` s textem o životě skladatele. Připomeňme, že formát všech těchto prvků, jako je velikost, odsazení, okraje, atd., je důkladně nastaven ve stylopisu.

Nyní je potřeba ještě znovu zkontrolovat, zda vzhled stránky vyhovuje našim představám. Pokud ne, věnujme čas tomu, abychom vzhled dobře upravili, dříve než přistoupíme k implementaci dalších sekcí. Pokud již vše vypadá dle našich představ, můžeme přistoupit k implementací dalších sekcí. Další sekce tvoříme kopírováním odladěného

souboru `index.php`. Dále tedy vytvoříme soubor `dilo.php`, který odpovídá sekci Dílo. Kód může vypadat takto (Řešený příklad 141):

ŘEŠENÝ PŘÍKLAD 141

```
    <!DOCTYPE html>
    <html>
      <head>
        <meta charset="UTF-8">
        <title>Claude Debussy - francouzský hudebník - dílo
        </title>
        <link rel="stylesheet" href="style.css">
      </head>
      <body>
     <?php
     $menu = 2;
      include("header.php");
     ?>
          <aside>
              <img src=figures/noty.jpg width=150><p>
              <img     src=figures/debussy-syrinx-autograph.jpg
width=150><p>
          </aside>
          <section>
            <article>

               <h2>Dílo</h2>
                Od 90. let 19. století si Debussy vytvořil svůj
vlastní styl, ovlivněný dobovým symbolismem Stéphana Mallarmé
a často označovaný jako impresionismus, jakkoli se Debussy sám
tomuto označení bránil. Psal hlavně kratší díla, klavírní,
komorní i orchestrální hudbu a písně, často na slova Paula
Verlaine. 1890 napsal Bergamskou suitu pro klavír, jejíž třetí
věta, Au clair de lune na Verlainovu báseň, patří k
nejznámějším. Publikoval ji až po úpravách roku 1905 a řada
skladatelů ji později upravovala pro orchestr.
    <p>Roku 1894 napsal jednu ze svých nejslavnějších skladeb,
Preludium k Faunovu odpoledni, krátkou symfonickou báseň pro
orchestr, inspirovanou Mallarméovou básní. V letech 1893–1902
složil operu Pelleas a Melisanda, v níž mizí dominantní úloha
sólisty, který je postaven na roveň sborovému zpěvu. Z
orchestrální tvorby jsou nejdůležitější jeho symfonické skicy
Moře, cyklus Obrazy a skladba Oblaka z třídílného titulu
nazvaného Nokturna. Významná jsou též jeho drobná klavírní
díla, především cyklus 24 skladeb Preludia, obsahující mj.
slavné tituly "Potopená katedrála", "Dívka s vlasy jako len",
či "Mlhy", dále klavírní cykly Obrazy, Rytiny, Pro klavír,
Masky a další. Klavírní Preludia byla nedávno zpracována pro
orchestr současným belgickým skladatelem Lucem Brewaeysem.

            </article>
          </section>
```

```
        <?php
  include("footer.html");
 ?>

  </body>
</html>
```

Vzhled sekce Dílo je pak takovýto (Obrázek 20):

Obrázek 20: Vzhled sekce Dílo

Kód sekce Audio a video, který uložíme pod názvem `aav.php`, je tento (Řešený příklad 142):

