České vysoké učení technické v Praze

Fakulta elektrotechnická

Bakalářská práce

Databázový systém pro hodnocení studentů

Jakub Trmota

Vedoucí práce: Ing. Jiří Roubal, Ph.D.

Studijní program: Kybernetika a měření

srpen 2007

České vysoké učení technické v Praze, fakulta elektrotechnická

Katedra řídicí techniky | Školní rok: 2006/2007

Zadání bakalářské práce

Student: Jakub T r m o t a

Obor: Kybernetika a měření

Název tématu: Databázový systém pro hodnocení studentů

Zásady pro vypracování:

1. Navrhněte strukturu databáze pro registraci a hodnocení studentů.
2. Naprogramujte webové formuláře pro uživatele: student, učitel a administrátor.
3. Napište k tomuto systému manuál.

Seznam odborné literatury:

- Jiří Kosek, *PHP – tvorba interaktivních internetových aplikací*, Grada publishing s. r. o., 1999
- Welling Luke, Laura Thomson, *PHP a MySQL - rozvoj webových aplikací*, 2003

Vedoucí bakalářské práce: Ing. Jiří Roubal, Ph.D.

Datum zadání bakalářské práce: zimní semestr 2006/07 (změna zadání květen 2007)

Termín odevzdání bakalářské práce: srpen 2007

Prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

Doc. ing. Boris Šimák, CSc.
děkan

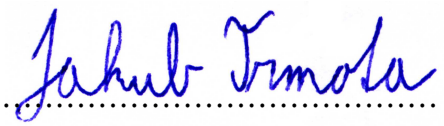
V Praze, dne 27. 7. 2007

Poděkování

Zde bych rád poděkoval všem lidem, kteří mi připravili ideální podmínky pro psaní této práce, hlavně mé mamince a přítelkyni. Rád bych poděkoval ing. Halaškovi za poskytnutí vzorového exportu dat, bez kterého bych aplikaci nemohl psát a pak svému vedoucímu, za trpělivost a ochotu, se kterou mi pomáhal řešit nástrahy, které mě před započetím psaní této práce provázeli.

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 24.8.2007 

podpis

Abstract

This work is about creating database system for classification students. Describes common object framework, all methods and properties. Also is here list of used modules and short manual for aplication.

Abstrakt

Tato práce pojednává tvorbě databázového systému pro hodnocení studentů. Popisuje obecný aplikační objektový framework, všechny jeho objekty, metody a vlastnosti. Dále je přiložen výpis použitých modulů a nakonec jednoduchý manuál s ovládáním aplikace.

Obsah

1. Úvod

Mým úkolem je přepracování aplikace OTA1 sloužící k hodnocení studentů. Původní systém, vyžadující ruční registraci všech studentů, je nahrazen automatickým importem dat z databáze KOS. Aplikaci je možné použít pro všechny předměty. Dříve bylo třeba pro každý předmět vytvořit vlastní kopii aplikace.

1.1 Zásady pro vypracování

Zadání:

1) Navrhněte strukturu databáze pro registraci a hodnocení studentů.
2) Naprogramujte webové formuláře pro uživatele: student, učitel a administrátor.
3) Napište k tomuto systému manuál.

Součástí zadání je i automatický import dat z databáze KOS z XML souboru, který obsahuje informace o studentech, učitelích a rozvrhu. Po domluvě není možné do dat ručně zasahovat. To znamená, že co není v KOSu, není ani v aplikaci a není to ani možné do aplikace dodatečně přidat. Kontrola a import dat z KOSu probíhá každý den.

1.2 Štábní kultura práce a aplikace

Při tvorbě aplikace jsem se snažil pojmenovávat všechny třídy, funkce a proměnné anglickými výrazy, na rozdíl od návrhu databáze, kde používám výrazy české.

V této práci budu všechny ukázky kódu, názvy funkcí, tříd, proměnných a souborů psát `tímto neproporciálním fontem`. Všechny názvy tabulek a sloupců budou navíc psány *`kurzívou`*.

2. Popis problému

Největším problémem při plánování aplikace se ukázalo získání zdrojových dat z databáze KOS. Nejprve jsem počítal s tím, že aplikace bude mít přímo přístup do databáze KOS. Toto řešení se ukázalo jako zbytečně složité pro naše potřeby. Nakonec jsem po konzultaci s inženýrem Halaškou došel k závěru, že pro tuto aplikaci využiji export dat z XML souboru, který vytvořil Radek Jun ve své diplomové práci *Použití XML a XSLT při tvorbě statických stránek nad KOS*. Vstupem aplikace je tedy jediný XML soubor, který obsahuje všechny potřebné informace a studentech, učitelích a rozvrhu. Tyto data by měla být kvůli aktuálnosti do aplikace importována nejlépe každý den.

Následně již bylo třeba pouze podle zmíněného XML souboru navrhnout databázi a vytvořit formuláře, pro její ovládání. Aby aplikace byla co nejvíce robustní a zároveň modulární, rozhodl jsem se pro její vytvoření naprogramovat obecný aplikační framework s možností používání zásuvných modulů pro řešení konkrétních situací. Díky tomu by aplikace měla být jednoduše rozšiřitelná a to kýmkoliv, kdo dokáže alespoň základně programovat v PHP.

Součástí aplikace jsou moduly důležité pro ovládání aplikace a pak modul pro hodnocení studentů.

3. Řešení aplikace

Celá aplikace je napsána v jazyce PHP ve verzi 4 a jako databáze byla vybrána MySQL ve verzi 4.1 a to především s ohledem na rychlost při práci nad několik sty tisíci řádky. Zobrazovaná data splňují normu XHTML 1.1 STRICT a jsou formátována pomocí CSS2. V celé aplikaci není použit ani jeden obrázek a je použit layout, který by měl aplikaci zpřístupnit i pro tělesně postižené uživatele.

Bohužel jazyk PHP 4 ještě neumožňuje v nastavení tříd definovat veřejné/privátní vlastnosti a metody, všechny jsou veřejné. V popisu aplikace tedy s tímto počítám, a pokud budu o nějaké vlastnosti či metodě psát, že je pouze privátní, znamená to, že je taková z pohledu samotné aplikace, ale ve skutečnosti je veřejná.

Celá aplikace a databáze je uložena v kódování UTF-8 hlavně s ohledem na budoucnost. Všechny vstupy v jiném kódování jsou automaticky převedena na UTF-8.

3.1 Struktura aplikace

Aplikace se skládá z několika částí. První z nich je obecný aplikační framework. Dále použité moduly, definice layoutu s příslušným kaskádovým stylem, externí soubor s javascriptovými funkcemi, adresář s nápovědou, dočasný adresář pro stažení dat pro import dat, adresář pro upload uživatelských dat, adresář s logem aplikace obsahující chybové hlášení a v neposlední řadě adresář se soubory určenými k pravidelnému spouštění pomocí cronu.

3.1.a Podrobná struktura aplikace

`/common`

Obsahuje soubory s aplikačním frameworkem.

`/cron`

Obsahuje soubory pro pravidelné spuštění.

- `import.php` - slouží k importu dat z KOSu (doporučeno spouštět při nejmenší zátěži serveru, například o půlnoci).

/css

Obsahuje soubory s kaskádovými styly.

- `layout.css` – hlavní soubor s definicí kaskádových stylů.
- `print.css` – soubor s definicí stylů pro tisk.

/help

Obsahuje soubory s nápovědou

- `napoveda_student.pdf` – soubor s nápovědou pro studenty.
- `napoveda_administrator.pdf` – soubor s nápovědou pro administrátora aplikace.
- `napoveda_cvicici.pdf` – soubor s nápovědou pro cvičící.
- `napoveda_prednasejici_administratorpredmetu.pdf` – soubor s nápovědou pro přednášející a administrátory předmětu.

/js

Extetní soubory s javascriptovým kódem.

- `main.js` – obsahuje javascriptové funkce použité v aplikaci.

/layout

Soubory s definicí layoutu.

- `layout.html` – obsahuje použitý layout ve formátu XHTML.

/log

Adresář s logem aplikace, do kterého se zapisují chyby, SQL dotazy apod.

- `log.txt` – obsahuje log.

/modules

Všechny moduly pro použití v aplikace. Více samostatná kapitola o modulech.

/temp

Adresář pro dočasné soubory. Zde použito pro import dat.

`/upload`

Slouží pro ukládání uživatelských souborů nahraných na serveru.

`/ [kořenový adresář]`

Obsahuje soubory, starající se o prezentaci dat uživateli.

- `index.php` – defaultní soubor při načtení aplikace a zobrazení úvodu.
- `help.php` – obsahuje odkazy na nápovědu pro studenty.
- `password_change.php` – po přihlášení umožňuje uživateli změnit si heslo.
- `password_regenerate.php` – umožňuje uživateli vygenerovat si nové heslo, když zapomene původní (z bezpečnostních důvodů si administrátor aplikace nemůže vygenerovat nové heslo).
- `service.php` – stará se o zobrazování předmětů a k nim přiřazeným modulům.
- `logout.php` – bezpečně odhlásí uživatele ze systému.

