UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH

PRÍRODOVEDECKÁ FAKULTA

ÚSTAV INFORMATIKY

DIPLOMOVÁ PRÁCA

JavaServer Pages

Košice 2006 **Slavomíra Krajčovičová**

Prehlásenie

Túto diplomovú prácu som vypracovala samostatne pod odborným vedením vedúceho práce RNDr. Jozefa Studenovského, CSc. a konzultanta Mgr. Petra Gurského, s použitím uvedenej literatúry.

.....

Slavomíra Krajčovičová

Poďakovanie

Ďakujem vedúcemu mojej diplomovej práce RNDr. Jozefovi Studenovskému, CSc. a konzultantovi Mgr. Petrovi Gurskému za usmernenie, cenné rady a pripomienky, ktoré mi poskytli pri vypracovávaní diplomovej práce.

Abstrakt

Táto práca sa venuje tvorbe dynamických webových stránok technológiou JavaServer Pages. Obsahuje prehľad súčasnej situácie vo vyučovaní JSP na vybraných slovenských fakultách. Súčasťou diplomovej práce je učebný text, ktorý sa skladá z troch lekcií. Tie postupne popisujú možnosti použitia JSP a základy tvorby JSP stránok, spracovávanie dát z formulárov a spoluprácu s databázou. Súčasťou teoretických častí lekcií sú aj obrázky a príklady pre lepšie pochopenie učiva. Učebný text obsahuje cvičenia, úlohy a autotesty na precvičenie a upevnenie získaných vedomostí a zručností. Práca ďalej obsahuje didaktické pokyny pre výučbu a vlastné skúsenosti s výučbou na PF UPJŠ a odporúčania pre prax.

Abstract

This work deals with dynamic web pages created by technology JavaServer Pages. It includes actual situation in teaching JSP at Slovak faculties. It includes coursebook organized to three lessons. It describes ways of using JSP, creating basic JSP web pages, form data processing and collaboration with a database. Coursebook includes figures, examples, tests and didactic instructions, too. Work describes my experiences in teaching JSP at PF UPJŠ and recommendation for practice.

Obsah

1. Úvod

Žijeme v dobe, kedy sa internet stáva nevylúčiteľnou súčasťou každodenného života. Stále viac bežných činností je možné vykonávať elektronickou formou. Pomocou internetu vykonávame činnosti, ktoré sú diskrétne alebo sú špeciálne pre každého jednotlivca. Preto je žiaduce, aby sa vytvárali webové stránky s individuálnym prístupom, stránky interaktívne, prispôsobené požiadavkám a potrebám jednotlivých používateľov. Do popredia sa preto dostávajú dynamické webové stránky.

Na tvorbu dynamických webových stránok slúži viacero programovacích jazykov. Jednou z možností je použiť JavaServer Pages (JSP).

V mojej diplomovej práci sa venujem práve tvorbe dynamických webových stránok pomocou technológie JavaServer Pages. Práca obsahuje prehľad súčasného stavu vo vyučovaní JSP, učebný text, didaktické pokyny, vlastné skúsenosti s výučbou a odporúčania pre prax.

Učebný text pozostáva z troch lekcií. Prvá z nich je venovaná základom práce s JSP, zameriava sa na zvládnutie vytvárania jednoduchých JSP stránok. V druhej lekcii je popísaná práca s formulármi a spracovávanie získaných dát. Tretia lekcia sa zaoberá spoluprácou s databázou. Každá lekcia obsahuje príklady, úlohy a autotest.

Súčasťou diplomovej práce je aj CD s elektronickou verziou učebného textu a všetkými skriptami použitými v texte. On-line verzia učebného textu je prístupná na adrese http://klud.ics.upjs.sk/~slavka/praca.

2. Prehľad súčasného stavu vo vyučovaní technológie JSP

Počítačová gramotnosť sa v dnešnej dobe považuje za nevyhnutnosť pre úspešné uplatnenie sa v profesionálnom živote. Preto v rámci každého študijného odboru na našich univerzitách absolvujú študenti výučbu informatiky a predmety, v ktorých sa učia ovládať informačné a komunikačné technológie.

V zisťovaní súčasného stavu, v ktorom som sa zaoberala situáciou vo vyučovaní tvorby dynamických www stránok pomocou technológie JSP, som sa zamerala na fakulty pripravujúce budúcich informatikov a učiteľov informatiky.

Na väčšine týchto fakúlt sa vyučujú predmety, v ktorých sa študenti oboznamujú s metódami tvorby dynamických www stránok. V rámci týchto predmetov sa preberajú rôzne klientské aj serverové technológie. Zo serverových technológií sa vo väčšine prípadov zameriavajú na jazyk PHP. Zoznamujú sa však aj s technológiou JSP a jej možnosťami. Vyučovaniu JSP sa venujú podrobnejšie na troch fakultách.

2.1. Prírodovedecká fakulta UPJŠ v Košiciach

Na tejto fakulte sa vyučuje Seminár z programovania v sieťach. Je vyučovaný v zimnom semestri v rozsahu 3 (3h seminár). Predmet je odporúčaný pre študentov piateho ročníka odboru informatika.

V rámci tohto predmetu majú študenti možnosť oboznámiť sa s rôznymi technológiami pre programovanie v sieťach. Jeden z trojhodinových seminárov je venovaný technológii JavaServer Pages.

2.2. Fakulta riadenia a informatiky Žilinskej Univerzity

Výučba JSP je na tejto fakulte zahrnutá v predmete Vývoj aplikácií pre internet a intranet. Tento predmet sa vyučuje v rozsahu 2 + 2 (2h prednáška, 2h laboratórne cvičenia) týždenne. Cieľom tohto predmetu je oboznámiť študentov s najnovšími technológiami vytvárania informačných systémov v prostredí Internetu a naučiť ich pracovať s vývojovými nástrojmi na tvorbu takýchto systémov.

Prednášky sú zamerané na prehľad základných protokolov Internetu, popisné jazyky HTML, XML, SGML, tvorbu interaktívnych stránok, formuláre, tvorba CGI

skriptov, klientské aj serverové technológie. V rámci serverových technológií je preberaná aj JSP.

V rámci cvičení sa študenti zoznamujú s internetom z užívateľského hľadiska, funkciami prehliadačov, editormi pre tvorbu www dokumentov, tvorbou jednoduchých i zložitejších www stránok, v rámci toho vytváranie stránok aj pomocou technológie JSP.

2.3. Fakulta elektrotechniky a informatiky Technickej Univerzity v Košiciach

Na tejto fakulte sa vyučuje predmet Technológie Javy pre študentov v druhej etape inžinierskeho štúdia v odbore Výpočtová technika a informatika. Predmet sa vyučuje v rozsahu 2 + 2 (2h prednáška, 2h cvičenia). Cieľom predmetu je oboznámiť študentov s jazykom Java, s jeho syntaxou a sémantikou, knižnicami tried a tvorbou aplikácií v jazyku Java. Preberajú sa alfanumerické, grafické aplikácie a sieťové aplikácie.

Jazyk PHP je na našich vysokých školách viac prezentovaný, ale aj technológia JSP má svoje miesto v informatických predmetoch.

3. Popis učebného textu

Rozvoj informačných a komunikačných technológií sa prejavuje vo väčšine oblastí ľudskej činnosti. Podpisuje sa aj v zmenách metód, foriem aj obsahu výučby v našich školách. Dostupnosť informačných technológií nám umožňuje zefektívňovať a modernizovať vyučovanie. Jednou z možností je aj projekt Virtuálnej univerzity, ktorá sa buduje na našej fakulte.

Tento učebný text, ako súčasť Virtuálnej univerzity, je určený pre vysokoškolských študentov. Poslúžiť môže aj samoukom, ktorý už majú aspoň nejakú skúsenosť s tvorbou webových stránok a jazykom JAVA a rozhodli sa s JSP zoznámiť.

3.1. Vstupné vedomosti a zručnosti

Pre zvládnutie učebného textu je potrebná znalosť jazyka HTML a programovacieho jazyka JAVA. Potrebné sú aj základné programátorské vedomosti a zručnosti. Pre prácu s databázou je potrebná znalosť jazyka SQL.

3.2. Výstupné vedomosti a zručnosti

Po zvládnutí učebného textu by mal študent vedieť :

- tvoriť jednoduché JSP stránky
- zabezpečiť interakciu webovej stránky prostredníctvom formulárov
- spracovávať údaje získané z formulárov
- ukladať získané informácie do tabuliek databázy
- opätovne pracovať s uloženými dátami
- poznať možnosti použitia JSP a princíp fungovania
- aplikovať vedomosti pri tvorbe jednoduchých aj zložitejších webových aplikácií

3.3. Štruktúra učebného textu

Učebný text sa skladá z troch lekcií. Každá lekcia obsahuje teoretický výklad. Táto teoretická časť je obohatená o riešené príklady, na ktorých čitateľ vidí praktické použitie nových poznatkov. Na precvičenie zručností a upevnenie vedomostí slúžia na konci každej lekcie cvičenia a úlohy. V rámci každej lekcie má študent možnosť overiť si svoje získané vedomosti v autoteste.

3.4. Didaktické pokyny k jednotlivým lekciám

3.4.1. Lekcia 1

Prvá lekcia sa zaoberá základmi práce s technológiou JavaServer Pages. Ťažiskom lekcie je predstavenie tejto technológie, jej možností a popísanie princípu fungovania. Študenti sa zoznámia s tvorbou jednoduchých JSP stránok. V lekcii sú popísané základné príkazy potrebné na tvorbu JSP stránok.

Ciele :

- poznať princíp fungovania JSP
- poznať možnosti využitia technológie JSP
- vedieť tvoriť jednoduché JSP stránky
- vedieť používať implicitné objekty a direktívy na stránkach

3.4.2. Lekcia 2

Druhá lekcia je venovaná spracovávaniu dát z formulárov. Formuláre sú súčasťou jazyka HTML, preto je pravdepodobné, že sa študenti s ich tvorbou už stretli. Považujem však zopakovanie vytvárania formulárov za užitočné. Hlavnou úlohou tejto lekcie je popísať, ako získať údaje zo vstupných polí formulára a spracovávanie dát od užívateľa.

Ciele :

- vedieť tvoriť formuláre
- poznať rozdiel medzi metódou POST a GET
- vedieť spracovávať údaje z jednotlivých polí formulára
- vedieť tvoriť JSP stránky, ktoré pracujú s údajmi získanými z formulára

3.4.3. Lekcia 3

Táto lekcia sa zameriava na vysvetlenie spolupráce JSP s databázovým systémom. Technológia JSP vie spolupracovať s väčšinou obľúbených databázových systémov. Výklad je orientovaný na spoluprácu s MySQL. Hlavnou úlohou lekcie je popísať možnosť uchovávania dát v databáze a následne ďalšej manipulácie s nimi.

Ciele :

- poznať princíp webovej databázovej architektúry
- poznať príkazy na vytvorenie a ukončenie spojenia s databázou, príkazy na aktualizáciu databázy a získavanie záznamov
- vedieť tvoriť jednoduché webové aplikácie s využitím získaných vedomostí a zručností

3.5. Spôsob hodnotenia

Hlavnou časťou hodnotenia by malo byť vypracovanie a odovzdanie samostatného projektu. Odporúčam ponúknuť témy projektov a nechať výber témy projektu voľne na študentoch. Hodnotiť by sa mala využiteľnosť v praxi, originalita, náročnosť riešenia a úprava.

Návrh tém projektov :

1. Webová aplikácia na zisťovanie spätnej väzby od návštevníkov pre tvorcu webovej lokality.
2. Databáza s vyhľadávaním filmových titulov.
3. Webová aplikácia pre študentskú diskusnú skupinu.
4. Webová aplikácia pre správu študijných výsledkov.

4. Učebný text

4.1. Lekcia 1 – Základ tvorby JSP stránok

4.1.1. Možnosti použitia JSP

Predstavte si, že vediete doučovací krúžok z matematiky. Pripravujete svojich žiakov na prijímacie skúšky na strednú školu. Chcete vytvoriť www stránku s matematickými testami, kde by sa mohli študenti prihlasovať a testovať sa. Vaša stránka by mala byť individuálne prispôsobená každému podľa jeho doterajších výsledkov, alebo iných špeciálnych požiadaviek. Žiaci by si mohli vyberať rôzne testy, nechať si ich hneď vyhodnotiť, prípadne sa neskôr vrátiť k problémovým úlohám, nechať si poradiť alebo pozrieť riešenie. A samozrejme, vy by ste mali vidieť dosiahnuté výsledky svojich žiakov. Tvorenie, napríklad takejto, dynamickej stránky umožňuje technológia JSP.

JSP – JavaServer Pages je jednou z technológií na vytváranie dynamických www stránok. Teda stránok prispôsobujúcich sa používateľovi, stránok interaktívnych, reagujúcich na impulzy používateľa.

Metóda programovania JSP spočíva v kombinácii jazyka HTML na tvorenie statického obsahu stránky a programovacieho jazyka JAVA, pomocou ktorého môže programátor vytvárať dynamické súčasti stránky.

Java Server Pages je serverová technológia. To znamená, že všetky požiadavky používateľa, či príkazy programátora sa vykonávajú a spracovávajú na strane servera (serverovom počítači). Výhodou je, že takto vytvorené dokumenty môžu zdieľať viacerí používatelia.

V porovnaní so statickými stránkami nám JSP poskytuje širšie možnosti, ako sú napríklad:

- vytváranie stránok prispôsobených špeciálnym požiadavkám jednotlivých používateľov
- získavanie údajov z formulárov a ich ďalšie spracovanie, a teda interakcia s používateľom prostredníctvom formulárov
- spolupráca s databázou
- kontrola toku spracovania dát

Technológia JSP nám umožňuje plne využívať programovací jazyk JAVA, objekty modelu JavaBeans, vkladať vlastné, vopred pripravené javovské triedy.

Technológia JSP je len jedným z prostriedkov na tvorbu dynamických stránok. Porovnajme výhody aj nevýhody jednotlivých prostriedkov.

Výhodou JSP proti ASP (Active Server Pages) je, že dynamická časť kódu je písaná v jazyku JAVA a nie vo VBScripte alebo inom jazyku určenom pre ASP. JAVA je komplexnejší jazyk a je prenositeľná, teda nie ste obmedzovaní použitým operačným systémom či serverom. Výhodou ASP je vysoká podpora komerčných programov a jednoduchá tvorba stránok. Na druhej strane pri tvorbe stránky pomocou vývojového programu dostaneme stránky, ktoré sú väčšie čím spomaľujú počítače s pomalším prístupom na internet.

Výhodou JSP proti PHP je že môžeme niektoré súčasti kódu (triedy) používať aj v iných programoch napísaných v JAVE. Pokiaľ si niekto vyberá, ktorú z týchto dvoch technológií použiť, tak v prípade, že je už zbehlý v JAVE, pri prechode na JSP by nemal mať žiadne problémy. Jazyk PHP je potrebné sa učiť od základov, aj keď niektoré základné programové štruktúry sú veľmi podobné jazyku JAVA. Jazyk PHP je podstatne viac používaný hlavne preto, že nevyžaduje deklarovanie typov premenných. Nevyžaduje teda pretypovávanie a konvertovanie.