ŘEŠENÝ PŘÍKLAD 142

```
    <!DOCTYPE html>
    <html>
      <head>
        <meta charset="UTF-8">
        <title>Claude Debussy - francouzský hudebník - audio a
video
        </title>
        <link rel="stylesheet" href="style.css">
      </head>
      <body>
     <?php
```

```
      $menu = 3;
      include("header.php");
     ?>
          <aside>
           <img      src=figures/Claude+Debussy+debussy1898.jpg
width=150><p>
                                    <img
src=figures/Claude+Debussy+With+Stravinsky+1911.png
title="Debussy s významýmným ruským skladatelem Stravinskym."
width=150><p>
               <img src=figures/debsatiesm.jpg title="Debussy a
Eric Satie." width=150><p>

          </aside>
          <section>

            <article>
                         <h2>Audio a video</h2>
                        <p><b>Pagodes</b> - klavírní skladba s
naprosto nezaměnitelným zvukem. Již od prvních tónu je slyšet
Debussy v nejčistší podobě. Ve své době byla skladby
považována za novodobý mysticismus.
                        <p   align=center><iframe   width="420"
height="315"          src="//www.youtube.com/embed/lswHSnJ0Rlw"
frameborder="0" allowfullscreen></iframe>
                        <p><b>Arabesque No. 1</b> - první ze
dvou Debussyho arabesek. Hravá skladba pro piáno, která už
naznačuje pozdější Debussyho výrazný styl.
                        <p   align=center><iframe   width="420"
height="315"          src="//www.youtube.com/embed/Yh36PaE-Pf0"
frameborder="0" allowfullscreen></iframe>
                        <p>   <b>Prélude à "L'après-midi d'un
faune"</b> - Preludium k Faunovu odpoledni - první výrazná
Debussyho skladba pro orchestr. Skladba se zcela vyhýbá
standardním hudebním postupům (viz citát záhlaví stránek) a
tehdejším hudebním teoretikům a analytikům není vůbec jasné,
jak tato skladba může vůbec "fungovat". Jinými je toto
považováno za projev geniality autora, např. Maurice Ravel
odpovídá na otázku, jakou skladbu by si nechal zahrát na
pohřbu (tehdy se to považovalo za největší poctu):
<i>"Preludium k Faunovu odpoledni. Protože ta jediná je
dokonalá."</i>
                        <p  align=center>  <iframe  width="420"
height="315"          src="//www.youtube.com/embed/ZVlyXh87b1g"
frameborder="0" allowfullscreen></iframe>

            </article>
          </section>

            <?php
       include("footer.html");
```

```
  ?>

   </body>
</html>
```

Vzhled této sekce je pak tento (Obrázek 21):

Obrázek 21: Vzhled sekce Audio a video

Kód pro sekci Zajímavosti je tento (Řešený příklad 143):

ŘEŠENÝ PŘÍKLAD 143

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Claude Debussy - francouzský hudebník -
zajímavosti
    </title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
 <?php
 $menu = 4;
  include("header.php");
 ?>
      <aside>
          <img src=figures/debussy-karik.jpg width=150><p>
          <img     src=figures/110812-CapletDebussy-225.jpg
title="Debussy s Capletem." width=150><p>
```

```
            <img            src=figures/Deb-kreslo-baba.png
width=150><p>

          </aside>
          <section>
            <article>

              <h2>Citáty</h2>
               <p><i>"Neposloucháme dost ony tisíce přírodních
zvuků kolem sebe a nepátráme dost po oné tak rozmanité hudbě,
kterou nám příroda nabízí s takovým nadbytkem. Obklopuje nás a
my v ní žijeme, aniž jsme ji vzali na vědomí. tady je podle
mého nová cesta. Ale – věřte mi – sotva jsme ji sám zahlédl,
protože to, co zbývá ještě udělat, je ohromné."</i>
               <p><i>"Člověk se uchází o pochvalu obecenstva,
jehož velká část se skládá z hlupáků. Je v tom kus směšného a
ironického protikladu."</i>
                 <p> Citát, který je v záhlaví stránek, je
Debussyho jasným názorem na to, jak se stavěl v jeho době
standardním zavedeným hudebním postupům. Toto je zřejmě
vlastní každému velkému hudebníkovi.