3.2 Import dat

Data se do aplikace importují každý den[1] z XML souboru, který obsahuje informace o všech předmětech, studentech a učitelích. O import se stará soubor `/cron/import.php`, kterému se z bezpečnostních důvodů musí předat parametr `password`, který je zadán v nastavení aplikace a musí se skriptu předat formulářovou metodou GET[2].

Při importu je nejprve stažen soubor ze zaheslované stránky[3]. Jedná se o komprimovaný soubor, který je následně rozbalen, načten a importován. Import probíhá tak, že na začátku jsou všechny řádky, které již v tabulce existují, označeny, že jsou právě importovány. Pokud je importován řádek, který již v databázi existuje, upraví se podle nových dat a nastaví se jako naimportovaný. Řádek, který v databázi ještě neexistuje, se přidá a také se nastaví jako naimportovaný. Na konci importu se

[1] Záleží na nastavení, každý den je doporučeno.
[2] /cron/import.php?password=nejake_heslo
[3] V době psaní aplikace nebyl zřízen přístup na adresu https://www.feld.cvut.cz/education/rozvrhy-export/rz.xml.zip, k dispozici jsem měl jeden ukázkový export a zaheslovaný přístup jsem simuloval na lokálním serveru Apache 2.2.

z databáze vymažou všechny řádky, které nejsou označeny jako „již naimportované“. Toto řešení jsem použil, aby se při neočekávaném ukončení importu neztrácela data.

Na konci importu se všem novým studentům a učitelům vytvoří uživatelský účet a vygeneruje se nové heslo, které se odešle na jejich školní emailovou adresu.

K tomu jedna poznámka: v budoucnu se počítá s ověřováním uživatelů přes centrální server, poté se tabulka s uživatelskými účty zruší, proto zde není vyřešen problém s hromaděním uživatelských účtů studentů, kteří již ukončili studium.

Při prvním importu dat se také vytvoří uživatelský účet pro administrátora aplikace a odešle se mu heslo na email, zadaný v nastavení aplikace. Uživatelské jméno pro administrátora aplikace je `administrator`.

Upozorňuji, že celý import dat je velice časově náročný a z tohoto důvodu je doba běhu skriptu pro import nastavena na jednu hodinu.

3.3 Databáze

Definice tabulek jsou v příloze. Zde jen krátce o struktuře názvů tabulek a sloupců. Tabulky a její sloupečky jsem na rozdíl od aplikace pojmenovával všechny pouze českými názvy a to z důvodu rychlejší orientace při práci s daty přímo v databázi, nikoliv přes aplikaci.

V aplikaci jsou tři základní typy tabulek:

- **Tabulky s rozvrhem** – jejich jména začínají písmenem *r*
- **Tabulky pro přihlášení a práva k předmětům** – jejich jména začínají na písmeno *p*
- **Tabulky pro moduly** (ne každý modul potřebuje tabulku) – jejich jména začínají na písmeno *m*

Každá tabulka začíná příslušným písmenem (viz výše) a pořadovým číslem. Od toho se pak následně odráží název sloupečku s primárním klíčem. Příklad pro tabulku se semestry. Její název je:

```
r01_semestry
```

a její sloupeček s klíčem je *id_r01*. To mi zvláště při použití cizích klíčů v jiné tabulce velice zlehčuje orientaci.

Tabulky pro moduly jsou zvláštní tím, že pokud vyžaduje modul více tabulek, přidává se ještě jedno pořadové číselné označení. Pokud má tedy modul dvě tabulky, budou jejich názvy takovéto: *m01_01_tabulka1* a *m01_02_tabulka2* a jejich klíče potom *id_m01_01* a *id_m01_02*. Tím je na první pohled jasné, že tabulky náleží jednomu modulu.

Všechny tabulky musí mít sloupec s kódem semestru. Pokud tabulka nepotřebuje nastavení kódu semestru, musí být zařazena do výjimky u modulu pro export a výmaz semestrů (tak jako je teď tabulka *p01_prihlaseni*) a také u importu předchozích semestrů v nastavení modulů. Zároveň tabulky k modulům s daty, které nebudete chtít importovat do dalších semestrů (typicky data s výsledky studentů) musí mít sufix *_data* ve svém názvu. Při importu nastavení do dalších semestrů budou takovéto tabulky ignorovány.

3.4 Aplikační framework

Aplikaci jsem se od začátku snažil psát co nejobecněji, hlavně kvůli jednoduché úpravě a snadnému rozšíření. Z tohoto důvodu vznikl objektový aplikační framework, sdružující všechny obecné funkce, které jsou využívány pro celou aplikaci s možností nahrání modulů, které by měly řešit konkrétní situace. Každý modul může jednoduše přistupovat ke všem objektům frameworku a stejně tak jednotlivé objekty mohou spolupracovat či dokonce ke své správné činnosti vyžadovat jiný objekt.

Celý framework a potažmo i celá aplikace se ovládají pomocí jediného konfiguračního souboru, který je schopen předat či nastavit vlastnosti všech závislých objektů.

O inicializaci frameworku na každé stránce se stará třída nazvaná `common()`,která je umístěna v souboru `common.init.php` Pomocí ní lze na každé stránce vytvořit objekt s frameworkem, který si automaticky naimportuje nastavení a vytvoří všechny závislé moduly.

3.4.a Obecné objekty použité v aplikaci:

- `db (common.db.php)` – poskytuje připojení k databázi a obsahuje funkce pro volání SQL dotazů a jednoduchému vkládání, úpravě a mazání záznamů z databáze.
- `import (common.import.php)` – používá se pouze k importování dat z KOSu ve formě XML souboru do databáze.
- `login (common.login.php)` – vytváří a spravuje přihlášení do aplikace.
- `layout (common.layout.php)` – celá aplikace je napsána tak, že odděluje aplikační vrstvu od vrstvy prezentační. Všechny funkce (až na nutné výjimky, a ty vracejí obecný XHTML kód), vracejí pouze data a samotné formátování je přidáno až při jejich volání. O konečnou prezentaci výsledků do validní XHTML podoby se stará tento objekt.
- `modules (common.modules.php)` – objekt sloužící k načítání předmětů a jím příslušejícím modulům. Dále slouží k výběru studentů a cvičících pro konkrétní předmět.

Níže následuje popis všech objektů a možných nastavení. Vlastnosti a metody, které jsou využitelné pouze pro vnitřní chod objektu, se v popisu nevyskytují, pokud je nepovažuji důležité k pochopení funkce objektu.

3.4.b Nastavení pomocí settings

Provádí se v souboru `/common/common.settings.php`. Pomocí tohoto souboru můžete ovlivňovat nastavení celé aplikace. Nejedná se o zvláštní objekt, ale o prostý výčet direktiv pro nastavení. Do souboru se zapisují následujícím způsobem:

```
$this->settings['objekt'][ 'nastaveni1'] = hodnota;
```

Jako `objekt` se musí použít název objektu, do kterého se má nastavení naimportovat, `nastaveni1` je pak název proměnné, která se v objektu vytvoří. Nastavení se pak do objektu vloží jako jeho další vlastnost a to takto:

```
$objekt->nastaveni1 = hodnota;
```

Následuje popis nastavení pro všechny objekty. Všechny hodnoty jsou typu string, není-li uvedeno jinak.

Nastavení pro common

- `$this->settings['common']['applicationName']` – název aplikace.
- `$this->settings['common']['applicationUrl']` – kompletní url, z kterého je aplikace přístupná.
- `$this->settings['common']['path']['root']` – absolutní cesta k aplikaci.
- `$this->settings['common']['log']['file']` – soubor, do kterého bude zapisován log aplikace.
- `$this->settings['common']['log']['message']['errorOpen']` – chybová zpráva, která se zobrazí, pokud se log nepodaří otevřít.
- `$this->settings['common']['administrator']['email']` – emailová adresa administrátora aplikace.
- `$this->settings['common']['email']['address']` – sufix pro emailové adresy uživatelů, zde @fel.cvut.cz.
- `$this->settings['common']['email']['headers']` – hlavička odesílaným emailů.

Profily

Při vytváření hlavního objektu aplikačního frameworku, je potřeba zvolit profil, ve kterém jsou definovány objekty, které se mají pro konkrétní stránku načíst a jaké se mají před odesláním výstupu odeslat hlavičky.

Profily se definují následovně:

```
$this->settings['common']['profiles']
[‘nazev_profilu’]['class'] = array('objekt1', 'objekt2');
```

a

```
$this->settings['common']['profiles']
['web']['header'] = array('hlavicka1', 'hlavicka2');
```

Aplikace obsahuje dva profily. Jeden pro web a jeden pro import. Profil web obsahuje následující objekty: `layout, db, login, modules`. Profil import pak inicializuje následující objekty: `import, db, login`.