Výhodou JSP proti Servletom je, že oddeľuje statický a dynamický obsah stránky. Kód JSP je tým pádom aj oveľa kratší a prehľadnejší. Pri spúšťaní JSP stránky sa jej obsah najprv preloží do servletu a až potom sa vykoná. JSP je vhodné použiť v prípade, že sa nemení celý obsah stránky. Je potrebné povedať, že to, čo sa dá vytvoriť v JSP, dá sa vytvoriť aj v Servletoch.

4.1.2. Dokument JSP

Vieme už, čo to vlastne technológia JavaServer Pages (JSP) je, akému účelu slúži, načo všetko sa dá použiť. Pozrime sa teraz na konkrétnom príklade, ako môže taká JSP stránka vyzerať. Popíšeme si, z akých častí sa skladá a čo sa pri jej zavolaní udeje.

Ako som už spomenula, stránka JSP je spojením HTML a programovacieho jazyka Java. Statická HTML stránka sa stáva dynamickou vložením príkazov v jazyku JAVA. Na to, aby server, ktorý náš dokument bude spracovávať, vedel, že ide o jazyk

JAVA a nie obyčajný text, je potrebné tieto príkazy uzatvoriť do špeciálnych značiek (tzv. tagov). Jednoduchý príkladom toho, ako sa to dá spraviť môže byť nasledujúca stránka.

```
//prva.jsp
<HTML>
<HEAD><TITLE>jednoduchá stránka JSP</TITLE></HEAD>
<BODY>
<BR>
  <%!int hodnota; %>
  Vaša
      <% hodnota=1;
         out.print(hodnota); %>. JSP stránka.<br>
</BODY>
</HTML>
```

Naša stránka, ktorej kód tu vidíme sa volá *prva.jsp*. Táto stránka je ukážkou dynamickej web stránky vytvorenej pomocou JSP. Všimnime si teraz, akú štruktúru má náš kód. V prvých riadkoch je použitý jazyk HTML, pre nás nič nové. Stránka bude mať hlavičku : „jednoduchá stránka JSP“. Zaujímavejšie je to až v tele dokumentu. Po značke na vloženie nového riadku je ďalšia časť kódu napísaná v JAVE. V tagoch je uzavretý javovský príkaz, ktorý deklaruje premennú typu `int` s názvom hodnota. V ďalšom riadku je obyčajný text. V ďalšej časti je opäť javovský kód uzavretý v špeciálnych značkách. Nachádza sa tam priradzovací príkaz, ktorý priradí do premennej `hodnota` číslo 1 a príkaz `out.print(hodnota)`, ktorý túto hodnotu „vytlačí“, čo vlastne znamená , že hodnota 1 bude zobrazená vo výslednej stránke, ktorá sa objaví v prehliadači používateľa. Ďalšie časti sú opäť obyčajným HTML kódom.

Na tomto jednoduchom príklade sme si ukázali, ako taký dokument môže vyzerať. Teraz je však určite namieste povedať, čo sa vlastne udeje po spustení tejto stránky. Na to si odpovieme v ďalšej časti.

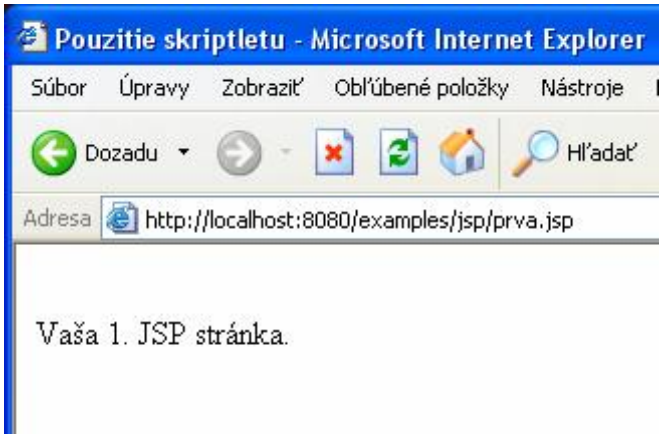
4.1.3. Testovanie JSP

Na to, aby sme zistili, či nami pripravený dokument vykonáva to čo má, potrebujeme ho nejakým spôsobom otestovať. V tejto časti si popíšeme jeden z možných postupov, ako to urobiť.

Ako prvé si musíme povedať, že na to, aby sme mohli nami vytvorený dokument testovať, potrebujeme kontajner JSP. Na testovanie môžeme použiť napríklad server

Tomcat, ktorý JSP správne implementuje. Popis inštalácie a potrebné nastavenia, ako aj inštalačné programy nájdete na priloženom CD v adresári install.

Ďalším dôležitým faktom je, že každý dokument vytvorený technológiou Java Server Pages musí mať príponu .jsp. Tento dokument je potrebné uložiť do adresára, ktorý je v kontajneri nastavený ako adresár jsp stránok. (Konkrétny popis, kam ukladať naše stránky je uvedený na priloženom CD v adresári install.) Po uložení našej stránky pod názvom prva.jsp môžeme ju v prehliadači spustiť.



Obrázok 1

Ak sme náš dokument uložil na správne miesto a spustili sme server tomcat, môžeme začať s testovaním. Spustíme si webový prehliadač a do poľa adresa napíšeme http://localhost:8080/examples/jsp/prva.jsp .

Na obrázku 1 môžete vidieť, čo sa zobrazí v okne klientského prehliadača po spracovaní požiadavky na stránku *prva.jsp* .

Takže teraz už vieme, že náš kód po spracovaní vytvorí stránku, na ktorej bude napísaný text „Vaša 1.JSP stránka.“

4.1.4. Ako funguje technológia JSP

Popíšeme si základné črty fungovania JSP, ako reaguje server na požiadavku používateľa.

1. Nezabúdajme na to, že náš dokument JSP je potrebné uložiť na správne miesto, teda do adresára, na ktorom beží kontajner JSP, dokument musí mať príponu .jsp (napr. prva.jsp).
2. Pri prvej klientskej požiadavke kontajner JSP preloží náš dokument na servlet (javovský program, vzniká _prva.java).

3. Kontajner tento servlet uloží do špeciálneho adresára a skompiluje (preloží do bajtového kódu, vzniká _prva.class).
4. Server pracuje s preloženým servletom (v našom prípade so súborom _prva.class), pri prvej požiadavke vytvorí nový objekt a inicializuje všetky premenné, vyhodnotí požiadavku a odošle výsledný dokument na klientský počítač.
5. Pri opätovných požiadavkách na daný dokument kontajner už len vyhodnocuje požiadavky a odosiela výsledný dokument, nevytvára nový objekt a neinicializuje nanovo premenné.
6. Ak bol dokument uvoľnený z pamäte (pre nedostatok miesta, vypnutie kontajnera, ...) , tak pri ďalšej požiadavke kontajner načíta a inicializuje súbor (_prva.class).

Vráťme sa k nášmu príkladu *prva.jsp* a povedzme si, čo sa s ním udeje po uložení.

Po uložení do adresára `<tomcat>\webapps\examples\jsp\prva.jsp` kontajner zistí vďaka správnej prípone , že ide o dokument jsp a preloží ho na servlet. Preloží ho do bajtového kódu a uloží si ho do špeciálneho adresára. Dokument obsahuje jedinú premennú s názvom `hodnota`. Po prvej klientskej požiadavke na daný dokument server vytvorí nový objekt a inicializuje premennú `hodnota`. Priradí jej hodnotu 1 a vytvorí výsledný dokument, ktorý obsahuje už len HTML kód a v ňom vloženú hodnotu 1 z premennej `hodnota`, a odošle ho na klientský prehliadač. V prípade opakovanej požiadavky server priradí tej istej premennej `hodnota` číslo 1 a odošle výsledný dokument, ktorý zobrazí na klientskom prehliadači text : „Vaša 1.JSP stránka.“

V prípade, že bude vypnutý kontajner JSP, alebo bude náš dokument uvoľnený z pamäte, po novej požiadavke klienta server vytvorí nový objekt a inicializuje novú premennú `hodnota`, ktorej priradí 1 a vytvorí výsledný dokument, do textu ktorého priradí hodnotu 1 premennej `hodnota`.

4.1.5. Základné programové bloky JSP

Vieme už, že pomocou technológie JSP môžeme vytvárať dynamické webové stránky. Popísali sme si, ako táto technológia funguje a čo potrebujeme k tomu, aby sme

nami vytvorené stránky mohli spúšťať. Teraz sa už môžeme pustiť do samotného vytvárania www stránok.

Ako už bolo spomenuté, okrem HTML kódu obsahujú dokumenty JSP aj kód v jazyku JAVA. Tento sa uzatvára do špeciálnych značiek a vytvára dynamický obsah stránky. Na programovanie dynamických častí stránok sa využíva viacero konštrukcií . V tejto časti si priblížime skriptovacie značky. Tvoria základné stavebné prvky dokumentu JSP. Skriptovacie značky v JSP sú tri : deklarácie, výrazy a skriptlety. Na zaznamenanie poznámok sa používajú komentáre JSP.

Postupne sa budeme venovať všetkým základným programovým blokom. Naučíme sa, akú majú jednotlivé elementy štruktúru, kedy je vhodné ich použiť. Ukážeme si na príkladoch ich funkčnosť a význam.

4.1.5.1. Deklarácie, výrazy a skriptlety

Povedali sme si, že pri tvorbe JSP stránok využívame programovací jazyk JAVA. Časti javovského programu sa na stránkach JSP vkladajú do jednotlivých skriptovacích značiek.

Ak sa chystáme tvoriť dynamické webové stránky, je pravdepodobné, že sa nevyhneme použitiu premenných. Definovať premenné, metódy a polia môžeme dvoma spôsobmi. Tieto definície môžeme vložiť do deklarácie alebo skriptletu. Popíšeme si, ako to urobiť a aký je rozdiel pri voľbe jednej či druhej možnosti.

Každá deklarácia v JSP musí byť uzavretá v špeciálnych tagoch `<%!` a `%>` . Medzi tieto tagy vkladáme deklarácie v jazyku JAVA. Ak by sme napríklad chceli deklarovať reťazec a vložiť do neho text, deklarácia by vyzerala takto :

```
<%! String ahoj=”Ahoj!”; %>
```

Tento príkaz vytvorí reťazec `ahoj` a vloží do neho text „Ahoj!“.
Takýto istý výsledok dosiahneme aj použitím nasledujúceho skriptletu :

```
<% String ahoj=”Ahoj!”; %>
```

A v čom je rozdiel? Deklarácie sa spúšťajú pri prvej návšteve stránky, alebo pri opätovnom spustení kontajnera JSP, či uvoľnení JSP stránky z pamäte. Skriptlety sa vyhodnocujú pri každej návšteve stránky. Rozdiel je preto v platnosti a životnosti premenných. Ak definujeme premennú v deklarácii, bude inicializovaná len pri prvej

návšteve stránky alebo vo výnimočnej situácii. Pri definovaní v skriptlete sa bude inicializovať nanovo pri každej návšteve stránky (vždy sa vytvorí nová inštancia).

Pri programovaní väčšinou potrebujeme používať viacero premenných či metód. Pri ich definovaní stačí použiť jednu deklaráciu alebo skriptlet, ktorý môže obsahovať viacero definícií. Príklad zloženej deklarácie si môžete pozrieť vo výpise:

```
<%! int pocet=2;
    public int getInt()
    { return pocet;}
%>
```

O deklaráciách treba povedať, že nevytvárajú žiaden výstup. Slúžia len na definovanie. Kombinujú sa s výrazmi a skriptletmi.

Tvorca dynamickej www stránky nevystačí len s deklaráciami. Potrebuje tvoriť výstup vo forme stránky. Čo v prípade, ak chceme na stránku umiestniť obsah premennej či metódy?

Ak chceme vložiť výstup priamo do výslednej www stránky (väčšinou ide o hodnotu premennej), môžeme použiť výraz JSP. Ďalšou možnosťou je použite skriptletu. O tej si povieme neskôr. Každý výraz musí byť uzavretý v tagoch `<%=` a `%>`. Medzi ne vložíme výraz v jazyku JAVA. Ak by sme napríklad chceli na stránku zobraziť hodnotu premennej `pocet` zvýšený o 1, výraz by vyzeral takto :

```
<%= pocet+1 %>
```

Keď používateľ navštívi stránku, ktorá obsahuje takýto výraz, server najprv zistí akú hodnotu má premenná `pocet` . Potom túto hodnotu zvýši o 1 a výsledok vloží do výsledného dokumentu, ktorý server vytvára a pošle ho späť klientskému prehliadaču. Ten hodnotu premennej `pocet` zväčšenú o 1 zobrazí na www stránke, kde ju uvidí používateľ.

Spojme teraz, čo už vieme o deklaráciách a výrazoch a vytvorme JSP stránku, na ktorej naše vedomosti využijeme. Príkladom stránky používajúcej deklarácie a výrazy je napríklad *ahoj.jsp*, ktorej zdrojový kód si môžete pozrieť vo výpise.

```
//ahoj.jsp
<HTML>
<HEAD><TITLE>Pozdrav</TITLE></HEAD>
<BODY>
<%! String ahoj="Ahoj!"; %>
<%= ahoj %> Vítam Ťa !
```

```
</BODY>
</HTML>
```

Všimnime si zdrojový kód stránky *ahoj.jsp*. Deklarácia obsahuje príkaz `String ahoj="Ahoj!";.` Pri spracovávaní tejto deklarácie serverom sa vytvorí reťazec `ahoj` a priradí sa mu text „Ahoj!“. Výraz JSP obsahuje premennú `ahoj`, čo znamená, že pri spracovávaní stránky sa do výsledného dokumentu vloží obsah reťazca `ahoj`. Ďalšie časti zdrojového kódu sú bežným HTML kódom. Ako bude vyzerať výsledná stránka ?

Čo uvidí návštevník našej stránky *ahoj.jsp* vo svojom prehliadači si môžete pozrieť na obrázku :

Obrázok 2

Určite ste si všimli jeden problém. A to, že prehliadač nezobrazil správne diakritiku, ktorú sme použili v našom zdrojovom kóde. Je to preto, že kontajner JSP jakarta-tomcat pri vytváraní výstupných dokumentov implicitne využíva kódovanie ISO-8859-1. My by sme chceli využiť kódovanie pre stredoeurópske jazyky. Jednou z možností, ako tento problém vyriešiť, je vloženie direktívy `page` do nášho dokumentu. Ak ju napíšeme v nasledujúcom tvare, nastavíme kódovanie pre stredoeurópske jazyky :

```
<%@ page contentType="text/html;charset=windows-1250"
%>
```

Ak doplníme zdrojový kód našej stránky *ahoj.jsp* o direktívu `page` v uvedenom tvare, text vo výslednom dokumente sa zobrazí správne . Ako bude vyzerať upravený dokument si môžete pozrieť vo výpise. Čo zobrazí klientský prehliadač, si môžete pozrieť na obrázku.

```
//ahoj.jsp
<%@ page contentType="text/html;charset=windows-1250"
%>
<HTML>
<HEAD><TITLE>Pozdrav</TITLE></HEAD>
<BODY>
<%! String ahoj="Ahoj!"; %>
<%= ahoj %> Vítam Ťa !
</BODY>
</HTML>
```

Obrázok 3

Dôležité je povedať, že každý výraz na stránke JSP musí byť napísaný samostatne. Výrazy sa vyhodnocujú pri každej návšteve stránky.

