UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMTATIKY
KATEDRA APLIKOVANEJ INFORMATIKY

NOVÝ DLHODOBÝ VIACÚČELOVÝ SKLAD ÚLOH NA CVIČENIA

BAKALÁRSKA PRÁCA

Andrej Jursa

Školiteľ: Mgr. Pavel Petrovič, PhD.

Študijný program: 9.2.9 APLIKOVANÁ INFORMATIKA

BRATISLAVA 2013

Abstrakt

Text ...

Čestne prehlasujem, že som túto prácu vypracoval samostatne,
s použitím literatúry a zdrojov uvedených v závere práce.

_____

Andrej Jursa

Poďakovanie

Rád by som sa poďakoval svojmu školiteľovi bakalárskej práce Mgr. Pavlovi Petrovičovi, PhD. za cenné rady, pripomienky a usmernenia počas celého procesu tvorby tejto práce.

Obsah

1 Úvod

Úvodná kapitola ...

2 Východiská

V tejto kapitole predstavím východiskové informácie ako je teória, existujúce systémy či východisková práca.

2.1 Prehľad teórie

2.1.1 E-learning a learning management system

E-learning je vzdelávací proces, využívajúci informačné a komunikačné technológie k tvorbe kurzov, k distribúcii študijných materiálov, komunikáciou medzi študentami a pedagógmi a k riadeniu štúdia. V e-learningu sa často používajú tzv. learning management systémy.

Learning management system alebo LMS je software na administráciu, dokumentáciu, sledovanie, spravovanie a predávanie vzdelávacích kurzov či tréningových programov.

Takýto LMS by mal byť schopný:

- centralizovať a automatizovať administráciu,
- používať samo-obslužné a samo-riadiace služby,
- zhromažďovať a rýchlo poskytovať vzdelávací obsah,
- podporovať prenositeľnosť a normy,
- upravovať obsah a povoliť znovupoužitie vedomostí,
- poskytovať online tréning a semináre,
- evidencia a správa študentov,
- evidencia a správa kurzov,
- správa študijných plánov,
- evidencia hodnotenia študentov,
- testovanie a preskúšanie študentov,
- správa prístupových práv,
- komunikačné nástroje (diskusné fórum, chat),
- úložisko výukových materiálov.

Väčšina LMS systémov je webovo-založená, čo uľahčuje prístup k vzdelávaciemu obsahu a správe tohto obsahu. LMS sú často používané vzdelávacími inštitúciami na zlepšenie a podporu vzdelávania v triedach a ponúkajú kurzy väčšiemu počtu študentov celosvetovo.

2.1.2 SCORM

Shareable Content Object Reference Model je referenčný model pre e-learning. Ide o súbor špecifikácií a štandardov, ktorých hlavnou úlohou je umožnenie používať obsah vytvorený v súlade so štandardom SCORM v ľubovolnom LMS. Ako z názvu vyplýva, ide o model zdielatelných obsahových

objektov (SCO), umožňujúce znovupoužitie vzdelávacieho obsahu vo všetkých produktoch alebo platformách, ktoré sú SCORMu prispôsobené. SCORM používa na popis objektov manifest, čo je popisný súbor v XML (eXtensible Markup Language). Aplikačný profil metadát popisujúcich SCORM objekty má 64 prvkov, no len malá časť z vyžadovaná povinne pre dosiahnutie zhody s referenčným modelom. Tieto sa delia do deviatich kategórií:

- všeobecná kategória,
- životný cyklus,
- meta-metadáta,
- technická kategória,
- vzdelávacia kategória,
- právna kategória,
- vzťahy,
- anotácie,
- klasifikácia.

Tento model je vytváraný americkou iniciatívou ADL (Advanced Distributed Learning Initiative) s odvolávaním sa na normy vytvárané konzorciami IEEE a IMS Learning Technology Standards.

