VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

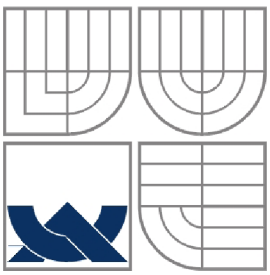
SYSTÉM PRO SPRÁVU JAZYKOVÝCH VERZÍ PORTÁLU VUT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

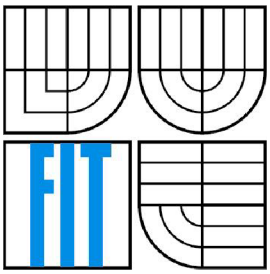
AUTOR PRÁCE
AUTHOR

MILAN PAVLÍČEK

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO SPRÁVU JAZYKOVÝCH VERZÍ PORTÁLU VUT

LANGUAGE VERSION TOOLS FOR WEB PORTAL OF BUT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

MILAN PAVLÍČEK

VEDOUCÍ PRÁCE
SUPERVISOR

JAROMÍR MARUŠINEC, Ing., Ph.D.

BRNO 2007

Zadání:

Systém pro správu jazykových verzí Portálu VUT

Language version tools for web portal of BUT

Vedoucí:

Marušinec Jaromír, Ing., Ph.D., CVIS VUT

Oponent:

Burget Radek, Ing., Ph.D., UIFS FIT VUT

Přihlášen:

Pavlíček Milan

Zadání:

1. Prostudujte problematiku vytváření vícejazyčných webových aplikací.
2. Prostudujte současný stav Portálu VUT a vývoje nových modulů s ohledem na práci s jednotlivými jazykovými verzemi.
3. Analyzujte, vývojáři kladené požadavky, na systém práce s jazykovými verzemi v současném vývojovém modelu.
4. Realizujte nejdůležitější části vyplývající z analýzy (včetně dokumentace pro překladatelky a vývojáře).
5. Integrujte nové řešení do informačního systému VUT.
6. Zhodnoťte výsledky vaší práce a navrhněte další postup implementace.

Část požadovaná pro obhajobu SP:

Bez požadavků.

Kategorie:

Web

Implementační jazyk:

PHP, SQL, Perl

Komerční software:

Oracle9i, 10g

Volně šířený software:

PHP, Perl

Literatura:

- Literatura Oracle
- Dokumentace k jazyku PHP
- Dokumentace k jazyku Perl

Licenční smlouva

Licenční smlouva je uložena v archívu Fakulty Informačních Technologií Vysokého Učení Technického v Brně.

Abstrakt

Hlavním cílem této práce je vytvořit systém pro správu jazykových verzí Portálu VUT. Nejprve popisuji problematiku tvorby vícejazyčných webových aplikací. Možnosti jejich implementace, využívající databázi i bez ní. Další částí je analýza současného stavu Portálu VUT. Popisuji jednotlivé servery, nástroje a hlavně řešení zajišťující vícejazyčnost. Dále se věnuji návrhu a implementaci požadovaného systému. Ten se skládá z několika samostatných skriptů a webové aplikace pro vývojáře a překladatele. Nakonec uvádím postup integrace tohoto systému do stávajícího Portálu VUT.

Klíčová slova

Jazykové verze, SVN, revize, parser, regulární výraz, HTML, Portál VUT

Abstract

The main purpose of this master's thesis is to create language version tools for web Portal of BUT. There are possibilities of applications using databases or without them. The next part of the thesis analyses current situation of the web Portal of BUT. I described individually servers, tools and mainly solution ensuring multilingual sites. Thereinafter I attend to design and implementation of required system. This system consists of some single scripts and web applications for web developers and translators. Finally I will describe the procedure of integration of this system to current web Portal of BUT.

Keywords

Language versions, SVN, revision, parser, regular expression, HTML, web Portal of BUT

Citace

Pavlíček Milan: Systém pro správu jazykových verzí Portálu VUT. Brno, 2007, diplomová práce, FIT VUT v Brně.

Systém pro správu jazykových verzí Portálu VUT

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Jaromíra Marušince, Ph.D. Další informace mi poskytli zaměstnanci CVIS VUT v Brně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Milan Pavlíček
31. 6. 2007

Poděkování

Tímto bych chtěl poděkovat panu Ing. Jaromírovi Marušinci, Ph.D. za umožnění zpracování diplomové práce na CVIS a hlavně Ing. Michalovi Juroszovi za užitečné podněty a rady při práci.

Obsah

Obsah 1
1 Úvod 3
2 Tvorba webů obecně 4
2.1 Apache HTTP Server 4
2.2 Jazyk HTML 4
2.3 Jazyk PHP 5
2.4 Databáze 5
2.4.1 Relační databáze 6
2.4.2 SQL 6
3 Tvorba vícejazyčných webů 8
3.1 Požadavky na vícejazyčnou podporu 8
3.2 Informační obsah uvnitř stránek 9
3.3 Informační obsah mimo stránky 11
3.3.1 Použití vlastního datového souboru a překládací funkce 11
3.3.2 Použití knihovny GNU Gettext 13
3.3.3 Použití šablonovacího systému s podporou slovníků 15
3.4 Aplikace využívající databázi 15
3.4.1 Možnosti návrhu DB 15
3.4.2 Balík Translation2 z knihovny PEAR 16
4 Portál VUT 18
4.1 Struktura serverů 18
4.2 Databáze 19
4.2.1 Oracle 19
4.2.2 Standard ODBC 19
4.3 SVN 21
4.4 Analýza webu 23
4.4.1 Struktura 23
4.4.2 Instance 24
4.4.3 Databáze 25
4.5 Analýza řešení vícejazyčnosti 27
4.5.1 Funkce pro překlad a datový soubor 27
4.5.2 Překladatelé 29
5 Návrh systému 30
5.1 Popis celého systému 30

5.2 Aplikace pro vývojáře a překladatele 31
5.2.1 Neformální specifikace 31
5.2.2 Diagram případů použití aplikace 32
5.2.3 E-R diagram 33
5.3 Návrh databáze 33
5.3.1 Tabulka uživatel 33
5.3.2 Tabulka aplikace 34
5.3.3 Tabulka soubor 34
5.3.4 Tabulka text 35
5.3.5 Tabulka textvsouboru 36
5.4 Sestavovací skript 37
5.5 Skript spouštěný po SVN operaci commit 37
6 Realizace 38
6.1 Regulární výrazy v PHP 38
6.2 Předávání hodnot mezi skripty 40
6.3 Webová aplikace 41
6.3.1 Soubor login.php 41
6.3.2 Soubor index.php 42
6.3.3 Popis vlastní aplikace 43
6.4 Sestavovací skript 46
6.5 Skript volaný po SVN operaci commit 49
7 Integrace 50
8 Závěr 52
Literatura a internetové zdroje 55

1 Úvod

Síť Internet je globální medium, které nám zpřístupňuje informace a služby téměř z celého světa. Jelikož však všichni lidé na Zemi nemluví stejným jazykem, nelze jej plně využít. Již od pradávna tlumočníci spojovali národy. Stejně tak i dnešní tvůrci internetových stránek mnohdy využívají jejich služeb, aby tím rozšířili skupinu svých uživatelů.

K překladu webových stránek či internetových aplikací nás mohou motivovat i jiné okolnosti. Jde například o zákony, ambice firmy podnikat na zahraničních trzích, atd. Daná problematika se však stále více dotýká i oblasti školství. Na Vysokém Učení Technickém v Brně studuje nebo má v plánu studovat i velké množství zahraničních studentů. Také řada dalších osob ze zahraničí se zajímá o tuto instituci. Proto i VUT se musí touto otázkou dále zabývat.

Internacionalizace, někdy označovaná numerickou zkratkou i18n z anglického internationalization, znamená přizpůsobení produktu, tedy i webové aplikace pro prostředí mimo to, ve kterém, či pro které byl produkt vyvinut. Nejčastěji máme na mysli pro jiný národ. Lokalizace, někdy označována l10n podle localization, v obecném pojetí znamená určení polohy, umístění objektu. V našem případě ji bereme ve významu překladu software z původního jazyka do jazyka místního. Rozdíl mezi těmito pojmy dle [13] je, že internacionalizace představuje adaptaci produktu pro potenciální použití kdekoliv, zatímco lokalizace je přidání speciálních vlastností pro použití v určitém prostředí. Tyto procesy se navzájem doplňují a musí být kombinovány pro dosažení možnosti globálního použití aplikace.

Spolu s překladem souvisí implementace. Je zapotřebí rozhodnout, jak budou přeložené informace, většinou texty uloženy a jak s nimi bude pracováno. Jaké jsou možnosti implementace vícejazyčných webových aplikací rozeberu v první části této diplomové práce.

Dále se věnuji současnému stavu Portálu VUT, jak jsou překlady implementovány dnes, v kapitole jsou analyzované funkce pro překlad a uložení textů.

Další částí práce je realizace vlastního systému na základě analýzy z předešlé kapitoly, je také nutné vytvořit dokumentaci pro vývojáře a překladatele.

V poslední části se věnuji zhodnocení práce a návrhu dalšího postupu či vylepšení.

Diplomová práce nenavazuje na Ročníkový ani Semestrální projekt, které jsem zpracovával na Fakultě Informačních Technologií. V ročníkovém projektu jsem se zabýval zabezpečením v bezdrátových sítích a realizací prolomení klíče u standardu WEP. V Semestrálním projektu jsem se také věnoval bezdrátovým sítím, ale zabezpečení pomocí WPA, které je jak se ukázalo mnohem bezpečnější. Nicméně právě nedostatky ve standardu WEP vedly ke vzniku WPA.

2 Tvorba webů obecně

Pro tvorbu webových stránek existuje v současnosti celá řada jazyků a prostředků. V této kapitole nastíním ty, které jsem pro tuto práci využil.

2.1 Apache HTTP Server

Apache HTTP Server je Open Source webový server, který je nejen kvůli své cenové dostupnosti nejčastěji používaným web serverem na světě. Tento projekt vznikl v roce 1993 na univerzitě v Illinois pod názvem NCSA HTTPd, kde byl ale po pěti letech zastaven. Nicméně už v průběhu tohoto vývoje byl projekt rozšířen mezi správce webových serverů, kteří si pro něj sami začali tvořit vlastní záplaty. Později vznikla webová konference pro koordinování těchto záplat. Odtud také vznikl název serveru (a patch = záplata).