Najvšeobecnejším programovacím blokom je skriptlet. Medzi počiatočný a koncový znak skriptletu môžeme vkladať ľubovoľný javovský kód (<% kód v Jave %>). Kód v jednom skriptlete môže, ale nemusí byť celistvý. Medzi skriptlety, ktoré spolu vytvoria samostatnú zmysluplnú časť kódu v Jave môžeme vložiť deklarácie, výrazy aj kód v jazyku HTML. Vo výpise si môžete pozrieť, ako skriptlet môže vyzerať:

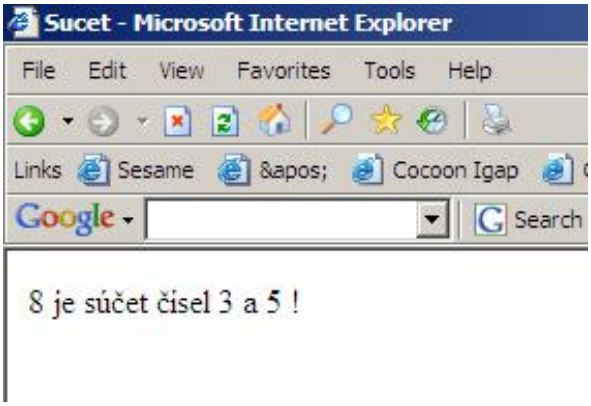
```
<% int a=3;
   int b=5;
   int c=a+b;
%>
```

V tomto skriptlete je jednoduchý javovský kód, ktorý vypočíta súčet čísel 3 a 5. Najprv dané premenné vytvorí a inicializuje na priradené hodnoty. Vypočíta ich súčet a výsledok vloží do premennej c.

Vyskúšajme doplniť skriptlet z výpisu na úplný zdrojový kód stránky, ktorá výsledok súčtu aj zobrazí. Tento skriptlet spojíme s výrazom a vytvoríme stránku *skriptlet.jsp* .

```
//skriptlet.jsp
<%@ page contentType="text/html;charset=windows-1250"
%>
<HTML>
<HEAD><TITLE>Sucet</TITLE></HEAD>
<BODY>
<% int a=3;
   int b=5;
   int c=a+b;%>
<%= c %> je sucet cisel <%= a %> a <%= b %> !
</BODY>
</HTML>
```

V zdrojovom kóde je napísaný vyššie spomínaný skriptlet. Za ním nasleduje výraz obsahujúci výslednú hodnotu premennej c. Výraz použijeme preto, aby sa hodnota premennej c zobrazila vo výslednom dokumente. Na konci náš dokument obsahuje HTML kód. Po zaslaní požiadavky na túto stránku ju server spracuje. Znamená to, že v skriptlete inicializuje premenné, urobí ich súčet a pomocou výrazu výsledok súčtu vloží do výsledného dokumentu spolu s HTML kódom. Dokument odošle späť klientskému prehliadaču. Na obrázku môžete vidieť, čo zobrazí prehliadač.



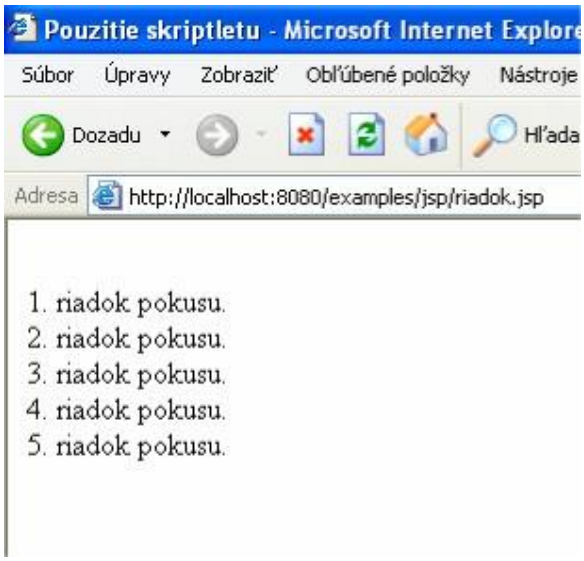
Obrázok 4

Skriptlet je možné rôznymi spôsobmi spájať s ostatnými súčasťami stránky. Javovské príkazy v skriptlete, ktoré majú spoločne niečo vykonať, môžu byť prerušené vložením deklarácie, výrazu alebo HTML kódu. Príkladom stránky JSP, v ktorej sú skriptlety poprekladané ďalšími súčasťami stránky, je stránka `riadok.jsp` . Zdrojový kód dokumentu *riadok.jsp* je uvedený vo výpise:

```
//riadok.jsp
<HTML>
<HEAD><TITLE>Pouzitie skriptletu</TITLE></HEAD>
<BODY>
```

```
<BR>
<% for (int i=1;i<=5; i++){
%>
<%= i %>. riadok pokusu.<br>
<%
   }
%>
</BODY>
</HTML>
```

Ak sa pozrieme na zdrojový kód, uvidíme, že obsahuje dva skriptlety, medzi ktorými je vložený výraz JSP a HTML text. V prvom skriptlete je začiatok príkazu `for` a v druhom sa tento príkaz ukončuje. Vidíme, že cyklus `for` sa vykoná 5 krát. Pri každom vykonaní sa do výsledného dokumentu vloží hodnota premennej `i` a HTML text. Ako bude vyzerať výsledná stránka v prehliadači po odoslaní požiadavky na stránku *riadok.jsp* a spracovaní tejto požiadavky serverom ukazuje obrázok:



Obrázok 5

Ak sa vám nepáči takéto prekladanie skriptletov s inými súčasťami stránky, môžete celú stránku napísať ako skriptlet, v ktorom použijete príkaz `out.print()` (prípadne `out.println()`). Spracovanie tohto príkazu vytvorí rovnaký výsledok ako použitie výrazu. Zdrojový kód stránky potom bude vyzerať takto :

```
//riadok.jsp
<HTML>
<HEAD><TITLE>Pouzitie skriptletu</TITLE></HEAD>
<BODY>
<BR>
```

```
<% for (int i=1;i<=5; i++){
       out.print(i) ;
       out.println(". riadok pokusu.<br>");
   }
%>
</BODY>
</HTML>
```

Výsledný dokument bude vyzerať úplne rovnako.

4.1.5.2. Komentár v JSP

Ak budete pracovať na nejakej zložitejšej úlohe, je zrejmé, že ju nevyriešite bez prestávky. Ak sa k rozpracovanej úlohe vrátite po dlhšom čase, určite vám pomôže, ak budete mať k dispozícii poznámky o doteraz urobenej práci. Na to nám poslúžia komentáre JSP. Komentár v JSP má takúto syntax : <%-- JSP komentár --%>. Slúži na rovnaký účel ako normálny komentár HTML. Rozdiel je v tom, že sa tento komentár nikdy neodosiela na klientskú stranu. Pozrime sa na predošlom príklade, ako možno komentáre JSP použiť.

```
//riadok.jsp
<HTML>
<HEAD><TITLE>Pouzitie skriptletu</TITLE></HEAD>
<BODY>
<BR>
<% for (int i=1;i<=5; i++){
%>
<%= i %>. riadok pokusu.<br>
<%
   }
%>
<%-- vypíše na stránku 5 riadkov --%>
</BODY>
</HTML>
```

4.1.6. Implicitné objekty

Je pravdepodobné, že pri vytváraní webových stránok sa niektoré objekty a ich metódy využívajú častejšie. Získavanie informácii o používateľovi, jeho prehliadači a špecifikách, ktoré ho charakterizujú, následné prispôsobenie výslednej stránky sú pri programovaní webových stránok veľmi dôležité k tomu, aby sa stránky zaradili k dynamickým. V technológii JSP na často opakujúce sa príkazy môžete využiť niektorý zo skupiny implicitných objektov.

Implicitný objekt je identifikátor, pomocou ktorého kontajner JSP sprístupňuje objekt príslušnej triedy. Pomocou nich vieme zistiť, či upraviť rôzne zaujímavé vlastnosti stránky. Nie je potrebné ich definovať. Implicitné objekty na stránkach JSP sú tvorené automaticky pri preklade dokumentu JSP na servlet. Tieto objekty sú dostupné vo všetkých výrazoch a skriptletoch v celom dokumente. Nie sú dostupné v deklarácii, ale v deklarácii si ich môžeme zadefinovať a zavolať v skriptlete.

Implicitnými objekty v JSP sú :

- request
- response
- out
- session
- application
- config
- page
- pageContext
- exception

Jeden s implicitných objektov sme už použili vo vyššie uvedenom príklade *riadok.jsp.* Použili sme príkaz `out.print()`. Mohli sme tak urobiť preto, lebo `out` je implicitným objektom. Objekt `out` sa odkazuje na text odosielaný na klientský počítač. Obsahuje dve dôležité metódy: `print()` a `println()`. Obe tieto metódy pridávajú text do dokumentu zobrazovaného vo webovom prehliadači. Jediný rozdiel je v tom, že `out.println()` pridá k výstupu znak nového riadku.

Niektoré ďalšie implicitné objekty si popíšeme pri konkrétnych príkladoch, v ktorých ich použijeme.

4.1.7. Direktívy JSP

S jednou direktívou sme už pracovali v predošlej časti textu. Pri tvorbe stránky *ahoj.jsp* sme použili direktívu `page`. Tá zabezpečila, že sa text v klientskom prehliadači zobrazil s nami požadovanou diakritikou.

Direktíva JSP je inštrukcia, pomocou ktorej oznamujeme kontajneru, ako má zostaviť určité pasáže programového kódu, ktoré sa stanú súčasťou nového programu v jazyku JAVA. Direktíva začína značkou <%@ a končí značkou %>. Medzi týmito znakmi je zapísaný názov direktívy a za ním je uvedený jeden alebo viacero atribútov v tvare názov=hodnota. Teda formát direktívy JSP vyzerá takto :

```
<%@  názov_direktívy názov_atrib1=hodnota_atrib1;... %>
```

Direktívami JSP sú:

- include
- page
- taglib

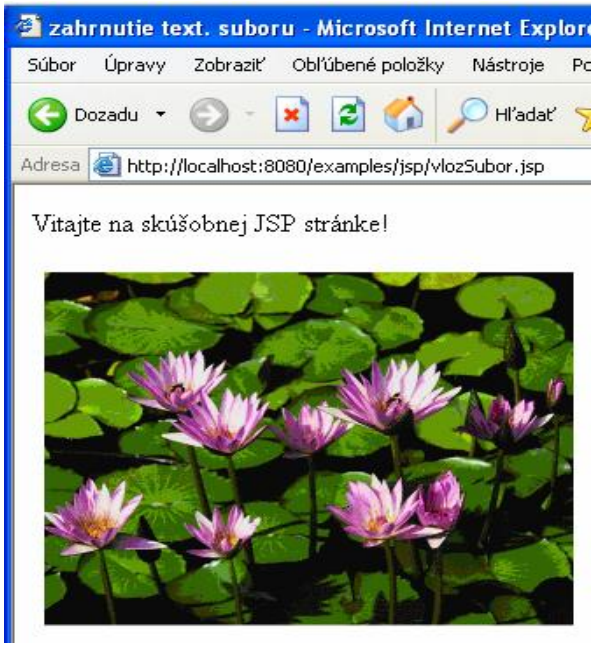
Pri vytváraní webových stránok sa nám môže ľahko stať, že určitý text budeme chcieť zobrazovať na všetkých, alebo aspoň na viacerých z nich. Dobrým príkladom je poznámka o autorských právach, o tvorcovi alebo správcovi danej lokality. Aby sme nemuseli tento text prepisovať na každú novú stránku, kde by sa mal objaviť, ponúka nám technológia JSP direktívu `include`.

Direktíva `include` nám slúži na vkladanie obsahu textového súboru na akúkoľvek stránku na miesto, kde chceme, aby sa obsah súboru objavil. Príkladom JSP stránky, na ktorú sa bude vkladať obsah iného súboru je dokument *vlozSubor.jsp*, ktorého zdrojový kód je uvedený vo výpise. Textový súbor, ktorý je zahrnutý do JSP stránky je uvedený pod ním.

```
//vlozSubor.jsp
<HTML>
<HEAD><TITLE>vloženie text. súboru</TITLE></HEAD>
<BODY>
<%@ page contentType="text/html;charset=windows-1250"
%>
<%@ include file="hlavicka.txt" %>
<BR>
<BR><img src="obrazok.gif"></th>
</BODY>
</HTML>
```

```
//hlavicka.txt
Vitajte na skúšobnej JSP stránke!
```

Pozrime sa bližšie na uvedený zdrojový kód. Použili sme 2 direktívy : `page` a `include`. Ako už vieme, direktíva `page` nastaví kódovanie pre stredoeurópske jazyky, aby sa text zo súboru objavil so správnou diakritikou. Direktíva `include` zahrnie na stránku nami požadovaný súbor *hlavicka.txt*. Pomocou HTML príkazov sme ďalej na stránku vložili obrázok. Výslednú stránku, ktorá bude zobrazená v prehliadači, si môžete pozrieť na obrázku:



Obrázok 6

Treba povedať, že `include` slúži len na vkladanie textových dokumentov. Ak chcete pomocou `include` zahrnúť súbor obsahujúci aj iné elementy (napríklad obrázky), vytvorte najprv HTML dokument s požadovaným obsahom. Ten potom môžete pomocou `include` zahrnúť na JSP stránku.

Direktíva `taglib` sa používa pri tvorbe a využívaní vlastných užívateľsky definovaných funkcií.

4.1.8. Cvičenia

1. Vytvorte JSP dokument, ktorý pri prvej návšteve užívateľa vypíše text "Vitajte, ráčte vstúpiť!" a pri opätovnej návšteve text "Vitajte znova!"

