

Webová aplikace pro generování zadání a kontrolu výsledků laboratorních cvičení

Petr Žák



Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav aplikované informatiky

akademický rok: 2005/2006

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Petr ŽÁK**

Studijní program: **B 3902 Inženýrská informatika**

Studijní obor: **Informační technologie**

Téma práce: **Webová aplikace pro generování zadání a kontrolu
výsledků laboratorních cvičení**

Zásady pro vypracování:

1.Vytvořte webovou aplikaci pro generování jedinečného zadání pro laboratorní práci předmětu Analýza a simulace technologických procesů.

2.Provedte příslušnou rešerši o databázových systémech a programovacích jazycích vhodných pro vytvoření této aplikace.

3.Databáze zadání by měla být propojena i s databází výsledků, takže se vyučujícímu přímo ukáže u každého studenta nejen zadání, ale i výsledky.

4.Pro získání seznamů studentů přihlášených do daného předmětu použijte napojení na informační systém MOODLE popř. zkuste aplikaci zabudovat přímo do tohoto systému.

Rozsah práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] <http://cz.php.net/> – on-line manuál k programovacího jazyka PHP

[2] <http://moodle.org/> – webové stránky informačního systému MOODLE

[3] Matlab programming. The Mathworks Inc., 2000

Vedoucí bakalářské práce:

Ing. Jiří Vojtěšek
Ústav řízení procesů

Datum zadání bakalářské práce:

14. února 2006

Termín odevzdání bakalářské práce:

16. června 2006

Ve Zlíně dne 14. února 2006


prof. Ing. Vladimír Vašek, CSc.
pověřený děkan




doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Cílem této práce je vytvořit webovou aplikaci pro generování zadání do předmětu Analýza a simulace technologických procesů. V teoretické části osvětlím některé aspekty databázových systémů a bude zde taky řeč o databázových a programovacích jazycích. V praktické části se zaměřím na popis a funkci jednotlivých částí aplikace, která komunikuje s Moodle.

Klíčová slova: databázové systémy, databázové a programovací jazyky, Moodle

ABSTRACT

The goal of this work is create a web application for setting generation to the object of Analysis and simulation of technological processes. In theoretical part I will throw light on the matter for some aspects of database systems and it will be discussed about database and programming languages too. In practical part I will focus the description and function of single parts of application which is communicating with Moodle.

Keywords: database systems, database and programming languages, Moodle

...

Skutečnost je taková, že velbloudi jsou mnohem inteligentnější než delfini.* Jsou o tolik bystřejší, že si rychle uvědomili, že nejprozíravější věc, kterou může chytré zvíře udělat, pokud nechce, aby jeho potomci strávili většinu života na laboratorním stole s elektrodami v mozku, přesunem námořních min ze základny A na loď B, nebo pod přihlouplým dozorem biologů, je udělat všechno pro to, aby to ti zas... lidé nezjistili. Takže se už dávno a dávno uchýlili ke svému životnímu stylu, který s sebou nese povinnost nosičů a píchání tyčemi, ale také jistotu potravy, ošetření a možnost občas lidem beztrestně naplivat do očí.

A tenhle konkrétní velbloud, výsledek milion let dlouhého výběrového vývoje, na jehož základě vzniklo stvoření, které bylo schopno spočítat zrnka písku, po kterém kráčelo, uzavřít nozdry dle libosti a přežít pod žhavým sluncem pouště mnoho dnů bez vody, se jmenoval Ty Mrcha.

A ve skutečnosti to byl největší matematik na světě.

Ty Mrcha si myslel: zdá se, že tady roste rozměrová nestabilita, která se podle vnějších projevů pohybuje v rozmezí od nuly do pětáctyřiceti stupňů. To je zajímavé. Tak by mě zajímalo, co to způsobuje? Tak dejme tomu, že v se rovná 3. Pak nechť se τ rovná π lomeno 4. Žvýkžvýkžvýk. Nechť je κ lomeno y diferenciální tenzor odporu Smrdutého Ksindla** týkající se oblasti se čtyřmi imaginárními spinovými koeficienty...

* Pozn. Autora: *Nikdy nevěřte druhu, který se neustále usmívá. Něco s ním není v pořádku.*

** Pozn. Autora: *Smrdutý Ksindl byl proslulý jako největší velbloudí matematik všech dob, který rozpracoval matematiku osmirozměrného prostoru, a to ve chvíli, kdy s uzavřenými nozdrami ležel uprostřed písečné bouře.*

Děkuji vedoucímu práce, panu Ing. Jiřímu Vojtěškovi za podporu a Ing. Tomáši Dulíkovi za cenné informace.

OBSAH

ÚVOD.....	8
I TEORETICKÁ ČÁST	9
1 DATABÁZOVÉ SYSTÉMY.....	10
1.1 DATABÁZE	10
1.2 SYSTÉM PRO OVLÁDÁNÍ SOUBORŮ.....	10
1.3 DATABÁZOVÉ SYSTÉMY SE SYSTÉMEM ŘÍZENÍ BÁZE DAT	11
1.4 DATOVÉ MODEL Y	11
1.4.1 Hierarchický model.....	11
1.4.2 Síťový model.....	12
1.4.3 Relační model.....	13
1.5 STRUKTURA DATABÁZOVÉHO SYSTÉMU	13
1.6 UŽIVATELÉ DATABÁZOVÉHO SYSTÉMU	14
1.7 MOODLE.....	14
2 DATABÁZOVÉ JAZYKY	15
2.1 JAZYK SQL	15
2.1.1 Jazyk pro definici dat (DDL)	15
2.1.2 Jazyk pro manipulaci s daty (DML).....	15
3 PROGRAMOVACÍ JAZYKY	16
3.1 HTML.....	16
3.2 PHP.....	16
3.3 CSS STYLY.....	17
3.4 XML	17
3.4.1 XML parser	18
II PRAKTICKÁ ČÁST.....	20
4 STRUKTURA WEBOVÉ APLIKACE.....	21
5 POPIS JEDNOTLIVÝCH ČÁSTÍ.....	23
5.1 INSTALACE	23
5.2 PŘÍPRAVA APLIKACE	23
5.3 PROSTŘEDÍ PRO STUDENTA (SKRIPT STUDENT.PHP)	25
5.3.1 Vložit/upravit výsledek	25
5.3.2 Vložit soubor	25
5.3.3 Vymazání souborů	26
5.4 PROSTŘEDÍ PRO VYUČUJÍCÍHO (SKRIPT TEACHER.PHP).....	27
5.4.1 Menu	27
5.4.2 Panel pro výběr a třídění zadání.....	28
5.4.3 Tabulka.....	28

5.5	STRUČNÝ POPIS OSTATNÍCH SKRIPTŮ.....	28
ZÁVĚR.....		30
SEZNAM POUŽITÉ LITERATURY		31
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK		32
SEZNAM OBRÁZKŮ		33

ÚVOD

V dnešní době je internet samozřejmostí prakticky pro každého z nás. Pomocí internetu např. vyhledáváme dopravní spoje, platíme účty, objednáváme zboží z internetových obchodů atd. Setkáváme se tak s velkým množstvím informací, které nám nějakým způsobem poslouží k našemu pohodlí nebo potřebám. Ale kde je takové množství dat uloženo a kdo nebo co se stará o správu těchto dat? Odpovědi na tyto otázky jsou databázové systémy.