                              <h2>Řekli o něm</h2>
                              <p>Debussymu se naprosto a
skvěle podařilo obklopit patřičnou atmosférou Maeterlinckovo
drama. Výsledek mohl udivit i soudce nejvíc cvičené. Mnozí z
nich vykládali, že tam není vůbec hudby, zatímco je tam naopak
samá hudba, ale hudba tak přirozeně vtělená v akci, tak
přirozeně tryskající ze situací dekorací a jazyka, hudba tak
blízká hudbě řeči, že se stává nemožným v celkovém dojmu,
vytvořeném z tohoto zvukového splynutí, odloučit ji od textu,
který prolíná. – Paul Dukas o Debussyho opeře Pélleas a
Mélisanda
                              <p>Jas Péllease, Nokturen,
předehry k Faunovu odpoledni všude oslnil umělce a poznenáhlu
zvítězil u citlivého a inteligentního obecenstva. Vlivy
nezbytné u každého tvůrčího umělce, jež ho živily v jeho
prvních pokusech, byly stráveny a vytvořily osobnost silnou a
zároveň i harmonickou. Dokonce i druhá umění, básnictví s
Verlainem a Mallarmé, malířství s celou školou
impresionistickou tvoří k našemu hudebníkovi jakoby doprovod.
Proto také nelze dnes najít hudebníka, ketrý by tak či onak
nepocítil na sobě mistrův stín a který by mohl říci, že mu nic
nedluží. – Arthur Honegger

            </article>
            <article>
            </article>
          </section>
```

```
        <?php
    include("footer.html");
  ?>

    </body>
  </html>
```

Vzhled je pak takovýto (Obrázek 22):

Obrázek 22: Vzhled sekce Zajímavosti

Další sekce slouží pro přihlášení registrovaných uživatelů. Pro neregistrované uživatele sekce nabízí možnost registrace. Kód této sekce Přihlásit (`members.php`) pak vypadá takto (Řešený příklad 144):

ŘEŠENÝ PŘÍKLAD 144

```
    <!DOCTYPE html>
    <html>
      <head>
        <meta charset="UTF-8">
        <title>Claude Debussy - francouzský hudebník - členská
sekce
        </title>
        <link rel="stylesheet" href="style.css">
      </head>
      <body>
     <?php
     $menu = 5;
```

```
      include("header.php");
     ?>
          <aside>
              <img src=figures/Claude+Debussy+Debussy33.jpg
width=150><p>
          </aside>
          <section>
            <article>
                <form name="prihlaseni" action="overeni.php"
method="post">
                <h2>Přihlášení</h2>
                <table>
                <tr>
              <td>Uživatelské jméno:</td>
                <td><input type="text" name="login" value=""
></td>
            </tr>

    <tr>
      <td>Heslo:</td>
      <td><input type="password" name="password" value=""
></td>
    </tr>

    <tr>
      <td></td>
      <td align=right><input type="submit"
value="Odeslat"></td>
    </tr>
    </table>
    </form>
    <p>Nemáte přihlašovací údaje. Zde se můžete <a
href=register.php>registrovat.</a>
            </article>
          </section>

            <?php
      include("footer.html");
     ?>

      </body>
    </html>
```

Tento kód není nic jiného, než známy přihlašovací formulář z předchozí kapitoly vypadající takto (Obrázek 23):

Obrázek 23: Vzhled sekce pro přihlášení

Pokud uživatel ještě na stránkách registrován není, může se registrovat skrze odkaz na `register.php`. Ten má takovýto kód (Řešený příklad 145):

ŘEŠENÝ PŘÍKLAD 145

```
    <!DOCTYPE html>
    <html>
      <head>
        <meta charset="UTF-8">
        <title>Claude Debussy - francouzský hudebník - členská
sekce
        </title>
        <link rel="stylesheet" href="style.css">
      </head>
      <body>
     <?php
     $menu = 5;
     include("header.php");
     ?>
          <aside>
              <img     src=figures/Claude+Debussy+Debussy33.jpg
width=150><p>
          </aside>
          <section>
            <article>

    <form name="registrace" action="zapis.php" method="post">
    <h2>Registrace</h2>
    <table>
```