Nastavení pro `import`

- `$this->settings['import']['password']` – heslo, které se musí předat parametrem `password` při volání souboru `import.php` zaheslováno pomocí `MD5`.
- `$this->settings['import']['download']['username']` – uživatelské jméno pro přístup k souboru s exportem dat z KOSu
- `$this->settings['import']['download']['password']` – viz výše, pouze se jedná o heslo.
- `$this->settings['import']['download']['url']` – url adresa s exportem dat z KOSu[4].
- `$this->settings['import']['download']['zip']` – adresář, kam se má stáhnout komprimovaný soubor s exportem dat z KOSu.
- `$this->settings['import']['download']['extractTo']` – adresář, určující kam se má soubor extrahovat.
- `$this->settings['import']['download']['xml']` – celá cesta k extrahovanému souboru XML.
- `$this->settings['import']['departments']` – array, pole s čísly kateder, které se mají naimportovat do databáze a jím příslušné předměty.
- `$this->settings['import']['email']['subject']` – předmět emailu s novými hesly.
- `$this->settings['import']['email']['text']` – text emailu s novými hesly. `%login%` se nahradí za uložené přihlašovací jméno a `%password%` za vygenerované heslo.

[4] V době psaní práce https://www.feld.cvut.cz/education/rozvrhy-export/rz.xml.zip

Nastavení pro db

- `$this->settings['db']['server']` – adresa databázového serveru.
- `$this->settings['db']['username']` – uživatelské jméno pro připojení k databázovému serveru.
- `$this->settings['db']['password']` – heslo pro připojení k databázovému serveru.
- `$this->settings['db']['database']` – databáze s daty aplikace.
- `$this->settings['db']['charset']` – kódování, ve kterém jsou data uložena.
- `$this->settings['db']['showError']` – boolean, zobrazovat ve výpisu chyby.
- `$this->settings['db']['showErrorQuery']` – boolean, zobrazit SQL dotaz, který chybu způsobil.
- `$this->settings['db']['logError']` – boolean, zapisovat chyby do logu.
- `$this->settings['db']['logSql']` – boolean, zapisovat do logu všechny SQL dotazy[5].
- `$this->settings['db']['message']['errorConnect']` – chybová hláška, která se zobrazí, pokud se aplikace nedokáže připojit k databázi.

Nastavení pro `login`

- `$this->settings['login']['time']` – integer, počet sekund nečinosti, po kterém dojde k automatickému odhlášení.

Nastavení pro `modules`

- `$this->settings['modules']['modules']['administrator']` – array, pole s moduly, určenými administrátorovi aplikace.

[5] Vhodné pro testování, ale zpomaluje aplikaci a velikost logu se rychle rozrůstá, doporučeno nastavit na false

- `$this->settings['modules']['modules']['common']` – array, pole s moduly společnými pro všechny předměty.
- `$this->settings['modules']['modules']['all']` – array, pole s moduly, které lze přiřadit předmětům.
- `$this->settings['modules']['modules']['description']` – array, asociativní pole s popisem modulů, které lze přiřadit předmětům. Jako klíč v poli je použit název modulu, jako hodnota popis modulu.

Nastavení pro `layout`

- `$this->settings['layout']['fileName']` – soubor s definicí layoutu.
- `$this->settings['layout']['footer']` – patička všech stránek.

3.4.c Objekt `common`

Jedná se o hlavní objekt, jehož vytvořením se inicializuje celý aplikační framework. Framework se inicializuje následovně:

```
require_once('./common/common.init.php');


$common = new common([profile], [needLogin]);
```

Do funkce `require_once` je nutné zadat cestu k souboru `common.init.php`, který obsahuje definici s objektem. Pomocí konstrukce `new` se následně objekt vytvoří. Při vytváření objektu můžete zadat dva nepovinné parametry, *`profile`* označuje profil s nastavením objektů k inicializaci a s hlavičkami určenými k odeslání před vlastním výstupem. Pokud nezadáte, použije se defaultně „web“. Parametr *`needLogin`* nabývá hodnoty true nebo false. Pokud je true, je nutné být při přístupu na stránku přihlášen[6], pokud je false, přihlášeni být nemusíte. Pokud parametr vynecháte, nastaví se false. Chcete-li nastavit parametr *`needLogin`*, nemůžete vynechat parametr *`profile`*.

[6] Je nutné v profilu použít objekt login

Vlastnosti

Kromě vlastností, naimportovaných do objektu z nastavení, obsahuje objekt tyto vlastnosti:

- `pathCommon` – obsahuje absolutní cestu k aplikačnímu frameworku.
- `pathFile` – obsahuje absolutní cestu k souboru, v kterém je aplikační framework inicializován.
- `pathWeb` – obsahuje relativní cestu z kořenového adresáře aplikace k aktuálnímu souboru.
- `fileName` – obsahuje název souboru, z kterého je framework inicializován.

Metody

Při vytvoření objektu se zavolá konstruktor (`common()`), kterému se předají příslušné parametry *`profile`* a *`needLogin`*. Následuje:

1. nastavení proměnné `pathCommon`
2. nastavení proměnné `pathFile`
3. nastavení proměnné `fileName`
4. načtení souboru s nastavením
5. odesláním hlaviček podle nastaveného profilu
6. import nastavení pro `common`
7. nastavení proměnné `pathWeb`
8. inicializace objektů podle nastaveného profilu

Inicializace objektů

Každý objekt musí být umístěn ve vlastním souboru a v adresáři `/common`. Soubor musí mýt tvar `common.`*`objekt`*`.php`, kde *`objekt`* je název objektu uvedeného v profilu v nastavení a musí se shodovat s názvem třídy umístěné v souboru.

Každý objekt může obsahovat dvě aplikační metody, `__run()`, která se volá při inicializaci objektu a `__destroy()`, která se volá, při uvolňování objektu. Typické použití je například při připojení a odpojení databáze. Následně je do

objektu vložena vlastnost `__common`, která obsahuje odkaz na inicializační objekt, přes který je možné se dostat k ostatním objektům.

Následuje příklad přístupu k objektům:

V souboru, kde je inicializován framework pomocí `$common = new common('web', false)` se k objektu `db` dostanu takto:

```
$common->db->query('…');
```

Z objektu `login` se do objektu `db` dostanete takto:

```
$this->__common->db->query('…');
```

Metoda `destroy()`

Je doporučeno ji volat na konci každého skriptu. Zavolá metody `__destroy()` všech inicializovaných objektů a ukončí aplikaci.

Metoda `loadCommonSettings()`

Načte nastavení frameworku.

Metoda `loadCommonClass(`*`class`*`)`

Načte soubor s objektem *`class`* a zavolá `makeClass(`*`class`*`)`.

Metoda `makeClass(`*`class`*`)`

Vytvoří nový objekt *`class`*, nastaví vlastnost `__common`, importuje nastavení pomocí `setSettings(`*`class`*`)` a zavolá metodu `__run()`, pokud je definována.

Metoda `setSettings(`*`class`*`)`

Importuje nastavení do objektu.

Metoda `destroyClass(`*`class`*`)`

Zruší objekt a zavolá metodu `__destroy()`, je-li definována.

Metoda `sentHeaders()`

Odešle všechny hlavičky definované v příslušném profilu.

Metoda `showError(`*`error`*`)`

Zobrazí chybové hlášení *`error`*.

Metoda `writeToLog(text)`

Zapíše *`text`* do souboru s logem.

Metoda `relativePath(sourceDir, targetDir)`

Vrátí relativní cestu mezi adresářem *`sourceDir`* a *`targetDir`*.

3.4.d Objekt db

Tento objekt se stará o připojení do databáze a vykonávání SQL dotazů. Podle nastavení zobrazuje a loguje chyby či všechny SQL dotazy.

Vlastnosti

Kromě předaných nastavení obsahuje objekt jedinou vlastnost a tou je `connection`, ve kterém je umístěn odkaz na aktuální připojení do databáze, které se může předat databázovým funkcím.

Metody

Metoda `__run()`

Připojí se k databázi podle zadaného nastavení.

Metoda `__destroy()`

Odpojí se od databáze.

Metoda `select(select, from, [where], [order], [limit], [group], [having])`

Zavolá SQL dotaz SELECT. Povinné parametry jsou *`select`* a *`from`*. Kde *`select`* může být buďto řetězec nebo pole se sloupci, které se mají vybrat. Ostatní parametry jsou typu string. Nepovinné parametry jsou *`where`*, *`order`*, *`limit`*, *`group`*, *`having`*. Názvy parametrů odpovídají kontrukci SQL příkazu SELECT. Vrací identifikátor výsledku.

Metoda `insert(values, table)`

Vykoná SQL příkaz INSERT. Parametr *`values`* je asociativní pole sloupec => hodnota, *`table`* je název tabulky. Vrací identifikátor výsledku.

Metoda `update(set, table, [where], [limit])`

Vykoná SQL příkaz UPDATE. Parametr *`set`* je asociativní pole sloupec => hodnota, *`table`* je název tabulky. Nepovinné parametry *`where`* a *`limit`* odpovídají opět konstrukci SQL příkazu UPDATE. Vrací identifikátor výsledku.