Již rok po svém oficiálním uvedení (1995,1996) se v nasazení Apache vyrovnal dříve nejpoužívanějšímu web serveru od firmy Sun. Od té doby, je nepřetržitě na první pozici v uveřejňovaných seznamech [19]. V žebříčku z průzkumu z července 2007, je Apache nasazen na 49,98% aktivních stránek celého světa.

Distribuce jsou k dispozici pro velkou škálu platforem jako jsou Windows, Linux, atd.. Pro studium tohoto webového serveru je vhodné využít přímo jeho dokumentaci [5], či jednu z mnoha publikací , které se instalaci a správě tohoto serveru věnují [20].

2.2 Jazyk HTML

Jazyk HTML (Hypertext Markup Language = značkovací jazyk pro hypertextové dokumenty) je jazyk který se používá pro tvorbu internetových stránek. Tento jazyk má svoji přesnou syntaxi, její zvláštností je, že je sice přesně definovaná, ale zároveň velmi přizpůsobivá. I když jsou v dokumenty podstatné syntaktické chyby, nesplňující pravidla HTML, nebrání to zobrazení a prohlížení tohoto dokumentu. V takovém případě není samozřejmě zaručena správnost dokumentu. HTML je textového formátu a je v tomto formátu používán. To značí, že není nikde kompilován do binární nebo jakékoliv jiné podoby. V důsledku to znamená, že jakmile vytvoříme dokument v některém z textových editorů, je to jeho finální tvar, který dokáže prohlížeč načíst a interpretovat.

Všechny příkazy jazyka jsou tagy uzavřené do špičatých závorek Některé tagy jsou párové. Vše co je mimo špičaté závorky je vlastní obsah stránky.Příklad jednoduchého dokumentu HTML je v uveden v Kód 1.

```
<html>
   <head>
      <title>Titulek stránky</title>
   </head>
   <body>
      <h1>Nadpis stránky</h1>
      <p>Toto je tělo dokumentu</p>
   </body>
</html>
```

Kód 1: Příklad dokumentu HTML

2.3 Jazyk PHP

Původním tvůrcem z roku 1994 je Rasmus Lendorf, když vytvořil jednoduchý systém pro evidenci přístupu ke svým stránkám. Ten dostal název Personal Home Page Tools. Celosvětovou proslulost si získal až systém PHP/FI 2.0. Nyní PHP zaujímá významné postavení při tvorbě webových stránek. Jedná se o tzv. server side skriptovací jazyk - je zabudován na straně serveru. To má za následek, že pracuje uvnitř dokumentu HTML a umožňuje dynamicky měnit vzhled a obsah tohoto dokumentu. Prohlížeč následně zobrazuje právě výstup PHP skriptu.

Jádro jazyka PHP má velmi dobře optimalizovanou dobu odezvy potřebnou pro webové aplikace. Z těchto předpokladů vychází jeho jednoduchost, přirozený způsob práce se stále se zvyšujícím počtem databázových serverů a také jedna z důležitých předností - nezávislost na platformě. Velmi často se používá ve spojení s Apache Serverem a databází MySQL. V případě nasazení na operačním systému Linux se vžilo označení LAMP, pro platformu Microsoft Windows označení WAMP. Jazyk je od začátku vyvíjen jako open-source (software s otevřeným zdrojovým kódem).

Vlastní jazyk se skládá z operátorů, obvyklých doplňků řídících struktur, druhů proměnných, deklarací funkcí a deklarací tříd a objektů.

Samotné provádění PHP je možné ve dvou variantách a to buď jako interpret CGI skriptů nebo jako modul webového serveru. První možnost spouštění jako CGI skript. Interpret se musí spouštět při každém požadavku, což u větších systémů vede ke snížení výkonu webového serveru. Druhá možnost je zkompilovat PHP jako modul k webovému serveru tak, že běží jako jeden z jeho procesů, což má za následek mnohem větší výkon než varianta s CGI. V dnešní době máme k dispozici velkou škálu modulů a knihoven pro tento jazyk, které je možné získat přes Internet. Tímto je vlastní programování nových systémů značně jednodušší. Více viz.[1].

2.4 Databáze

Lidé již od svého počátku zaznamenávají nějaká data. Ať už to byly výjevy z lovů vyobrazené na stěnách jeskyní, kroniky středověku, či záznamy vývoje akcií na burze. Objem ukládaných dat se

neustále zvětšuje. Proto vznikla nutnost ukládat související data o velkých objemech strukturovaně na jednom místě. Dříve se používaly papírové kartotéky, nyní však používáme databáze – oproti papírové verzi jsou data uložena na elektronickém datovém médiu.

2.4.1 Relační databáze

To jak jsou data strukturována a jakým způsobem k nim přistupujeme, což určuje daný databázový systém, označujeme jako datový model. Ten upřesňuje charakter jak databázových řídících systémů, tak i aplikací pro něž je zvláště vhodný. Relační databázový model, který je dnes nejrozšířenější, redukuje a zjednodušuje starší datové modely. V relační databázi je základním organizačním prvkem tabulka. Veškeré operace pracují s těmito tabulkami.

Tyto tabulky jsou tvořeny řádky a sloupci. Řádky odpovídají jednotlivým záznamům, označujeme je jako datové věty. Sloupce odpovídají atributům, jednotlivým prvkům těchto datových vět. To znamená, že všechny datové věty v jedné tabulce mají stejnou strukturu atributů danou strukturou tabulky. Každý řádek v tabulce obsahuje v každém sloupci přesně datovou hodnotu. Konkrétní implementace datových typů z hlediska rozsahu hodnot, používaného označení či vůbec existence daného datového typu závisí na konkrétním databázovém systému (SŘBD), ale prakticky na všech databázových systémech máme k dispozici celá a reálná čísla, řetězce, datum booleovské hodnoty, vlastní výčtové typy a obvykle typy pro rozsáhlá textová či binární data. Tyto jsou využitelné pro uložení textů, viz dále.

V rámci celé tabulky mají všechny hodnoty dat v jednom sloupci stejný datový typ. V jedné tabulce musí mít každý sloupec jiný název kvůli identifikaci, stejně tak musí být v rámci jedné databáze unikátní jména pro jednotlivé tabulky. V různých databázových systémech může nebo nemusí být omezen počet řádků v tabulce. V relační databázi má každá tabulka daný sloupec nebo kombinaci sloupců, jejíž hodnoty jednoznačně identifikují řádek v tabulce. Toto označujeme jako primární klíč tabulky.

Primární klíč obsahuje jedinečnou hodnotu pro každý řádek, proto žádné dva řádky tabulky s primárním klíčem nejsou přesným duplikátem jiného řádku. Sloupec, jehož hodnota v jedné tabulce odpovídá hodnotě primárního klíče v jiné tabulce nazýváme cizí klíč. Primární klíč a cizí klíč společně vytváří relaci typu rodič - potomek mezi tabulkami, které je obsahují. Pokud je tabulka spojena s více tabulkami může obsahovat více cizích klíčů. Více viz [22].

2.4.2 SQL

Historie jazyka SQL spadá do rozmezí 70. a 80. let. V letech 1974 až 1975 probíhal ve firmě IBM výzkum týkající se využití relačních databází. Pro tento projekt byla vytvořena sada příkazů, kterými se relační databáze ovládala. Tato sada dostala název SEQUEL (Structured English Query Language).

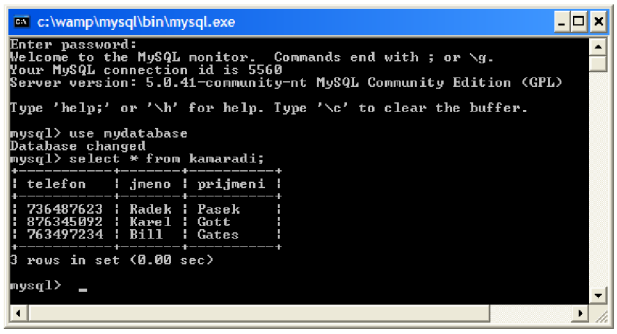
V databázových systémech jiných firem se používaly různé verze jazyka SEQUEL, který se přejmenoval na SQL (Structured Query Language).

Jelikož začal vzrůstat význam relačních databází, vyžádal si standardizaci jazyka. První standard byl přijat v roce 1986 a byl pojmenován SQL86. Časem se však projevily některé nedostatky. Opravená verze jazyka SQL je z roku 1992 a nese název SQL92 . Ten je v oblasti relačních databází používán a platný dodnes. Zatím nejnovějším standardem je SQL3 (SQL-99), který reaguje na potřeby nejmodernějších databází s objektovými prvky.Více viz.[22].

Dělení jazyka SQL

Jazyk v sobě zahrnuje nástroje pro tvorbu databází (tabulek) a dále nástroje na manipulaci s daty (vkládání dat, aktualizace, mazání a vyhledávání informací). SQL patří do kategorie tzv. deklarativních programovacích jazyků, což v praxi znamená, že kód jazyka SQL nepíšeme v žádném samostatném programu (jako např. u jazyka C, nebo Pascal), ale vkládáme jej do jiného programovacího jazyka, který je již procedurální.

Se samotným jazykem SQL můžeme pracovat pouze v případě, že se terminálem připojíme na SQL server a na příkazový řádek zadáváme přímo příkazy jazyka SQL jak je zobrazeno na Obr.1.



Obr.1: SQL konzole

Samotný jazyk je složen z několika částí:

- **DDL** - První částí jazyka SQL je Data Definition Language.Tento se používá pro vytváření databázových schémat a katalogů.
- **SDL** - Další část zastupuje Storage Definition Language. Definuje způsob ukládání tabulek.
- **VDL** - View Definition Language, určující vytváření pohledů. Tyto si lze představit jako virtuální tabulky složené z různých jiných tabulek. Je určen převážně pro návrháře a správce.
- **DML** - Poslední je Data Manipulation Language. S ním pracují nejvíce koncoví uživatelé a programátoři databázových aplikací. Obsahuje základní příkazy SELECT, INSERT, UPDATE a DELETE.