2.1.3 Unit testing

V programovaní je unit testing (voľne preložené ako jednotkové testovanie) metóda, ktorou sa individuálne časti zdrojového kódu, množiny jedného či viacerých modulov programu spojených kontrolnými dátami, procedúry a funkcie testujú aby sa overilo, že pracujú tak ako bolo zamýšľané. Intuitívne, unit je možno vidieť ako najmenšiu testovateľnú časť aplikácie.

V procedurálnom programovaní môže byť takýmto unitom aj celý modul, ale bežnejšie sú to samotné procedúry a funkcie. V objektovo-orientovanom programovaní unitom často býva celý interface, ako sú triedy, ale môžu to byť aj individuálne metódy triedy.

Unit testy sú vytvárané programátormi, prípadne občas testermi, počas procesu vývoja programu.

Pri testovaní sa používajú takzvané test case (testovacie prípady), čo sú množiny podmienok a premenných, pomocou ktorých sa určuje, či aplikácia alebo softvérový systém pracuje správne. Ideálne je, ak sú tieto testovacie prípady na sebe nezávislé, pričom pri testovaní môžu na zlepšenie izolácie jedného od druhého používať rôzne nekompletné metódy, makety objektov, falošné alebo skúšobné dáta či objekty.

Unit testy sa používajú v takzvanom testami-riadenom vývoji (test-driven development, TDD), metodológiách Extrémneho programovania alebo Scrum, kde sa skôr ako sa píše kód programu, napíšu sa najprv testy, ktoré musí

program spĺňať. Ak potom napísaný program spĺňa všetky definované testy, je považovaný za kompletný. Ak unit test zlyhá, je možné zvažovať len dve možnosti, buď je chyba v kóde, ktorý testu nevyhovel, alebo je chyba v teste. Unit testy do veľkej miery napomáhajú k objaveniu práve chýb v kóde, keďže pomáhajú presnejšie vymedziť, pri akých vstupoch sa aká časť kódu správa inak ako je od nej očakávané.

2.2 Prehľad existujúcich systémov

2.2.1 Moodle

Moodle je open source systém na tvorbu elektronických výukových kurzov na internete. Ide o modulárny learning management system, ktorý poskytuje funkcionalitu pre:

- pridávanie a správu študijných materiálov vo forme HTML stránok, súborov na stiahnutie, FLASH animácií, štruktúrovaných prednášok a podobne,
- podporuje diskusné fóra s možnosťou odoberania príspevkov e-mailom,
- vytváranie a správa úloh pre účastníkov kurzov,
- podporuje automaticky vyhodnocované testy zložené z rôznych typov testovacích úloh,
- slovníky a databázy, na ktorých napĺňaní sa môžu podieľať aj účastníci kurzov,
- podporuje ankety a hlasovania,
- vzdelávací obsah je v špecifikáciách SCORM alebo IMS Content Package.

Moodle tiež umožňuje evidenciu študijných výsledkov účastníkov kurzov / študentov, taktiež zaznamenáva činnosť používateľov v podrobných protokoloch a súhrnných štatistikách. Moodle je napojitelný na iné systémy, napríklad autentifikačné systémy ako LDAP, komunikačné ako IMAP či systémy pre správu obsahu ako Postnuke.

Na rozdiel od systému LaMSfET, ktorý je hlavným východiskom tejto práce, systém Moodle nepodporuje dlhodobé skladovanie úloh a ich znovupoužitie či klonovanie do nových zostáv úloh. Tak isto je Moodle narozdiel od LaMSfETu orientovaný všeobecne, nepodporuje automatické spracovanie odovzdaných riešení, napr. formou testovania riešenia formou unit testov.

2.2.2 Chamilo

Chamilo je open-source e-learning a content management system, zameraný na globálne zlepšenie prístupu k vzdelaniu a vedomostiam. Vytvára ho asociácia Chamilo Association, ktorá sa rovnako stará aj o jeho propagáciu, údržbu komunikačných kanálov týkajúcich sa systému a vytvárajú sieť poskytovateľov služieb a prispievateľov do tohto softvéru.

Momentálne existujú dve verzie Chamilo, verzia 1.*, postavená na Dokeos

LMS, a verzia 2.0, ktorá je kompletne novým softvérom.