Riešenie :

```
//vstup.jsp
<%@ page contentType="text/html;charset=windows-1250"
%>
<html>
<HEAD>
<BODY>
<% if (session.isNew())%>
     Vitajte, ráčte vstúpiť!
<% else %>Vitajte znova!
</body>
</html>
```

2. Vytvorte JSP dokument, v ktorom zistíte typ užívateľského prehliadača. Využite implicitný objekt request.

Riešenie :

```
//request.jsp
<HTML>
<HEAD><TITLE>rok</TITLE></HEAD>
<BODY>
<%@ page contentType="text/html;charset = windows-1250"
%>
prehliadač : <BR>
<%= request.getHeader("User-Agent")%> <br><br>
</BODY>
</HTML>
```

4.1.9. Úlohy

1. Vytvorte JSP dokument, ktorý vypíše do prvého riadku výstupnej stránky súčin čísel 2 a 4, do druhého riadku ich rozdiel a do tretieho ich podiel aj s odpovedajúcim textom tak, ako bolo uvedené v príklade o súčte vo výkladovej časti.
2. Vytvorte JSP dokument, ktorý vypíše na každý z desiatich riadkov pozdrav. V párnych riadkoch bude pozdrav "Ahoj" a v nepárnych "Čau".
3. Vytvorte JSP dokument, v ktorom zistíte IP adresu klientského počítača. Využite implicitný objekt `request` a jeho metódu `getRemoteAddr()`.
4. Vytvorte JSP dokument, v ktorom zistíte aký je aktuálny rok a na obrazovku vypíšete text : "Píše sa rok : " a zistený rok. Využite direktívu page, v ktorej importujte triedu java.util.Calendar. (pomôcka : na zistenie aktuálneho roku slúži príkaz Calendar.getInstance().get(Calendar.YEAR)

4.1.10. Autotest

1. Dokument tvorený technológiou JavaServer Pages ukladáme s príponou :
   a) .jsp
   b) .java
   c) .html

Správna odpoveď : A

Počet bodov : 1b

2. Výsledný dokument odosielaný na klientský prehliadač je :
   a) súbor s príponou .class

b) obyčajná HTML stránka

c) stránka obsahujúca kombináciu JAVY a HTML

Správna odpoveď : B

Počet bodov : 1b

3. Príkaz <%! string meno="Slávka"%> :

a) vytvorí reťazec meno a vloží do neho text Slávka

b) vytvorí výstup na stránku

c) sa vykoná pri každej návšteve stránky

Správna odpoveď : A

Počet bodov : 1b

4. Text uzavretý v tagoch <% ... %> :

a) musí byť súvislý zmysluplný javovský kód

b) sa môže zobraziť na výstupnej stránke

c) sa vyhodnocuje pri prvej návšteve stránky alebo opätovnom spustení kontajnera JSP

Správna odpoveď : B

Počet bodov : 1b

4.2. Lekcia 2 - Práca s formulármi

Jednou z úloh tvorcu dynamických webových stránok je interaktivita s užívateľom. Na prispôsobenie stránok individuálnym požiadavkám je potrebné od návštevníka stránky získať údaje. Najčastejšie využívanou možnosťou na získavanie informácií je použitie formulárov. Formuláre umožňujú návštevníkovi odosielať údaje serveru.

Určite ste sa už s formulármi pri používaní internetu stretli. Ak ste napríklad použili niektorý z vyhľadávacích strojov, kľúčové slovo vyhľadávania ste zadávali do textového poľa formulára. Alebo pri prihlasovaní do emailovej schránky ste vaše identifikačné údaje vkladali do polí formulára. Tie ste odoslali serveru, kde boli overené.

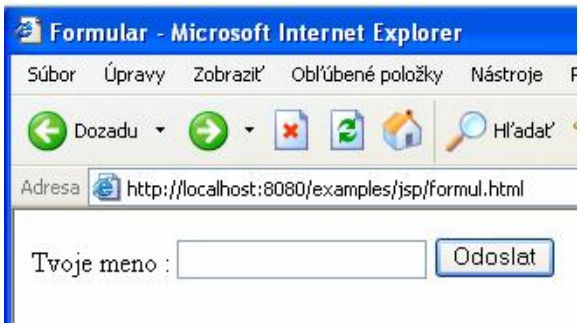
Vytváranie praktických formulárov nám ponúka jazyk HTML. Elementy HTML nám neposkytujú možnosť spracovania získaných údajov. Preto sa vo väčšine prípadov údaje posielajú serveru. Existuje viacero serverových technológií, ktoré vedia spracovávať formuláre. Jednou z nich je technológia JSP.

Získavaním a spracovávaním dát od používateľa sa venuje táto lekcia. Povieme si, ako získať pomocou formulárov údaje od návštevníkov našich stránok, ako tieto dáta spracovať a ďalej využiť.

4.2.1. Vytváranie formulárov

Povedali sme si, že formuláre budeme využívať na získavanie údajov od používateľa našich stránok. Vieme už aj to, že formuláre budeme vytvárať pomocou príkazov HTML. Nami pripravené formuláre umiestnime na našu stránku a po ich vyplnení sa údaje odošlú na server. Tu sa dáta budú spracovávať.

Každý formulár musí byť uzavretý medzi špeciálne značky (tagy) `<form>` a `</form>`. Medzi tieto príkazy vkladáme vstupné polia na získavanie nami požadovaných informácií. Vstupné polia vkladáme do formulára pomocou príkazu `<input>` s príslušnými atribútmi. Najdôležitejšími atribútmi tohto príkazu sú `type` a `name`. `Type` slúži na určenie typu poľa a `name` na pomenovanie tohto poľa. Pre ďalšie spracovanie údajov je veľmi dôležité každému vstupnému poľu priradiť meno. Aby mohli byť údaje serveru odoslané, náš formulár musí obsahovať tlačidlo na odoslanie. Príklad jednoduchého formulára na zistenie mena používateľa si môžete pozrieť na obrázku. Zdrojový kód tohto formulára je uvedený vo výpise pod obrázkom.



Obrázok 7

```
//formul.html
<HTML>
<HEAD><TITLE>Formular</TITLE></HEAD>
<BODY>
<FORM action="spracovat.jsp" method="GET">
Tvoje meno:
<INPUT type="text" name="meno">
<INPUT type="submit" name="odoslat" value="Odoslat">
</FORM>
```

```
</BODY>
</HTML>
```

Všimnite si zdrojový kód dokumentu *formul.html*. Príkaz `<form>` obsahuje dva atribúty: `action` a `method`. Už bolo uvedené, že HTML nám neumožňuje údaje z formulára spracovávať. Preto sa tieto údaje posielajú na spracovanie JSP dokumentu. Na určenie URL (kompletné alebo relatívne URL) tohto dokumentu slúži prvý z uvedených atribútov - `action`. To znamená, že po odoslaní formulára sa vykoná JSP dokument *spracovat.jsp*. Druhý z atribútov – `method` určuje to, akým spôsobom budú dáta z formulára serveru posielané. Podrobnejšie o možnostiach posielania dát si povieme v nasledujúcej časti.

Všetky príklady vytvárajúce a spracovávajúce formuláre si vyžadujú dokument HTML, ktorý daný formulár vytvorí, umožní vyplniť a odošle a JSP dokument, ktorý ho spracuje. Tieto dva dokumenty je možné zlúčiť do jedného súboru. Tento súbor musí mať príponu *.jsp*. Kvôli prehľadnosti sa odporúča vytvárať radšej dva súbory.

4.2.2. Spracovávanie formulárov

Ak používateľ navštívi stránku *formul.hmtl* objaví sa v prehliadači formulár. Do neho môže používateľ zadať svoje meno a po stlačení tlačidla *Odoslat* sa tento údaj odošle na server, kde bude spracovávaný dokumentom *spracovat.jsp*. Akým spôsobom sa dáta serveru odošlú?

4.2.2.1. Metódy GET a POST

Akým spôsobom sa budú dáta serveru posielať vieme rozhodnúť my, pri tvorbe nášho formulára. Poznáme dve metódy: `POST` a `GET` . To, ktorá z nich sa pri odosielaní použije, nastavíme atribútom `method` v príkaze `<form>`.

Metóda `GET` posiela dáta tak, že ich pridá na koniec URL adresy, na ktorú sa odosielajú. Prehliadač pridá na koniec URL adresy otáznik a za ním informácie v tvare `názov=hodnota`. Názov je hodnota atribútu `name` vo vstupnom poli (v našom prípade *meno*) a hodnota je obsah poľa, ktorý zadal používateľ (v našom formulári meno návštevníka). Ak formulár obsahuje viacero vstupných polí, za informácie o jednotlivých poliach prehliadač pridá znak & a vloží informácie o poli nasledujúcom. Náš formulár obsahuje ešte tlačidlo *Odoslat*. Informácie o stlačení tlačidla budú tak isto

prostredníctvom URL oznámené serveru. URL adresa po stlačení tlačidla na odoslanie bude v našom príklade vyzerať takto :

```
http://localhost:8080/examples/jsp/spracovat.jsp?meno=S
lavka&odoslat=Odoslat
```

Dĺžka prenesených dát pomocou metódy `GET` je obmedzená prehliadačom (zväčša na 2048 znakov).

Metóda `POST` pracuje inak ako metóda `GET`. Rozdiel je v tom, že údaje z formulára sú posielané v tele HTTP požiadavky, nie ako časť URL adresy. Ďalší podstatný rozdiel je v tom, že dĺžka prenášaných dát je teoreticky neobmedzená. Určité obmedzenia môžu byť dané prehliadačom alebo webovým serverom.

Atribút `method` príkazu `form` nie je povinný. Pri vytváraní formulára nemusíme zadávať, akou metódou majú byť dáta serveru prenášané. V takomto prípade prehliadač implicitne používa metódu `GET`. V prípade prenášania citlivých údajov táto metóda nie je vhodná, keďže sa všetky prenášané dáta zobrazujú v poli pre URL adresu. V takomto prípade je lepšie použiť metódu `POST`, aj keď ani tá nie je samostatne bezpečná. Dáta nie sú bezprostredne viditeľné, ale nie sú kryptované. Ktorú z metód v jednotlivých prípadoch použiť je na rozhodnutí samotného programátora.

4.2.2.2. Prístup k údajom z formulára

V prvej lekcii sme sa dozvedeli o tom, že technológia JSP umožňuje programátorovi používať pri tvorbe stránok implicitné objekty. Jeden z nich využijeme pri práci s údajmi z formulára.

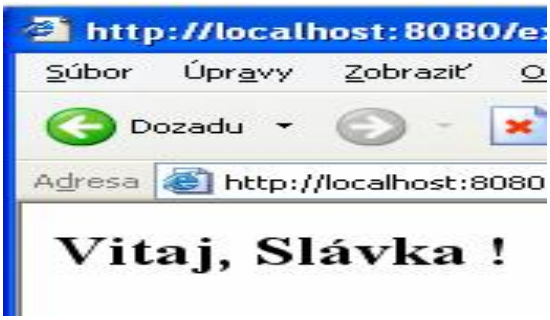
Na získavanie údajov z formulára používame implicitný objekt `request`. Tento objekt okrem iných zaujímavých metód obsahuje metódu `getParameter()`. Do zátvorky tejto metódy vkladáme hodnotu atribútu `name` vstupného poľa, ktorého obsah chceme získať.

Ak by sme chceli získať text, ktorý návštevník zadal do textového poľa pre zistenie mena v našom formulári, použili by sme príkaz `request.getParameter("meno")`. Takto získané údaje môžeme napríklad vložiť do premennej alebo umiestniť na výslednú webovú stránku tak, ako to urobí dokument *spracovat.jsp*.

```
//spracovat.jsp
<%@ page contentType="text/html;charset=windows-1250"
```

```
%>
<% request.setCharacterEncoding("windows-1250"); %>
<h2>Vitaj, <%=request.getParameter("meno") %> !
</h2>
```

Po zadaní mena Slávka do textového poľa a stlačení tlačidla *Odoslat* sa informácie z formulára odošlú dokumentu *spracovat.jsp*, ktorý sa vykoná. *Spracovat.jsp* obsahuje HTML text a výraz JSP, ktorý na výslednú stránku vloží hodnotu získanú z textového poľa formulára. Výsledná stránka, ktorú návštevník uvidí vo svojom prehliadači, je na obrázku:



Obrázok 8

Pozorný čitateľ si určite v zdrojovom kóde všimol použitie inej metódy objektu `request`. Pri volaní metódy `request.getParameter` („menoPrvku“) získame vždy hodnotu z daného prvku formulára ako string v kódovaní ISO-8859-1. Na nastavenie iného kódovania sme použili metódu `setCharacterEncoding()`. Tá definuje nami zvolené kódovanie znakov celej požiadavky.

4.2.3. Spracovávanie jednotlivých prvkov formulára

V tejto časti textu budeme pracovať s najčastejšie používanými prvkami formulára. Uvedieme spôsob ich vytvárania, získavania údajov z jednotlivých vstupných polí a na príkladoch si ukážeme niektoré z možností ďalšieho využitia získaných dát.

4.2.3.1. Textové pole

Textové pole slúži na zadávanie krátkeho textu. Patrí k najčastejšie používaným prvkom formulára. Vytvoríme ho tak, že v príkaze `<input>` priradíme atribútu `type` hodnotu `"text"`. Na určenie dĺžky textového poľa slúži atribút `size`. Ak chceme

zobraziť v poli úvodný text, priradzujeme tento text atribútu `value`. Atribútom `maxsize` môžete nastaviť maximálny počet znakov, ktoré bude môcť zadať návštevník.

V JSP dokumente, ktorému budú informácie z formulára odosielané, získame obsah textového poľa použitím príkazu `request.getParameter("menoPoľa")`. Možnosť vloženia textového poľa do formulára sme si uviedli v dokumente *formul.html.* Spôsob získania obsahu poľa v dokumente *spracovat.jsp*.

4.2.3.2. Pole pre zadávanie hesla

Pre vytvorenie vstupného poľa na zadávanie hesla používame príkaz `<input>` a jeho atribútu `type` priradíme hodnotu "`password`". Prvok formulára typu `password` je veľmi podobný textovému poľu. Jediný rozdiel je v tom, že pri zadávaní textu sa z bezpečnostných dôvodov na obrazovke zobrazujú miesto znakov hviezdičky alebo bodky. Formulár obsahujúci pole pre zadávanie hesla je viac ako vhodné posielať metódou `POST`, inak by sa heslo objavilo v poli pre URL adresu.

Text zadávaný do poľa pre zadávanie hesla je prístupný rovnakým spôsobom ako v textovom poli.

Na ukážku využitia poľa pre zadávanie hesla upravíme súbor *formul.html,* do ktorého pridáme tento prvok. Návštevníkovi zobrazíme uvítaciu stránku len v prípade, ak zadá správne heslo. Inak mu prístup k stránke odoprieme.

```
//formul2.html
<HTML>
<HEAD><TITLE>Formular</TITLE></HEAD>
<BODY>
<FORM action="overenie.jsp" method="POST">
Tvoje meno:
<INPUT type="text" name="meno"> <BR>
Heslo:
<INPUT type="password" name="heslo">
<INPUT type="submit" name="odoslat" value="Odoslat">
</FORM>
</BODY>
</HTML>
```

```
//Overenie.jsp
<%@ page contentType="text/html;charset=windows-1250"
%>
<% request.setCharacterEncoding("windows-1250");
   if(request.getParameter("heslo").equals("pristup")){
```

```
%>
        <h2>Vitaj, <%=request.getParameter("meno")%>!
        </h2>
<%
    }
    else {
        response.sendError(403);
    }
%>
```

V tomto príklade bude jediným správnym heslom slovo *pristup*. Po vyplnení mena a tohto hesla sa v okne používateľovho prehliadača zobrazí uvítacia stránka rovnako, ako pred tým (vo výpise stránky *spracovat.jsp*). Ak heslo nebude zhodné so slovom *pristup*, v prehliadači sa objaví oznam o nedostatočných právach pre zobrazenie stránky.