3 Tvorba vícejazyčných webů

V této kapitole nejdříve proberu obecně problematiku vícejazyčných webových aplikací, dále požadavky na ně a následně přejdu k jednotlivým možnostem implementace.

Běžně přicházíme do styku s mnoha typy internetových webových aplikací. Ty mohou být různého druhu a rozsahu. Jako například osobní statické informativní stránky s pár texty, fotografiemi a dokumenty, internetový obchod s dlouhodobými texty – informace o postupu nákupu a neustále se měnícími položkami a jejich cenami. Nebo stále se rozvíjející internetový portál velké instituce s mnoha moduly, velkým množstvím dat a počtem uživatelů. A právě, jak každá jednotlivá aplikace využívá jinou formu implementace, také implementace vícejazyčné podpory může, nebo spíše musí být provedena ke svému použití odpovídajícím způsobem. Možnosti implementace spolu s příklady a jejich vhodným použitím, jsou hlavní náplní této kapitoly.

Ukázky kódů jsou v jazyce PHP, případě použití databáze je využita MySQL.

3.1 Požadavky na vícejazyčnou podporu

Na každé programové dílo či jeho část, tedy i vícejazyčnou podporu internetové aplikace, jsou kladeny různé požadavky plynoucí z toho, jak bude program využit. Hlavní požadavky na vícejazyčný web jsou tyto:

Neomezená množina jazyků

Na světě existuje obrovské množství jazyků a národních abeced, jejich rozdílnost nezná mezí. Dříve se pro každou národní abecedu používalo více kódování (Kameničtí, Windows 1250, PC Latin 2), což s sebou neslo problém spolupráce více aplikací, či různých OS. Použití nebylo standardizováno. Pokud bychom navíc tvořili vícejazyčný web za takových podmínek, je velmi pravděpodobné, že uložené informace nebudou správně zobrazeny. Abychom se tomuto vyhnuli, využijeme Standard Unicode [4], který bez ohledu na program, platformu či jazyk znak identifikuje jednoznačně v rámci celého světa. A to tak, že znakům přiřazuje čísla až do 65 tisíc (216 b). Znaková sada Unicode se dá zapisovat různými způsoby:

- **UTF-8** píše anglické a programátorské značky jedním bajtem (znakem), ostatní světová písmenka dvěma bajty.
- **UTF-16** píše všechny značky a písmenka dvěma bajty.

Meta html tag pro nastavení UTF-8 kódování je:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

Díky tomu, že je tento standard velmi rozšířen a podporován širokou řadou firem a organizací, mimo jiné i těch, zabývajících se tvorbou DB či prohlížečů webových stránek, máme problém s nasazení více jazyků na jednom serveru nebo jedné DB vyřešen. Musíme pouze zkontrolovat, zda toto kódování naše prostředky podporují, což by ale v dnešní době již neměl být problém.

Konzistence a duplicita dat

Při každém uchovávání informací požadujeme, aby data byla po celou dobu, kdy je potřebujeme využívat, konzistentní. Tedy soudržná, platná, aktuální. Nežádoucí je také mít informace uloženy vícekrát. U vícejazyčných webů předpokládáme minimálně trojici:

```
((identifikátor_textu),(text_v_jazyce_A),(text_v_jazyce_B))
```

Pokud dojde ke změně jednoho z textů, je zapotřebí zabezpečit korekturu textu v jazyce druhém.

Snadná údržba

Při tvorbě systému musíme už dopředu předpokládat, že jednou dojde k jeho rozšíření nebo jiné změně. U vícejazyčné aplikace nás může potkat nutnost rozšíření o další jazyky.

Nízké operační nároky

Tímto máme na mysli nízké nároky na výpočetní výkon a paměťový prostor. V každém počítačovém systému je potřeba tyto požadavky brát na zřetel.

3.2 Informační obsah uvnitř stránek

Zde se věnuji aplikacím, které nepoužívají datový soubor ani databázi pro uložení textových informací. Toto řešení je vhodné u jednoduchých stránek, které mají například pouze informativní poslání a nejsou obměňovány velmi často.

V tomto případě máme pro každou stránku (soubor s jejím zdrojovým kódem) množinu jejich mutací. Soubory potom rozlišujeme buď pomocí jména, rozšířeného např. o postfix určující jazyk, či podle umístěním v odlišném adresáři. Jak si označíme jazyk je zcela na nás, ale i zde bychom se měli držet standardizovaného označení jazyků ISO 639 [7], jehož výběr je v Tab. 1.

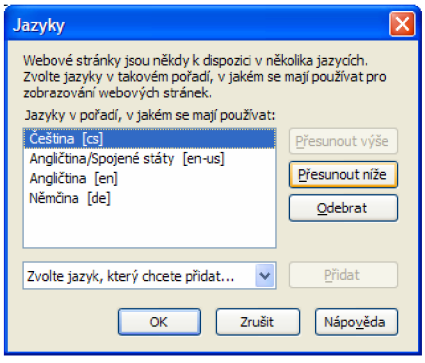
Kód	Jazyk	Kód	Jazyk
CS	Sestina	ES	Španělština
SK	Slovenština	PT	Portugalština
EN	Angličtina	RU	Ruština
DE	Němčina	HU	Maďarština
IT	Italština	PL	Polština
FR	Francouzština	EO	Esperanto

Tab. 1: Výběr některých jazykových kódů

V jakémkoliv případě uspořádaní souborů musíme zajistit správné přepnutí jazyka pomocí explicitně uvedeného odkazu na daný soubor. Tento způsob s sebou přináší nutnost kontrolování odkazů a jejich neustálou aktualizaci v případě jakékoliv změny, jako je např. přesun souborů do jiného adresáře. Toto je ostatně problém jakéhokoliv statického webu. V daném případě je vhodné, aby server sám rozpoznal jakou jazykovou verzi má uživateli (prohlížeči) předat. Pro tento účel je možno využít modulu mod_negotiation, který je standardně obsažen v serveru Apache.

Použití Apache modulu mod_negotiation

Tento modul umožňuje automatické vyhodnocení obsahu na základě nastavení prohlížeče, a to díky údajům z hlaviček požadavku od klienta. Tyto údaje přidává do hlaviček internetový prohlížeč, kde si je může uživatel také upravit. Nastavení preferovaných jazyků spolu s jejich pořadím je vidět na Obr.2



Obr. 2: Panel pro vybrání jazyka

Modul rozhoduje podle těchto údajů:

- Preferovaný typ média
- Preferovaný typ jazyka
- Preferovaná znaková sada
- Preferované kódovaní

Apache Server ovládá dvě metody vyhodnocování obsahu:

- MultiViews (nastavení vlastnosti adresáře)
- Typová mapa (použití handleru type-map)

MultiViews se používal hlavně u Apache 1.x. Jedná se o vlastnost adresáře, která se mu musí nastavit pomocí direktivy DirectoryIndex. Poté musíme mít dané soubory rozšířeny o další příponu, odpovídající kódu jazyka nebo znakové sadě. Jako např.: index.html.cs, index.html.en.Navíc musíme nastavit direktivu DirectoryIndex určující soubor z adresáře.

Typová mapa je dokument, který je spojen s handlerem `type-map`. Tento soubor má buď příponu `.html` nebo `.html.var`. Soubor obsahuje jednotlivé varianty, které obsahují požadované parametry, jako jsou délka souboru, typ kódování, kód jazyka, typ MIME souboru. Opět musíme mít soubory rozšířeny o doplňující přípony, nastavit direktivu `DirectoryIndex` a navíc také `AddHandler`. Vyhodnoceni obsahu pomocí Typové mapy se používá u Apache 2.x.

Řešení, kdy máme řadu souborů pro jeden obsah, ale pro různé jazykové verze, není příliš vhodné. Tyto soubory totiž musíme neustále překládat a udržovat v konzistentním stavu. Je jej možno akceptovat opravdu jenom u jednoduchých stránek nebo jako provizorní řešení. Pro rozsáhlejší weby je vhodnější jazykovou verzi stránky generovat automatizovaně podle nastavené proměnné určující jazyk. Tato proměnná je pak automaticky nastavena pomocí informací z hlaviček požadavků od klienta nebo pomocí uživatelova výběru: typicky vlaječka určující jazyk. Tímto se tedy dostáváme k aplikacím, které mají informace uložené mimo vlastní soubor.

3.3 Informační obsah mimo stránky

Jde o stránky, které jsou generovány pomocí nějakého skriptu (PHP). Data jsou potom uložena mimo tento skript. Je sice možné, aby byla uložena přímo ve skriptu a pomocí proměnné se určovalo, který řetězec se zobrazí. Při větším počtu textů ale toto řešení není vhodné.

3.3.1 Použití vlastního datového souboru a překládací funkce

Tento způsob má uložené informace, které se vyskytují ve více jazykových mutacích v jednom nebo více pomocných datových souborech. Informace jsou potom překládány pomocí vlastní překladatelské funkce, která umí data z datového souboru získat.

Daný datový soubor musí mít určitou strukturu. Je možné například i jako datový soubor použít skript, který je danou stránkou zavolán. Tento skript potom naplní datovou strukturu, kterou využívá překladatelská funkce. Tímto datovým typem může být asociativní pole, kdy první index je identifikátor textu. Položky pole jsou opět pole s jednotlivými verzemi textu. Ukázka takového datového souboru je v Kód 2.

```
<?php // soubor texty.php
        global $texty; // pole $texty bude globální proměnná
        $texty['nazev_stranky'] = array(
              'Testovaci stránka',
              'The testing page',
              'Die Testseite');
        $texty['uvod_text'] = array(
              'Vítejte, toto je vícejazyčná stránka',
              'Welcome, this is a multilingual page',
              'Willkommen, Das ist eine mehrsprachinge Seite');

              // .. dalsi data ..
?>
```

Kód 2: Ukázka datového souboru

Takto například víme že v proměnné `$texty['uvod_text'][2]` máme německou verzi úvodního textu.