Chamilo podporuje:

- správu kurzov, používateľov a tréningových cyklov vrátane SOAP webových služieb pre vzdialenú správu,
- je kompatibilný so SCORM 1.2,
- viac-inštitučný mód s centrálnym manažmentom,
- testy študentov, ktoré môžu byť časovo obmedzované / kontrolované,
- natívne používa UTF-8 medzinárodné znakové sady a časové zóny,
- zaznamenáva pokroky používateľov / študentov,
- vytvára sociálnu sieť študentov.

Tak ako Moodle, aj Chamilo je veľmi všeobecne orientovaný softvér, ktorý nepodporuje nijak dlhodobé uchovávanie úloh. Jeho výhodou je množstvo komunitných a komunikačných funkcií, a vcelku jednoduché manažovanie obsahu, dokonca podporuje video konferencie. Nevýhodou je, že nie je takmer vôbec vhodný na vyučovanie informatických predmetov.

2.2.3 LaMSfET

Systém LaMSfET (Longterm and Multipurpose Storage for Excercise Tasks – Dlhodobý a viacúčelový sklad úloh na cvičenia) je webovo-orientovaný software podobný LMS, ktorý vznikol ako bakalárska práca študenta aplikovanej informatiky Petra Jurča v roku 2009 a ďalej bol nasadený, vyvíjaný a udržovaný samotným autorom a Mgr. Pavlom Petrovičom, PhD. na univerzitnej doméne capek.ii.fmph.uniba.sk, kde slúži ako kvázi-LMS pre podporu predmetov Programovanie 4, 5, Programovacie paradigmy a pod.

Tento systém sa na rozdiel od iných LMS vyznačuje snahou dlhodobo uchovávať úlohy a ich riešenia, zaraďovať úlohy do zostáv úloh a tieto znovu-používať pri cvičeniach.

LaMSfET umožňuje úlohy označovať značkami z hierarchickej štruktúry značiek, čo umožňuje správcom / učiteľom jednoducho nájsť staršie úlohy na požadovanú tému a použiť ich v novej zostave úloh, prípadne urobiť klon takejto úlohy a pozmeniť v nej hodnoty pred vložením do novej zostavy úloh.

Systém má integrované jednoduché testovanie riešení úloh týkajúcich sa programovacieho jazyka Java. Toto testovanie funguje na princípe očakávaného výstupu z programu po načítaní dát zo štandardného vstupu.

Systém má integrovanú aj podporu pre unit testy Javových programov, táto podpora je v ňom však integrovaná vcelku nesystémovo, čo znamená, že na vytváranie unit testov sa nedá použiť administrátorské rozhranie. Tento stav sa budem snažiť vylepšiť v tejto práci, tj. aby bolo možné z administrácie ku konkrétnym úlohám vygenerovať JUnit testy.

V prípade použitia jednoduchých testov je systém schopný po otestovaní odovzdanej úlohy navrhovať počet bodov.

LaMSfET nie je plnohodnotným LMS, má len veľmi úzku paletu možností zacielenú hlavne na vytváranie, uchovávanie, znovu-používanie úloh, možnosť spravovať študentov, kurzy a skupiny, odovzdávať úlohy a hodnotiť ich, prípadne vyberať študentské úlohy ako vzorové riešenia. Systému chýba takmer akákoľvek interakcia vyučujúcich so študentami, keďže systém neobsahuje žiadne diskusné fóra, chat, ani len komentáre k zostavám či úlohám od študentov. Akákoľvek takáto komunikácia s musí diať mimo systém. LaMSfET takisto nedovoľuje učiteľom vytvoriť informačnú nástenku kurzu, čo býva zvykom v LMS systémoch.

2.3 Prehlad technológie

2.3.1 JUnit

JUnit je unit testing framework pre programovací jazyk Java. Tento framework je dôležitou súčasťou test-driven development metodológie v tomto jazyku a je súčasťou unit testing frameworkov z rodiny xUnit.