V teoretické části zde budou popsány základní věci (toto nejsou skripta, proto nečekejte vyčerpávající popis) a principy, které se databázových systémů týkají. Proto by měl případný čtenář po přečtení této práce nabýt pocitu, že o databázových systémech už něco ví, popřípadě se možná doví něco nového. S databázovými systémy také souvisí databázové a programovací jazyky, o kterých zde bude také řeč. Protože aplikace je webového charakteru, budou zde popsány programovací jazyky, které byly za tímto účelem použity.

Praktická část bude zaměřena na tvorbu a popis jednotlivých částí webové aplikace, která je vytvořena převážně pomocí jazyka PHP. Webová aplikace je vytvořena tak, aby byla přehledná a její ovládání jednoduché. Pro grafickou úpravu je použit stylový jazyk CSS, který umožňuje efektivně a rychle měnit vzhled aplikace.

Aplikace komunikuje s výukovým systémem Moodle, ze kterého získává potřebné údaje (id uživatelů, jejich jména atd.) a jsou potřebné pro správný běh aplikace. Jakým způsobem aplikace komunikuje s Moodle, bude popsáno v praktické části. Aplikace je vytvořena jako externí program, tzn., že není přímo zabudována do Moodle jako samostatný modul. Rozhodl jsem se tak z toho důvodu, že jsem chtěl použít svoje postupy, o kterých jsem byl přesvědčen že budou dobré a snadno programovatelné. Avšak postupem času jsem zjistil, že tomu tak docela není a že vytvoření modulu, který by byl do Moodle zakomponován, by usnadnilo některé programátorské postupy a rozhodně by aplikace byla výhodnější i po stránce bezpečnosti.

I. TEORETICKÁ ČÁST

1 DATABÁZOVÉ SYSTÉMY

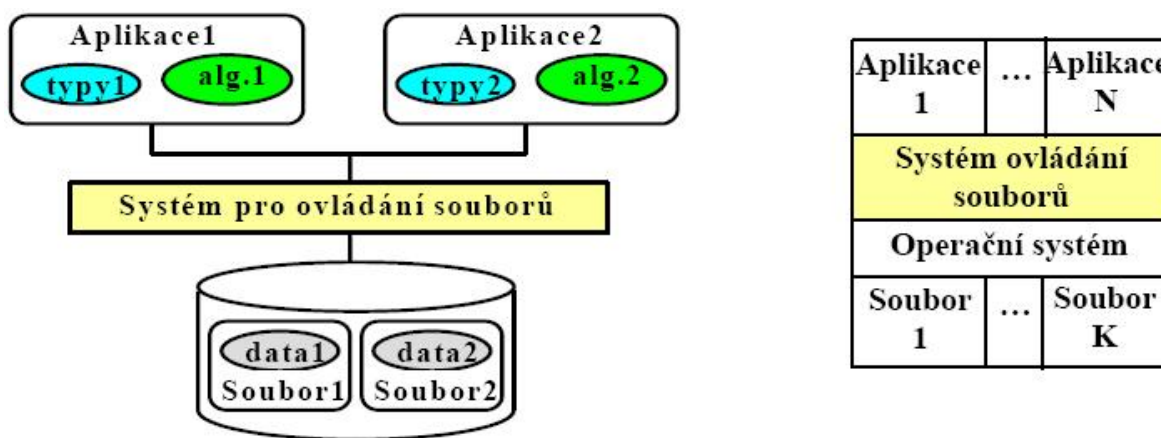
Databázový systém si můžeme představit jako program, který se stará o správu dat, které jsou uloženy v databázi. S těmito daty provádí různé operace, jako např. zobrazení, modifikaci, mazání a ukládání apod. Databázové systémy můžeme podle vývoje rozdělit na dvě základní skupiny, a to databázové systémy pro ovládání souborů a databázové systémy se správou řízení báze dat. Nejdříve bychom si měli ale říct, co to ta databáze vlastně je.

1.1 Databáze

Na pojem databáze existuje nejedna definice, ale všechny v podstatě říkají totéž. Databázi si můžeme představit jako místo (tabulku), kde jsou uloženy potřebné informace z dané oblasti (např. bankovníctví), se kterými se dále (opakovaně) pracuje v rámci databázového systému.

1.2 Systém pro ovládání souborů

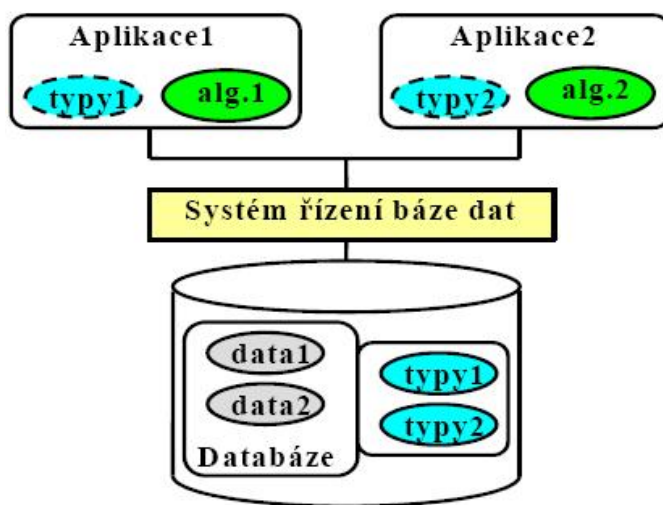
Tyto databázové systémy se začaly realizovat s příchodem operačních systémů (OS) se systémem souborů. Aplikace obsahovaly popis dat i vlastní algoritmus. Data byli uložena v souborech a operace s nimi se prováděly pomocí speciálních příkazů jazyka (tehdy Cobol, Pascal,...). Modernější OS přinesly vzájemnou komunikaci při práci více uživatelů. Pro pohodlnější a efektivnější práci se dodávali specializované knihovní funkce, které umožňovali definování a přístup na úrovni záznamů. Příkladem souborového databázového systému, který se dnes používá, by mohl být MS Access.



Obr. 1. Systém pro ovládání souborů

1.3 Databázové systémy se systémem řízení báze dat

Koncem šedesátých a začátkem sedmdesátých let vznikly první systémy, poskytující úplnější řešení databázových požadavků. Dostali název systémy řízení báze dat (SŘBD). SŘBD je účinný nástroj pro správu velkého množství dat. Umožňuje uchovat rozsáhlé množství dat, které existují nezávisle na jakémkoli procesu, který s daty pracuje, umožňuje modifikovat data pomocí dotazovacího jazyku a podporuje souběžný přístup různých procesů (transakcí) k datům najednou.