Dále je potřeba si vytvořit překládající funkci, vyhledávající v globálním poli text podle dodaného klíče a zvoleného jazyku aplikace z proměnné určující jazyk požadovaný uživatelem. Tato proměnná může být opět nastavena skriptem z hlaviček zpráv od klienta. Funkce by měla umět ošetřit nepřítomnost daného textu v souboru, kdy vrací programátorem určenou varovnou hlášku nebo implicitní jazykovou verzi textu. V samotném zdrojovém kódu již tuto funkci voláme pro použití jakéhokoliv výpisu na výstup. Použití pro zobrazovaný text:

```
<h1><?php echo preloz('uvod_text');?></h1>
```

Tento způsob můžeme použít ale například i pro zobrazení různých národních variant obrázků. V českém překladu `logo_adresa` bude adresa souboru s českým, v anglickém s mezinárodním logem organizace:

```
<img src="<?php echo preloz('logo_adresa');?>" />
```

Je pouze na programátorovi, jak si dané řešení navrhne a implementuje. Musí se starat o aktualizaci, údržbu slovníku. Opakované vložení textů do datového souboru z výše uvedeného příkladu pod stejným klíčem způsobí jeho přepsání, je pouze na programátorovi, aby si nevědomky nezměnil výstup dříve vytvořených skriptů, používajících stejný klíč. V každém případě při zpracování skriptu se načítá celý obsah datového souboru, i pokud potřebujeme pouze jeho zlomek. To zbytečně zatěžuje paměť, kterou skript při svém běhu zabírá. Toto řešení tedy není vhodné pro

objemné weby. Potom je potřeba datový slovník rozdělit podle složení aplikace a zajistit nahrání požadovaného slovníku pro danou část webu.

3.3.2 Použití knihovny GNU Gettext

Knihovna GNU Gettext [8] je komplexní nástroj pro překlání – internacionalizaci a lokalizaci programových děl. Zároveň poskytuje služby pro údržbu těchto překladů. Podpora této knihovny je v programovacích jazycích jako C, C++, ObjectiveC, Python, PHP a mnohých dalších.

V PHP je nejprve potřeba ale knihovnu Gettext aktivovat. Buď kompilovat PHP s parametrem `--with-gettext`, nebo aktivací modulu `php_gettext.dll` v `php.ini`. Dále potřebujeme pomocné nástroje, které jsou v distribuci.

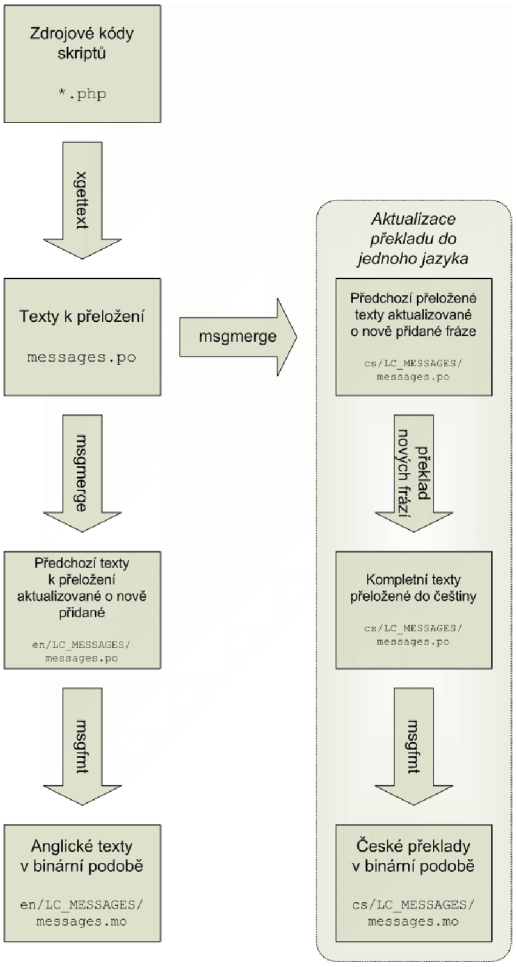
Použití knihovny je v PHP skriptech stejné jako v minulém řešení. Opět využíváme překladovou funkci a datový soubor. Nejprve musíme ve skriptu do proměnné `$lang` uložit verzi jazyka, kterou chceme zobrazit. Dále podle této proměnné nastavíme hodnoty systémové proměnné a lokalizace a určíme adresu k souborům s překladem.

Při samotném překladu využíváme funkci `gettext()` či alias funkce `_()`. Možnost volání obou funkcí však není zaručena. Z vlastních zkušeností mohu říci, že je závislá na verzi knihovny, kterou používáme. Nicméně funkce `gettext()` fungovala na všech testovacích distribucích. Parametr této funkce je ale na rozdíl od předchozího řešení přímo text v primárním jazyce aplikace nebo v jazyce, ve kterém aplikaci vyvíjíme.

Knihovna také poskytuje funkci `ngettext()` pro různé tvary textů. Toho využijeme, pokud je text závislý na číselné hodnotě s ním spojené. Je potom nutné pro jednotlivé jazyky upravit soubor s překlady. Například v češtině se pro oznámení o počtu nových emailů použijí 3 různé varianty textů (pro množiny: {1},{2..4},{0,5..}). Tuto funkci samozřejmě nemusíme využívat a můžeme si varianty textů vybírat sami zdrojovým kódem našeho skriptu.

Překlad jazyků si již řešíme mimo PHP skripty ručně pomocí dalších funkcí knihovny GNU Gettext. Pro analýzu textu, která nám vybere všechny řetězce k překladu, použijeme příkaz `xgettext`, které předáme vstupní soubor nebo seznam souborů či celý adresář. Ve výstupu – souboru s příponou `.po` musíme určit kódování. Dále zde máme části `msgid` – texty z originálu s původním textem a části `msgstr`, kam zapíšeme překlad do dalšího jazyka. Pro každý jazyk je potřeba udělat binární překlad tohoto souboru `.po`, tím dostaneme soubor `.mo`, který musí být uložen jednotlivě pro každý jazyk v adresářové struktuře určené ve skriptu, jak je zmíněno výše. Tento překlad provádíme příkazem `msgfmt`.

Pro aktualizaci textů po změnách využíváme příkaz `msgmerge`, který přidá nové texty ze souboru `.po` do jeho starší verze v adresáři pro danou verzi jazyka. Tyto aktualizace je potřeba provést pro všechny jazykové verze, dále je pro ně potřeba doplnit překlady a vytvořit binární soubory `.mo`. Následně je potřeba restartovat server. Celý proces aktualizace vidíme na Obr. 3



Obr. 3: Schéma aktualizace překladů

Tento proces aktualizace může být pro rozsáhlejší aplikace zdlouhavý a zbytečně pracný, proto existují nástroje, které umožňují správu těchto překladů. Například KBabel [10], který je šířen pod licencí GNU GPL. Tento nástroj nebo spíše skupina nástrojů tvoří hlavně editor `.po` souborů, poskytující editační, vyhledávací, porovnávací, navigační, kontrolní a další funkce. Dále obsahuje manager `.po` souborů a slovník. Obrázek editoru je v příloze.

Podporu knihovny GNU Gettext mají v sobě i některé šablonovaní systémy, jako například Y.A.T.S [11].

Knihovna Gettext je velmi propracovaná a dá se s výhodou použít. Data pro běh aplikace jsou po překladu uložena v binární podobě, což je ve prospěch rychlosti aplikace. Dále rychlosti napomáhá, že data s překlady jsou uloženy ve vyrovnávací paměti www serveru. Podle [9] je zpomalení překládané aplikace oproti originální dokonce neměřitelné. Toto řešení má bohužel také své nevýhody. Rozšíření nemusí být na serveru k dispozici, a také spouštění příkazů pro překlad nemusí být vždy dostupné. Navíc i přes dostupné programy pro editaci a management překladů, nemusí toto být vždy vhodné řešení. Pokud překlad nezajišťuje přímo tvůrce aplikace je problém

s distribucí těchto souborů a překladatelé nemusí být se soubory ve speciálním formátu, jako zadání překladu příliš spokojeni. A nakonec, jelikož jsou data ve vyrovnávací paměti serveru, je nutné jej při každé změně restartovat, což nemusí být také umožněno.

3.3.3 Použití šablonovacího systému s podporou slovníků

Pokud vytváříme aplikaci tímto způsobem, tak v šablonách stránek nejsou plné znění textů, ale pouze klíče ukazující na jednotlivé výrazy do překladového slovníku. Každému jazyku odpovídá vlastní textový soubor se strukturou klíč – překlad. Naše aplikace je potom zodpovědná za přepínání jazyků. Šablonovací systém poté sám určí na základě parametru, který soubor zvolit. Například na Internetu hodně používaný šablonovaní systém Smarty [12] v sobě podporu slovníků má implementovanou.

3.4 Aplikace využívající databázi

Jak bylo řečeno v předcházející kapitole, existují weby bez databáze, většina však nějakou využívá či využívat může. Je tedy pohodlné je použít nejen jako skladiště informací obsahového charakteru, jako jsou data narození, výše zůstatku na bankovním kontě atd., ale také jazykových verzí textů dané aplikace.

Pro vyhnutí se problémům s různým kódováním znaků, potřebujeme databázi, která umožňuje uložit data ve formátu UTF-8. S tím si bez problému poradí například, MySQL, ORACLE či PostgreSQL. Uvedené příklady budou postaveny na databázi MySQL.

Implementace aplikace, která má texty v databázi, je opět na programátorovi. Opět můžeme předpokládat volání nějaké překladatelské funkce, která načítá data, nikoliv však ze souboru, ale z databáze.Implementací je hodně, proto se spíše zaměřím na možnosti návrhu databáze a použití balíku Translation2 z knihovny PEAR.

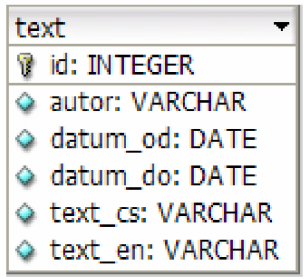
3.4.1 Možnosti návrhu DB

Pro uložení textů můžeme zvolit několik variant struktury. Budeme předpokládat text, u kterého potřebujeme zaznamenat identifikátor, autora, datum vydání, platnost do, česká varianta, anglická varianta.