Metoda `delete(table, [where], [limit])`

Zavolá SQL příkaz DELETE. Parametrem *`table`* zadáváte tabulku. Nepovinné parametry *`where`* a *`limit`* zase odpovídají konstrukci SQL příkazu DELETE. Vrací identifikátor výsledku.

Metoda `query(sql)`

Zavolá libovolný SQL dotaz *`sql`*. Vrací identifikátor výsledku.

Metoda `numRows()`

Vrátí počet řádků v posledním dotazu.

Metoda `lastId()`

Vrátí automaticky generované ID posledního příkazu INSERT.

Metoda `nextRecord([result])`

Vrátí další záznam z posledního dotazu nebo false, pokud už další záznam není. Pokud zadáte nepovinný parametr *`result`*, nevrací řádky z posledního dotazu, ale z dotazu, jehož výsledek předáme.

Metoda `escapeString(text)`

Vrátí *`text`* s ošetřenými nebezpečnými znaky pro použití v SQL dotazech a zabraňuje útoku SQL injection. Pro funkci `insert` a parametr *`values`* a pro funkci `update` parametr a *`set`* se používá automaticky.

3.4.e Objekt `import`

Tento objekt je využíván pouze při importu dat z KOSu v souboru `/cron/import.php`. Pro svůj běh vyžaduje inicializované objekty `db` a `login`.

Vlastnosti

Kromě nastavení neobsahuje žádné vlastnosti.

Metody

Metoda `downloadFile()`

Stáhne soubor s exportem ze zaheslované stránky.

Metoda `unzipFile()`

Extrahuje XML soubor ze staženého archivu.

Metoda `importFile()`

Naimportuje data z XML souboru.

Metoda `idExists(`*`name`*`, `*`value`*`, `*`table`*`)`

Vrací true, pokud v tabulce *`table`* existuje záznam, který má ve sloupci *`name`* hodnotu *`value`*. Jinak vrací false.

Metoda `rowExists(`*`items`*`, `*`table`*`, `*`id`*`)`

Vrací ID řádku, pokud v tabulce *`table`* existuje s hodnotami z asociativního pole *`items`* (sloupec => hodnota), jinak vrací false.

Metoda `importTableBegin(`*`semester`*`, `*`table`*`)`

Zpracuje všechny řádky tabulky *`table`* pro zadaný semestr *`semester`* pro začátek importu.

Metoda `importTableEnd(`*`semester`*`, `*`table`*`)`

Vymaže z tabulky *`table`*, všechny řádky, které nebyly v nových datech z KOSu.

3.4.f Objekt `login`

Slouží k přihlášení se na stránky a uchovávání přihlášení mezi stránkami. Pro svůj běh vyžaduje inicializovaný objekt `db` a dokáže spolupracovat s objektem `layout`.

Vlastnosti

Kromě vlastností importovaných z nastavení obsahuje následující vlastnosti:

- `isLogin` – vrací true, pokud je uživatel práve přihlášen.
- `username` – vrací uživatelské jméno aktuálně přihlášeného uživatele.
- `name` – vrací reálné jméno aktuálně přihlášeného uživatele.

- `rights` – vrací číselné označení práv přihlášeného uživatele pro konkrétního předmět.
- `rightsDescription` – pole, obsahující popis práv pro jednotlivé typy uživatelů.
- `errorDescription` – pole chybových hlášek, které mohou vzniknout při přihlášení

Metody

Metoda `__run()`

Při spouštění zapne v aplikaci podporu SESSION, v kterých se uchovávají informace o přihlášení. Pokud je inicializován objekt `layout`, předá mu přihlášeného uživatele. Není-li nikdo přihlášen, přidá posledně přihlášeného uživatele.

Pokud má být uživatel přihlášen, otestuje přihlášení. Přihlašuje-li se uživatel, zkusí uživatele přihlásit. Není-li uživatel přihlášen a pokouší-li se dostat na stránku, kde by přihlášen měl být, přesměruje ho na stránku `index.php`.

Metoda `__destroy()`

Uloží SESSION. Toto by zde nemuselo být, PHP se o vše stará automaticky.

Metoda `makeLogin()`

Zkontroluje, zda existuje uživatel se zadaným uživatelským jménem a heslem, pokud ano, uloží jeho uživatelské jméno do COOKIE, aby uživateli mohla aplikace toto jméno nabídnout při příštím přihlášení. Následně vygeneruje náhodné číslo a vytvoří hash z loginu uživatele, jeho IP adresy, z hodnoty USER_AGENT jeho prohlížeče a z náhodného čísla. Následně uloží do databáze tento hash, náhodné číslo a aktuální čas.

Pokud je inicializován objekt `layout`, předá mu informaci, že uživatel je v pořádku přihlášen, jeho uživatelské jméno a reálné jméno.

Metoda `testLogin()`

Vytáhne z databáze has, náhodné číslo a uložený čas pro přihlášeného uživatele. Následně vytvoří nový hash, uživatelova loginu, jeho IP adresy, hodnoty USER AGENT prohlížeče a náhodného čísla z databáze. Poté nový hash porovná

s tím, který je v databázi. Pokud se shodují, ověření uživatele projde. Následně zkontroluje, jestli uložený čas není menší než aktuální, o víc než je zadáno v nastavení. Pokud je, uživatele odhlásí pro dlouhou nečinnost. Pokud není, vygeneruje nové náhodné číslo, vytvoří nový hash jako v `makeLogin()` a spolu s aktuálním časem uloží vše do databáze.

Pokud je inicializován objekt `layout`, předá mu informaci, že uživatel je pořádku přihlášen, jeho uživatelské jméno a reálné jméno.

Metoda `logout([redirect])`

Odhlásí bezpečně uživatele. Pokud je *`redirect`* rovno true (standardně je), přesměruje ho zároveň na `index.php`.

Metoda `error(id, [username])`

Přesměruje uživatele na `index.php` se zadanou chybovou hláškou podle *`id`*. Pokud je zadán nepovinný parametr *`username`*, je předáno i aktuální uživatelské jméno.

Metoda `getName(username)`

Zjistí reálné jméno uživatele s uživatelským jménem *`username`*.

Metoda `setRights([semester], [subject])`

Nastaví uživateli práva pro zadaný semestr *`semester`* a předmět *`subjet`*. Oba parametry se mohou vynechat. Pak testuje pouze práva pro administrátora, která ani na jednom nezávisí.

Metoda `getStudentNameFromLogin(login)`

Vrátí reálné jméno studenta, podle zadaného uživatelského jména *`login`*.

Metoda `ifLoginToIndex()`

Metoda, která se volá na stránkách, kam nemají přístup přihlášení uživatelé, jsou přesměrováni na `index.php`. Například pro stránku s generováním nového hesla.

Metoda `changePassword(`*`oldPassword, newPassword1, newPassword2`*`)`

Změní heslo aktuálně přihlášenému uživateli. Musíte zadat správné staré heslo *`oldPassword`* a dvakrát nové heslo, *`newPassword1`* a *`newPassword2`*.

Metoda `regenerateCode(`*`username`*`)`

Vygeneruje kontrolní kód pro vygenerování nového hesla, uloží ho do databáze a odešle ho uživateli *`username`* na email.

Metoda `regeneratePassword(`*`username, code`*`)`

Vygeneruje nové heslo uživateli *`username`*, uloží ho do databáze a pošle uživateli na email. Musí být ale zadán správně kontrolní kód *`code`*, který byl minulou funkcí poslán uživateli na email.

Metoda `generatePassword(`*`[length]`*`)`

Vygeneruje náhodné heslo o délce *`length`*. Není-li zadáno, je délka 8 znaků.

3.4.g Objekt `layout`

Při psaní aplikace jsem chtěl úplně oddělit prezentační vrstvu od aplikační. Nejenom pomocí kaskádových stylů. Všechny funkce (až na výjimky) tudíž nevrací XHTML výstup, ale ten je doplněn až v samotných skriptech. Když už byl vymyšlený framework, chtěl jsem prezentování dat řešit pouze tak, že v každém skriptu, který bude framework využívat, se všechen výstup umístí do předem připravené šablony. A o tohle jsem se nechtěl programově starat, vše jsem chtěl, aby se dělalo naprosto automaticky. Abych nemusel výstup ukládat do žádné proměnné a psal ho stejně, jako bych odesílal prohlížeči kompletní XHTML stránku, ale ve skriptu budu odesílat pouze její tělo. Řešení jsem nalezl v PHP využitelném bufferování výstupu. V metodě `__destroy()` objektu `login` je současný výstup ještě před odesláním prohlížeči automaticky uložen z bufferu, následně vymazán, dosazen do XHTML šablony a pak je vše odesláno prohlížeči.

Toto řešení mi následně umožnilo jednoduše přesměrovávat i na konci skriptů, kdy už by byl normálně odeslán nějaký výstup a konstrukce `header('location: url')` by nešla použít.

Tento objekt pro svůj běh nevyžaduje žádný jiný objekt a ani s žádným nespolupracuje. Chtěl jsem, aby ostatní objekty spolupracovali s ním.