Obr. 2. Systém řízení báze dat

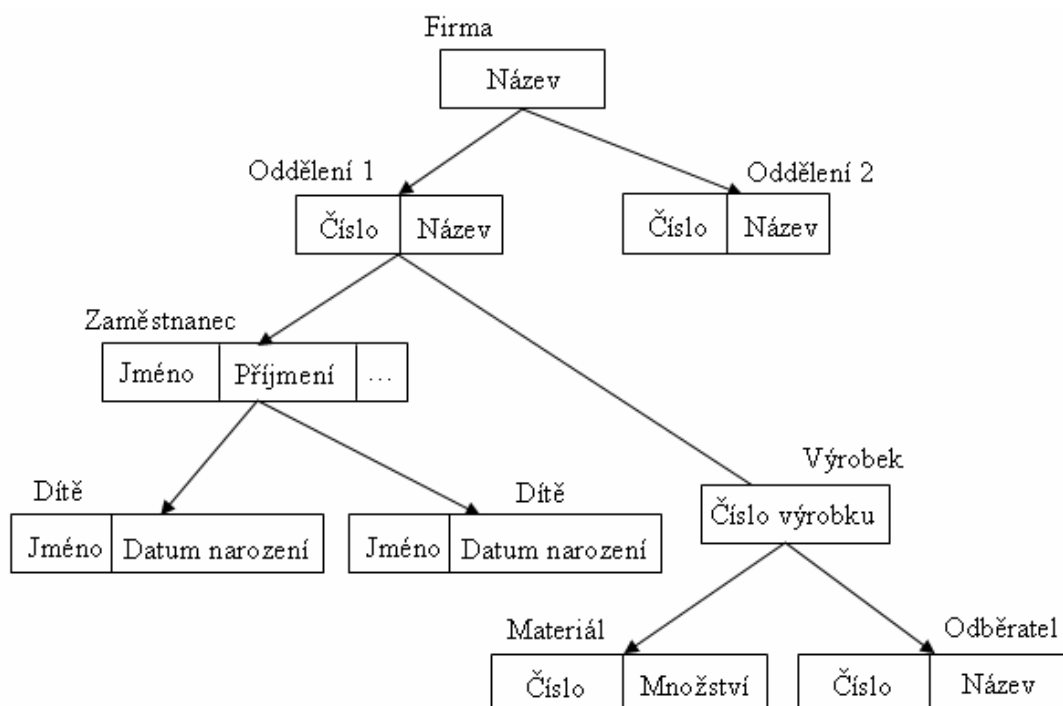
1.4 Datové modely

S vývojem databázových systémů vznikaly i různé datové modely. Datové modely popisují logickou strukturu a uložení dat v databázích a vztahy mezi nimi.

1.4.1 Hierarchický model

V hierarchickém modelu jsou data organizovány ve stromové struktuře. Uplatňují se zde vztahy rodič-potomek, které jsou zobrazovány 1:N. Potomek tedy může mít pouze jednoho rodiče. Jako příklad mějme firmu, která má více provozoven a v každé se vyrábí jiný výrobek. Ta si vede záznamy jak o zaměstnancích, tak i o jednotlivých provozovnách. U zaměstnance potřebuje znát jeho jméno, příjmení, adresu a kupodivu si vede záznamy i o jeho dětech, které obsahují jméno a datum narození. Zaměstnanec a údaje o dítěti vytváří hierarchii, ve které data o zaměstnanci reprezentují rodičovský segment a data o dítěti segment potomka. Když bude mít zaměstnanec 4 děti, pak budou 4 segmenty potomka

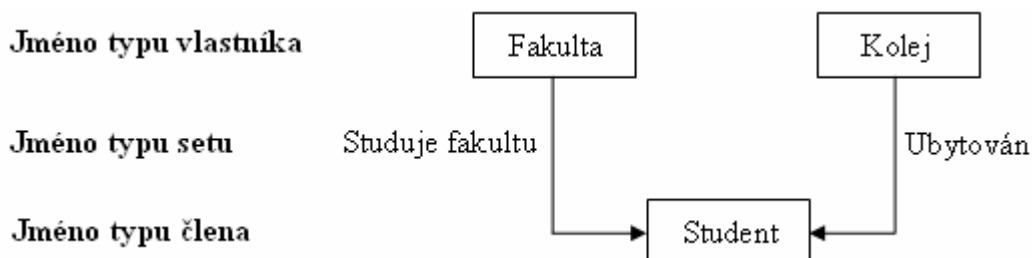
spojeny s jedním segmentem rodiče. Podobně by to vypadalo i u výrobků, které jsou z nějakého materiálu a mají např. odběratele. Hierarchické SRBD byly populární od konce šedesátých let, představením IBM systému řízení informací (Information Management System – IMS) do let sedmdesátých.



Obr. 3. Hierarchický model

1.4.2 Síťový model

Síťový model je obdobou hierarchického, jenom s tím rozdílem, že mimo vztahů 1:N umožňuje vytvářet i vztahy M:N. Tento model formálně definovala v roce 1971 CODASYL (Conference on Data Systems Languages). Základem je modelování je vytvoření tzv. setu, který se skládá z typu záznamu vlastníka (owner), jména typu setu a typem záznamu člena (member) setu.



Obr. 4. Síťový model

1.4.3 Relační model

Teoretický základ relačního modelu položil E. F. Codd v roce 1970. Data v relačním modelu jsou uloženy v tabulkách, kde řádky reprezentují záznam jednoho typu a sloupce jejich atributy. U tohoto typu modelu se využívá pro manipulaci s daty relační algebry, např. sjednocení, průnik, výběr aj.

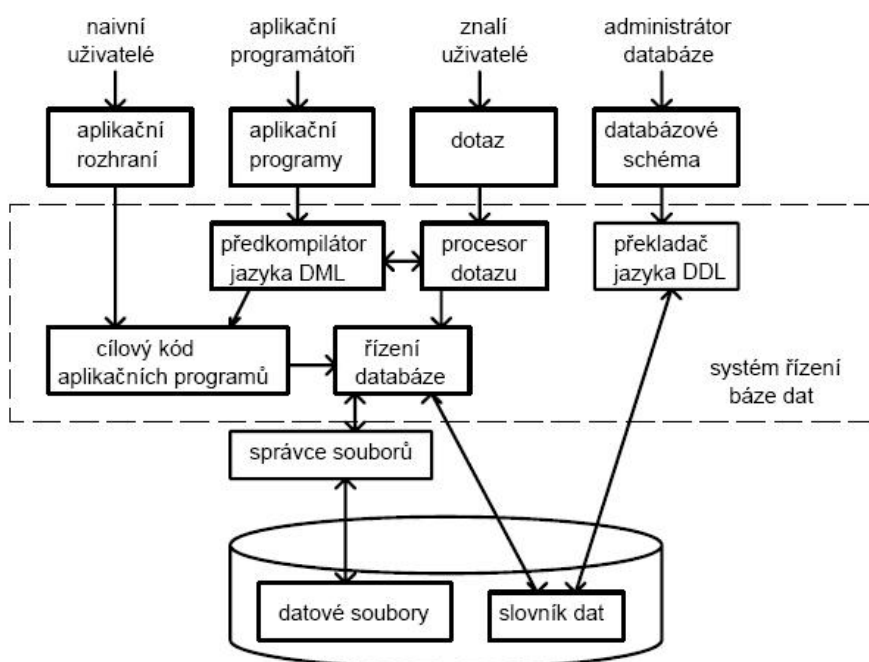
č_klienta	jméno	adresa
100	Pavla Zelená	Hájkova 1, Brno
200	Jan Novák	Cejl 8, Brno

č_úctu	stav	majitel
800	10856	100
557	117543	200
486	57800	100

Obr. 5. Relační model

1.5 Struktura databázového systému

Databázový systém sestává z různých částí, viz. Obr. 6. [3]. O některých z nich ještě bude řeč.



Obr. 6. Struktura databázového systému

1.6 Uživatelé databázového systému

S databázovými systémy přichází do styku různí uživatelé. Podle toho, kdo s databázovým systémem přijde do styku se uživatelé dělí do následujících kategorií [11]:

- Administrátor databáze – zajišťuje centrální kontrolu nad daty a programy, které je využívají, plní rozhodnutí správce dat. Mezi jeho funkce patří definice schématu, definice paměťové struktury a přístupové metody, modifikace schématu a fyzické organizace, přidělování práv přístupu.
- Aplikační programátor – vytváří aplikační programy využitím prostředků jazyka pro manipulaci s daty vloženými do hostitelského jazyka, nebo použitím problémově orientovaných neprocedurálních jazyků.
- Znalý uživatel – nepíše programy, ale je schopen formulovat požadavky v databázovém jazyce pro manipulaci s daty.
- Naivní uživatel – komunikuje se systémem pouze prostřednictvím aplikačního programu.