Použití jedné tabulky

Tento způsob ukládá všechny texty do jedné tabulky, její návrh je zobrazen na Obr. 4. V tomto řešení je pro každý text použit jeden řádek tabulky. Pro každý jazyk je vybrán jeden sloupec. Přidáním dalšího jazyka musíme přidat další sloupec tabulky a také obslužný programový kód. Proto je toto řešení nevhodné pro projekty u kterých předpokládáme rozšiřování jazyků. Naopak je vcelku vhodné

u jednoduché aplikace, kde dopředu víme, že nám budou stačit třeba pouze dva jazyky. Proto taková struktura bývá dost často nasazena v ostrém provozu.

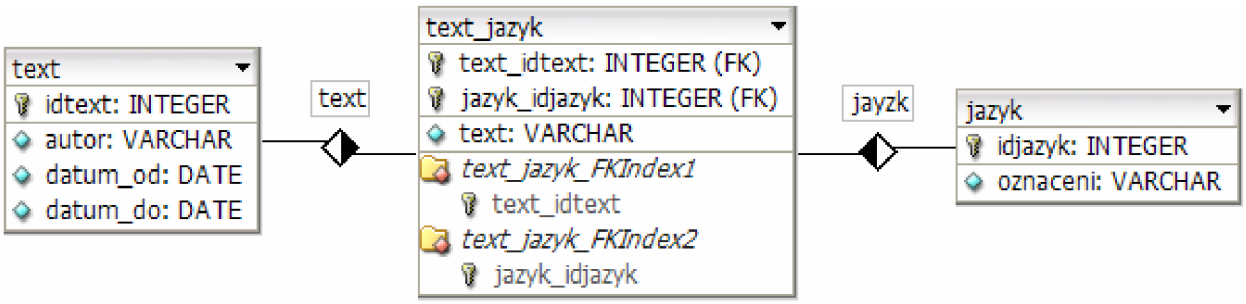


Obr. 4: Použití jedné tabulky pro uložení více variant textu

Použití tří tabulek

Pokud bychom se snažili vylepšit předchozí řešení tak, že bychom si vytvořili zvláštní tabulku jazyk pro vyznačení jazykové verze textu, v původní tabulce text by přibyl atribut jazyk a místo skupiny sloupců bychom zde měli jenom text jeden. Došlo by zbytečně k opakování společných atributů skupiny překladu jednoho textu. Z toho také plyne hrozba nekonzistence.

Proto je vhodné rozdělit si původní tabulku na tabulky tři, viz Obr. 5. Tím od sebe oddělíme atributy, které jsou potřeba překládat a ty, které jsou pro všechny jazykové varianty konstantní. Takový návrh je vůči proměnlivému počtu jazyku odolný. Pokud bude překladatelská funkce aplikace správně napsaná, tedy, že bude umět obsloužit i nepřeložené texty pro určený jazyk. A i samotná aplikace bude skupinu podporovaných jazyků získávat z tabulky jazyk, po jednoduchém přidání záznamů, není v aplikace potřeba nic měnit. Nad takovou databází je samozřejmě možné postavit další aplikaci, kterou budou využívat překladatelé pro její správu.



Obr. 5: Použití tří tabulek pro uložení více variant textu

3.4.2 Balík Translation2 z knihovny PEAR

Balík Translation2 [14] je balík z knihovny PEAR (PHP Extension and Application Repository)[15] , která obsahuje open-source kód pro uživatele PHP. Balíčky jsou zde sdružovány v stromové struktuře. Translation2, který je pokračováním balíčku Translation, se nachází ve skupině

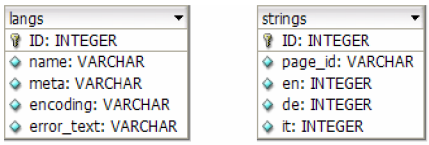
Internationalization. Předchozí Translation, je označována jako třída pro tvorbu vícejazyčných stránek, nynější verze je popsána jako třída pro management vícejazyčných aplikací.

Třída je navrhnuta s ohledem na redukci dotazů do databáze. Třída Translation2_Admin poskytuje služby přidávání nebo odebírání jak řetězců pro překlad, tak i jazyků. Jsou podporovány tyto ovladače zdrojů dat:

- PEAR::DB
- PEAR::MDB
- PEAR::MDB2
- gettext
- PEAR::DB_DataObject – zatím ve vývoji
- XML

Dříve bylo potřeba pro možnost využití balíku Translation2 doinstalovat základní balík PEAR. Dnes je již tato knihovna součástí PHP.

Nejprve je potřeba nastavit informace pro připojení k databázi, tedy host, jméno a typ databáze, uživatelské jméno a heslo. Dále, pokud je naše struktura tabulek odlišná od standardní, musíme ji definovat. Standardní definice je zobrazena na Obr. 6.



Obr. 6: Standardní struktura pro knihovnu Translation2

Je možné mít jednotlivé tabulky pro každý jazyk, místo jedné společné. Dále si volíme driver databáze.

Hlavní metody jsou:

- **`get()`** – vybírá přeloženy řetězec ze zdroje. Pokud je řetězec prázdný – není přeložený a je použita volba (v Translation2 se označuje decorator), je vrácena hodnota proměnné `$defaultText`.
- **`getPage()`** – dodá celou skupinu překládaných řetězců, v případě potřeby vyhledá nouzový jazyk a nahradí parametry.
- **`getRawPage()`** – dodá stránku ze zdroje, bez jakéhokoliv formátování a záměnou parametrů.
- **`getLang()`** – pro získání metadat
- **`getStringID()`** – získáme pomocí řetězce jeho identifikátor. Tuto metodu použijeme pokud chceme řetězec přeložit, ale neznáme jeho stringID.

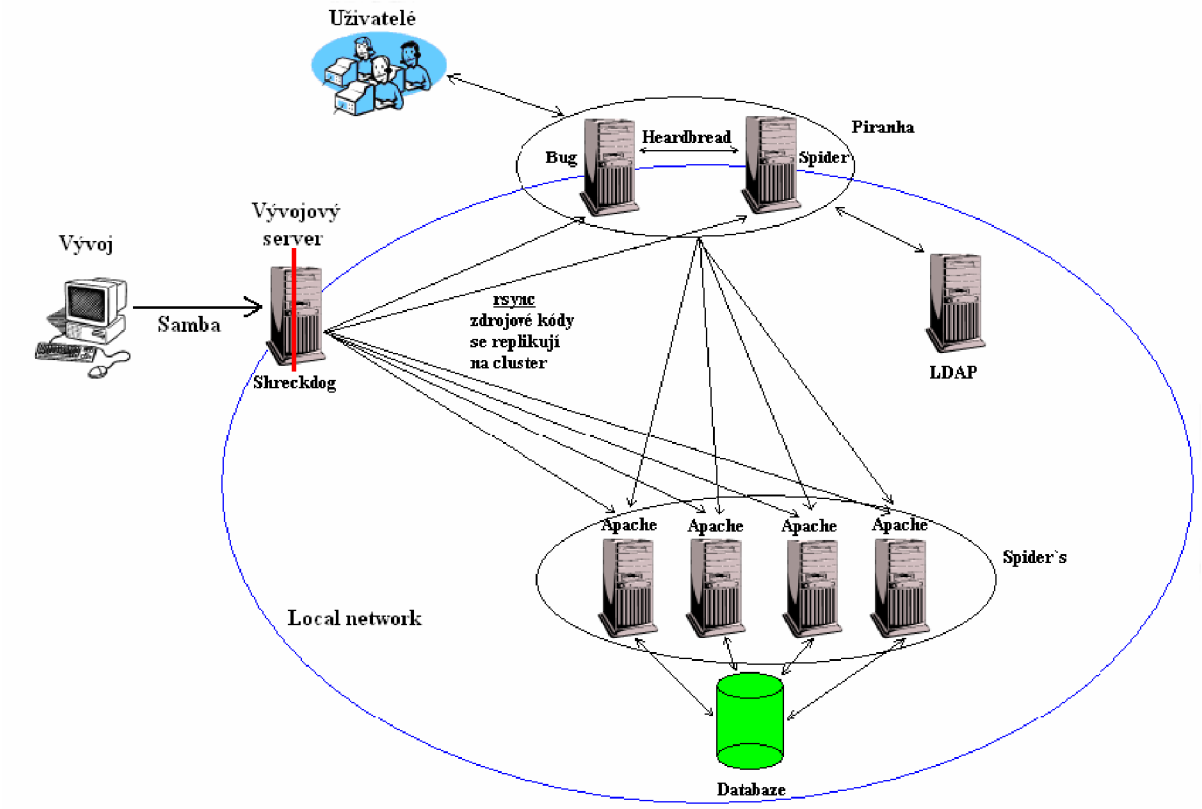
4 Portál VUT

V této kapitole proberu analýzu stávajícího stavu Portálu VUT [2] se zaměřením jak na celkový návrh a implementaci, tak na řešení zajištění vícejazyčnosti webových aplikací. Informace pro tuto část jsem čerpal z informací od zaměstnanců CVIS (Centrum Výpočetních a Informačních Služeb) VUT v Brně a na centrálním úložišti dokumentace - wiki stránkách [16] .

Portál VUT slouží jak cizím návštěvníkům pro získávání všeobecně dostupných informací, tak zaměstnancům, studentům a pedagogům VUT pro přístup k jejich osobních informací, ty jsou přístupné až po přihlášení.

4.1 Struktura serverů

Centrální webový portál VUT je provozován na technologii clusteru. Jedná se o řešení společnosti Redhat Inc. s označením Piranha. Na oddělení CVIS, které se o tyto servery a celý Portál stará, je provozováno řešení, které je zobrazené na Obr. 7. Základ tvoří jeden server s názvem Bug, který slouží jako vstupní brána. Tento server si neustále kontroluje dostupnost všech serverů s názvem Shrecks. Na nich běží samostatné servery Apache. Na každém z nich je celý obsah portálu samostatně, takže výpadek jednoho nezpůsobí nefunkčnost celého portálu nebo některé jeho části. Server Bug podle momentálního zatížení nebo dostupnosti jednotlivých serveru provádí směrování příchozího uživatele na webový server s nejmenším vytížením, kde je navázána relace, která je aktivní až do ukončení spojení. Tímto je rovnoměrně rozdělována zátěž jednotlivých webových serverů tak, aniž byt uživatel něco poznal. Server Bug má zálohu, a to v podobě serveru s názvem Spider. Mezi těmito stroji probíhá neustála komunikace, ve které se navzájem kontrolují, aby v případě nestandardní situace nebo poruchy mohl reagovat záložní stroj a převzít funkci hlavního vstupní brány. Hlavní vývojový server, který je zde provozován, má jméno Shreckdog. Běží na něm služba SVN – viz. dále a slouží pouze k vývoji stávajících i nových portálových aplikací. Tímto způsobem vývojář nepřijde do přímého styku se servery Shreck, na něž jsou replikovány až odladěné zdrojové kódy. Tento krok zajišťuje dobrý výkon a stabilitu aplikací běžících na webovém clusteru.