Na overenie prichádzajúceho hesla sme použili príkaz `if`, v ktorom porovnávame dva reťazce. Prvý z reťazcov obsahuje text z poľa pre zadávanie hesla, ktorý sme získali pomocou príkazu `request.getParameter("heslo")`. Druhý reťazec je heslo *prístup*, ktoré musí návštevník stránky zadať. Na porovnanie sme použili metódu `equals`. Ak heslo nie je správne, zobrazí sa chybová hláška. To sme zabezpečili použitím príkazu `response.sendError(403)`.

`response` je implicitný objekt. `sendError()` je jedna z jeho metód, ktorá slúži na odosielanie chybových hlášok. Objekt `response` v sebe ukrýva informácie o odpovedi servera na prichádzajúcu požiadavku. Jeho metódy slúžia na nastavenie vlastností nami pripravovanej odpovede. Ak napríklad chceme používateľa presmerovať na inú adresu, na pôvodnú stránku umiestnime tento skriptlet:

```
<% response.sendRedirect("nova_URL_adresa"); %>
```

4.2.3.3. Tlačidlo pre odoslanie a vynulovanie obsahu formulára

Aby údaje z formulára mohli byť odoslané na server, musíme do formulára vložiť tlačidlo na odoslanie. Vložíme ho príkazom `<input>` s atribútom `type="submit"`.

Ak chceme umiestniť na tlačidlo vlastný text, priradíme ho atribútu `value`. V prípade, že jeden formulár bude obsahovať dve tlačidlá na odoslanie, priradíme im mená v atribúte `name`. Potom budeme vedieť rozoznať, ktorým bol formulár odoslaný. Je to vhodné napríklad vtedy, ak chceme formulár spracovať dvomi spôsobmi.

Napríklad pri vypracovávaní autotestu môžeme okrem tlačidla na vyhodnotenie ponúknuť aj tlačidlo pre pomoc.

Ak chceme používateľovi uľahčiť prácu v prípade, že potrebuje zadané údaje vymazať a vyplniť formulár odznova, môžeme do formulára vložiť tlačidlo na vynulovanie obsahu formulára. Tlačidlo bude vložené, ak atribútu `type` priradíme `reset`. Stlačenie tohto tlačidla sa serveru neoznamuje.

4.2.3.4. Prepínacie tlačidlá

Ak chceme používateľovi ponúknuť voľbu jednej z viacerých možností, mali by sme použiť prepínacie tlačidlo. Vložíme ho príkazom `<input>` s atribútom `type="radio"`. Všetky prepínacie tlačidlá patriace k tej istej odpovedi musia mať rovnakú hodnotu atribútu `name` a rôznu hodnotu atribútu `value`.

Na stránke JSP, ktorá formulár spracováva, pracujeme s hodnotou `value` vybraného prepínacieho tlačidla. Túto hodnotu získame použitím príkazu `request.getParameter("`*meno*`")`, kde *meno* je hodnota `name` daného tlačidla.

Príklad použitia prepínacieho tlačidla je uvedený vo výpise :

```
//pohlavie.html
<HTML>
<HEAD><TITLE>formular</TITLE></HEAD>
<BODY>
Oznámte svoje pohlavie <BR>
 <FORM action="spracuj2.jsp" method="POST">
   <INPUT type="radio" name="pohlavie" value="z">
       žena<BR>
   <INPUT type="radio" name="pohlavie" value="m">
       muž<BR>
   <INPUT type="submit" name="odoslat" value="Odoslat">
 </FORM>
</BODY>
<HTML>

//spracuj2.jsp
<%@ page contentType="text/html;charset=windows-1250"
%>
<%
  request.setCharacterEncoding("windows-1250");
  if (request.getParameter("pohlavie").equals("z")){
%>
       <img src="obrazok1.gif">
<%
  }
```

```
  else{
%>
       <img src="obrazok2.gif">
<%
  }
%>
```

Prvý z uvedených súborov *pohlavie.html* vytvorí formulár, v ktorom návštevník oznámi svoje pohlavie. Po stlčení tlačidla na odoslanie sa údaje pošlú druhému súboru *spracuj2.jsp*. Tento vyhodnotí, aké pohlavie bolo zvolené a podľa výsledku zobrazí jeden z dvoch obrázkov.

4.2.3.5. Zaškrtávacie polia

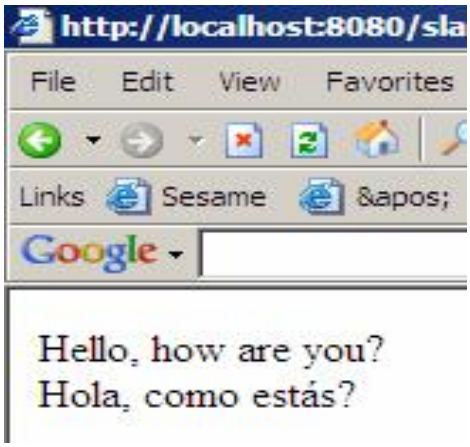
Zaškrtávacie polia sa do formulára vkladajú príkazom `<input>` s atribútom `type="checkbox"`. Môžu nadobudnúť jednu z dvoch logických hodnôt áno/nie. Zaškrtávacie polia majú vo formulári tvar štvorčekov. Zaškrtnutím štvorčeka používateľ priradzuje danému poľu pravdivú hodnotu. Ak je políčko zaškrtnuté, na server sa odošle hodnota atribútu `value`. O tom , ktoré zaškrtávacie pole bolo vybrané nás informuje atribút `name`. Na zistenie, aké cudzie jazyky ovláda používateľ, nám poslúži jednoduchý formulár *jazyk.html* (obrázok 9).

```
//jazyk.html
<HTML>
<HEAD><TITLE>formular</TITLE></HEAD>
<BODY>
Aký jazyk ovládate ? <BR>
 <FORM action="spr.jsp" method="post">
  <INPUT type="checkbox" name="jazyk1" value="ang">
     anglický <BR>
  <INPUT type="checkbox" name="jazyk2" value="nem">
     nemecký <BR>
  <INPUT type="checkbox" name="jazyk3" value="span">
     španielsky <BR>
  <INPUT type="submit" name="odoslat" value="Odoslat">
 </FORM>
</BODY>
<HTML>
```

Obrázok 9

Po označení príslušného jazyka (jazykov) a stlačení tlačidla na odoslanie, sa údaje odošlú dokumentu *spr.jsp*. Tento ich spracuje a v klientskom prehliadači sa objaví pozdrav v označenom jazyku (obrázok 10).

```
//spr.jsp
<%@ page contentType="text/html;charset=windows-1250"
%>
<%
  request.setCharacterEncoding("windows-1250");
  if ((request.getParameter("jazyk1")!=null) &&
  (request.getParameter("jazyk1").equals("ang")))
  {
%>
       Hello, how are you? <BR>
<%
  };
  if ((request.getParameter("jazyk2")!=null) &&
  (request.getParameter("jazyk2").equals("nem"))){
%>
       Hallo, wie geht es? <BR>
<%
  };
  if ((request.getParameter("jazyk3")!=null) &&
  (request.getParameter("jazyk3").equals("span")))
  {
%>
       Hola, como estás? <BR>
<%
  };
%>
```



Obrázok 10

4.2.3.6. Textová plocha

Ak očakávame zadanie dlhšieho textu, je vhodné použiť textovú plochu. Do formulára ju vložíme príkazom `<textarea>`. Jej meno vložíme do atribútu `name`. Rozmery textovej plochy určíme atribútmi `rows` a `cols`. Pri vkladaní tohto prvku musíme použiť aj ukončovací príkaz `</textarea>`. Text, ktorý má byť implicitne zobrazený v textovej ploche, sa zapisuje medzi začiatočný a koncový príkaz.

V dokumente JSP, ktorý formulár spracováva, sa s údajmi získanými z textovej plochy pracuje rovnako, ako pri textovom poli. Napríklad na získanie spätnej väzby od návštevníka našich stránok by sme mohli použiť tento skript:

```
//plocha.html
<HTML>
<HEAD><TITLE>Formular</TITLE></HEAD>
<BODY>
<FORM action="spracuj4.jsp">
<TEXTAREA rows="10" cols="30" name="nazor">Aké sú tieto
stránky?</TEXTAREA>
<INPUT type="submit" name="odoslat" value="Odoslat">
</FORM>
</BODY>
</HTML>
```

```
//spracuj4.jsp
<%@ page contentType="text/html;charset=windows-1250"
%>
Váš názor :

<%  request.setCharacterEncoding("windows-1250");
    out.println(request.getParameter("nazor"));
%>
```

```
<BR>
Ďakujem za príspevok!
```

4.2.3.7. Zoznamy

Zoznamy využívame, ak chceme používateľovi ponúknuť jednu z viacerých možností. Slúžia podobne, ako zaškrtávacie polia. Na vytvorenie zoznamu požívame príkaz `<select>`. Ak by sme chceli používateľovi umožniť výber viacerých možností, použijeme atribút `multiple`. Atribútom `name` určíme meno zoznamu. Štandardne je zobrazená jedna voľba. Atribútom `size` môžeme nastaviť, koľko možností má byť viditeľných.

Ostatné sú prístupné v rolovacom zozname. Jednotlivé položky sa do zoznamu pridávajú príkazom `<option>`. Ak je text položiek príliš dlhý, je výhodné použiť atribút `value` s nami priradenými hodnotami.

Ak sme použili atribút value, na server sa odosiela jeho hodnota. Inak je to text uvedený v príkaze `<option>`. Údaje sú posielané na server rovnako ako v prípade zaškrtávacích polí.

//skola.html

```
<HTML>
<HEAD><TITLE>Formular</TITLE></HEAD>
<BODY>
<FORM action="spracuj5.jsp">
 Ste študentom :
 <SELECT NAME="skola">
  <OPTION value="TU">Technická Univerzita
  <OPTION value="UPJS">Univerzita Pavla Jozefa Šafárika
  <OPTION value="EU">Ekonomická Univerzita
  <INPUT type="submit" name="odoslat" value="Odoslat">
</FORM>
</BODY>
</HTML>
```

//spracuj5.jsp

```
<%@ page contentType="text/html;charset=windows-1250"
%>
Vaša škola :
<%
  request.setCharacterEncoding("windows-1250");
  if (request.getParameter("skola").equals("TU"))
       out.println("Technická Univerzita");}
  if (request.getParameter("skola").equals("UPJS"))
       out.println("Univerzita Pavla Jozefa Šafárika");
  if (request.getParameter("skola").equals("EU"))
```

```
        out.println("Ekonomická Univerzita");
%>
```

4.2.4. Ošetrovanie výnimiek

Formuláre sú častou súčasťou dynamických webových stránok. Pri získavaní informácií prostredníctvom formulárov musíme myslieť na to, že ich vypĺňa používateľ. Ľahko sa môže stať, že dáta budú chybné, neúplné, alebo ich používateľ vôbec nezadá. Pri tvorbe stránok JSP by mal programátor myslieť aj na bezpečnosť a ošetrenie neočakávaných udalostí.

Ak by ste napríklad odoslali formulár zo stránky *pohlavie.html* bez zaškrtnutia niektorej voľby, v okne prehliadača by sa objavila odstrašujúco vyzerajúca chybová hláška. Rovnaká situácia by nastala po spustení stránky *spracuj3.jsp* samostatne. Takýmto nepríjemnostiam sa dá predísť. Jazyk JAVA je na ošetrovanie výnimiek prispôsobený. Problémový kód sa vkladá do chráneného bloku `try`. Tie sa dajú použiť aj na stránkach JSP. Na ošetrenie prípadných chýb môžeme využiť aj direktívu `page`. Jej atribút `errorPage` chráni ľubovoľný kód pred akýmikoľvek výnimkami. Vo výpise si môžete pozrieť praktické použitie tohto atribútu.

```
//spracuj2.jsp
<%@ page contentType="text/html;charset=windows-1250"
        errorPage="chyba.jsp"%>
<%
  request.setCharacterEncoding("windows-1250");
  if (request.getParameter("pohlavie").equals("z"))
  {
%>
       <img src="obrazok1.gif">
<%
  } else
  {
%>
       <img src="obrazok2.gif">
<%
  }
%>
```

```
//chyba.jsp
<%@ page contentType="text/html;charset=windows-1250"
         isErrorPage="true"%>
<%if (request.getParameter("pohlavie")==(null))%>
Nevybrali ste žiadnu možnosť!<BR>
<a href="pohlavie.html" >Znovu</a>
```

Do zdrojového kódu stránky *spracuj3.jsp*, ktorá spracováva formulár zo stránky *pohlavie.html* sme pridali príkaz `errorPage="chyba.jsp"`. Ak používateľ odošle formulár bez zvolenia jednej z možností, hodnota *null* pri pokuse získať dáta z formulára v JSP dokumente vyvolá výnimku. Riadenie programu presmeruje atribút `errorPage` direktívy `page` na stránku *chyba.jsp*. V takejto situácii používateľ uvidí stránku, ktorá ho upozorní na chybu a umožní mu vrátiť sa späť k formuláru.

4.2.5. Cvičenia

1. Vytvorte JSP dokument, v ktorom klient zadá číslo do textového pola a po stlačení tlačidla sa číslo umocní a výsledok sa vypíše na stránku.

Riešenie :

```
//mocnina.jsp
<%@page contentType="text/html; charset=windows-1250"%>
<html>
<HEAD>
<BODY>
<form action="mocnina.jsp" method="GET">
  zaklad: <input type="text" length="20" name="zaklad">
<br>
          <input type="submit" value="Umocnit"></form>
<% int a=0;
   if (request.getParameter("zaklad")!=null)
   a=Integer.parseInt(request.getParameter("zaklad"));
   int c =a*a;
%>
<P>vysledok = <%= c %>
</body>
</html>
```

2. Vytvorte webovú stránku s hádanku. Pomocou formulára sa opýtajte návštevníka na správnu odpoveď a oznámte mu, či uhádol.

Riešenie :

```
//hadaj.html
<HTML>
<HEAD><TITLE>Hádanka dňa</TITLE></HEAD>
<form action="hadanka.jsp" method=post>
  Uhádnite :
  Kto chodí ráno po štyroch, cez deň po dvoch a večer
  po troch ?<br>
  <input type="radio" name="odp" value="a"> pes<br>
  <input type="radio" name="odp" value="b"> človek<br>
  <input type="radio" name="odp" value="c"> dva psy<br>
```

```
  <input type=submit value=Odoslat>
</form>
</HTML>
```

//hadanka.jsp

```
<%@ page contentType="text/html;charset=windows-1250"
%>
<%
  request.setCharacterEncoding("windows-1250");
  if (request.getParameter("odp").equals("b")){
%>
      Uhádli ste!
<%
  } else{
%>
      Radšej skúste znova...
<%
    }
%>
```

4.2.6. Úlohy

1. Vytvorte webovú stránku, ktorá predstavuje jednoduchý internetový obchod. Návštevníkovi ponúknite 5 CD titulov. Návštevník si bude vyberať druh a počet CD. Po stlačení tlačidla na odoslanie sa mu zobrazí druh a počet vybraného tovaru.
2. Vytvorte JSP dokument obsahujúci prepínacie tlačidlo s troma možnosťami. Každá z nich bude ponúkať iný druh vtipu. Po vybratí jednej z možností a odoslaní požiadavky sa do textového poľa vypíše vtip z vybranej kategórie.
3. Vytvorte formulár html predstavujúci matematický test s minimálne tromi otázkami a JSP dokument, ktorý daný test vyhodnotí a vypíše testujúcemu sa dosiahnutý výsledok.