Jelikož v praktické části bude řeč o výukovém informačním systému Moodle, tak se o něm krátce zmíním.

1.7 Moodle

Moodle (Modular Object-Oriented Dynamic Learning Environment) je volně šiřitelný Open-Source programový balík, který se vyvíjí od roku 1999, slouží pro podporu výuky prostřednictvím online kurzů a obsahuje mimo jiné docela rozsáhlou databázi. Tento program je stále vyvíjen komunitou vývojářů z celého světa a je optimalizován na různé systémy, jako např. Unix, Linux, Windows aj.

2 DATABÁZOVÉ JAZYKY

V předchozí kapitole jsem se zmínil o systému řízení báze dat. Jazyk, kterému SŘBD „rozumí“, se někdy označuje jako dotazovací. Typickým dotazovacím jazykem, který se používá v databázových systémech, je SQL.

2.1 Jazyk SQL

SQL (Structured Query Language) byl vyvinut společností IBM v r. 1974. Má dvě hlavní části – první „část“ slouží pro definování datové struktury (např. vytváření tabulek, indexů) a druhá část slouží pro manipulování s daty. Označují se jako DDL (Data Definition Language) a DML (Data Manipulation Language).

2.1.1 Jazyk pro definici dat (DDL)

Pro definování dat slouží v SQL sada příkazů. Nejpoužívanější je příkaz pro vytváření nových objektů CREATE, příkaz ALTER pro změny existujících objektů a příkaz DROP pro odstranění objektů.

Příklad:

```
CREATE TABLE zamestnanci  
(prijmeni varchar(20) not null, jmeno varchar(20) not null, plat int(5) not null);
```

V tomto případě se vytvoří tabulka s názvem zaměstnanci. První sloupec má název prijmeni, druhý jmeno a třetí plat. Za názvy sloupců následují typy vkládaných dat, v tomto případě varchar, jako libovolný znak o délce 20 znaků a int, jako celočíselný údaj o délce 5 znaků. Not null zde značí, že údaj bude nenulový.

2.1.2 Jazyk pro manipulaci s daty (DML)

Pro manipulování s daty souží v SQL nejčastěji příkazy SELECT pro výběr dat, INSERT pro vkládání dat, UPDATE pro modifikování dat a DELETE pro mazání.

Příklad:

```
SELECT jmeno, prijmeni FROM zamestnanci WHERE plat <= 10000;
```

Příkaz SELECT vybere jméno a příjmení zaměstnance z tabulky zamestnanci, kteří mají menší plat než 10000.

3 PROGRAMOVACÍ JAZYKY

Databázové systémy jsou naprogramovány v různých jazycích. V duchu této práce se tedy zaměřím na jazyky, které souvisí s webem a kterými jsou napsány některé databázové systémy.

3.1 HTML

HTML (Hypertext Markup Language) se používá na tvorbu webových stránek. Předchůdcem HTML byl SGML, který sloužil pro manipulaci s elektronickými dokumenty, ale pro jeho složitost byl v roce 1990 navržen jazyk HTML. Základem HTML jazyka jsou příkazy, tzv. tagy, které určují, jak se mají dané data zobrazit. Dokument HTML má svoji základní syntaxi, která vypadá takto:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>Titulek</title>
</head>
<body>
...
</body>
</html>
```

HTML je co se týká příkazů hodně benevolentní, tzn., že pokud narazí na nějaký příkaz, který nezná, tak ho ignoruje a pokračuje dále. V dnešní době je používána verze HTML 4.01, která už se dále nevyvíjí a je nahrazen novějším XHTML.

3.2 PHP

PHP (původně Personal Home Page, následně pak zkratka pro PHP: Hypertext Preprocessor) je skriptovací jazyk, který se vykonává na straně serveru. Za zrodem tohoto jazyka (1994) stojí Rasmus Lerdorf, který si naprogramoval rychlý hack v Perlu jako počítaadlo na své stránky. PHP je používán hlavně u webových aplikací, protože díky svým možnostem umožňuje snadnou komunikaci se serverem nebo databázemi.

Příklad skriptu:

```
<?php echo "Today is ".Date("l").",".Date("j.n.Y") ?>
```

Vypíše např.: Today is Tuesday, 5.8.2006

3.3 CSS styly

V této době jdou ruku v ruce s PHP a HTML CSS styly, proto se o nich taky zmíním. CSS styly se používají k definování vlastností a vzhledu elementů v HTML (XHTML, XML) a určují, jak mají být zobrazeny. Výhodou těchto stylů je to, že pro určitý element můžeme definovat jeho vzhled v celém dokumentu jediným zápisem pro tento element. Zápis CSS stylu je nazýván pravidlem a každé pravidlo se skládá ze dvou částí – selektoru a bloku deklarací. Selektor může být např. prvek table a deklarace např. width: 100%, kde width je vlastnost a 100% je hodnota.

Obecný zápis CSS stylu:

```
selektor {vlastnost1: hodnota_vlastnosti1; vlastnost2: hodnota_vlastnosti2;}
```

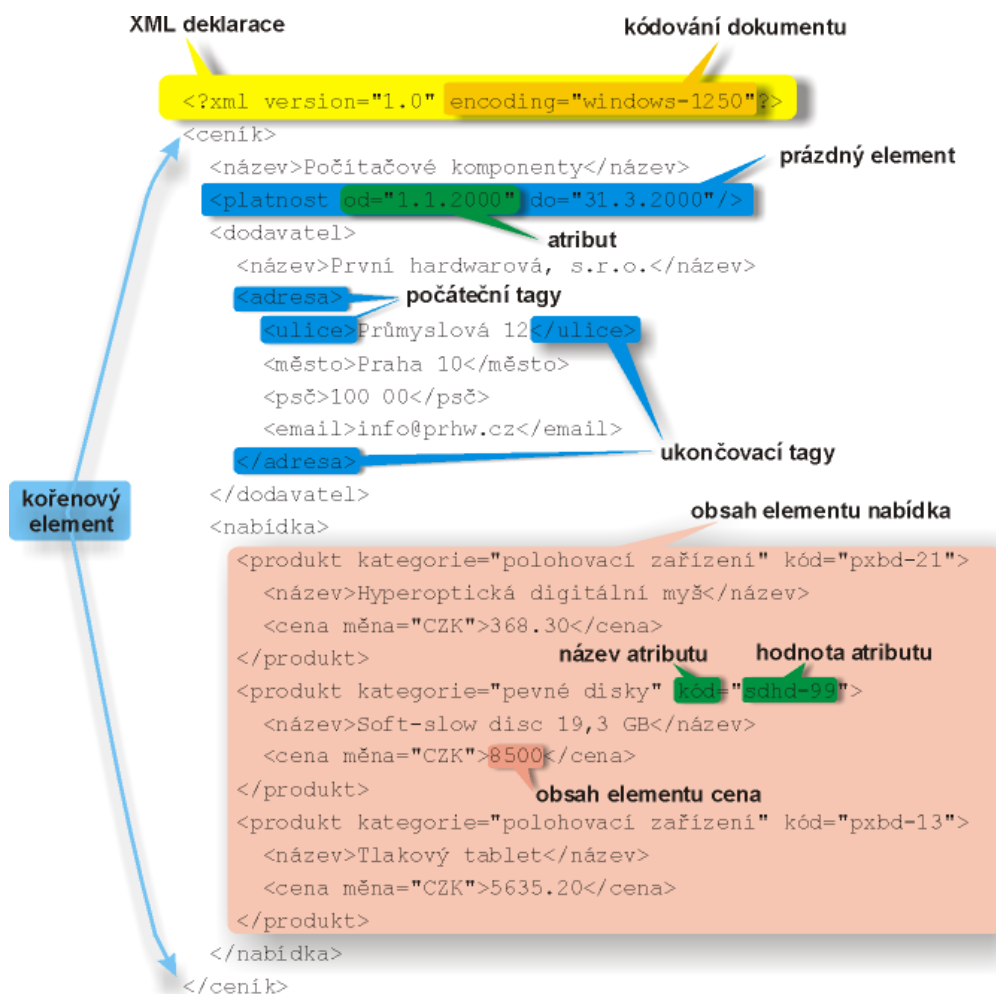
Konkrétní příklad:

```
table {background-color: lightblue; border: 1px; border-style: solid;}
```

Pokud chceme definovat styly v aktuálním dokumentu, stačí je zapsat mezi tagy <style> a </style>, nebo přímo do patřičného elementu pomocí zápisu "style=vlastnost: hodnota". Pokud máme externí soubor s CSS styly, můžeme ho definovat v hlavičce dokumentu, a to následovně: < link rel="stylesheet" type="text/css" href="styl.css" >.

3.4 XML

XML (eXtensible Markup Language) byl vyvinut a standardizován konsorciem W3C a převážně se používá k ukládání dat do XML dokumentů, které umožňují snadné předávání dat mezi aplikacemi. Svou syntaxí se podobá HTML, ale oproti němu umožňuje lépe specifikovat data. Jeho síla spočívá v definování vlastních elementů, což je v rámci manipulace s daty a jejich prezentování snadnější a přehlednější. DTD (definice typu dokumentu) umožňuje snadněji nalézt chyby v dokumentu a pomocí DTD můžou být definovány elementy a atributy, které se v dokumentu můžou používat. Strukturu xml dokumentu [12] můžeme vidět na Obr. 7.



Obr. 7. Struktura XML dokumentu

Poměrně snadné získávání dat z XML dokumentů umožňují tzv. parseři, proto se o nich taky zmíním.

3.4.1 XML parser

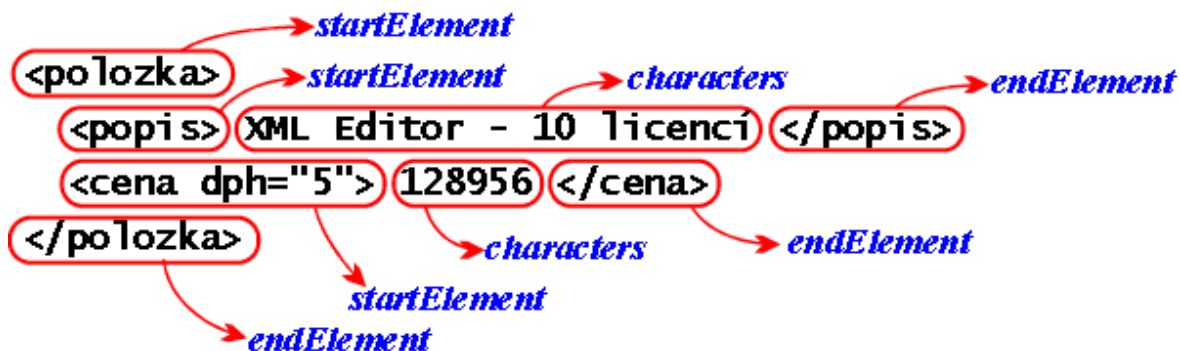
Pod pojmem parser se rozumí sada knihovních funkcí, které slouží pro získávání dat z XML dokumentů. Existují dva základní druhy parserů, resp. rozhraní:

1. událostmi řízené rozhraní
2. rozhraní se stromovou strukturou

Událostmi řízené rozhraní

Nejznámější událostmi řízené rozhraní je SAX (Simple API for XML). Je založeno na tom principu, že postupně čte XML dokument a při narazení např. na značky (`<`, `>`), textový

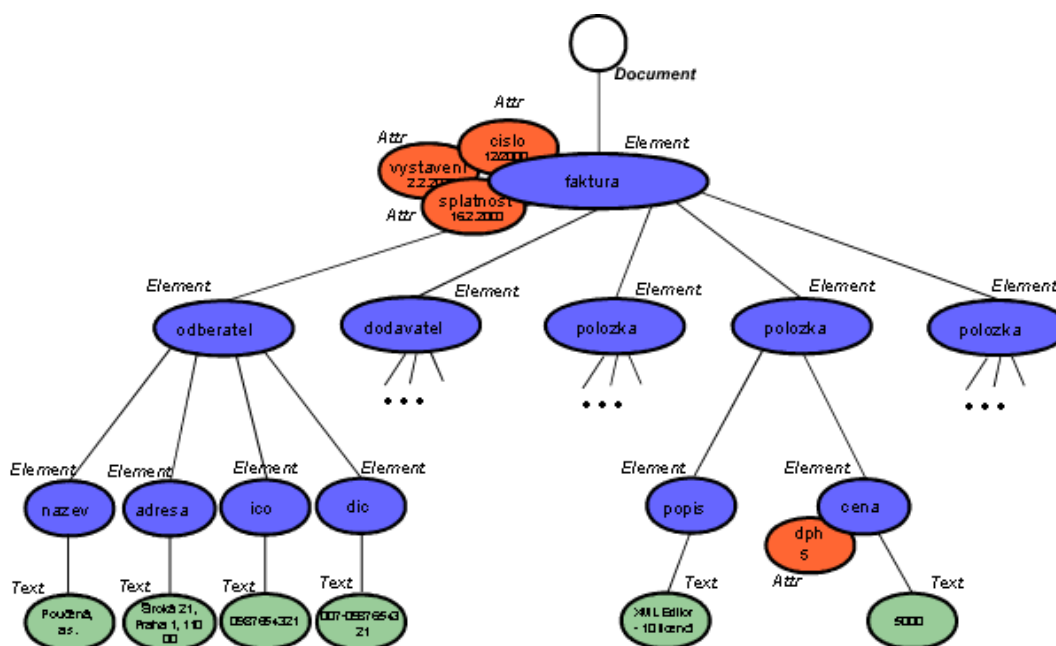
obsah elementů apod. volá obslužné funkce, které tyto data zpracují. Výhodou tohoto rozhraní je rychlost a má malé paměťové nároky. Nevýhodou je naopak nutnost zpracovat XML dokument během jednoho průchodu.



Obr. 8. Událostmi řízené rozhraní SAX

Rozhraní se stromovou reprezentací

Nejznámější rozhraní, které pracuje se stromovou reprezentací dokumentu, je DOM (Document Object Model). Toto rozhraní načítá celý dokument do paměťové struktury, ze které je možné přistupovat k jakémukoli prvku. Struktura je tvořena objekty, kde každý objekt znázorňuje jeden element. Výhodou této struktury je přístup k jakékoliv části dokumentu: Nevýhodou je, že má vysoké nároky na paměť (celý dokument se do ní musí uložit) a má nižší rychlost oproti rozhraní SAX.



Obr. 9. Stromová reprezentace rozhraním DOM

II. PRAKTICKÁ ČÁST

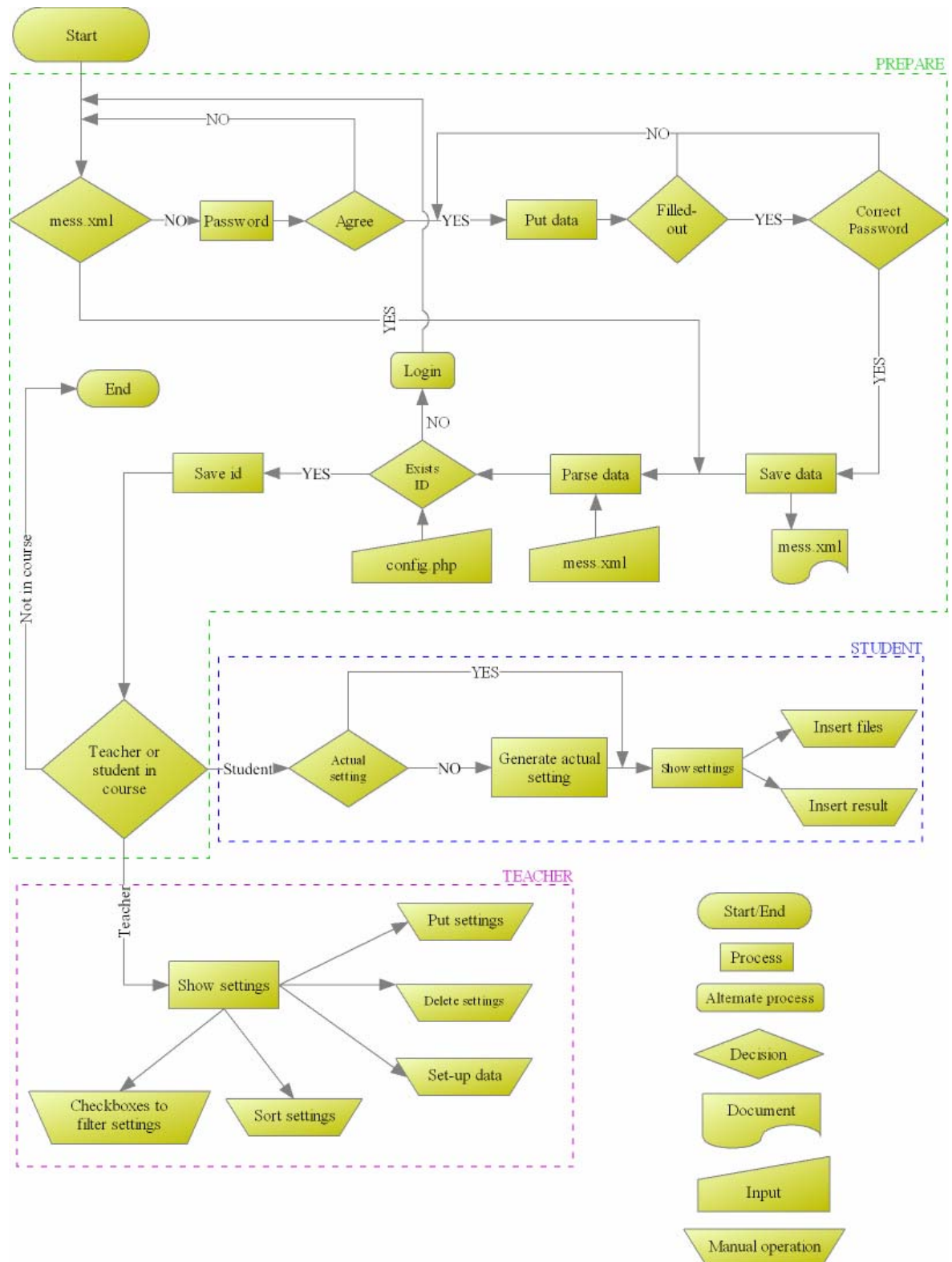
4 STRUKTURA WEBOVÉ APLIKACE

Nejdříve bych se chtěl zmínit o tom, co se vlastně od aplikace očekává. Aplikace má sloužit k usnadnění zadávání individuálních úkolů studentům a jejich správu. Představme si situaci, kdy učitel chce dát studentům úkol, ale každý student bude pracovat s jinými hodnotami. Učitel si připraví seznam zadání a na papírky napíše čísla zadání, ze kterých budou studenti losovat. Studenti tak chodí po jednom a losují čísla zadání a učitel si u každého studenta poznamená, jaké zadání si vylosoval.

Aplikace má „nahradit tuto metodu“ přidělování zadání studentům. Každý student, který je zapsán do tohoto předmětu si po spuštění aplikace náhodně vygeneruje zadání. Zadání se mu tak zobrazí, má možnost provádět správu tohoto zadání (vložit výsledek, popř. odevzdat soubory, které by byly součástí zadání) a vidět, kolik bodů za úkol dostal. Učitel po spuštění aplikace uvidí seznam studentů a u každého studenta přidělené zadání. Zobrazí se mu samozřejmě studentův výsledek (pokud byl vložen) a může si zobrazit odevzdané soubory (pokud byly nějaké odevzdány).

Aplikace je udělána jako externí a není tak zakomponována přímo do systému Moodle, ale pouze s ním komunikuje a získává tak potřebné informace o přihlášených studentech, popř. učitelích prostřednictvím jejich id (toto id je získáno přes skript config.php).

Celou aplikaci je možné pomyslně rozdělit do tří částí. První část by se dala nazvat jako příprava aplikace. V této části aplikace zjistí potřebné údaje (o přihlášeném uživateli, existenci dat pro správu aplikace, ...), ty vyhodnotí a na jejich základě se zachová dále. Druhou část tvoří prostředí pro studenta a třetí tvoří prostředí pro vyučujícího. Struktura může být vidět na Obr. 8.



Obr. 10. Struktura webové aplikace

5 POPIS JEDNOTLIVÝCH ČÁSTÍ

5.1 Instalace

Jelikož se jedná o externí aplikaci, adresář wapgeza (Webová Aplikace Pro GEnerování ZAdání) nakopírujeme do složky moodle/ a spuštění aplikace provedeme přímým odkazem na tento adresář, kde se spustí skript *index.php*.

5.2 Příprava aplikace