Obr. 7: Schéma webového clusteru VUT

4.2 Databáze

4.2.1 Oracle

Primárním zdrojem dat je několik instancí databáze Oracle. V současnosti bychom těžko našli výkonnější databázový stroj. Jedna z instancí je používána jako ostrá verze, další jsou pro testovací a vývojové užití. Databáze běží na samostatném serveru pro tento účel vyhrazený. Kromě klasického schématu tabulek, klasických a materializovaných pohledů se ještě používají procedury a funkce, které jsou vytvořeny jazykem PL/SQL. Procedury a funkce jsou vynikajícím prostředkem k dolování informací z databáze, jelikož hlavní dotazy jsou centralizované v databázi a klientské aplikace volají pouze rozhraní procedury.

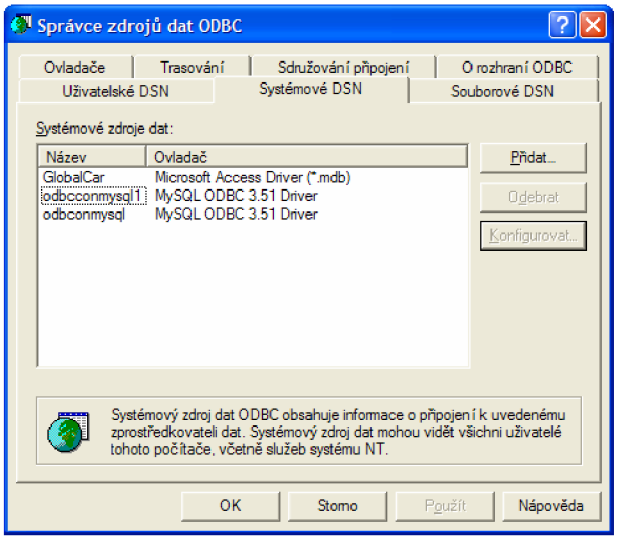
4.2.2 Standard ODBC

Celá řada relačních databázových systémů (Oracle a MySQL nevyjímaje) podporuje standard ODBC. Ten sice umožňuje práci s databází určitého výrobce bez možnosti využití jím implementovaných speciálních funkcí, které mohou tuto práci zefektivnit, zato však jsme při využití tohoto standardu na konkrétním typu databáze nezávislí. Výhoda je tedy zřejmá v možnosti připojení k několika druhům databází, různých výrobců bez nutnosti změny kódu ve skriptech. Je teda zaručena absolutní

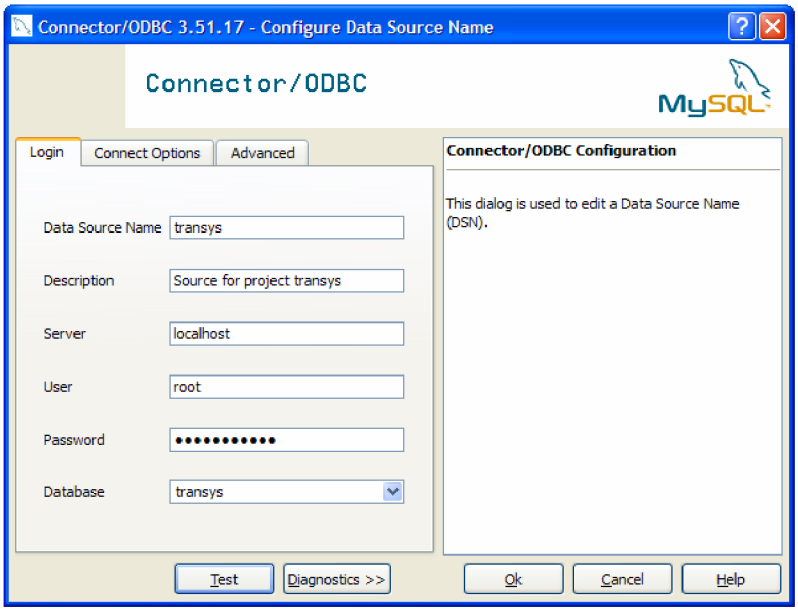
přenositelnost. Vývojář může aplikaci vyvinout na vlastní databázi na lokálním počítači – typicky MySQL a potom aplikaci přenést do reálného provozu beze změny kódu. Jedinou podmínkou je nutnost stejné datové struktury.

Standard ODBC je založen na tom, že mezi aplikací – skriptem a databázovým systém je vložena ještě jedna mezivrstva – ovladač, který zprostředkovává obecné databázové dotazy konkrétnímu serveru. Zároveň standard definuje schéma příkazů SQL, které je možné v ODBC použít. Je potřeba aby na systému, kde PHP běží, byl definovaný datový zdroj ODBC pro příslušný databázový systém a příslušnou databázi. Je nejprve nutné nainstalovat ovladač pro příslušný databázový systém. Na Obr. 8 je vidět Správce zdrojů dat ve Windows XP a na Obr. 9 MySQL ODBC Connector – ovladač pro tuto databázi.

V PHP je knihovna pro práci s ODBC zdroji a vztahuje se na ni licence GPL.



Obr. 8: Správce zdrojů dat ODBC ve Windows XP



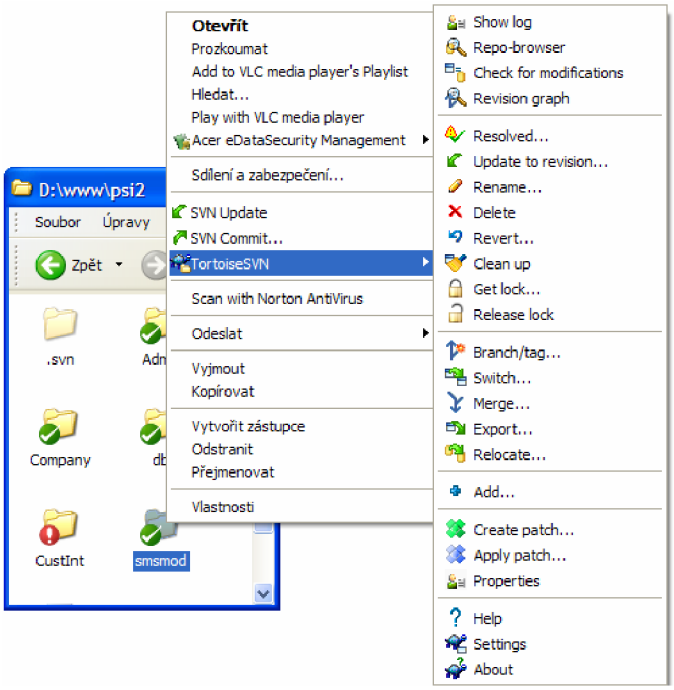
Obr. 9: MySQL ODBC Connector

4.3 SVN

Subversion je software, který má za úkol pomáhat programátorovi udržet přehlednost projektu, na kterém pracuje. Patří do oblasti SCM (Source Code Management) aplikací. Používáme jej tam, kde na projektu pracuje více osob. Díky tomu, že zaznamenává jakékoliv změny na projektu, je jako funkcí pro řešení konfliktů velmi využitelným nástrojem.

Mezi nejznámější a zároveň volně dostupné SCM systémy patří CVS (Concurrent Version System), jehož nástupcem je SVN.

Celý systém je postaven na dvouvrstvé architektuře klient – server. Navíc je přístupný pro různé druhy platforem, jako jsou Linux, Windows i jiné, a to jak pro server tak i klienta. Na oddělení vývoje webových aplikací CVIS je používána kombinace server na operačním systému Linux a klient TortoiseSVN na platformě Windows. Tento klient jak můžeme vidět na Obr.10 se plně integruje do systému, a proto samotná práce je velmi pohodlná a intuitivní.



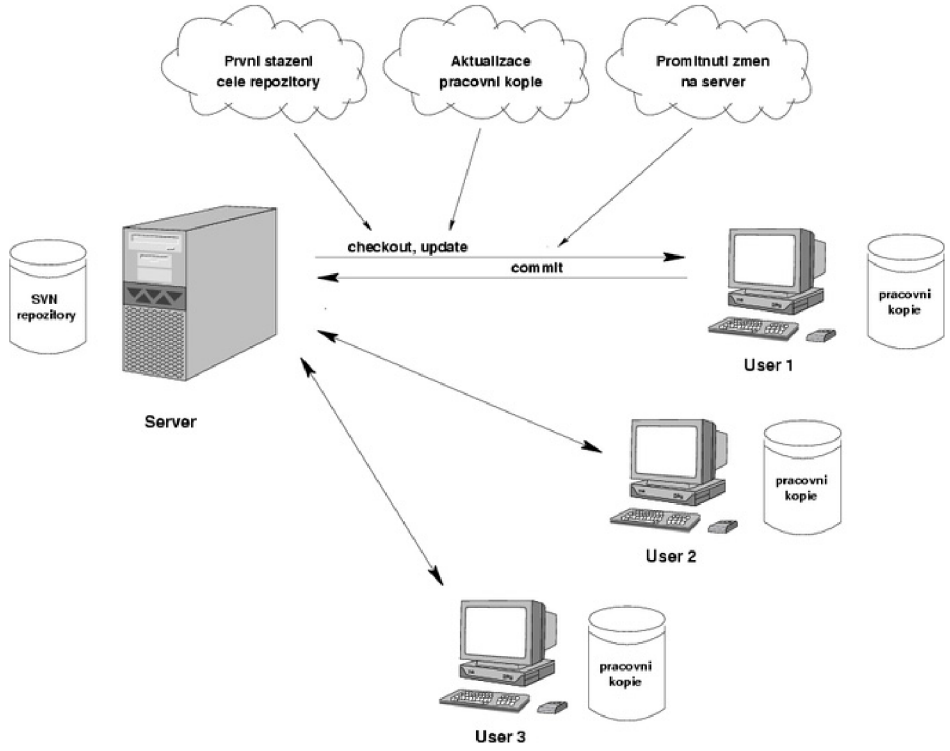
Obr. 10: Integrace TortoiseSVN klienta do Windows XP

Nástroj je založen na dvou částech. První je databáze „Repository“, kde jsou uloženy zdrojové kódy projektu. Není zde uložena pouze poslední verze, ale i všechny předešlé změny, které jsou samozřejmě přístupné všem uživatelům. Druhou stranou je pracovní kopie „working copy“, která je kopií některého ze stavu databáze a je určena k lokálním úpravám. Tímto je zajištěno, že uživatel nikdy nepracuje přímo se zdrojovými kódy, ale pouze s kopiemi.Práce se skládá z několika operací:

- **checkout** – Provádí se pouze jednou a slouží k vytvoření pracovní kopie na lokální disk počítače z databáze.