4.2.7. Autotest

1. Technológia JSP
   a) umožňuje spracovávať formuláre JSP len ak sú posielané metódou post.
   b) nevie pracovať s formulármi.
   c) umožňuje spracovávať formuláre HTML.

Správna odpoveď : C

Počet bodov : 1b

2. Metóda request.getParameter
   a) vracia string v kódovaní "windows-1250"

b) používame na získanie hodnôt z formulára

c) vracia hodnotu z premennej formulára vždy v nami požadovanom type

Správna odpoveď : B

Počet bodov : 1b

3. Akú metódu na odosielanie sme použili, ak sa po odoslaní formulára objavila táto URL adresa http://localhost:8080/examples/jsp/mocnina.jsp?zaklad=3?

   a) GET

   b) FILES

   c) POST

Správna odpoveď : A

Počet bodov : 1b

4. Atribút ACTION v príkaze <FORM> určuje

   a) skript, ktorý bude spracovávať údaje z formulára.

   b) názov použitej metódy na odoslanie.

   c) definuje druh vstupného poľa formulára.

Správna odpoveď : A

Počet bodov : 1b

4.3. Lekcia 3 - Spolupráca s databázou

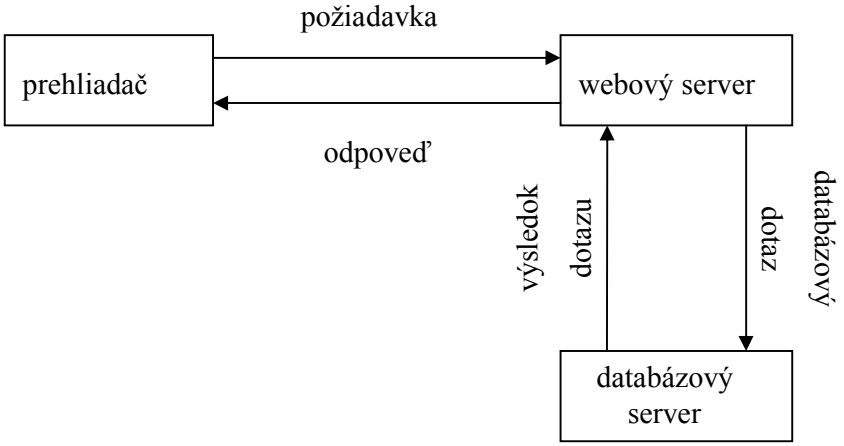
V predchádzajúcej lekcii sme sa naučili, ako získať údaje od návštevníkov našich stránok. Je prirodzené si tieto dáta zapamätávať, aby sme sa k nim neskôr vedeli vrátiť a opäť s nimi pracovať. Ak spravujeme webový server, je pravdepodobné, že potrebujeme uchovávať dáta väčšej veľkosti so zložitejšou štruktúrou. Tieto dáta potrebujeme vyhľadávať, často podľa nejakých podmienok. Vtedy je výhodné ukladať dáta do databázy. JavaServer Pages spoluprácu s databázou pri generovaní stránok ponúka.

Databáza je prostredie na ukladanie a opätovné použitie dát. Je to prostredie s vlastným spracovateľským strojom, jazykom a podobne. Podstatné je, že sa stará o množstvo užitočných vecí. Má napríklad zabudovaný mechanizmus pre súbežný prístup. Táto lekcia sa venuje spolupráci s databázou. Naučíme sa vytvoriť spojenie s databázou, vytvárať a mazať tabuľky a hlavne pracovať s údajmi z databázy. Budeme pracovať s databázou MySQL.

4.3.1. Server a databáza

Databáza nie je prostriedkom JSP. Je to samostatné prostredie, s ktorým JSP spolupracuje. Server založený na technológií JSP môže využiť prostriedky a možnosti javovských knižníc. Konkrétne balíček *java.sql* umožňuje preddefinovaný prístup k väčšine používaných databázových systémov. Triedy v tomto balíčku sú známe pod názvom JDBC. Prostredníctvom JDBC môžeme posielať príkazy v jazyku SQL mnohým druhom databáz.

Ako táto spolupráca funguje? Klasická webová architektúra pozostáva z dvoch prvkov. Prehliadača, ktorý vytvára požiadavky a webového servera, ktorý tieto požiadavky spracováva a vytvára odpoveď. Pri webovej databázovej architektúre je situácia o niečo zložitejšia. Ak si návštevník vyžiada JSP stránku, alebo stlačí tlačidlo na odoslanie, prehliadač vytvorí požiadavku a odošle ju serveru. Kontajner JSP na serveri začne spracovávať dokument JSP. Pri spracovávaní sa stretne s kódom, ktorý volá databázu. Vytvorí spojenie s databázou a odošle jej dopyt. Databázový server dopyt spracuje a výsledok odošle späť. Na serveri sa spojenie ukončí, dokončí sa spracovávanie požiadavky a klientskému prehliadaču sa odošle výsledná HTML stránka. Ten túto stránku zobrazí návštevníkovi. Princíp tejto práce je znázornený na obrázku:



Obrázok 11

Toto je základný princíp fungovania spolupráce s databázou. Je rovnaký v prípade, že webový a databázový server sú, ale aj nie sú na tom istom počítači. Princíp prístupu k databáze nie je ovplyvnený ani druhom databázy, ktorú používame.

4.3.2. Tvorba databázového spojenia

Ako už bolo viac krát spomenuté, databáza nie je súčasťou JSP. Preto je možné ju využívať až potom, ako sa k nej pripojíme. V tejto časti si popíšeme príkazy na vytvorenie a ukončenie spojenia s databázovým serverom.

Pri tvorbe JSP stránok, ktoré budú vedieť posielať požiadavky databázovému serveru je treba použiť niektoré príkazy v jazyku JAVA. Na pripojenie sa k databáze slúžia tieto príkazy :

```
Class.forName(" názov databázového ovládača");
Connection connection = DriverManager.getConnection
("protokol:názov_zdroja_dát","používateľské_meno","heslo");
```

Prvý príkaz určuje databázový ovládač. To je softvér, ktorý umožňuje komunikáciu s príslušnou databázou. Na ovládač sa odkazujeme názvom javovskej triedy. Pri našom testovaní použijeme triedu `sun.jdbc.odbc.JdbcOdbcdrive`, známu ako most JDBC-ODBC. Prekladá databázové požiadavky JDBC programov v JAVE na požiadavky ODBC. Rozhranie ODBC, ktoré je alternatívou JDBC, je aplikačné rozhranie, ktoré umožňuje aplikácii prístup k rôznym databázam v rôznych systémoch.

Druhý z uvedených príkazov otvára dátové spojenie. Argumenty metódy `getConnection()` objektu typu `Connection` obsahujú informácie, ktoré sú zadávané pri konfigurácii databázového servera. Prvý argument udáva protokol použitý na komunikáciu s databázovým ovládačom a meno databázy konfigurovanej na serveri. Druhý a tretí argument je overovací, na zistenie oprávnenosti prístupu k databáze. Všetky potrebné informácie o inštalácii a konfigurácii databázového servera a príslušného ovládača nájdete v prílohe.

Príklad stránky, ktorá vytvorí spojenie s databázou je uvedený vo výpise (názov zdroja dát je „dip“, používateľské meno je „root“ a heslo „q“).

```
//spojenie.jsp
<%@ page contentType=text/html;charset=windows-1250" %>
<% Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   Connection connection = DriverManager.getConnection
   ("jdbc:odbc:dip","root","q");
%>
Spojenie vytvorené!
<% connection.close();
%>
```

Ak sa podarí pripojiť na príslušnú databázu, prehliadač zobrazí stránku s oznamom o úspešnom spojení. Ďalším príkazom, ktorý sme použili na stránke *spojenie.jsp*, je príkaz `connection.close()`. Tento príkaz slúži na ukončenie otvoreného spojenia s databázou. Treba uviesť, že ukončovanie je dôležité, pretože databázy majú obvykle malý počet povolených otvorených pripojení (rádovo 20).

4.3.3. Príkazy pre prácu s databázou

Už vieme, ako vytvoriť spojenie s databázou. Môžeme začať samotnú spoluprácu. V tejto časti si predstavíme príkazy potrebné na :

- vytvorenie a vymazanie databázovej tabuľky
- vkladanie dát do tabuľky
- načítavanie dát z tabuľky
- vymazávanie a úprava záznamov tabuľky

4.3.3.1. Vytvorenie a vymazanie tabuľky

Na to, aby sme mohli uchovávať dáta, potrebujeme v databáze vytvoriť tabuľky s vhodnou štruktúrou. Tabuľky na stránkach JSP vytvárame príkazmi jazyka SQL. Stránka JSP môže v databáze vykonávať príkazy SQL vďaka objektu typu `Statement`. Nový objekt tohto typu vytvoríme na základe existujúceho spojenia volaním metódy `createStatement()` objektu `connection`.

Na aktualizáciu databázy prostredníctvom stránok JSP slúži metóda `executeUpdate()` objektu `Statement`. V rámci nej majú byť použité aj príkazy na tvorbu a mazanie požadovaných tabuliek. Ako to môže vyzerať v praxi, ukazujú nasledujúce príklady :

```
//vytvorTabulku.jsp
<%@ page contentType ="text/html;charset=windows-1250"
%>
<%@ page import="java.sql.* " %>
<H1>Vytvorenie tabulky zviera</H1>
<% Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   Connection connection = DriverManager.getConnection
   ("jdbc:odbc:dip","root","q");
   Statement statement = connection.createStatement();

   statement.executeUpdate
   ("create table zviera(meno varchar(15),druh
   varchar(15),vek integer)");
```

```
   statement.close();
   connection.close();
%>
```

```
//zrusTabulku.jsp
<%@ page contentType ="text/html;charset=windows-1250"
%>
<%@ page import="java.sql.* " %>
<H1>Zrušenie tabuľky zviera</H1>
<% Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   Connection connection = DriverManager.getConnection
   ("jdbc:odbc:dip","root","q");
   Statement statement = connection.createStatement();

   statement.executeUpdate ("drop table zviera" );

   statement.close();
   connection.close();
%>
```

V oboch výpisoch sme najprv vytvorili spojenie s databázou pomocou príkazov, ktoré sme si už popísali. Ďalej sme pracovali s objektom `statement` a jeho metódou `executeUpdate()`. V prvom príklade sme príkazmi SQL vytvorili tabuľku *zviera* so stĺpcami *meno*, *druh* a *vek*. V druhom príklade sme túto tabuľku zrušili. Poslednými dvomi príkazmi sme ukončili aktualizačnú prácu a spojenie s databázou. Je treba upozorniť, že vo výpise sú príkazy SQL rozdelené do viacerých riadkov len kvôli čitateľnosti. Pri písaní zdrojového kódu, v prípade ich rozdelenia na viacero reťazcov, je potrebné spojiť ich operátorom plus. Netreba zabúdať na ukončenie práce s objektom `statement` príkazom `statement.close()` ešte pred ukončením spojenia.

4.3.3.2. Vkladanie záznamov do tabuľky

Na vkladanie záznamov do tabuľky nám opäť poslúži metóda `executeUpdate()`. V rámci nej použijeme príkaz INPUT. Na ukážku funkčnosti si uvedieme jednoduchý príklad, v ktorom do tabuľky *zviera* vložíme dva riadky.

```
//naplnTabulku.jsp
<%@ page contentType ="text/html;charset=windows-1250"
%>
<%@ page import="java.sql.*,java.text.NumberFormat" %>
<H1>Naplenie tabulky zviera</H1>
<% Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   Connection connection = DriverManager.getConnection
   ("jdbc:odbc:dip","root","q");
```

```
   Statement statement = connection.createStatement();

   statement.executeUpdate("insert into zviera values
   ('Dunco','pes',6)" );
   statement.executeUpdate("insert into zviera values
   ('Micka','macka',3)" );

   statement.close();
   connection.close();
%>
```

V praxi je málo pravdepodobné, že by sme záznamy do tabuľky vkladali takto, v rámci tvorby stránky. Do databázy budeme ukladať dáta získané zo vstupných polí formulára alebo iných zdrojov. Ako je možné získať a uložiť údaje z formulára uvedieme v nasledujúcom príklade. V databáze vytvoríme tabuľku *človek* so štyrmi stĺpcami *id*, *meno*, *priezvisko* a *vek*. Ďalej vytvoríme HTML stránku s formulárom, do ktorého sa budú vkladať jednotlivé záznamy a dokument JSP, ktorý bude údaje spracovávať a ukladať do tabuľky.

SQL príkaz na vytvorenie tabuľky:

```
create table clovek (id int, meno varchar(15),
priezvisko varchar(30),vyska int);
```

//vlozenieDoTab.html

```
<form action="vlozit.jsp" method=post>
<h1>Vlozenie udajov do tabulky clovek</h1>
  ID: <input type=text name=id><br>
  MENO: <input type=text name=meno><br>
  PRIEZVISKO: <input type=text name=priezvisko><br>
  VYSKA: <input type=text name=vyska><br>
<input type=submit value=Vlozit>
</form>
```

//vlozit.jsp

```
<%@ page contentType="text/html;charset=windows-1250"
%>
<%@ page import="java.sql.*,java.text.NumberFormat" %>
<%! int i;
    String m;
    String p;
    double v;
   %>
<% Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   Connection connection = DriverManager.getConnection
   ("jdbc:odbc:dip","root","q");
   Statement statement = connection.createStatement();
   PreparedStatement pS;
   pS = connection.prepareStatement
```

```
    ("insert into clovek values (?,?,?,?)");
    request.setCharacterEncoding("windows-1250");
    i = Integer.parseInt(request.getParameter("id"));
    m = request.getParameter("meno");
    p = request.getParameter("priezvisko");
    v=Double.parseDouble(request.getParameter("vyska"));
    pS.setInt(1,i);
    pS.setString(2,m);
    pS.setString(3,p);
    pS.setDouble(4,v);
    pS.executeUpdate();

    statement.close();
    connection.close();
%>
```

V zdrojovom kóde pri tvorbe stránky *vlozit.jsp* sme použili objekt typu `PreparedStatement`. Objekt tohto typu slúži ako pripravený príkaz, ako určitá šablóna databázového príkazu. (Jeho užitočnosť by sa prejavila lepšie pri viacnásobnom použití.) Po vyplnení šablóny databázový ovládač jednoducho zostaví požadovaný príkaz. Prázdne kolónky (otázniky v pripravenej šablóne) vyplníme príkazom `pS.set`*`NázovTypu`*`(`*`poradové_číslo`*`,`*`hodnota`*`)`. Na miesto prvého argumentu v zátvorke zadávame poradové číslo parametra pripraveného príkazu (otáznika), kam sa údaje vložia. Na miesto druhého argumentu zadávame hodnotu získanú od používateľa, ktorá sa po vykonaní príkazu vloží do tabuľky. Napríklad príkaz `pS.setInt(1,i)` vloží na miesto prvého parametra databázového príkazu hodnotu príslušného identifikačného čísla získaného z prvého vstupného poľa formulára.