Vlastnosti

Kromě importovaných vlastností z nastavení můžeme nastavit:

- `title` – titulek stránky.
- `header` – hlavička stránky.
- `login` – asociativní pole s nastavením informací o ne/přihlášení. Obsahuje klíče `isLogin (boolean)`, `username`, `name` a `rights`.
- `subjects` – asociativní pole se seznamem předmětů (název => url).
- `subjectsActual` – název aktuálně vybraného předmětu.
- `services` – asociativní pole se seznamem služeb pro předmět (název => url).
- `servicesActual` – název aktuálně vybrané služby.

Metody

Metoda `__run()`

Zapne bufferování výstupu a načte šablonu. Následně v šabloně nahradí: `%folder%` za relativní cestu k souboru, `%applicationName%` za název aplikace, `%applicationUrl%` za celé url k aplikaci a `%footer%` za patičku.

Metoda `__destroy()`

Uloží současný bufferovaný výstup do proměnné a v šabloně nahradí zbývající data: `%title%` za titulek, `%header%` za hlavičku, `%login%` za box s nastaveným přihlášením či nepřihlášením, `%subjects%` za seznam předmětů (vynechá u administrátora aplikace), `%services%` za seznam služeb pro konkrétní předmět (nebo pro administrátora) a nakonec nahradí `%text%` za uložený výstup.

Metoda `deleteBuffer()`

Vymaže předčasně bufferovaný výstup, bez toho, aby byl uložen.

3.4.h Objekt `modules`

Tento objekt se stará o kompletní správu předmětů, jím příslušných akcí, nahráváním modulů a výběru studentů a cvičících podle předmětů a cvičení.

Objekt pro svůj běh vyžaduje objekty `db` a `login` a spolupracuje s objektem `layout`.

Vlastnosti

Kromě importovaných vlastností z nastavení se z objektu můžete dozvědět:

- `actualSemester` – kód aktuálního semestru.
- `actualSubject` – ID aktuálně vybraného předmětu.
- `actualSubjectCode` – kód aktuálně vybraného předmětu.
- `actualSubjectName` – název aktuálně vybraného předmětu.
- `actualSubjectSeminar` – ID semináře pro přihlášeného studenta pro vybraný předmět.
- `actualModule` – aktuálně používaný modul.
- `actualAction` – aktuálně zavolaná akce.
- `actualActionName` – název aktuálně zavolané akce.
- `subjects` – asociativní pole s předměty příslušejícími přihlášenému studentovi nebo učiteli. Klíč je název předmětu, hodnota je url adresa pro aktivaci předmětu
- `services` – asociativní pole s akcemi příslušejícími přihlášenému uživateli a vybranému předmětu. Klíč je název služby, hodnota je url pro zavolání služby.

Metody

Metoda `__run()`

Nejprve načte z databáze aktuální semestr. Poté zkontroluje, zda je vybrán předmět a pokud ano, nastaví jeho ID, kód, jméno a ID semináře. Pokud je vybrán modul a k němu akce, obojí uloží. Nakonec načte předměty a moduly.

Metoda `loadSubjects()`

Pokud je uživatel přihlášen a není to administrátor aplikace, načte pro uživatele předměty, na které je zapsán, případně které cvičí či přednáší. Uloží je do pole `subjects` i s url pro aktivaci a pokud je definován objekt `layout` předá mu toto pole.

Metoda `loadModules()`

Pokud je přihlášen administrátor aplikace, načte moduly nastavené v nastavení pouze pro administrátora. Pokud přihlášený uživatel není administrátor, načte společné moduly pro všechny předměty zadané v nastavení a nakonec načte konkrétní moduly předmětům.

Metoda `loadOneModule(`*`module`*`)`

Fyzicky načte modul *`module`* do objektu `modules`. Ten bude přístupný přes `module_`*`nazevmodulu`*, kde *`nazevmodulu`*, je zadaný *`module`*. Ten se musí shodovat se zadanými názvy v nastavení.

Pokud existuje v modulu metoda `__load()`, zavolá ji a nastaví modulu proměnnou `__modules`, kde je odkaz na objekt `modules`, z kterého se dá dál přes `__common` dostat k ostatním objektům. Více o struktuře modulů samostatná kapitola Moduly.

Nakonec do `services` z definice modulu načte příslušné služby pro konkrétního uživatele a jeho práva a pokud je definován objekt `layout`, předá mu toto pole.

Metoda `loadAction()`

Nejprve ověří, zda má uživatel právo na zavolání akce a pokud ano, načte soubor s příslušnou akcí. Jestli není vybrána žádná akce, vybere první akci v seznamu služeb. Pokud dojde k chybě, přesměruje na `index.php`.

Metoda `form()`

Vrací kompletní url na odkaz pro aktuálně načtený modul a akci.

Metoda `getSubjectCode(`*`subjectId`*`)`

Vrací kód předmětu pro `subjectId`.

Metoda `getSubjectName(`*`subjectId`*`)`

Vrací název pro *`subjectId`*.

Metoda `getSubjectSeminar(`*`subjectId, username`*`)`

Vrací ID semináře, na který je student zapsán.

Metoda `getRights()`

Podle práv nastavených v objektu `login` vrací pole s právy, které má konkrétní uživatel a podle kterých přiřazuje akce ze služeb.

Metoda `toIndex()`

Okamžitě přesměruje uživatele na `index.php`.

Metoda `toFirstAction()`

Přesměruje uživatele na první akci ve službách pro konkrétní předmět.

Metoda `toIndexError()`

Přesměruje uživatele na index.php s chybovou hláškou. Typicky, pokud by uživatel chtěl zobrazit akci, na kterou nemá práva.

Metoda `getSeminars(`*`subjectCode, [type]`*`)`

Vrací asociativní pole se semináři pro přihlášeného uživatele pro konkrétní předmět zadaný pomocí kódu *`subjectCode`*. Pokud nezadáte type, vybere všechny semináře. Pokud jako type zadáte *`don`*, vybere pouze semináře, kde je uživatel cvičící. Pokud zadáte *`don_other`*, vybere, ty, kde uživatel není cvičící. Zadáte-li *`student`*, vybere všechny semináře, kam je uživatel zapsán jako student a zadáte-li *`student_other`*, vyhledá všechny semináře, kde uživatel není zapsán jako student. Jinak vrací prázdné pole.

Výsledné pole je ve tvaru, kde klíč je id konkrétního semináře a hodnota je kdy a kde se seminář koná.

Metoda `getStudents(`*`seminar`*`)`

Vrátí asociativní pole všech studentů z konkrétního semináře *`seminar`*. Předáváte ID semináře. Vrací pole, kde klíč je uživatelské jméno studenta a hodnota je jeho jméno.

Metoda `getAllStudents(`*`subject`*`)`

Vrátí asociativní pole všech studentů z předmětu `subject`. Předáváte ID předmětu. Vrací pole, kde klíč je uživatelské jméno studenta a hodnota je jeho jméno.

Metoda `getDons(`*`seminar`*`)`

Vrátí asociativní pole všech cvičících z konkrétního semináře *`seminar`*. Předáváte ID semináře. Vrací pole, kde klíč je uživatelské jméno cvičícího a hodnota je jeho jméno.

Metoda `getAllDons(`*`subject`*`)`

Vrátí asociativní pole všech cvičících z předmětu `subject`. Předáváte ID předmětu. Vrací pole, kde klíč je uživatelské jméno cvičícího a hodnota je jeho jméno.

4. Moduly

Aplikace je velice jednoduše rozšiřitelná pomocí modulů. Všechny moduly musí být uloženy do adresáře `/modules`. Každý modul se skládá ze souboru s inicializačním objektem, který má tvar `module.`*`nazev`*`.php`, kde *`nazev`* je název modulu, uvedený v nastavení. Tento soubor musí být umístěn v přímo v adresáři `/modules`. Soubor obsahuje třídu s názvem `module_`*`nazev`*, kde *`nazev`* musí být stejný jako v názvu souboru. Tato třída se inicializuje a připojí k objektu `modules`, kde je pak přístupná přes `modules->module_nazev`.

Modul může mít libovolné vlastnosti a metody. Kromě toho existuje jedna vlastnost a metoda vyhraněná pro běh modulu. Pokud modul obsahuje metodu `__load()` je tato metoda zavolána při inicializaci modulu. Zároveň se při inicializaci vytvoří odkaz na objekt `modules` pomocí vlastnosti `__modules` přes kterou se dá následně pomocí vlastnosti `__common` dostat k celému aplikačnímu frameworku.