- **update** – Příkaz provádí synchronizací změn v databázi s naší lokální kopií. Samozřejmě se neprovede přepsání našich změn těmi, které jsou v databázi, ale klient na tento rozdílový stav upozorní a je pouze věcí uživatele zda změny potvrdí nebo si nechá svou verzi.
- **commit** – Příkaz provede změnu do centrální databáze. Tato změna je označena jako revize. Při každém nahrání modifikovaných souborů do databáze je číslo revize automaticky inkrementováno. Proto je také dost vhodné uvést čeho se naše změna týká, co bylo upraveno a proč.

Hlavní smysl uchovávání historie jednotlivých změn je možnost se vrátit o jeden nebo více kroků zpět k verzi, která fungovala, jelikož nám to šetří spoustu času u hledání chyb nebo vlastních přepisů nejen u rozsáhlejších projektů. Snadné porovnání dvou rozdílných verzí nás zavede na ono místo, kde se nachází možný zdroj problémů.



Obr. 11: Operace nástroje Subversion

Po každé operaci commit je zapsán do log souboru záznam (Ukázka na následující straně) o číslu revize, autoru, datu a čase operace commit a seznam změněných souborů spolu s jejich stavovým kódem tzv. SVN status viz Tab. 2. Dále pokud byla zapsána, poznámka od autora, který operaci commit provedl, je také zaznamenána.

```
r2914 | user1 | 2007-03-14 10:58:49 +0200 (Wed, 14 Mar 2007) | 4 lines
Changed paths:
   M //trunk/_base/templ/gm_vizitka_cv_templ.php
   A //trunk/_base/templ/gm_vizitka_hledat_templ.php
   D //trunk/_base/templ/gm_vizitka_pub_templ.php
   R //trunk/_base/templ/gm_vizitka_templ.php

U všech souboru pripsany komentare
```

Ukázka záznamu SVN po operaci commit

SVN status	Význam
M	Soubor byl modifikován
A	Soubor/adresář byl přidán
D	Soubor/adresář byl odstraněn
R	Soubor/adresář byl přejmenován nebo přesunut

Tab. 2: SVN statu

V pravém dolním rohu většiny aplikací Portálu VUT je vidět popis revize této aplikace. Pokud je popis tvořen dvěmi dvojtečkou oddělenými čísly, tak to znamená, že čísla revizí jednotlivých souborů této aplikace se nacházejí v rozsahu těchto dvou čísel. Pokud popis revize obsahuje na konci M, tak to znamená, že některý soubor z této aplikace byl modifikován.

Popis revize jednotlivých aplikací jednotlivých instancí není zjišťován online, ale aktualizuje se jednou za minutu. Protože se všechny aplikace nacházejí v jednom repositáři a mají společné základní knihovny, tak se číslo revize aplikace počítá z čísla revize adresáře aplikace a čísla revize adresáře základních knihoven. Více o Subversion a SVN je možné získat ve zdrojích [17],[18],[23],[24].

4.4 Analýza webu

4.4.1 Struktura

Struktura portálu se odvíjí od jeho hlavních součástí a požadovaných funkcí. Obsahuje několik souborů – knihoven s obecnými funkcemi, které se používají v celém projektu, např. funkce pro ověření uživatele, pro zpracování názvů dokumentů a jejich zkrácení na únosnou mez, pro výpis jazykového ekvivalentu výrazu či názvu, pro vložení oddělovací čáry či pro vložení titulu stránky dle jazykové mutace.

Hlavním jádrem je soubor index.php přes který se parametrem určuje stránka, kterou má zobrazit nebo aplikace, kterou má spustit. Tento soubor ověřuje přítomnost stránky či aplikace, dále vkládá podle kontextu menu vlevo, či informační panel vpravo. Také načítá příslušný kaskádový styl

stránky. Neméně důležité jsou hlavičkový a patičkový soubor. Tyto soubory obsahují obecný HTML kód definující základní vlastnosti rozdělení portálu. Je kompletně napsán pro zformátování pomocí CSS souboru a to včetně obrázků.

4.4.2 Instance

Z důvodů kontinuálního vývoje Portálu a kooperace stálé skupiny vývojářů CVIS s externími pracovníky a studenty, je v provozu více instancí webových aplikací. Tyto jsou rozebrány v Tab. 3.

verze	databáze	url	revize	správce	přístup	počet
ostra	CDBX	https://www.vutbr.cz/	A	zaměstnanci	všichni	1 (a)
test	CISD	https://ent.ro.vutbr.cz/	B	Zaměstnanci	všichni	1
vyvoj	CISD	https://ent.ro.vutbr.cz/vyvoj/	X	Zaměstnanci	povolené IP (1), (2)	6-10 (b)
ostratest	CDBX	https://ostratest.ro.vutbr.cz/_base/www_base/	X	zaměstnanci	povolené IP (2)	0 (c)
student	CISB	https://3wtest.ro.vutbr.cz/jmeno studenta/	C	studenti	povolené IP (3)	?

Tab. 3: Instance web aplikací

Vysvětlivky:

- (1) – Společná instance pro všechny zaměstnance, dále má každý svoji vlastní.
- (2) – Občas je do této instance povolen i dočasný přístup dalším osobám. Toto je ale pouze mimořádná situace, kdy je potřeba rychle vyřešit nějaký problém.
- (3) – Instance jednotlivých studentů a externích spolupracovníků. Většinou jsou povoleny default IP adresy, další si mohou přidat programátoři.
- (a) – Ostrá instance běžící na web clusteru, okolí se instance jeví jako celistvá, ale zdrojové kódy jsou ve skutečnosti na čtyřech, resp. šesti místech. V budoucnosti se počítá s diskovým polem.
- (b) – Každý vývojář má jednu až dvě vlastní instance. Dále jsou zde instance all a z-fresh.
- (c) - Instance ostratest má stejné zdrojové kódy jako instance vývoj/all.

Popis verzí:

- **Ostra** - Ostré verze aplikací dostupné pro návštěvníky a uživatele. Zdrojové kódy se synchronizují z adresáře na vývojovém serveru cca. Každé dvě minuty.
- **Test** - Vývojové instance určená pro testování, zde probíhá testování rozsáhlejších změn, než jsou vystaveny do ostré verze.
- **vyvoj/all** - Společná instance pro všechny zaměstnance, používaná pouze pro rychlé úpravy, které není nutné nebo možné testovat na testovací instanci. Zdrojové kódy pro instanci ostratest jsou stejné jako zde.
- **vyvoj/z-fresh** - Zde se neprovádí žádné úpravy. Je zde vždy poslední revize z repositáře. Instance je vhodná pro kopírování aplikací na jiná místa.

- **vyvoj/…** - Ostatní instance jednotlivých zaměstnanců používané pro řečení jednotlivých úkolů, které projdou fázi testování.
- **student/*** - Jednotlivé instance studentů a externích spolupracovníků.
- **Ostratest** - Verze určená pro testování. Instance má stejné zdrojové kódy jako instance vyvoj/all, ale připojuje se na databázi CDBX. V současné době, kdy se pravidelně synchronizuje databáze CISD, již není velmi využívána.

4.4.3 Databáze

Jedná se o centrální úložiště dat v rámci celého VUT. Na univerzitě celkově studuje a pracuje téměř 20 000 osob, což samo o sobě znamená, že jen jejich osobní záznamy zaujímají velký objem dat. Další velmi rozsáhlou skupinu dat tvoří informace týkající se studia, vědy a výzkumu. Počet nově přihlášených a úspěšných studentů každým rokem vzrůstá, proto má celkový objem dat samozřejmě lineárně vzrůstající tendenci.

Data jsou členěna v rámci logických celků, které obsahují jednotlivá databázová schémata. Tato schémata jsou dále dělena na úroveň tabulek obsahujících vlastní data.

Příklady existujících schémat:

- **BRUTISADM** – Rozsáhlé schéma, které obsahuje například osobní informace, informace o organizačních jednotkách VUT, data z oblasti vědy a výzkumu.
- **STO1** – Schéma seskupující informace týkající se studia. Jedná se o studijní programy a obory, výsledky studia jednotlivých studentů, také data týkající se rozvrhů.
- **Apollo** – Data pro nastavení uživatelských prostředí informačního systému.
- **ApoloAdm** – Obsahuje všechny SQL dotazy, které jsou pak používány pro práci s daty v rámci informačního systému.

Na oddělení CVIS jsou provozovány celkem tři kopie centrálního datového skladu:

- **CDBX** - obsahuje ostrá data a také nad ní pracují informační systémy.
- **CISC** - slouží k testování nových aplikací a komponent v rámci vývojového oddělení. Data jsou zde totožná s ostrou databází.
- **CISB** - je vývojová databáze, ve které nejsou data shodná s ostrou, ale jsou značně změněna jelikož tuto databázi využívají např. studenti pro vývoj svých projektů pro VUT a mohlo by dojít k zneužití těchto dat. Jelikož se tato práce řadí do kategorie studentských prací, byl mi umožněn přístup právě na tuto databázi. Samotný přístup k databázi se řídí dost přísnými pravidly, které mají za úkol zamezit zneužití dat, která jsou zde uložena. Pro samotný přístup slouží přihlašovací iniciály, které jsou jedinečné pro každého uživatele, ale toto samo o sobě nestačí, jelikož je povolen přístup pouze z určitých IP adres.