Treba pripomenúť, že všetky údaje získané z polí formulára sú typu `String`. Preto je potrebné prekonvertovať ich na typ prislúchajúci stĺpcu, do ktorého tieto dáta patria, ešte pred ich ukladaním do tabuľky.

4.3.3.3. Načítanie záznamov z tabuľky

Ku práci s databázou neodmysliteľne patrí aj posielanie požiadaviek (dopytov), na ktoré databáza odpovedá. Odpoveď väčšinou predstavujú záznamy, ktoré vyhovujú podmienkam dopytu.

Na získavanie informácií z databázy slúži metóda `executeQuery()` objektu typu `Statement`. Táto metóda vracia objekty typu `ResultSet` (výsledná sada). V tomto objekte typu `ResultSet` sú uložené dáta získané spracovaním dopytu. Výsledná sada je zoradená vo výsledkovej tabuľke. Na používateľov počítač sa

s výslednou sadou posiela aj kurzor, ktorý ukazuje na riadok výsledkovej tabuľky. Po zavolaní metódy `executeQuery()` sa kurzor nachádza pred prvým záznamom. Každé ďalší prechod cyklom `while (resultset.next())` posunie kurzor o riadok nižšie. Hodnoty z riadku, na ktorý ukazuje kurzor získame metódou `getString(), getInt(), getFloat()` a iných, kde do zátvorky vložíme názov atribútu. Ak by sme chceli načítať všetky záznamy z tabuľky človek, poslúžila by nám nasledujúca JSP stránka.

```
//vybrat.jsp
<%@ page contentType ="text/html;charset=windows-1250"
%>
<%@ page import="java.sql.*,java.text.NumberFormat" %>
<% Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection connection = DriverManager.getConnection
("jdbc:odbc:dip","root","q");
Statement statement = connection.createStatement();
ResultSet resultset;
request.setCharacterEncoding("windows-1250");

resultset = statement.executeQuery
("select id, meno, priezvisko, vyska from clovek ");
while (resultset.next())
{
  out.print(resultset.getInt("id"));
  out.print(" ");
  out.print(resultset.getString("meno"));
  out.print(" ");
  out.print(resultset.getString("priezvisko"));
  out.print(" ");
  out.print(resultset.getFloat("vyska"));
  out.print("<br> ");
}
resultset.close();
statement.close();
connection.close();
%>
```

Po spracovaní tejto stránky webový prehliadač vypíše všetky záznamy z tabuľky *človek*. Aby sme získali všetky záznamy, použili sme cyklus `while` s podmienkou `resulteset.next()`. Metóda `next()` objektu typu `ResultSet` vracia hodnotu `true`, ak pod aktuálnym riadkom tabuľky výslednej sady existuje ešte ďalší. V opačnom prípade vracia hodnotu `false`. Cyklus skončí po vypísaní údajov z posledného riadka tabuľky. Prácu s objektom `resultset` ukončíme ako prvú príkazom `resultset.close()`.

4.3.3.4. Úprava záznamov

Pri uchovávaní údajov v databáze musíme myslieť na to, že nič netrvá večne. Je pravdepodobné, že uložené informácie budeme časom chcieť upraviť, doplniť alebo vymazať.

Ako už vieme, na aktualizáciu databázy máme k dispozícii metódu `executeUpdate()` objektu typu `Statement`. V rámci nej môžeme používať SQL príkazy ako DELETE, UPDATE a pod. V nasledujúcom výpise si ukážeme príklad vymazania záznamov s príslušným identifikačným číslom z tabuľky *človek*.

```
//zaznam.html
<HTML>
<HEAD><TITLE>Vymazanie záznamu</TITLE></HEAD>
<form action="vymazanie.jsp" method=post>
  Id záznamu, ktorý chcete vymazať:
  <input type=text name=id><br>
  <input type=submit value=Odoslat>
</form>
</HTML>

//vymazanie.jsp
<%@ page contentType="text/html;charset = windows-1250"
%>
<%@ page import="java.sql.*" %>
<H1>Vymazanie záznamu z tabuľky človek</H1>
<%! int id;
%>
<% Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   Connection connection = DriverManager.getConnection
   ("jdbc:odbc:dip","root","q");
   Statement statement = connection.createStatement();
   PreparedStatement pS;
   pS = connection.prepareStatement
   ("delete from clovek where id = ?");
   ResultSet resultset;

   request.setCharacterEncoding("windows-1250");
   id = Integer.parseInt(request.getParameter("id"));
   pS.setInt(1,id);
   pS.executeUpdate();
%>
Záznam bol úspešne vymazaný. <BR><BR>
Ostávajúce záznamy : <BR>
<%
   resultset = statement.executeQuery
   ("select * from clovek ");
```

```
    while (resultset.next()){
        out.print(resultset.getString("meno"));
        out.print(" ");
        out.print(resultset.getString("priezvisko"));
        out.print(" ");
        out.print(resultset.getDouble("vyska"));
        out.print("<br> ");
    }
    resultset.close();
    statement.close();
    connection.close();
%>
```

Ak návštevník stránky zadá do poľa formulára príslušné identifikačné číslo a formulár odošle, zavolá sa *vymazanie.jsp*. Tento JSP dokument vymaže príslušný záznam a vypíše všetky záznamy, ktoré ostali uložené v tabuľke. (Treba upozorniť, že sme v príklade vychádzali z ideálnych podmienok. Bolo by potrebné ošetriť výnimky ako napríklad odoslanie formulára bez vyplnenia príslušného poľa `id`.)

4.3.4. Praktický príklad

Pokúsme sa teraz využiť získané vedomosti a vytvoriť jednoduchú praktickú webovú aplikáciu. Táto aplikácia bude predstavovať testovací systém pre študentov. Študent vypracuje test, zadá požadované údaje a test odošle. Ten sa vyhodnotí, oznámia sa študentovi výsledky a dosiahnuté výsledky sa uložia. Výsledky všetkých žiakov budú prístupné pre učiteľa.

Ako prvé je potrebné vytvoriť vhodný databázový model. Vytvoríme nasledujúce tabuľky.

```
Create table ziak
(id int primary key, meno varchar(10), priezvisko
varchar(20), body int);

Create table otazka
(id_ziak int, cislo_otazky int, spravne int);
```

Teraz vytvoríme úvodnú stránku, na ktorej si návštevník zvolí, či chce vypracovať test alebo zobraziť dosiahnuté výsledky žiakov.

```
//prakticky.html
<html>
<head><title>Testovanie</title></head>
<body>
<a href="test.html">Vypracovať test</a><br>
```

```
<a href="vysledky.html">Ukázať výsledky</a>
</body>
</html>
```

Ďalej vytvoríme pomocou formulára test, ktorý budú študenti vypracovávať (súbor *test.html*). Stránka *vyhod.jsp* bude údaje z testu po odoslaní formulára spracovávať. Počet získaných bodov sa zobrazí používateľovi a výsledky sa uložia do databázy.

```
//test.html
<html>
<head>
<title>test</title>
</head>

<body>
<p align="center" class="style1" >Test</p>
<form action="vyhod.jsp" method="post">
<ol>
  Tvoje meno : <input type="text" name="meno" ><br>
  Tvoje priezvisko : <input type="text"
  name="priezvisko" ><br>
  <br>
  <li>Cyklus for používame : </li>
  (1 bod)<br>
  <br>
  <input type="radio" name="rb1" checked value="1">
  ak vieme počet opakovaní<br>
  <input type="radio" name="rb1" value="2">
  iba ak to vyplýva zo zadania príkladu<br>
  <input type="radio" name="rb1" value="3">
  vždy v kombinácii s iným cyklom<br>
  <br>
  <li>Príkaz inc(i); : </li>
  (1 bod)<br>
  <br>
  <input type="radio" name="rb2" checked value="1">
  zníži hodnotu premennej i o 1<br>
  <input type="radio" name="rb2" value="2">
  zvýši hodnotu premennej i o 1<br>
  <input type="radio" name="rb2" value="3">
  umocní premennú i<br>
  <br>
  <li>Deklaračná časť programu sa začína slovom : </li>
  (1 bod)<br>
  <br>
  <input type="radio" name="rb3" checked value="1">
  begin<br>
  <input type="radio" name="rb3" value="2">
```

```
  procedure<br>
  <input type="radio" name="rb3" value="3">
  var<br>
  <br>
  <table width="468" border="0">
    <tr>
     <td width="232"><ol>
     <input name="submit" type="submit" value="pošli ">
     </ol></td>
     <td width="226"><ol>
     <input name="reset" type="reset" value="začni
      znova ">
     </ol></td>
    </tr>
  </table>
</ol>
</form>
</body>
</html>
```

//vyhod.jsp

```
<html>
<head>
<title>vyhodnotenie</title>
</head>
<%@ page contentType="text/html;charset=windows-1250"
%>
<%@ page import="java.sql.*,java.text.NumberFormat" %>
<body>
<%! int id;
    String m;
    String p;
    int b;
%>
<% Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   Connection connection = DriverManager.getConnection
   ("jdbc:odbc:dip","root","q");
   Statement statement = connection.createStatement();
   PreparedStatement pS1;
   pS1 = connection.prepareStatement
   ("insert into ziak values (?,?,?,?)");
   PreparedStatement pS2;
   pS2 = connection.prepareStatement
   ("insert into otazka values (?,?,?)");
   ResultSet resultset;
   resultset = statement.executeQuery
   ("select id from ziak");
   while (resultset.next()){
     id = resultset.getInt("id");
     }
   request.setCharacterEncoding("windows-1250");
```

```
%>
<%!  int hodnota;
     int o;
%>
<%   hodnota=0;
     o=0;
     if (request.getParameter("meno").equals(""))
       out.println("nezadali ste meno");
     else {
       if request.getParameter("priezvisko").equals(""))
         out.println("nezadali ste meno");
       else{
         if (request.getParameter("rb1").equals("1")){
          hodnota=hodnota+1;
          o=1;
         }
         pS2.setInt(1,id+1);
         pS2.setInt(2,1);
         pS2.setInt(3,o);
         pS2.executeUpdate();
         o=0;
         if (request.getParameter("rb2").equals("2")){
          hodnota=hodnota+1;
          o=1;
         }
         pS2.setInt(1,id+1);
         pS2.setInt(2,2);
         pS2.setInt(3,o);
         pS2.executeUpdate();
         o=0;
         if (request.getParameter("rb3").equals("3")){
          hodnota=hodnota+1;
          o=1;
         }
         pS2.setInt(1,id+1);
         pS2.setInt(2,3);
         pS2.setInt(3,o);
         pS2.executeUpdate();
         request.setCharacterEncoding("windows-1250");
         m = request.getParameter("meno");
         p = request.getParameter("priezvisko");
         b = hodnota;
         pS1.setInt(1,id+1);
         pS1.setString(2,m);
         pS1.setString(3,p);
         pS1.setInt(4,b);
         pS1.executeUpdate();
         resultset.close();
         statement.close();
         connection.close();
%>
```

```
Dosiahnutý počet bodov je :
<%out.print(hodnota);
%><br>
Maximálny počet bodov je : 3
</body>
<%   }
    }
%>
<br>
<a href='test.html'> späť na test</a>
```

Teraz vytvoríme oblasť aplikácie pre učiteľa. Po odoslaní požiadavky na zobrazenie výsledkov sa objaví stránka *vysledky.html*. Táto stránka bude vlastne prihlasovací formulár. Po zadaní správneho hesla („*domace*“) sa zobrazia učiteľovi výsledky žiakov zotriedené od najslabšieho (súbor *over.jsp*). Na tejto stránke bude mať učiteľ možnosť nechať si zobraziť podrobnosti o výsledkoch jednotlivých žiakov. Ak si túto možnosť vyberie, zobrazí sa stránka *odpoved.html*. Tu sa bude zadávať `id` daného žiaka. Túto požiadavku bude spracovávať *odpoved.jsp*. *Odpoved.jsp* zobrazí učiteľovi správnosť jednotlivých odpovedí vybraného žiaka.

//vysledky.html

```
<html>
<body>
<form action="over.jsp" method=post>
  Zadajte heslo: <input type=password name=heslo><br>
  <input type=submit value=Odoslat>
</form>
</body>
</html>
```

//over.jsp

```
<%@ page contentType="text/html;charset=windows-1250"
%>
<%@ page import="java.sql.*,java.text.NumberFormat" %>
<% request.setCharacterEncoding("windows-1250");
   if (request.getParameter("heslo").equals("domace")){
%>
<h2>Dosiahnuté výsledky žiakov : </h2><br>
<% Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   Connection connection = DriverManager.getConnection
   ("jdbc:odbc:dip","root","q");
   Statement statement = connection.createStatement();
   ResultSet resultset;
   request.setCharacterEncoding("windows-1250");
   resultset = statement.executeQuery
   ("select id, meno, priezvisko, body from ziak order
   by body");
```

```
   while (resultset.next()){
       out.print(resultset.getInt("id"));
       out.print(" ");
       out.print(resultset.getString("meno"));
       out.print(" ");
       out.print(resultset.getString("priezvisko"));
       out.print(" ");
       out.print(resultset.getFloat("body"));
       out.print("<br> ");
}
resultset.close();
statement.close();
connection.close();
%>
<br>
<a href="odpoved.html">ukázať jednotlivé odpovede</a>
<% } else {
%>
<% response.sendError(403); %>
<%  }
%>
```

//odpoved.html

```
<HTML>
<HEAD><TITLE>Zobrazenie záznamu</TITLE></HEAD>
<form action="odpoved.jsp" method=post>
  Id žiaka, ktorého výsledky chcete zobraziť:
  <input type=text name=id><br>
  <input type=submit value=Odoslat>
</form>
</HTML>
```

//odpoved.jsp

```
<%@ page contentType ="text/html;charset= windows-1250"
%>
<%@ page import="java.sql.*,java.text.NumberFormat" %>
<%! int i;%>
<% Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   Connection connection = DriverManager.getConnection
   ("jdbc:odbc:dip","root","q");
   Statement statement = connection.createStatement();
   ResultSet resultset;
   request.setCharacterEncoding("windows-1250");
   i=Integer.parseInt(request.getParameter("id"));
   resultset = statement.executeQuery
   ("select id_ziak,cislo_otazky,spravne from otazka");
   while (resultset.next()){
      if (resultset.getInt("id_ziak")==i){
         out.print("číslo otázky : ");
         out.print(resultset.getInt("cislo_otazky"));
         out.print(" počet bodov : ");
```

```
            out.print(resultset.getFloat("spravne"));
            out.print(" <br>");}
      }
      resultset.close();
      statement.close();
      connection.close();
%>
```

4.3.5. Cvičenia

1. Vytvorte webovú stránku, na ktorej sa bude dať v jednotlivých záznamoch v tabuľke *človek* na základe určenia priezviska človeka zmeniť jeho výška uložená v tabuľke.