Příklad přístupu v objektu modulu k objektu db:

```
$this->__modules->__common->db->query();
```

Tímto způsobem je modul již plně funkční a může vykonávat definovanou činnost. Pokud však chcete přes modul ovládat formuláře či posílat jiný výstup uživateli, musíte definovat akce. Ty se definují ve vlastnosti `__action`. Jedna se o asociativní pole, asociativních polí. Hlavní klíč musí být stejný jako práva uživatelů, kteří mohou k akci přistupovat (`student`, `don`, `subject_admin`, `lecturer`, `administrator` – postupně, student, cvičící, administrátor předmětu, přednášející a administrátor aplikace – přednášející má automaticky práva administrátora předmětu a cvičícího a administrátor předmětu má automaticky práva cvičícího). Ke každému klíčí se přiřadí další asociativní pole, kde klíč značí název akce a hodnota popis. Příklad:

```
var $__actions = array(

'subject_administrator' => array('akce1' => 'Název
akce 1),
```

```
'don' => array('akce2' => 'Název akce 2'),
'student' => array('akce3' => 'Název akce 3', 'akce4' =>
'Název akce 4')

    );
```

Pokud definujete akce, musíte v adresáři `/modules` vytvořit adresář s názvem `module.nazev` a do něj umístit soubor podle názvu akce *`nazev_akce`*`.php`. Do tohoto souboru můžete vkládat přímo výstup, který se odešle do prohlížeče po dosazení do XHTML šablony. V souboru se přes konstrukci `$this` dostanete k objektu `modules`. Pokud chcete přistupovat do objektu modulu ze souboru akce, musíte použít `$this->module_nazev->nejakeMetoda()`.

4.1 Existující moduly

Následuje jednoduchý popis modulů, dodávaných spolu s aplikací. Detailnější popis použití naleznete v kapitole s manuálem.

4.1.a Modul `administrator`

Speciální modul, který definuje akce pouze pro administrátora aplikace. Umožňuje vybrat aktuální semestr, exportovat celý semestr jako sadu SQL příkazů a nakonec vymazat celý semestr.

4.1.b Modul `lecturer`

Tento modul umožňuje přednášejícímu přidělit jednomu cvičícímu předmětu práva pro administraci předmětu, která má každý přednášející automaticky.

4.1.c Modul `text`

Studentům umožňuje zobrazit text o předmětu, cvičícím ho měnit a administrátorovi aplikace změnu úvodního textu celé aplikace.

4.1.d Modul `email`

Umožňuje odesílat hromadné emaily studentům i cvičím pro jednotlivé semináře nebo všem studentům zapsaným na předmět.

4.1.e Modul `select`

Dovolí administrátorovi předmětu přiřadit předmětu konkrétní moduly.

4.1.f Modul `import`

Umožňuje administrátorovi předmětu import starých definic modulů do nového semestru.

4.1.g Modul `points`

Dovoluje definovat všechny možnosti pro zadávání bodů, vybírání úloh, tvoření skupin a odevzdávání úloh elektronickou cestou.

5. Manuál

K vlastní aplikaci se mi píše manuál velice těžko. Snažil jsem se psát aplikaci tak, aby k ní žádný manuál nebyl potřeba. Ale co přijde jasné mě, určitě nepřijde jasné všem uživatelům. Kopie manuálu pro jednotlivé typy uživatelů je umístěna v aplikaci v adresáři `/help`.

5.1 Obecné

Rozvržení aplikace sestává ze čtyř částí. Úplně nahoře je hlavička a nadpisem o aktuální akci. Klinutím na název aplikace se vrátíte zpět na úvod. V levé části aplikace je formulář pro přihlášení, případně seznam předmětů a příslušných služeb. Dolní část zabírá patička stránky s odkazy na úvod a emailem na administrátora aplikace. Konečně největší část zabírá samotný prostor pro prezentování obsahu.

5.1.a Přihlášení

Pro přihlášení zadejte vaše uživatelské jméno (je shodné pro celý FEL) a heslo, které Vám přišlo na školní email do formuláře na levé straně aplikace. Úspěšné přihlášení je indikováno tím, že zmizí formulář a zobrazí se Vaše jméno a uživatelské jméno. Pokud přihlášení nebylo úspěšné, zobrazí se chybová hláška.

5.1.b Odhlášení

Pro bezpečné odhlášení z aplikace se doporučuje při ukončení práce stisknout tlačítko odhlásit, umístěné v boxu po levé straně aplikace.

5.1.c Zapomenuté heslo

Pokud zapomenete Vaše heslo, můžete si vygenerovat heslo nové. V levém boxu s formulářem klikněte na tlačítko zapomenuté heslo. Následně zadejte Vaše uživatelské jméno a klikněte na ověřit. Pokud je v pořádku uživatelské jméno ověřeno, přijde Vám na školní email bezpečnostní kód, který zadáte na další stránce. Po kliknutí na tlačítko vygenerovat nové heslo, se Vám nové heslo vytvoří a pošle na školní email.

5.1.d Změna hesla

Pokud si přejete změnit heslo, přihlaste se a vyberte tlačítko změnit heslo z boxu přihlášení v levé straně aplikace. Na zobrazené stránce zadejte Vaše staré

heslo a pro kontrolu dvakrát heslo nové. To musí mít minimálně pět znaků! Pokud jsou zadané údaje správné, bude Vám po kliknutí na tlačítko změnit heslo, heslo změněno.

5.2 Uživatel administrátor

Jedná se o speciální účet, určený pouze k ovládání aplikace. Uživatelské jméno, které nelze změnit je `administrator`. Tento uživatel si také nemůže nechat vygenerovat nové heslo, pouze si může heslo změnit.

Po přihlášení se zobrazí v levé části menu se službami:

- **Uprav úvodní text** – umožňuje nastavit text, který je zobrazen v úvodu aplikace.
- **Aktuální semestr** – nastaví pro aplikaci aktuální semestr.
- **Export semestru** – exportuje semestr do SQL souboru.
- **Vymazat semestr** – vymaže semestr z aplikace.

5.2.a Uprav úvodní text

Umožňuje změnit text, který je zobrazen na úvodní stránce aplikace. Můžete použít všechny formátovací značky jazyka HTML.

5.2.b Aktuální semestr

Data se importují do aplikace automaticky. Zde si můžete nastavit, jaký semestr se bude v aplikaci používat jako aktuální.

Vyberte semestr z roletky a klikněte na nastavit jako aktuální

5.2.c Export semestru

Umožňuje exportovat data celého semestru jako soubor SQL. Vyberte semestr z roletky a klikněte na exportovat semestr.

5.2.d Vymazat semestr

Každý semestr se skládá až z několika set tisíc záznamů. Pro šetření místem v databázi je doporučeno staré semestry z databáze vymazat. Před tímto krokem je doporučeno semestr exportovat!

Vyberte semestr z roletky a klikněte na vymazat semestr.

5.3 Uživatel student

Po přihlášení se Vám zobrazí seznam předmětů, na který jste zapsán. Pokud vyberete předmět, na který jste zapsán jako student, zobrazí se Vám pod předmětem následující služby:

- **O předmětu** – zobrazí, text s informacemi o předmětu.
- **Poslat email** – umožňuje odesílat hromadné emaily, všem studentům a cvičícím pro vybraný předmět, případně pro konkrétní cvičení.

Tyto služby jsou aktivní pro všechny předměty. Pokud má předmět přidělen modul s bodováním, máte přístupné i tyto služby:

- **Moje výsledky** – zobrazí Vaše výsledky a zda máte nárok na zápočet.
- **Všechny výsledky** – zobrazí výsledky všech studentů předmětu.
- **Semestrální práce** – umožňuje vytváření skupin, rezervaci úloh a elektronické odevzdání úloh.

5.3.a Poslat email

Nahoře vyberte cvičení, jehož studentům nebo cvičícím chcete odeslat email nebo všechny studenty či cvičící pro konkrétní předmět. Následně zaškrtněte uživatele, kterým chcete email odeslat, zadejte jeho předmět a text a klikněte na tlačítko odeslat email.

5.3.b Moje výsledky

Zobrazí body, které Vám udělil cvičící. Kliknete-li na tlačítko detail, uvidíme popis bodování, minimum a možné maximum bodů.

5.3.c Semestrální práce

Zobrazí všechny seminární práce pro daný semestr. Vidíte termín odevzdání, je-li zadán, požadované minimum a maximum bodů.

Je-li možnost vybrat si úlohu, klikněte na tlačítko vybrat úlohu. Zobrazí se Vám seznam úloh. Chcete-li si zamluvit úlohu klikněte na tlačítko zamluvit úlohu. Máte-li úlohu zamluvenou a chcete-li si vybrat jinou, stačí kliknout u jiné úlohy na zamluvit úlohu. Z původní úlohy budete automaticky odhlášeni. Chcete-li si pouze

odhlásit úlohu, stačí kliknout na uvolnit. Máte-li vytvořenou skupinu, promítnou se všechny změny všem členům skupiny.

Je-li možné, vytvořit pro odevzdání úlohy skupinu, klikněte na tlačítko vytvořit skupinu. Zobrazí se vám seznam uživatelů z Vašeho cvičení. Chcete-li do skupiny přidat uživatele, klikněte vedle jeho jména na tlačítko přidat do skupiny. Nahoře na stránce vidíte Vaši skupinu. Chcete-li uživatele odebrat, klikněte na tlačítko odebrat ze skupiny, vedle jeho jména. Skupinu můžete upravovat pouze do odevzdání práce!