JUnit sa linkuje k programu ako JAR v čase kompilácie, používa sa ako balíček (package), pre verziu JUnit 3.8 a skoršie verzie je to balíček junit.framework, pre verzie JUnit 4 a novšie je to už balíček org.junit.

JUnit vo verzii 4, používajúci balík org.junit, sa programuje ako jednoduchá trieda, ktorá nemusí byť odvodená od žiadnej inej triedy (ako tomu bolo vo verzii 3.8). Táto trieda potom obsahuje verejné metódy bez návratovej hodnoty, ktoré majú pred svojím zápisom pridanú anotáciu @Test. Takáto trieda je JUnit frameworkom chápaná ako test, a každá takáto trieda je vykonaná pri testovaní. Celá trieda obsahujúca tieto metódy je chápaná ako test fixture.

Anotácia @Test podporuje dva voliteľné parametre:

- @Test(expected=<Exception>.class), kde <Exception> je názov triedy výnimky, ktorú očakávame, že test vráti, v prípade, že test vráti inú výnimku alebo žiadnu výnimku, je považovaný za zlyhávajúci test,
- @Test(timeout=<milisekundy>), kde nastavíme časové obmedzenie behu testu, ak potom test pobeží dlhšie ako je nastavené, bude vyhlásený za zlyhaný test.

Okrem metód, ktoré sú označené anotáciou @Test a definujú jednotlivé testy, je možné vytvoriť ešte ďalšie metódy s anotáciami:

- @BeforeClass, takáto metóda bude vykonaná ako prvá, pred prvým testom,
- @AfterClass, takáto metóda bude vykonaná ako posledná, po poslednom teste,
- @Before, takáto metóda bude vykonaná pred vykonaním každého jedného testu,
- @After, takáto metóda bude vykonaná po vykonaní každého jedného testu.

Testová metóda sa skladá s programového kódu, ktorý môže iniciovať nejakú situáciu, objekt, premennú a pod., a potom s funkcií assert*, ktoré vyhodnocujú testované programové jednotky. Takéto assert* funkcie sú napríklad:

- assertTrue(boolean condition), ktorá vyhodnocuje, či condition je True,
- assertFalse(boolean condition), podobne ako predchádzajúca funkcia, táto očakáva, že condition bude False,
- assertEquals(Object expected, Object actual), ktorá vyhodnocuje rovnosť očakávanej (expected) a aktuálnej (actual) hodnoty, kde aktuálnu hodnotu vráti testovaná programová jednotka,
- assertArrayEquals(Object[] expecteds, Object[] actuals), ktorá vyhodnocuje rovnosť dvoch polí,
- atď...

Tieto assert funkcie môžu voliteľne na prvom mieste zadať identifikujúci správu ako textový reťazec, ktorá bude zobrazená ak test zlyhá.

Rôzne integrované vývojové prostredia pre Javu, ako je napríklad Eclipse, majú v sebe zabudovaný mechanizmus spúšťania a prehľadného zobrazovania výsledku testov. Ak nejaký test v tomto rozhraní nie je splnený, po dvojkliku na test sa vyznačí assert, ktorý nebol splnený.

Okrem assert* funkcií môžeme použiť aj assume* funkcie, ktoré pri zistení chyby zastavia test ale neoznačia ho ako zlyhaný, ale ako ignorovaný.

2.3.2 Smarty 3

Smarty 3 je php template system (php šablónovací systém), ktorý primárne slúži na oddelenie aplikačnej logiky od prezentačného kódu. Smarty generuje výslednú štruktúru dokumentu, prevažne html, z predlohy, ktorá obsahuje špeciálne značky pre smarty. Takéto predlohové súbory, šablóny, majú štruktúru výsledného dokumentu, napr. html, no zvyčajne majú tieto súbory príponu .tpl. Smarty spracováva tieto špeciálne značky v kóde a nahrádza ich iným kódom, typicky php kódom.

Smarty používa vlastné značky, typicky zabalené v znakoch { a }. Tieto značky môžu byť buď výrazy, ktoré sa vyhodnotia a ich hodnota sa zobrazí, priradenia, volanie funkcií a pod. Spolu takto riadia tok programu pri prezentácii dát.