Pro správnou komunikaci s Moodle je potřeba znát některé informace, jako je např. název serveru kde se Moodle nachází, přihlašovací jméno a heslo ke školním databázím, atd. Aplikace tedy nejdříve zjistí, jestli jsou tyto informace dostupné ze souboru *mess.xml* (viz dále). Jestli tento soubor existuje, data jsou z něj parsována a uložena do proměnných SESSION, se kterými pak budou skripty pracovat. Po prvním spuštění aplikace však tyto informace ještě mít nebudeme (soubor *mess.xml* ještě neexistuje), takže tyto údaje budeme muset nejdříve zadat. Zobrazí se formulář s dotazem na heslo. Po vložení správného hesla nás aplikace pustí dál a to na formulář, do kterého se zadají následující údaje:

- * Název serveru – pro navázání spojení s databází pomocí SQL
- * Uživatelské jméno a heslo – pro přihlášení na server
- * Název databáze Moodle – kvůli SQL dotazům
- * Název tabulky – všech uživatelů, vyučujících, studentů, kurzů a tabulky pro uložení zadání
- * Název kurzu
- * Přihlašovací jméno vyučujícího

Po vložení údajů všech se ještě ověří uživatelské heslo, jestli bylo zadáno správně a pokud ano, data z formuláře se uloží do souboru *mess.xml* a do proměnných SESSION, se kterými budou skripty pracovat.

Vyplněte všechny údaje

Server:	
Uživatelské jméno:	
Uživatelské heslo:	
Uživatelské heslo znovu:	
Název databáze moodlu:	
Název tabulky všech uživatelů:	
Název tabulky vyučujících:	
Název tabulky studentů:	
Název tabulky kurzů:	
Název tabulky pro zadání:	
Název kurzu:	
Vaše přihlašovací jméno:	

Odeslat

Obr. 11. Formulář pro vložení údajů

Dále následuje ověření, jestli je uživatel přihlášený v Moodle. Ověřuje se vlastně jen id přihlášeného uživatele (které je získáno ze skriptu *config.php*, který se nachází v adresáři moodle), a to pomocí dat získaných z formuláře (nebo ze souboru *mess.xml*) následujícím způsobem:

1. Máme získané *id* přihlášeného uživatele. Vyhledá se uživatelské jméno v tabulce všech uživatelů s tímto *id* a následně se ověří, jestli souhlasí s uživatelským jménem vyučujícího (bylo zadáno do formuláře). Pokud ano, spustí se skript pro vyučujícího.
2. Pokud nesouhlasí, z tabulky kurzů se zjistí *id* kurzu (název kurzu byl zadán do formuláře).
3. Vyhledá se *id* uživatele z tabulky studentů, kde se uživatelské *id* (*userid*) rovná našemu získanému *id* a *id* kurzu (*course*) se rovná získanému *id* kurzu. Pokud takové *id* bylo nalezeno, jedná se v tomto případě o studenta daného kurzu a je spuštěn skript pro studenta.
4. Jestli není přihlášen vyučující ani student daného kurzu, je aplikace ukončena.

5.3 Prostředí pro studenta (skript student.php)

Když je přihlášen student daného kurzu, aplikace první ověří, jestli už má vygenerované zadání. Pokud ne, zobrazí se formulář pro vygenerování. Po kliknutí na formulář je studentovi vygenerováno náhodné zadání z databáze. Při této akci se taky vytvoří složka ve tvaru *příjmení_jméno* studenta (příjmení a jméno se dá zjistit pomocí *id* studenta z databáze), která slouží pro ukládání souborů, které student odevzdá.

Po této operaci se studentovi zobrazí vygenerované zadání v tabulce a na pravé části obrazovky menu. V tabulce tak vidí, jaké zadání mu bylo vygenerováno, znění zadání, svůj výsledek a ohodnocení vyučujícím.

V menu může student volit mezi následujícími možnostmi:

- * Vložit/upravit výsledek
- * Vložit soubor
- * Vymazat soubor

5.3.1 Vložit/upravit výsledek

Po kliknutí na tento odkaz se studentovi zobrazí formulář, kam napíše svůj výsledek. Pokud výsledek vložil dříve, ten se ve formuláři ukáže a student ho má možnost upravit. Po odeslání formuláře se výsledek uloží do databáze.

5.3.2 Vložit soubor

Při nahrávání (uploadu) souborů si nejdříve musíme uvědomit následující věci:

1. jak nahrát (uploadovat) soubor na server
2. jak uložit informace o odevzdaných souborech a jak je znovu získat

Uploadování souboru až takový problém není, stačí vytvořit formulář, kde nastavíme typ elementu `<input>` jako soubor a nějak tento element pojmenujeme. Příklad by mohl vypadat takto: `<input type="File" name="soubor1">`. Jako akci ve formuláři zvolíme soubor se skriptem pro studenta, v tomto případě *student.php* a metodu posílání dat jako Post (např. `<form`

action="student.php" Metod="Post">). Po odeslání formuláře získáme informace o vložených souborech funkcí `is_uploaded_file()`, která ověří, jestli byl nějaký soubor formulářem odeslán. V tomto případě chceme získat informaci o souboru *soubor1*, kterou nám funkce `is_uploaded_file()` poskytne. Soubor pak stačí jen zkopírovat funkcí `copy("soubor","cesta")` do předem vytvořené složky, o které jsem se zmínil při generování zadání.