Umožňuje-li práce elektronické odevzdání, klikněte na tlačítko odevzdat úlohu. Zobrazí se Vám formulář, do kterého vyberete soubor s Vaší prací a klikněte na odevzdat práci. Soubor se uloží na server. Pokud máte založenu na úlohu skupinu, přiřadí se soubor všem členům skupiny.

5.4 Uživatel cvičící

Stejná práva jako cvičící mají také přednášející a administrátor předmětu. Každý cvičící u předmětu vidí následující služby:

- **O předmětu** – zobrazuje informace o předmětu.
- **Poslat email** – umožňuje odesílat hromadné emaily, všem studentům a cvičícím pro vybraný předmět, případně pro konkrétní cvičení.

Tyto služby jsou aktivní pro všechny předměty. Pokud má předmět přidělen modul s bodováním, máte přístupné i tyto služby:

- **Udělit body a zkontrolovat odevzdané práce** – umožňuje přidělit studentům body za jednotlivé úlohy a zkontrolovat odevzdané práce.
- **Výsledky studentů** – zobrazuje výsledky studentů a umožňuje jejich export do CSV.

5.4.a Poslat email

Nahoře vyberte cvičení, jehož studentům nebo cvičícím chcete odeslat email nebo všechny studenty či cvičící pro konkrétní předmět. Následně zaškrtněte uživatele, kterým chcete email odeslat, zadejte jeho předmět a text a klikněte na tlačítko odeslat email.

5.4.b Udělit body a zkontrolovat odevzdané práce

Nahoře vyberte cvičení, jehož studentů chcete přidělit body nebo vyberte všechny studenty předmětu. Zobrazí se Vám seznam studentů. Vyberte studenta, jemuž chcete přidělit body a klikněte na zobrazit bodování. Vedle seznamu studentů, se Vám zobrazí všechny úlohy, kterým můžete studentovi přidělit body. Vidíte minimum a maximum bodů za úlohu a je-li odevzdána práce elektronicky, můžete si ji zobrazit odkazem zobrazit práci. V závorce vidíte, kdy byla práce odevzdána. Nakonec má každá úloha políčko, kam zadáte počet bodů a vyberete přidělit body.

5.4.c Výsledky studentů

Vyberte nahoře cvičení nebo všechny studenty. Následně se Vám zobrazí seznam studentů seřazený podle počtu bodů a podrobným rozpisem bodů za všechny úlohy a s informací, zda má student nárok na zápočet.

Chcete-li seznam exportovat, klikněte na tlačítko exportovat do CSV. Tento soubor můžete otevřít například v Excelu.

5.5 Uživatel přednášející a administrátor předmětu

Uživatelé přednášející a administrátor předmětu mají stejná práva. Jediný rozdíl je ten, že přednášející může určit jednoho cvičícího, který bude mít práva administrátora předmětu. To se provede vybráním služby **Nastavit administrační práva cvičícímu**. Následně vyberete ze seznamu cvičícího a kliknete na nastavit cvičícímu administrační práva.

Dále mají oba uživatelé na výběr z těchto služeb:

- **O předmětu - upravit** – umožňuje upravit text s informacemi o předmětu.
- **Vybrat moduly pro předmět** – přiřadí předmětům moduly.
- **Importovat staré nastavení** – vloží do aktuálního semestru nastavení předmětu ze starších semestrů.

Tyto služby jsou aktivní pro všechny předměty. Pokud má předmět přidělen modul s bodováním, máte přístupné i tyto služby:

- **Nastavení předmětu** – umožňuje nastavit minimum bodů potřebných pro získání zápočtu.

- **Nastavení bodování** – nastavení všech úloh, za které mohou studenti dostávat body.
- **Semestrální práce** – umožňuje nastavit semestrální práce.
- **Seznam úloh** – zde můžete nastavit úlohy pro semestrální práce.

5.5.a O předmětu - upravit

Umožňuje změnit text, který je zobrazen jako informace o předmětu. Můžete použít všechny formátovací značky jazyka HTML.

5.5.b Vybrat moduly pro předmět

Na stránce je zobrazen seznam modulů, které můžete předmětu přiřadit. Zaškrtněte moduly, které chcete přiřadit předmětu. Pokud nějaký modul odškrtnete, nevymaže se jeho nastavení.

5.5.c Importovat staré nastavení

Umožňuje importovat nastavení ze starého semestru do semestru aktuálního. Vyberte semestr a klikněte na importovat nastavení.

5.5.d Nastavení předmětu

Zde můžete nastavit minimum bodů, potřebných pro získání zápočtu. Zadejte počet bodů a klikněte na uložit.

5.5.e Nastavení bodování

Zde zadáváte úlohy, za které studenti dostávají body. Pro přidání vyplňte formulář dole. Povinné položky jsou název a pořadí, kde pořadí určuje pořadí úlohy v semestru. Volitelně můžete zadat popis úlohy, minimum bodů potřebných pro zápočet, maximum bodů (pouze informativní) a můžete vybrat semestrální práci, viz semestrální práce.

Pokud chcete úlohu upravit či vymazat, vyberte příslušné akce v tabulce úplně napravo.

5.5.f Semestrální práce

Zde můžete nastavit semestrální práce. Zadejte povinně kolik studentů může práci řešit ve skupině, volitelně pak od kdy do kdy se může práce odevzdávat, informace o souboru, který se má odevzdat (Informace o souboru), má-li se odevzdat

a část souboru, která se doplní do tohoto formátu `%login_studenta%_%id_semestralky%_Nazev_souboru_%puvodninazev%.xxx`. Nechte název souboru prázdný, pokud nechcete, aby se práce odevzdávala elektronicky.

Pokud chcete semestrální práci upravit či vymazat, vyberte příslušné akce v tabulce úplně napravo.

5.5.g Seznam úloh

Zde můžete zadat seznam úloh pro jednotlivé semestrální práce. Nejprve vyberte semestrální práci nahoře na stránce a zadejte vybrat semestrální práci. Následně pro přidání zadejte povinně název úlohy a počet úloh na jedno cvičení. Zadáte-li jako -1 je počet úloh neomezen. Vybere-li si pak úlohu skupina, odečte se počet úloh stejný s počtem členů skupiny. Volitelně můžete zadat popis úlohy.

Pokud chcete úlohu upravit či vymazat, vyberte příslušné akce v tabulce úplně napravo.

6. Závěr

Musím se přiznat, že celá práce byla psána v časovém presu. Vzorový export dat jsem dostal až v průběhu letošního června a bez něj jsem nebyl schopen začít aplikaci psát. Snažil jsem se psát vše tak, aby bylo minimalizováno riziko chyb rozdělením zdrojového kódu do hodně souborů, hlavně, aby se neztrácela přehlednost. Bohužel ke vzorovému export jsem nedostal žádný komentář a díky termínu (letní prázdniny), kdy jsem práci musel napsat, jsem nebyl schopen ani žádné detaily k exportu získat. Z tohoto důvodu v aplikaci počítám, že jednotlivá ID u všech tabulek v XML souboru se vztahují k jednomu konkrétnímu semestru. Pokud by tomu tak nebylo, mohlo by při importu dat z nového semestru dojít k neočekávanému chování celé aplikace. Jelikož ještě není vyřešen přístup k aktuálním exportům z KOSu, počítám, že aplikace ještě tento semestr nebude nasazena do ostrého provozu a věřím, že do jejího ostrého nasazení, se vše vyjasní a případně opraví.

Pevně věřím, že aplikace je napsána tak, aby byla co možná nejjednodušeji rozšiřována i od jiných osob než ode mě a bude sloužit ještě mnoho let aniž by bylo třeba její kompletní přepracování.

V rámci rozšíření by mělo dojít k aplikaci jednotného přihlášení přes centrální server. Dále bych rád aplikaci přepsal tak, aby využívala nové objektové možnosti jazyka PHP 5 a do aplikačního frameworku přidal možnost využití více jazykových mutací.

Použitá literatura

[1] J. Kosek, PHP – tvorba internetových aplikací, Grada publishing s. r. o., 1999

[2] W. Luke, L. Thompson, PHP a MySQL – rozvoj internetových aplikací, 2003

[3] A. Gutmans, Mistrovství v PHP 5, Computer Press, a.s., 2005

[4] PHP Manual, http://www.php.net/manual/en/index.php

[5] MySQL - pestrý svět databází, díl 1 – 40,
http://www.linuxsoft.cz/article.php?id_article=731

Příloha A – Struktura tabulek

Podtržený sloupec je primárním klíčem tabulky.