```
<tr>
<td>Jméno:</td>
<td><input type="text" name="jmeno" value="" ></td>
</tr>

<tr>
<td>Příjmení:</td>
<td><input type="text" name="prijmeni" value="" ></td>
</tr>

<tr>
<td>Město:</td>
<td><input type="text" name="mesto" value="" ></td>
</tr>

<tr>
<td>Uživatelské jméno:</td>
<td><input type="text" name="login" value="" ></td>
</tr>

<tr>
<td>Heslo:</td>
<td><input type="password" name="password" value="" ></td>
</tr>

<tr>
<td></td>
<td align=right><input type="submit" value="Odeslat"></td>
</tr>
</table>
</form>

        </article>
      </section>

        <?php
  include("footer.html");
 ?>

  </body>
</html>
```

Opět zde není nic jiného, než známý registrační formulář z předchozí kapitoly. Po přihlášení se dostaneme do členské sekce (`private.php`). Kód je jednoduchý (Řešený příklad 146):

ŘEŠENÝ PŘÍKLAD 146

```
    <!DOCTYPE html>
    <html>
      <head>
        <meta charset="UTF-8">
        <title>Claude Debussy - francouzský hudebník - členská
sekce
        </title>
        <link rel="stylesheet" href="style.css">
      </head>
      <body>
     <?php
     include('protection.php');
     $menu = 5;
     include("header.php");
     ?>
          <aside>
              <img     src=figures/Claude+Debussy+Debussy33.jpg
width=150><p>
          </aside>
          <section>
            <article>
                   <h2>Členská sekce</h2>
                   Zde můžete přidávat své příspěvky na stránky
o Claude Debussy...

            </article>
          </section>

            <?php
      include("footer.html");
     ?>

      </body>
    </html>
```

Nesmíme zapomenout na to, že tato sekce je neveřejná, a je tudíž třeba ji chránit před očima nepřihlášených uživatelů. To je zařízeno vložením skriptu `protection.php` (viz tučně) (Řešený příklad 147).

ŘEŠENÝ PŘÍKLAD 147

```
    <?php
     //nezapomeň nastartovat session!
     session_start();
     header("Cache-control: private");
```

```
     //Pokud není uživatel přihlášen, pošleme mu přihlašovací
formulář
     if ($_SESSION["user_is_logged"] != 1){
      header("Location: members.php");
      exit();
      }
    ?>
```

Výsledek pak vypadá takto (Obrázek 24):

Obrázek 24: Vzhled neveřejné sekce

Při kliknutí na odkaz Odhlásit pak přejdeme na sekci Přihlásit (`members.php`).

Věnujme pozornost tomu, jakým způsobem je odkaz na odhlášení zobrazen. Vše se děje skrze tento kód v hlavičce (Řešený příklad 148):

ŘEŠENÝ PŘÍKLAD 148

```
       <a <?php if ($menu == 5) {echo "id=bright";} ?>
        href="
        <?php if (isset($_SESSION) == 0) {session_start();}
        if (isset($_SESSION["user_is_logged"]) and
($_SESSION["user_is_logged"] == 1) ) {echo "logout.php";}
        else {echo "members.php";}
        ?>
        ">
        <?php
        if (isset($_SESSION["user_is_logged"]) and
($_SESSION["user_is_logged"] == 1) ) {echo "Odhlásit";}
        else  {echo "Přihlásit";}
         ?>
        </a>
```

Díky podmínce `(isset($_SESSION["user_is_logged"])` ověříme, zda existuje session proměnná `user_is_logged`. Pokud ano, ještě ověříme, zda je nastavena na hodnotu 1. Pokud platí obě podmínky, je uživatel přihlášen a v odkazu logicky vypíšeme Odhlásit a odkaz nastavíme na `logout.php`, který uživatele odhlásí. V opačném případě vypisujeme Přihlásit, které odkazem míří na přihlašovací formulář `members.php`. Kód `logout.php` pro odhlášení vypadá takto (Řešený příklad 149):

ŘEŠENÝ PŘÍKLAD 149