Smarty podporuje medzi-pamäť výsledkov, cache, pričom umožňuje v šablónach určiť, ktoré časti budú z cache vyňaté a vždy spracované s aktuálnymi dátami.

Smarty pri procese zobrazovania šablón kontroluje, či existuje platný cache súbor pre želanú šablónu. Ak existuje, bude zobrazený jeho obsah, v opačnom prípade smarty nájde požadovanú šablónu a vygeneruje z nej cache súbor, ktorý následne zobrazí. Toto platí v prípade, že je zapnuté používane cache, ale ak nie je, tak sa vždy generuje výstup so šablóny a dát jej

poskytnutej. V každom prípade, ak Smarty zistí, že šablóna ešte nie je skompilovaná, prevedie jej obsah na ekvivalentný obsah s tým, že všetky Smarty značky budú vymenené za php kód, ktorý vykoná ich funkcionalitu.

V procese vývoja aplikácie je Smarty väčšinou nastavené tak, aby v každom prípade kontroloval rozdiel šablón a ich skompilovaných variant a prekompilovával ich podla potreby. Táto funkcionalita Smarty je však spomalujúca, preto sa neodporúča aby bola zapnutá v produkčnom prostredí.

Smarty dovoľuje v šablónach robiť viacero pokročilých vecí, ako je:

- volanie iného súboru so šablónou, pomocou značky {include}, táto dokonca vytvára nový scope pre premenné,
- vytvárať si pomocné premenné pomocou značky {assign} alebo pomocou výrazu {$premenna = hodnota}, druhý spôsob dovoľuje priraďovať hodnoty aj do polí alebo vlastností objektov,
- rozširovať šablóny, pomocou značky {extends}, táto značka sa musí zapísať ako prvá v súbore a určuje iný súbor so šablónou, ktorý obsahuje pomenované bloky značkami {block}{/block}, súbor, ktorý túto šablónu rozširuje potom obsahuje rovnako pomenované {block} {/block} značky, týmto sa dajú vytvoriť layouty pre šablóny,
- definovať v šablóne funkcie, pomocou značky {function}, ktoré sa dajú neskôr volať, tieto funkcie môžu byť rekurzívne, platí, že musia byť vždy definované pred použitím, sú nenahraditeľné pri generovaní navigácií a pod. (keďže navigácia je väčšinou stromová štruktúra),
- dovoľuje aplikovať modifikátory na zobrazovaný obsah premenných, napríklad {$studenti.email|default:"Študent nezadal svoj e-mail."}, tieto modifikátory môžu byť jak plugin Smarty, tak aj php funkcie, ktoré na prvom mieste dostávajú vstup, na ktorom majú vykonať nejakú modifikáciu podla parametrov na druhom, treťom a ďalšom mieste, ak je to potrebné, dá sa modifikátor aplikovať aj na výsledok funkcie či textový reťazec, modifikátory sa dajú skladať,
- zobraziť prehľad dát, ktoré šablóna aktuálne má k sebe pripojené a ďalších informácií, na ktoromkoľvek mieste, pomocou značky {debug}.

Samotný objekt Smarty nám dovoľuje robiť napríklad tieto veci:

- priraďovať, dopĺňať či odstraňovať dáta pripojené k šablóne,
- kontrolovať cache pre šablónové súbory, mazať cache,
- dynamicky registrovať modifikátory, funkcie, bloky alebo značky kompilátora (ktoré do skompilovanej šablóny vkladajú obslužný php kód),
- dynamicky odregistrovať spomenuté pluginy,
- zobraziť spracovanú šablónu na výstupe alebo vrátiť celý výsledok spracovanej šablóny do premennej.