Struktura tabulky r01_semestry

id_r01: varchar(4)
nazev: varchar(20)
aktualni: int(1)

Struktura tabulky r02_katedry

id_r02: int(11)
id_r01: varchar(4)
kod: varchar(10)
nazev: varchar(100)
aktualizovano: int(1)

Struktura tabulky r03_ucitele

id_r03: int(11)
id_r01: varchar(4)
login: varchar(30)
jmeno: varchar(30)
prameni: varchar(30)
titul_pred: varchar(20)
titul_za: varchar(20)
aktualizovano: int(1)

Struktura tabulky r04_studenti

id_r04: int(11)
id_r01: varchar(4)
login: varchar(20)
jmeno: varchar(30)
prameni: varchar(30)
rocnik: int(1)
aktualizovano: int(1)

Struktura tabulky r05_predmety

id_r05: int(11)
id_r01: varchar(4)
id_r02: int(11)
kod: varchar(10)
nazev: varchar(50)
aktualizovano: int(1)

Struktura tabulky r06_mistnosti

id_r06: int(11)
id_r01: varchar(4)
cislo: varchar(10)
aktualizovano: int(1)

Struktura tabulky r07_zapsane_predmety

id_r07: int(11)
id_r01: varchar(4)
id_r05: int(11)
id_r04: int(11)
aktualizovano: int(1)

Struktura tabulky r08_rozvrh

id_r08: int(11)
id_r01: varchar(4)
den: int(1)
sudy_lichy: char(1)
hodina: int(2)
pocet_hodin: int(2)
id_r05: int(11)

Struktura tabulky r09_rozvrh_ucitele

id_r09: int(11)
id_r01: varchar(4)
id_r08: int(11)
id_r03: int(11)
aktualizovano: int(1)

Struktura tabulky r10_rozvrh_studenti

id_r10: int(11)
id_r01: varchar(4)
id_r08: int(11)
id_r04: int(11)
aktualizovano: int(1)

Struktura tabulky p01_prihlaseni

login: varchar(30)
heslo: varchar(32)
heslo_kod: varchar(32)
login_hash: varchar(32)
login_time: datetime
login_rand: int(11)

Struktura tabulky p02_admini_predmetu

id_p02: int(11)
id_r01: varchar(4)
login: varchar(30)
id_r05: int(11)

Struktura tabulky m01_moduly

id_m01: int(11)
id_r01: varchar(4)
predmet_kod: varchar(10)
modul: varchar(50)

Struktura tabulky m02_texty

id_m02: int(11)
id_r01: varchar(4)
predmet_kod: varchar(10)
text: text

Struktura tabulky m03_body_01_predmety

id_m03_01: int(11)
id_r01: varchar(4)
predmet_kod: varchar(10)
min_bodu: decimal(10,2)

Struktura tabulky m03_body_02_body

id_m03_02: int(11)
id_r01: varchar(4)
predmet_kod: varchar(10)
nazev: varchar(100)
popis: text
min_bodu: decimal(10,2)
max_bodu: decimal(10,2)
poradi: int(2)
id_m03_03: int(11)

Struktura tabulky m03_body_03_semestralky

id_m03_03: int(11)
id_r01: varchar(4)
predmet_kod: varchar(10)
pocet_studentu: int(2)
datum_odevzdani_od: date
datum_odevzdani_do: date

soubor_info: varchar(255)
soubor_nazev: varchar(30)

Struktura tabulky m03_body_04_ulohy

id_m03_04: int(11)
id_r01: varchar(4)
predmet_kod: varchar(10)
id_m03_03: int(11)
nazev: varchar(100)
popis: text
pocet_cviceni: smallint(6)

Struktura tabulky m03_body_05_odevzdane_data

id_m03_05: int(11)
id_r01: varchar(4)
predmet_kod: varchar(10)
id_m03_03: int(11)
login: varchar(30)
soubor: varchar(100)
datum_odevzdani: date

Struktura tabulky m03_body_06_ulohy_data

id_m03_06: int(11)
id_r01: varchar(4)
predmet_kod: varchar(10)
id_m03_03: int(11)
id_m03_04: int(11)
id_r08: int(11)
login: varchar(30)

Struktura tabulky m03_body_07_skupiny_data

id_m03_05: int(11)
id_r01: varchar(4)
predmet_kod: varchar(10)
id_m03_03: int(11)
username_1: varchar(30)
username_2: varchar(30)

Struktura tabulky m03_body_08_body_data

id_m03_08: int(11)
id_r01: varchar(4)
predmet_kod: varchar(10)
id_m03_02: int(11)

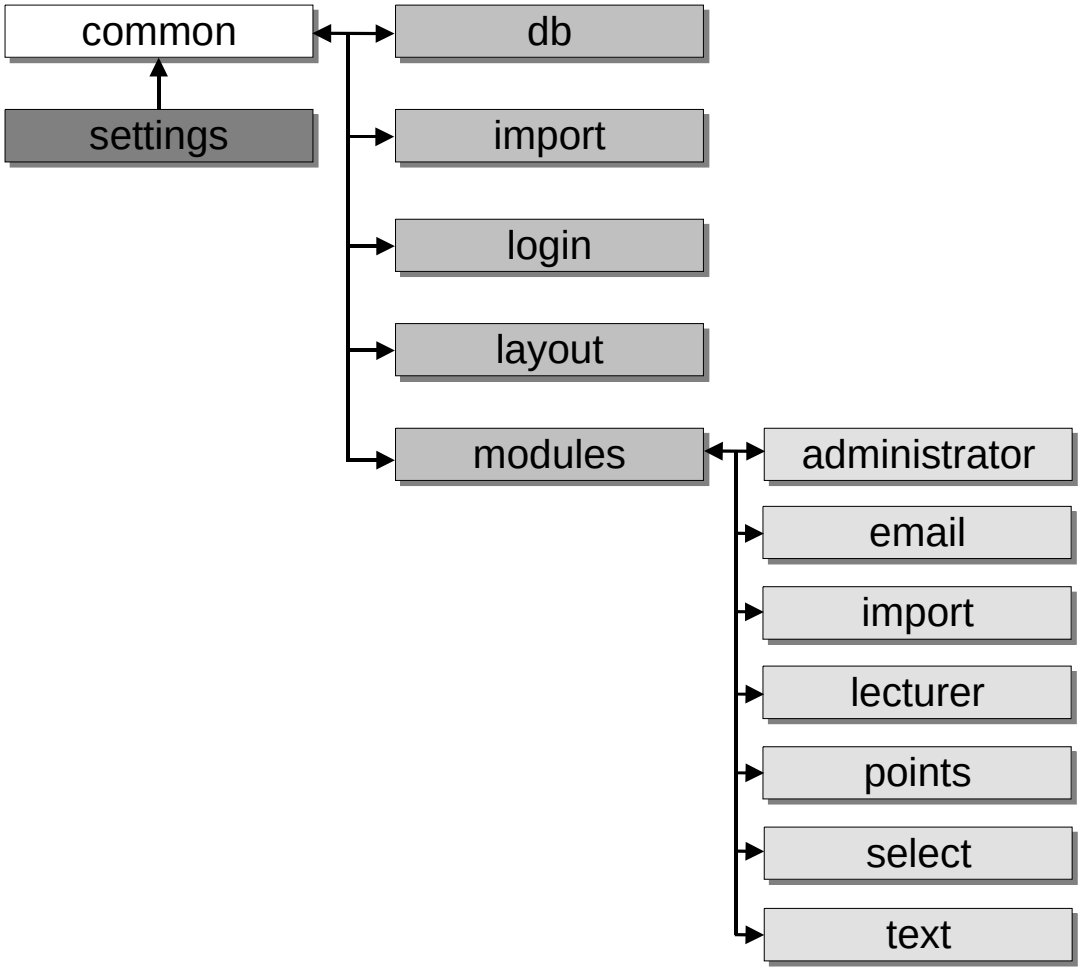
```
login: varchar(30)
body: decimal(10,2)
```

Příloha B - Struktura XML dokumentu

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<ROZVRH semestr=" " nazev_semestru=" ">
  <KATEDRY>
    <katedra id="" kod="" nazev_cz="" />
  </KATEDRY>
  <MISTNOSTI>
    <mistnost cislo=" " id="" />
  </MISTNOSTI>
  <PREDMETY>
    <predmet id="" kat_id="" kod=" " nazev=" " />
  </PREDMETY>
  <UCITELE>
    <ucitel id="" login="" jmeno=" " prijmeni=""  titul_pred=""
titul_za=" " />
  </UCITELE>
  <LISTKY>
    <listek id="" den_cis="" sudy_lichy=" " hodina="" pocet_hodin=""
predmet_id="" typ_vyuky=" " ucitel1_id="" ucitel2_id=""
mistnost_id="" />
  </LISTKY>
  <STUDENTI>
    <student id="" login="" jmeno="" prijmeni=" " rocnik="" />
  </STUDENTI>
  <LISTKY_STUDENTU>
    <listek_studenta listek_id="" student_id="" />
  </LISTKY_STUDENTU>
  <ZAPISY_PREDMETU>
    <zapis predmet_id="" student_id="" />
  </ZAPISY_PREDMETU>
</ROZVRH>
```

Vstupní dokument je v kódování ISO-8859-2 (Latin 2). PHP se při importu postará o automatický převod kódování UTF-8. V XML dokumentu je mnohem více parametrů, zde jsou uvedeny jen ty použité při importu.

Příloha C – Struktura aplikačního frameworku



Obrázek 1 - struktura aplikačního frameworku