//vyska.html

```
<HTML>
<HEAD><TITLE>Aktualizácia výšky záznamu</TITLE></HEAD>
<form action="aktualizacia.jsp" method=post>
  Priezvisko človeka, ktorého výšku chcete zmeniť:
  <input type=text name=priezvisko><br>
  Aktuálna výška: <input type=text name=vyska><br>
  <input type=submit value=Odoslat>
</form>
</HTML>
```

//aktualizacia.jsp

```
<%@ page contentType = "text/html;charset=windows-1250"
%>
<%@ page import="java.sql.*" %>
<H1>Aktualizácia záznamu z tabuľky človek</H1>
<%! String p;
    double v;
%>
<% Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   Connection connection = DriverManager.
   getConnection("jdbc:odbc:dip","root","q");

   Statement statement = connection.createStatement();
   PreparedStatement pS;
   pS = connection.prepareStatement
   ("update clovek set vyska=? where priezvisko= ?");
   ResultSet resultset;
   request.setCharacterEncoding("windows-1250");
   p = request.getParameter("priezvisko");
   v=Double.parseDouble(request.getParameter("vyska"));
   pS.setDouble(1,v);
   pS.setString(2,p);
   pS.executeUpdate();
%>
   Záznam bol úspešne zmenený. <BR><BR>
   Záznamy v tabuľke : <BR>
```

```
<% resultset = statement.executeQuery
   ("select * from clovek ");
   while (resultset.next()){
     out.print(resultset.getString("meno"));
     out.print(" ");
     out.print(resultset.getString("priezvisko"));
     out.print(" ");
     out.print(resultset.getDouble("vyska"));
     out.print("<br> ");
   }
   resultset.close();
   statement.close();
   connection.close();
%>
```

2. Vytvorte webovú stránku, na ktorej sa budú dať vykonávať všetky potrebné operácie so záznamami v tabuľke človek. (Stránka bude ponúkať možnosť vkladania záznamov do tabuľky, vyberať záznamy, po zadaní `id` dáta aktualizovať a vymazávať.)

//main.html

```
<HTML>
<HEAD><Title>praca s datami</Title></HEAD>
<body>
<form action="main.jsp" method=post>
<h1>Akú operáciu si želáte robiť?</h1>
  <input type="radio" name="odp" value="vloz">
  vkladanie záznamov<br>
  <input type="radio" name="odp" value="vyber">
  výpis záznamov<br>
  <input type="radio" name="odp" value="vymaz">
  vymazanie záznamu<br>
  <input type="radio" name="odp" value="zmen">
  aktualizácia záznamu<br>
  <input type=submit value=Odoslat>
</form>
</body>
</HTML>
```

//main.jsp

```
<%@ page contentType = "text/html;charset=windows-1250"
%>
<%@ page import="java.sql.*,java.text.NumberFormat" %>
<%  request.setCharacterEncoding("windows-1250");
    if (request.getParameter("odp").equals("vloz")){
%>
<%@ include file="vloz.html" %>
<%  }
    if (request.getParameter("odp").equals("vyber")){
```

```
%>
<%@ include file="vybrat.jsp" %>
<%  }
    if (request.getParameter("odp").equals("vymaz")){
%>
<%@ include file="vymaz.html" %>
<%  }
    if (request.getParameter("odp").equals("zmen")){
%>
<%@ include file="zmen.html" %>
<%  }
%>
```

Na vkladanie údajov, výpis záznamov, vymazávanie záznamov použijeme JSP stránky z teoretickej časti lekcie, ktoré doplníme o nasledujúci kód :

```
// <br>
<a href='main.html'>späť na úvodnú stránku</a>
```

//zmen.html

```
<html>
<head><title>aktualizácia záznamov</title></head>
<body>
<form action="zmen.jsp" method=post>
<h1>Aktualizácia údajov v tabuľke človek</h1>
  ID záznamu, ktorý chcete zmeniť :
  <input type=text name=id><br>
  Aktuálne údaje :<br>
  MENO: <input type=text name=meno><br>
  PRIEZVISKO: <input type=text name=priezvisko><br>
  VYSKA: <input type=text name=vyska><br>
  <input type=submit value=Vlozit>
</form>
</body>
</html>
```

//zmen.jsp

```
<%@ page contentType = "text/html;charset=windows-1250"
%>
<%@ page import="java.sql.*" %>
<H1>Aktualizácia záznamu z tabuľky človek</H1>
<%! int i;
    String m;
    String p;
    double v;
%>
<% Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   Connection connection = DriverManager.
   getConnection("jdbc:odbc:dip","root","q");
```

```
    Statement statement = connection.createStatement();
    PreparedStatement pS;
    pS = connection.prepareStatement
    ("update clovek set meno=?,priezvisko=?,vyska=?
    where id= ?");
    ResultSet resultset;
    request.setCharacterEncoding("windows-1250");
    i = Integer.parseInt(request.getParameter("id"));
    m = request.getParameter("meno");
    p = request.getParameter("priezvisko");
    v=Double.parseDouble(request.getParameter("vyska"));
    pS.setString(1,m);
    pS.setString(2,p);
    pS.setDouble(3,v);
    pS.setInt(4,i);
    pS.executeUpdate();
%>
    Záznam bol úspešne zmenený. <BR><BR>
    Záznamy v tabuľke : <BR>
<% resultset = statement.executeQuery
    ("select * from clovek ");
    while (resultset.next()){
      out.print(resultset.getString("meno"));
      out.print(" ");
      out.print(resultset.getString("priezvisko"));
      out.print(" ");
      out.print(resultset.getDouble("vyska"));
      out.print("<br> ");
    }
    resultset.close();
    statement.close();
    connection.close();
%>
<br>
<a href='main.html'>späť na úvodnú stránku</a>
```

4.3.6. Úlohy

1. Vytvorte v databáze tabuľku na uchovávanie výsledkov z písomiek (nech obsahuje meno, počet bodov, známku). Vytvorte webovú stránku, na ktorej sa budú dať vkladať záznamy do pripravenej tabuľky a po stlačení tlačidla na vyhodnotenie sa vypíšu všetky záznamy zoradené od najlepšieho výsledku po najslabší.
2. Upravte tretí príklad z lekcie 2 tak, aby sa správne odpovede načítavali z databázy.
3. Vytvorte webovú stránku pre domáce úlohy, na ktorú sa budú žiaci prihlasovať (jedno správne heslo). Z databázy sa načíta domáca úloha a žiak po vypracovaní odošle správnu odpoveď, ktorá sa s menom žiaka uloží do databázy.

4.3.7. Autotest

1. Vyberte nesprávne tvrdenie :
   a) JSP nám umožňuje prácu s databázou, pretože databáza je súčasťou JSP.
   b) Prostredníctvom JDBC môžete posielať príkazy v jazyku SQL mnohým databázam.
   c) JSP nám umožňuje spoluprácu s databázou, ale až po tom, ako sa na ňu pripojíme.

Správna odpoveď : C

Počet bodov : 1b

2. Čo je potrebné doplniť do príkazu Class.forName(" ... "); ?
   a) protokol:názov_zdroja_dát
   b) názov databázového ovládača
   c) užívatelské_meno

Správna odpoveď : B

Počet bodov : 1b

3. Čo vykonáva dokument xxx.jsp?

```
<%@ page contentType = "text/html;charset = windows-1250" %>
<%@ page import="java.sql.*,java.text.NumberFormat" %>

    <H1> xxx </H1>

    <% Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    Connection connection =
    DriverManager.getConnection("jdbc:odbc:dip","root","q");
    Statement statement = connection.createStatement();
    statement.executeUpdate ("create table vysledky (prvy varchar(15),druhy
    varchar(15),treti varchar(15),stvrty varchar(15),piaty varchar(50));" );

    statement.close();
    connection.close();
    %>
```

   a) vytvorí spojenie s tabuľkou výsledky
   b) vytvorí spojenie s databázou a vytvorí tabuľku výsledky
   c) vytvorí spojenie s tabuľkou výsledky a aktualizuje ju

Správna odpoveď : B

Počet bodov : 1b

5. Pedagogický experiment

5.1. Návrh a popis pedagogického experimentu

Každý pripravovaný učebný text je potrebné pred nasadením do praxe overiť na nejakej cieľovej skupine študentov.

V zimnom semestri školského roku 2005/2006 sa vyučoval predmet *Seminár z programovania v sieťach*. Predmet je odporúčaný pre študentov piateho ročníka odboru informatika. V rámci tohto predmetu sa študenti zoznamujú s rôznymi technológiami pre programovanie v sieťach, jednou z nich je technológia JSP.

Po dohode s vedúcim diplomovej práce a vyučujúcim tohto predmetu som realizovala pedagogický experiment k mojej diplomovej práci. Cieľovou skupinou bolo 12 študentov štvrtého a piateho ročníka odboru informatika.

Cieľom pedagogického experimentu bolo zistiť, ako organizovať výučbu a aké metódy pri výučbe použiť, aby sa čo najviac zefektívnila. Ďalším cieľom bolo overiť zrozumiteľnosť a primeranosť učebného textu.

Pred začatím samotnej výučby sme sa spoločne venovali inštalácii potrebného softvéru a nastaveniam systému. Začiatok výučby mal podobu výkladu. Úvodný výklad som venovala predstaveniu technológie JSP, možnostiam jej použitia a vysvetleniu princípu fungovania. Neskôr výučba prebiehala tak, že som sa snažila vystupovať v úlohe konzultanta. Študenti pracovali samostatne s pripravenými elektronickými učebnými textami. Vlastným tempom študovali teoretickú časť a riešené príklady, riešili pripravené úlohy a získané vedomosti si overovali autotestami.

Spätnú väzbu som od študentov získavala nasledujúcim dotazníkom:

1. Zaujala Vás problematika programovania dynamických www stránok pomocou technológie JSP ?
   a) áno, veľmi
   b) trochu
   c) vôbec nie
2. Odporúčali by ste vyučovať JSP na vysokej škole :
   a) ako samostatný predmet
   b) v rámci nejakého predmetu o sieťach alebo internete
   c) vôbec nevyučovať
3. Zrozumiteľnosť učebného textu je :
   a) nízka
   b) priemerná
   c) vysoká
4. Náročnosť danej problematiky je :
   a) nízka
   b) priemerná
   c) vysoká
5. Čo sa Vám na učebnom texte páčilo, prípadne nepáčilo?

5.2. Závery a odporúčania pre prax

Ukázalo sa, že tvorba dynamických webových stránok pomocou JavaServer Pages je pre študentov zaujímavá. Poskytuje im možnosť voľného experimentovania a realizovania tvorivých nápadov. Študenti by odporúčali vyučovať JSP v škole, po väčšinou v rámci nejakého predmetu o sieťach či internete. So zrozumiteľnosťou učebného textu bola väčšina z nich spokojná. Náročnosť bola vysoká len pre zopár študentov, ostatným sa zdala primeraná. Študenti by privítali viac riešených aj neriešených praktických príkladov.

Učiteľ musí venovať zvýšenú pozornosť príprave na vyučovanie. Je veľmi dôležité pripraviť kvalitné učebné texty s praktickými príkladmi. Na nich by mali študenti možnosť vidieť význam a užitočnosť sprístupňovaného učiva. Užitočné na elektronických učebných textoch je, že študenti môžu študovať priamo z obrazovky, alebo si v prípade potreby texty vytlačiť. V rámci príkladov je veľmi dôležité dať

študentom k dispozícii zdrojové kódy. Študenti potom majú možnosť otestovať ich funkčnosť, editovať ich a vyskúšať rôzne nové modifikácie, čím môžu látku lepšie pochopiť.

Veľmi dôležitá je motivácia študentov. Študenti by mali byť oboznámení s cieľom každej vyučovacej jednotky. Mali by správne chápať zmysel a praktické využitie preberaného učiva.

Odporúčam v maximálnej miere využívať metódy, pri ktorých študenti pracujú samostatne, rozvíja sa ich aktívnosť a tvorivosť. Ako veľmi vhodná sa javí projektová metóda. Snahou učiteľa by malo byť obmedziť výklad na minimum a vystupovať hlavne v úlohe konzultanta (najmä pri riešení príkladov). Nie je totiž možné dosiahnuť dostatočnú motiváciu a požadovaný výsledok len pomocou výkladu. Vhodná je aj dialogická metóda. Tá je však časovo náročná a nie vždy dostatočne realizovateľná.

Za dôležité pokladám vyhradiť priestor pre diskusiu a vysvetlenie prípadných nejasností vzniknutých počas práce na konci každej vyučovacej jednotky.

Pri vyučovaní tvorby webových stránok veľkú úlohu zohrávajú praktické príklady. Preto je potrebné vyčleniť dostatok času na precvičovanie, realizovanie tvorivých nápadov a experimentovanie.

Záver

Cieľom práce bolo vytvoriť elektronický učebný text k tvorbe JSP stránok. Tento učebný text je určený pre prvé oboznámenie sa s technológiou JavaServer Pages. Pozornosť som venovala popísaniu základných princípov fungovania a tvorby JSP stránok. Teoreticky aj na praktických príkladoch som ukázala najpoužívanejšie možnosti, ktoré JSP ponúka.

Snažila som sa vytvoriť učebný text, ktorý študentom pomôže v zorientovaní sa v možnostiach tvorby dynamických webových stránok pomocou JSP a uľahčí im prvé kroky v praktickom využívaní JSP.

V práci sa mi podarilo vytvoriť učebný text obsahujúci potrebnú teóriu, riešené a neriešené úlohy a autotesty. Všetky úlohy som sa snažila orientovať do praxe, aby študent videl význam a efektívnosť preberaného učiva.

Zároveň som učebný text overila v praxi a získala nové skúsenosti s výučbou. Väčšinu študentov tvorba dynamických webových stránok zaujala a odporúčali by túto technológiu vyučovať. Preto si myslím, že by budúci učitelia informatiky mali byť oboznámení s možnosťami JSP v rámci svojho štúdia.

Moja práca bude súčasťou Virtuálnej univerzity, preto si myslím, že nájde v budúcnosti uplatnenie a pomôže mnohým študentom.

Literatúra

1. Marry Hall: Core Servlets and JavaServer Pages, Sun Microsystem A Prentice Hall Tittle, 2004
2. Barry Burd : JSP – JavaServer Pages, Hungry Minds, 2004
3. http://interval.cz
4. http://java.sun.com
5. Zoznam vysokých škôl na Slovensku- http://www.univerzita.net

Príloha 1: CD

Na priloženom CD sa nachádzajú štyri adresáre : praca, install, skripty a stranka. Adresár praca obsahuje dokument dp.pdf s textom diplomovej práce. Adresár stranka obsahuje elektronický učebný text. Adresár skripty obsahuje všetky skripty použité v učebnom texte. Adresár install obsahuje všetky potrebné programy nevyhnutné na testovanie skriptov a off–line spustenie elektronického učebného textu. V tomto adresári nájdete dokument install.pdf popisujúci inštalácie programov a potrebné nastavenia.

Príloha 2: Webová stránka

On–line verzia učebného textu je prístupná na URL adrese : http://klud.ics.upjs.sk/~slavka/praca.