2.3.3 TinyMCE

TinyMCE alebo Tiny Moxiecode Content Editor je open sourcemultiplatformový webovo-orientovaný WYSIWYG editor, vytvorený v jazyku JavaScritp/HTML vyvýjany spoločnosťou Moxiecode Systems AB. Tento softvérový balík je schopný konvertovať HTML elementy textarea na plnohodnotný textový editor. Samotné TinyMCE je vytvorené tak, aby bolo ľahko nasaditeľné do CMS systémov, ako je Drupal, Joomla! či WordPress, ale aj do akýchkoľvek iných webových aplikácií, ktorých autor sa rozhodne toto riešenie použiť.

Editor podporuje štandardné formátovacie nástroje vo webovom prostredí, ako je hrúbka písma, sklon písma, podčiarkovanie, zoraďované a nezoraďované zoznamy, rozloženie textu, vkladanie obrázkov alebo videí, odkazov, nezalomiteľných medzier, kopírovanie, vystrihovanie a vkladanie textu a pod.

Editor podporuje používanie rôznych skinov, na prispôsobenie vzhľadu editora vlastnej aplikácii, prípadne dovoľuje upravovať samotné UI editora, pridávať a odoberať funkčné tlačidlá alebo vytvárať vlastné tlačidlá a ich funkcionalitu, či vytvoriť totálnu konverziu, kedy je možné celý interface editora prepracovať podla vlastných potrieb.

Editor navyše ponúka API na vytváranie vlastných pluginov a vývojári TinyMCE sami ponúkajú niekoľko prepracovaných pluginov na prácu so súbormi alebo obrázkami, tieto sú však komerčné.

Editor je do stránky možné integrovať pomocou obyčajného JavaScriptu, ale aj pomocou knižnice JQuery, pre ktorý ma vytvorené pluginy na integráciu.

2.3.4 JQuery / JQueryUI

JQuery je knižnica pre JavaScript navrhnutá tak, aby zjednodušovala skriptovanie HTML stránok na klientskej strane. Jej motto, píš menej, urob viac, vystihuje silu tohto nástroja. JQuery je open source projektom, pôvodným autorom je John Resig.

Syntax JQuery je navrhnutá tak, aby zjednodušovala navigáciu v dokumente, výber DOM elementov, vytváranie animácií, obsluhu udalostí a vytváranie Ajax aplikácií. Rovnako vývojárom umožňuje písať vlastné pluginy a teda im dovoľuje vytvárať abstrakciu nízko-úrovňovej interakcie a animácií.

Niektoré korporácie, ako je Microsoft, zabudovávajú JQuery do svojich produktov, konkrétne Microsoft pridal JQuery do Visual Studia pre ASP.NET AJAX framework a ASP.NET MVC framework.

JQuery teda prináša tieto výhody:

- výber DOM elementov pomocou multi-browser systému pre selektory Sizzle,
- traverzovanie a modifikovanie DOM s podporou pre CSS 1 až 3,
- obsluha udalostí,

- efekty a animácie,
- AJAX,
- rozšíriteľnosť pluginmi,
- multi-browser podpora.

JQueryUI je JavaScript knižnica, ktorá poskytuje abstrakciu pre nízko-úrovňovú interakciu a animácie, pokročilé efekty a vysoko-úrovňové widgety, ktorá je postavená na knižnici JQuery. Používa sa na vytvorenie interaktívnych webových aplikácií a ponúka celý rad widgetov a efektov:

- akordeón – harmonikové prepínanie obsahu so zachovaním výšky,
- automatické dokončovanie vkladania textu,
- tlačidlá, klasické aj prepínateľné, možno doplniť o ikony,
- kalendár na výber dátumu,
- dialógové okná,
- viac-úrovňové menu,
- progressbar,
- posuvníky s jazdcom,
- záložky,
- nástrojové tipy,
- a rôzne ďalšie widgety, efekty a pod.

2.3.5 CodeIgniter

CodeIgniter je open source framework pre PHP webové aplikácie založený na architektúre Model-View-Controller MVC. Poskytuje množstvo knižníc, ktoré pomáhajú rýchlejšie vytvárať dynamické webové aplikácie. Jadro frameworku je založené na PHP verzii 4.1, čo však nebráni písať v ňom aplikácie využívajúce výhody PHP 5 či framework rozširovať o podporu tejto verzie PHP.