Informace o uploadovaných souborech (jejich názvy) se ukládají do tabulky zadání v databázi, pod studenta, který je přihlášen. Názvy se ukládají jako řetězec v podobě *soubor1:soubor2:...:souborN*. Nejdříve se pomocí dotazu zjistí, jestli byly dříve nějaké soubory uploadovány. Jestli ano, do proměnné se načte řetězec se soubory, za který se přidají aktuální názvy souborů a pomocí dotazu `UPDATE` se tento nový řetězec uloží do databáze.

5.3.3 Vymazání souborů

Mazání souborů je založeno na podobném principu, jako upload. Po kliknutí na odkaz se pod každým souborem, který je v tabulce, zobrazí zatržítka (checkboxes). Ty slouží pro výběr souboru, který chceme smazat. Každý checkbox je pojmenován podle souborů, u kterých se nachází. Po vybrání stačí odeslat formulář, který spustí následující akce:

1. fyzicky odstraní soubory na serveru
2. odstraní informace o těchto souborech z databáze

Fyzické odstranění je otázka pár řádků. V první řadě musíme zjistit příjmení a jméno studenta; podle toho sestavíme cestu k souboru (víme, že název složky má tvar *příjmení_jméno*) a z odeslaného formuláře zjistíme pomocí metody `POST` názvy checkboxů, které jsou identické s názvy mazaných souborů. Příkazem `unlink("soubor")` se daný soubor ze složky smaže.

Odstranění informací se provede zavoláním dotazu pro získání řetězce se soubory. Řetězec se rozdělí na jednotlivé soubory pomocí funkce `for()`, které jsou uloženy do pole proměnných. Provede se porovnání, který soubor byl zatržen a ty pak jsou v následném sestavení řetězce s aktuálními soubory vynechány. Tento řetězec se opět uloží do databáze pomocí dotazu `UPDATE`.

5.4 Prostředí pro vyučujícího (skript `teacher.php`)

Po přihlášení vyučujícího se v tabulce zobrazí seznam studentů a u každého studenta jeho zadání. Nad tabulkou má vyučující panel ke třídění zadání a vpravo se zobrazí menu. Teď se podrobněji podívejme na tyto základní části.

5.4.1 Menu

Menu má jen tři položky, protože by měli vyučujícímu pro potřebné operace stačit. Jsou to:

- Vložit zadání
- Vymazat zadání
- Úprava dat

Vložit zadání

Tento odkaz umožňuje vyučujícímu vložit zadání z externího XML souboru dle dohodnutého DTD. Po kliknutí na odkaz se zobrazí formulář pro zadání souboru. Uploadování souboru se provede obdobně, jako v předešlé části vkládání souborů u studenta. Tzn., po vybrání souboru a odeslání formuláře se funkcí `is_uploaded_file()` zkontroluje, jestli byl odeslán z formuláře nějaký soubor, pokud ano, tak se funkcí `copy()` tento soubor zkopíruje do složky `wageza`. Po úspěšném zkopírování je zavolán skript *parser.php*, kde se tyto data rozparují a uloží do databáze (do tabulky, jejíž název se získal ze souboru *mess.xml*).

Vymazat zadání

Zde vyučující zadá, které zadání se má smazat (Pokud bylo zadáno více zadání, např. dva úkoly vyžadují dvě série zadání (2x60) – vybere se např. druhá série) a dané zadání se smažou.

Upravit data

Tento odkaz slouží pouze pro úpravu dat, které se zadávali při prvotním spuštění (uloženy v *mess.xml*).

5.4.2 Panel pro výběr a třídění zadání

Tento panel (formulář) slouží pro výběr a třídění zadání. Dá se říci, že je rozdělen na dvě části. Jedna část obsahuje checkboxy pro výběr zadání (volné a obsazené zadání) a druhá tři volitelné položky (radio) pro třídění zadání (podle čísla zadání, příjmení a data).

Výběr zadání:		Třídít podle:		
Obsazené zadání ☒	Volné zadání ☐	Číslo zadání ☒	Příjmení ☐	Data ☐

Obr. 12. Panel pro výběr a třídění zadání

Pro rychlou realizaci je zde použit JavaScript, konkrétně se testuje událost onclick(). Po kliknutí na příslušný prvek se zavolá objekt form.submit(), přičemž se odešle formulář a stránka se aktualizuje. Podle toho, na který prvek se kliklo se zformuje dotaz, který pak vybere nebo setřídí požadovaná data.

5.4.3 Tabulka

Do tabulky se zobrazí vybraná zadání, resp. studenti s konkrétními zadáními. U každého studenta tedy vyučující vidí: číslo zadání, znění, studentův výsledek, správný výsledek, body za jednotlivé zadání a může také zobrazit u daného studenta odevzdané soubory a samozřejmě může práci obodovat. Soubory se zobrazí po kliknutí na odkaz obdobným stylem, jako u studenta. Pokud bylo zadáno více úkolů (zadání), jsou tak u studentů všechna zobrazena.

5.5 Stručný popis ostatních skriptů

index.php

Tento skript slouží pro zjištění některých údajů, zejména jestli jsou dostupné základní informace v souboru *mess.xml*. Dále provádí na základě dostupných informací (*id*) rozhodnutí, jaké prostředí se spustí (student – vyučující).

options.php

Slouží pouze pro zadání, popř. úpravu údajů, které budou jednotlivé skripty potřebovat.

parser.php