```
<?php
 session_start();
 $_SESSION = array();
 session_destroy();
 //Ověř zda došlo ke korektnímu ukončení session
 if ($_SESSION["user_is_logged"]){
  echo "FATAL ERROR: Cannot terminate session!";
  } else {
   //pokud je vše o.k., pošli uživatelovi přihlašovací
dialog
   //(nebo jinou  vhodnou stránku podle vlastního uvážení)
   header("Location: members.php");
  }
?>
```

Tento PHP kód odstraní session proměnnou `user_is_logged` a přesměruje prohlížeč na příhlašovací stránku `members.php`.

ZÁVĚR

Tématika webových portálů je a zřejmě i zůstane jednou z klíčových v oboru informatika. Čte-li čtenář tyto řádky, lze předpokládat, že již má za sebou studium všech kapitol této studijní opory a je schopný jednoduchý webový portál vytvořit a zajistit jeho administraci, což bylo hlavním cílem autorů této prakticky zaměřené studijní opory. Předpokladem tedy je, že čtenáři, resp. studenti předmětu Portál a jeho správa A, pro které je tato studiní opora určena, se seznámili se základními metodami tvorby struktury webových stránek pomocí HTML resp. XHTML a CSS, základy skriptovacího jazyka PHP, tvorbou a správou databáze MySQL a propojení všech uvedených komponent do funkčního celku – webového portálu - umožňujícího prezentaci informací a komunikaci s uživatelem prostřednictvím webových formulářů.

Nutno podotknout, vedle prezentace základních znalostí a dovedností s výše uvedenými jazyky a nástroji, bylo cílem této studijní opory prezentovat metodiku a základní principy tvorby webových portálů i pro studenty, kteří nejsou oborově zaměřeni přímo na informatiku a mají předmět Portál a jeho správa A zapsaný jako volně volitelný, jinými slovy je tato problematika zajímá a chtějí o ní získat aspoň základní povědomí. Autoři věří, že se jim to povedlo.

SEZNAM POUŽITÉ LITERATURY

Adaptic, s.r.o. Tvorba WWW - o webdesignu, grafice a reklamě [online]. 2012, 2012-05-20 [cit. 2013-05-21]. Dostupné z: http://www.tvorba-webu.cz/php/

CASTRO, E. HTML, XHTML a CSS. Praha: Computer Press, 2007. ISBN 978-80-251-1531-2.

GILMORE, W. J. Beginning PHP 5 and MySQL: From Novice to Professional. CA United States: Apress, 2005. ISBN 1893115518.

HAUSER, M. HTML a CSS Velká kniha řešení. Praha: Computer Press, 2006. ISBN 80-251-1117-2.

JANOVSKÝ, D. Jak psát web. [online] [cit. 2013-01-20]. Dostupné z: http://www.jakpsatweb.cz/

JARÝ, V. Sessions v PHP. [online] [cit. 2013-04-22]. Dostupné z: http:// http://jary.borec.cz/prog/sessions.php

KOSEK, J. PHP - Tvorba interaktivních internetových aplikací. Praha: Grada, 1998. ISBN 80-7169-608-0.

KOVÁŘ, L. <HTML> učebnice. [online] [cit. 2013-05-12]. Dostupné z: http://ra-we.webpark.cz/uceb/index.html

TVORBA-WEBU.CZ. HTML učebnice. [online] [cit. 2013-03-10]. Dostupné z: http://www.tvorba-webu.cz/html/html_ucebnice.php

VANDERWEEËN, B. Coding a layout in HTML5 and CSS3. [online]. 2012, 2012-02-12 [cit. 2013-02-08]. Dostupné z: http://www.bene.be/blog/comments/coding_a_layout_in_html5_and_css3/

WELLING, L., THOMSON, L. MySQL - Průvodce základy databázového systému. Praha: Computer Press, 2005. ISBN 8025106713.