Framework má oddelené jadro systému s všetkými dodávanými knižnicami, helpermi, databázovými adaptérmi a jazykovými súbormi od samotnej aplikácie. Typicky v koreňovom adresári aplikácie postavenej na CodeIgniter sa nachádza adresár system (so súbormi jadra, knižnicami, atď) a adresár application (obsahujúci radiče, pohľady, modely, konfiguračné súbory, vlastné knižnice a pod.). Súbor index.php v koreňovom adresári je potom jediný vstupný bod pre aplikáciu, ktorý spúšťa jadro, a to na základe konfigurácie routeru obsluhuje užívateľské dopyty volaním príslušných radičov.

Vďaka architektúre MVC sa aplikácia napísaná v tomto frameworku skladá s troch hlavných na sebe nezávislých komponentov:

- radič (controller) je komponent, ktorý prijíma dopyty, spracúva ich a vráti odpoveď,
- model je komponent, ktorý obsluhuje dáta, bežne v databázy ale aj v

súborovom systéme,

- pohľad (view) je komponent, ktorý sa stará o prezentáciu dát používateľovi.

CodeIgniter vývojárom prináša podporu pre viaceré databázy, ako sú MySQL(i), PostgreSQL, MsSQL, SQLite, Oracle, Cubrid a PDO. Použitím ActiveRecord je možné aplikáciu prevádzkovať na ktorejkoľvek podporovanej databáze. Ďalej prináša knižnice na prácu so súbormi, formulármi, obrázkami, sessions, xml-rpc, zip, jazyky či jednoduchý template parser.

2.3.6 GeSHi

GeSHi alebo Generic Syntax Highlighter je free softvér umožňujúci zvýrazňovanie syntaxe zdrojového kódu veľkého množstva programovacích či značkových jazykov. GeSHi je napísaný v PHP, dostupný je samostatne či ako add-on pre rôzne webové aplikácie, ako MediaWiki, phpBB, Mambo atď.

GeSHi podporuje zvýrazňovanie syntaxe pre cca. 200 programovacích a značkových jazykov, pričom môže byť rozšírený o ďalšie jazyky dodaním vlastných definícií. Zdrojový kód, ktorý je spracovaný GeSHi je prevedený na html a zvýraznený pomocou kaskádových štýlov, výstup je kompatibilný s XHTML 1.1 a CSS úrovne 2. Zaujímavou funkcionalitou je aj generovanie liniek v zvýraznenom kóde na dokumentáciu pri niektorých programovacích jazykoch.

2.4 Zhodnotenie východiskovej práce

Počas písania tejto podkapitoly som analyzoval zdrojové kódy pôvodného systému LaMSfET, ako aj návrh fungovania systému.

Peter Jurčo, autor pôvodného systému, mal peknú myšlienku urobiť systém modulárny, čo prispieva k jeho flexibilite a mal by sa dať takto jednoduchšie upravovať. Keď som však dostal od môjho školiteľa pôvodné zdrojové kódy a databázovú štruktúru, rýchlo som zistil, že tu nie je všetko s kostolným poriadkom.

Po niekoľkých hodinách trápenia sa z rozbehnutím systému, čoho príčina bola množstvo chýb vrátených php procesorom a rovnako tak snaženie sa opraviť tieto chyby v častiach kódu, ktoré síce boli spúšťané, no nemali žiaden dopad na funkčnosť, som zistil, že celé fungovanie systému sa musí kompletne nastaviť v databáze, každá stránka, každý modul. S prázdnou databázou to však bol problém.

Systém je teda rozdelený do v databáze vytvorenej mapy stránok a k nim náležitých boxov (v podstate pod-modulov), tieto stránky majú jak ID číslo, tak aj krátky unikátny názov. Zaujímavé však je, že systém má v sebe hard-kódované názvy stránok, ktoré vynucuje na zobrazenie, ako je napr. login, task_list. Nikde nie je konfiguračná položka, ktorá by určovala východziu stránku ani stránku pre prihlásenie. Systém je natoľko modulárny, že s prázdnou databázou používateľ dostane len nezrozumiteľné chybové hlásenie a viac nič.