Jak už název napovídá, zde se provádí načítání dat do proměnných ze souborů *mess.xml* a zadání a umožňuje . Na tento soubor se odkazují následující skripty:

- *index.php* – odkáže se v případě, že existuje soubor *mess.xml*, aby uložil data z tohoto souboru do proměnných SESSION.
- *options.php* – po odeslání formuláře se data uloží do souboru *mess.xml*.
- *teacher.php* – pokud vyučující vloží soubor se zadáními, data z tohoto se uloží do databáze.

connection.php

V tomto skriptu je pouze uloženo spojení s databází do proměnné `$_SESSION['connection']` a tato proměnná je volána v dalších skriptech.

save.php

Slouží k uložení *id* přihlášeného uživatele, které je zjištěno ze skriptu *config.php*. *Id* se uloží do souboru *save.txt*. Skript se odkáže na *index.php*, ve kterém se *id* načte ze souboru, uloží se do proměnné SESSION a soubor se smaže.

ZÁVĚR

Každý, kdo dělá na trochu větším projektu (pojem „větší projekt“ každý z nás chápeme trochu jinak, ale pro mě tohle byl větší projekt) by si měl něco z té práce odnést. Proto i já nebudu výjimkou. Vyzkoušel jsem si naprogramovat externí aplikaci, která se stará o správu zadání a komunikuje s výukovým systémem Moodle a po stránce možností jsem ji udělal tak, aby byla jednoduchá a přehledná. Co se týká bezpečnosti a ošetření, tak není extrémně ošetřena proti případným „hackerským“ útokům, z toho důvodu proto ani nebyla vyzkoušena na školním serveru. Pokud bych se teď měl znovu rozhodnout, jak celý úkol provést, určitě bych nezvolil externí aplikaci, ale zabudoval bych ji přímo do systému Moodle. Důvod je docela jednoduchý – na vývoji Moodle, jak bylo zmíněno, se podílela celá řada vývojářů, proto po stránce bezpečnosti a komunikace by měl být naprogramován tak, aby splňoval tyto požadavky. Navíc, v systému Moodle jsou už vytvořeny některé moduly, které by stačilo jenom upravit, proto bych ušetřil několik (desítek) hodin programování. Proto je tato práce spíše mým poučením, že ne všechno, co se na první pohled zdá být složité (programování v rámci existujícího projektu) je opravdu tak složité. Mnohdy se ukáže pravý opak. Tato práce by proto mohla být námětem na další téma.

SEZNAM POUŽITÉ LITERATURY

Internetové zdroje:

- [1] Databázový svět [online]. [cit. 2006-06-11]. Dostupný z WWW: <<http://www.dbsvet.cz>>
- [2] Databázové systémy [online]. [cit. 2006-06-11]. Dostupné z WWW: <<http://www.umel.feec.vutbr.cz/~danekl/Archiv.PCK/!!!2001!!!/0115/index.html#uvod>>
- [3] Database Systems: The Complete Book [online]. [cit. 2006-06-11]. Dostupný z WWW: <<http://www-db.stanford.edu/~ullman/dscb/ch1.pdf>>
- [4] Databázové systémy [online]. [cit. 2006-06-11]. Dostupný z WWW: <<http://homen.vsb.cz/~sen76/db/db.html>>
- [5] Interval.cz – webdesign a e-komerce denně [online]. [cit. 2006-06-11]. Dostupný z WWW: <<http://interval.cz>>
- [6] Živě.cz – o počítačích a internetu [online]. [cit. 2006-06-11]. Dostupný z WWW: <<http://www.zive.cz>>
- [7] Databázové a informačné systémy [online]. [cit. 2006-06-11]. Dostupné z WWW: <<http://www.ipower.sk/dbs/etapy.html>>
- [8] Databázové systémy [online]. [cit. 2006-06-11]. Dostupné z WWW: <<http://www.fs.vsb.cz/books/dbacc20/dbacc01.htm#dbacc021>>
- [9] Database design – DDL & DML [online]. [cit. 2006-06-11]. Dostupné z WWW: <<http://www.tomjewett.com/dbdesign/dbdesign.php?page=ddlddl.php>>
- [10] Zend Technologies [online]. [cit. 2006-06-11]. Dostupné z WWW: <<http://www.zend.com/zend/art/intro.php#Heading4>>
- [11] Fakulta informačních technologií [online]. [cit. 2006-06-11]. Dostupné z WWW: <http://www.fit.vutbr.cz/study/courses/DSI/public/pdf/nove/1_uvod.pdf>
- [12] “Vše o WWW“ [online]. [cit. 2006-06-11]. Dostupné z WWW: <<http://www.kosek.cz>>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

OS	Operační Systém
SŘBD	Systém Řízení Báze Dat
IBM	Význam třetí zkratky.
CODASYL	Conference on Data Systems Language
SQL	Structured Query Language
DDL	Data Definition Language – jazyk pro definici dat
DML	Data Manipulation Language – jazyk pro manipulaci s daty
HTML	Hypertext Markup Language
PHP	PHP: Hypertext Preprocessor
CSS	
XML	eXtensible Markup Language
DTD	Definice Typu Dokumentu
SAX	Simple Api for XML
DOM	Dokument Object Model

SEZNAM OBRÁZKŮ

Obr. 1. <i>Systém pro ovládání souborů</i>	10
Obr. 2. <i>Systém řízení báze dat</i>	11
Obr. 3. <i>Hierarchický model</i>	12
Obr. 4. <i>Síťový model</i>	12
Obr. 5. <i>Relační model</i>	13
Obr. 6. <i>Struktura databázového systému</i>	13
Obr. 7. <i>Struktura XML dokumentu</i>	18
Obr. 8. <i>Událostmi řízené rozhraní SAX</i>	19
Obr. 9. <i>Stromová reprezentace rozhraním DOM</i>	19
Obr. 10. <i>Struktura webové aplikace</i>	22
Obr. 11. <i>Formulář pro vložení údajů</i>	24
Obr. 12. <i>Panel pro výběr a třídění zadání</i>	28