Situácia s databázou sa však vyriešila a po vytvorení účtu učiteľa / administrátora, som sa dostal do systému. Tak ako počas snahy systém rozbehnúť na mojom vývojovom webserveri (Apache 2.2, PHP 5.4.4), zobrazovalo množstvo chybových hlásení úrovne notice prípadne úrovne strict standard, ktoré efektívne znemožňujú prácu s veľkým množstvom funkcií systému a tie funkcie, ktoré akoby zázrakom ostávajú funkčné, sú týmito chybami rovnako negatívne postihnuté. Z tohto usudzujem, že autor systému bol v tom čase nováčikom v PHP a celý systém vytváral v produkčnom nastavení systému (žiadne chybové hlásenia). Jeho server a rovnako tak aj server capek.ii.fmph.uniba.sk, kde je systém momentálne nasadený, sú otrocky nastavené tak, aby tento veľmi zle naprogramovaný systém dokázali spustiť.

Mne teraz ostávajú tri možnosti:

1. otrocky nastaviť vlastný server tak, aby bol schopný spustiť LaMSfET bez chýb,
2. prejsť niekoľko tisíc riadkov systému LaMSfET a opraviť všetky alebo aspoň väčšinu chýb, na ktoré budem schopný naraziť,
3. pôvodný systém nechať tak a vytvoriť systém úplne nový, ktorý bude zohľadňovať funkcionalitu pôvodného systému.

Bod 2 určite efektívne zoberie veľký kus časového intervalu zostávajúceho na akékoľvek vlastné implementácie nových funkcií, bod 3 je na tom podobne ako bod 2, no zasa dáva priestor vytvoriť systém od základu, bez amatérskych chýb predchodcu programátora a rovno doň novú funkcionalitu zakomponovať. Pokračovať bodom 1 znamená na zlom stavať, čo sa môže len zle skončiť.

Celkovo môžem tento systém zhodnotiť ako vynikajúcu ideu skutočne amatérsky prevedenú do praxe. Za moju niekoľkoročnú prax programátora PHP som ešte nevidel horšie napísaný softvérový kód, ako je tento. Kódu samotnému chýba miestami štruktúra, sú v ňom použité nebezpečné konštrukcie, kód sa spolieha na automatickú registráciu premenných, často sa v ňom používajú neexistujúce premenné vplyvom preklepu autora kódu, funkcie a metódy často nemajú dokumentačné komentáre zatiaľ čo vnútri funkčných blokov sú zbytočné komentáre písane dvojjazyčne. Najhoršie, čo som v kóde zatiaľ našiel bolo posielanie SQL dotazov cez GET/POST metódy asynchrónnymi volaniami, alebo riešenie uploadu súborov pomocou platformovo závislých volaní funkcií z príkazového riadku.

3 Použitá literatúra

[1] Peter Jurčo: Dlhodobý viacúčelový sklad úloh na cvičenia, 2009, bakalárska práca, FMFI UK Bratislava

[2] JUnit framework api, junit.sourceforge.net, dosupné na:

http://junit.sourceforge.net/javadoc/

[3] Smarty template engine, www.smarty.net, dostupné na: http://www.smarty.net/docs/en/

[4] TinyMCE, Moxiecode Systems AB, dostupné na: http://www.tinymce.com/

[5] Moodle LMS, www.moodle.org, dostupné na: https://moodle.org/

[6] Chamilo LMS, Chamilo Association, dostupné na: http://www.chamilo.org/

[7] JQuery, JQuery team, dostupné na: http://www.jquery.com/

[8] JQueryUI, JQuery team, dostupné na: http://www.jqueryui.com/

[9] CodeIgniter, EllisLab Inc., dostupné na: http://ellislab.com/codeigniter

[10] GeSHi, Nigel McNie a Benny Baumann, dostupné na: http://qbnz.com/highlighter/index.php