Pro lepší přehlednost databáze jsou stanoveny konvence pojmenování DB objektů. Těmito konvencemi je nutné se řídit.

- **Primární klíče** - Tvoří se prefixem PK a názvem tabulky. Příklad: PK_NAZEV_TABULKY
- **Cizí klíče** - Tvoří se prefixem FK, názvem referované tabulky a názvem tabulky referující v tomto pořadí. V případě že je potřeba provázat dvě tabulky více než jedním cizím klíčem, rozlišujeme je připojením vhodného postfixu charakterizujícího klíč.
- **Indexy** - Tvoří se prefixy IX, IXUF, UQ, názvem tabulky a výčtu sloupců užitých v indexu. Prefixy mají následující význam: IX = index, IXUF = unikátní funkční index (neexistuje unikátní constraint), UQ = unikátní index, unikátní constraint.
- **Sekvence** - Tvoří se prefixem SQ a názvem tabulky (resp. s názvy tabulek oddělených podtržítky), která sekvenci využívají. Jako název sekvence je možné použít i název sloupce, který využívá sekvenci a je přítomný ve všech tabulkách využívajících tuto sekvenci.
- **Triggery** - Tvoří se prefixem TG, názvem triggeru. Je možné název doplnit postfixy složenými z písmen A, B, I, U, D, mající tento význam: A = after, B = before, I = insert, U = update, D = delete. V případě že je třeba rozlišit zda-li jde o trigger pro řádek nebo pro tabulku, připojujeme postfix R nebo T.
- **Kontroly** - Tvoří se prefixem CH a názvem kontroly.
- **Číselníky** - Tvoří se prefixem C a názvem tabulky.
- **Pohledy** - Tvoří se prefixem V a názvem pohledu. Název pohledu by měl intuitivně vypovídat o jeho obsahu, případně oblasti využití.
- **Materializované pohledy** - Tvoří se prefixem MV a názvem materializovaného pohledu. Název materializovaného pohledu by měl intuitivně vypovídat o jeho obsahu, případně oblasti využití.
- **Kurzory** - Tvoří se prefixem U a názvem kurzoru.
- **Lokální proměnné** - Tvoří se prefixem X a názvem proměnné.
- **Globální proměnné** - Tvoří se prefixem G a názvem proměnné.
- **Parametry procedur a funkcí** - Tvoří se prefixem a názvem parametru.

Dále jsou povinné tři atributy v tabulkách:

- **Upd_id** – Hodnota slouží k identifikaci uživatele, který provedl záznam nebo změnu příslušného záznamu v tabulce.
- **Upd_ts** – Obsahuje časovou informaci o vložení nebo modifikaci daného záznamu v tabulce, kterou provádí databázová funkce SYSDATE.
- **Status** – Určení statusu, ve kterém se nacházejí data v daném záznamu. Tento může nabývat těchto hodnot:

- 0 - data – synchronizace z jiných systémů
- 1 - data logicky smazána, nelze provést fyzický výmaz, jelikož existují návaznosti na tato data
- -1 - data určená k fyzickému vymazání po určité době
- 9 - platná data – nově vytvořený záznam

4.5 Analýza řešení vícejazyčnosti

V současné době se zajištění překladu do angličtiny provádí následovně: Vývojáři (zaměstnanci CVIS) prochází zdrojové kódy portálů a vybírají texty, které zatím nejsou přeloženy. Tyto texty ručně kopírují a vytváří soubor textů (většinou v Microsoft Excel) nebo texty vkládají přímo do emailu a posílají překladatelce. Překladatelka texty přeloží nebo se zeptá na upřesňující informace, které potřebuje k překladu. Nové texty jsou vloženy do zdrojových kódů. Překladatelce jsou poslány doplňující informace. Překladatelka počítá kolik textů přeložila a žádá o odměnu. Texty se nachází v jednom datovém souboru a to z důvodu snadnosti hledání a konzistence překladů. Soubor je však již příliš objemný (9000 řádků, 300 kB), což v době velké zátěže informačního systému způsobuje problémy.

4.5.1 Funkce pro překlad a datový soubor

Pro překlad se na stránkách Portálu používají tři překladatelské funkce ze souboru `func_tr.php`:

```
function insert_label( $id_label, $visible_en=1, $ret=0 )
function get_lang( $tr_mem_key, $in_lang=null )
function tr( $cs, $lang=null, $force_ucfirst_en=0 )
```

Všechny tyto funkce používají jeden datový soubor `func_tr_data.php`. Toto je běžný PHP skript, jehož úkolem je naplnit globální asociativní pole `$tr_mem`. Indexem pole je identifikátor textu. Jeho položky je opět pole s položkami pro českou a anglickou variantou textu. Viz příklad Kód 3.

```
$tr_mem['title_uvodni_info'] = array(
      'Úvodní informace',
      'Introductory information'
);
```

Kód 3: ukázka položky datového souboru

Všechny tyto funkce se používají v různých částech webu a mohou být volány s různým počtem parametrů určujících například jazyk, jehož varianta má být funkcí vrácena či vypsána na standardní výstup, zda je text v anglické verzi stránky viditelný atd. Pokud je funkce s některým parametrem volána, je použita přímo číselná hodnota či proměnná. Jako například:

```
<?php echo get_lang('unknown_passwd_hint',1); ?>
```

Jelikož výstupem překladatelských funkcí je řetězec, mohou být tyto funkce použity stejně jako řetězce. Mohou být tedy přiřazeny do nějaké proměnné, vytisknuty na standardní výstup, porovnány atd. Také povinný parametr těchto funkcí je řetězec, proto mohou být parametry překladatelských funkcí sestavovány pomocí operátoru „ . “ , který je v PHP používán pro spojování řetězců.

Nicméně se ale také vyskytují „problémová“ volání překladatelských funkcí, kdy jim jsou jako parametr pro překlad dodány proměnné či výstupy jiných funkcí, které jsou opět na nějakých proměnných závislé. Tyto volání se musí odstranit, protože nemůžeme jednoznačně určit text, který se má přeložit. Tento text je závislý na jednom konkrétním běhu skriptu – na obsahu použitých proměnných. Více o těchto problémových voláních bude napsáno v kapitole o realizaci a integraci.

To že jsou funkce tři a každá se chová jinak je dáno jejich použití v různých částech portálu, kde mohou být na formát výstupu kladeny různé požadavky. Hlavním a společným rysem funkcí tedy je, že vrací hodnoty z pole, které je naplněno z datového souboru. Pro názornost uvádím kód funkce `get_lang` v Kód 4. Tato funkce v případě nedodání parametru určující jazyk zvolí jazyk z globální proměnné. Pokud existuje varianta textu pro daný jazyk, je funkcí vrácen a funkce je ukončena. Pokud pro daný jazyk není text přeložen, funkce se snaží vrátit českou variantu textu, pokud ani tato není v datovém souboru – ve slovníku, je vrácen parametr funkce.

```
function get_lang( $tr_mem_key, $in_lang=null ){
      global $lang, $tr_mem; // in
      if ( !isset($in_lang) ) $in_lang = $lang;
      if ( isset($tr_mem[ $tr_mem_key ][ $in_lang ]) ) {
            return $tr_mem[ $tr_mem_key ][ $in_lang ];
      }
      if ( isset($tr_mem[ $tr_mem_key ][0]) ) {
            return $tr_mem[ $tr_mem_key ][0];
     }
     return $tr_mem_key;
}
```

Kód 4 : Funkce get_lang()

Datový soubor je optimalizovaný, jsou tu totiž sekce, které se načítají jenom pro určité části portálu: aplikace portal, Studis, Teacher. Děje se tak na základě globální proměnné určující právě probíhající aplikaci. V těchto sekcích jsou položky, které se vyskytují právě jenom v těchto aplikacích, proto nejsou načteny do paměti v případě ostatních aplikací.

4.5.2 Překladatelé

Cílem požadovaného systému není usnadnit práci jenom vývojářům, ale také překladatelům. Proto jsem si domluvil s ředitelkou ústavu jazyků Fakulty Strojního Inženýrství Mgr. Ditou Gálovou, která pro CVIS zabezpečuje odborné překlady textů schůzku, abych zjistit i její pohled tento projekt. Podle jejího sdělení se někdy překladu zúčastňuje i více osob, všichni jsou oproti dřívějšímu zajišťování překladů zaměstnanci VUT, konkrétně výše uvedeného ústavu. Za každý překlad je požadována odměna, která se účtuje za tzv. normo strany, kde jedna strana je 1500 znaků, mezery se do těchto znaků nepočítají.

Dále mi bylo sděleno, že z důvodů více možností překladů některých českých výrazů do AJ si ústav vytvořil slovník často používaných výrazů. Jeho použití zajistí, že na portálu nejsou některé výrazy přeloženy různě. Dále ze stejného důvodu u textu k přeložení dodat kontext, což si překladatelé vyžádají.

5 Návrh systému

Celá kapitola obsahuje popis návrhu nového systému. Jak je potřeba systém zprovoznit, jak bude pracovat samostatně na základě vytvoření nových verzí stránek a jaké bude poskytovat služby uživatelům skrz webovou aplikaci.

5.1 Popis celého systému

Systém pro správu jazykových verzí by měl pomoci vývojářům v evidenci textů používaných na Portálu. Kvůli rychlosti je zapotřebí, aby se dosavadní jediný soubor rozdělil na více. Jako vhodné řešení se jeví rozdělení podle jednotlivých aplikací. Těmito aplikacemi jsou:

- Studis
- Teacher
- Portal
- V budoucnu možná i další

Tyto aplikace jsou uloženy v adresářích stejného jména v kořenovém adresáři Portálu. V PHP skriptech portálu dojde ke změně načítání datového souboru právě na ten odpovídající jeho aplikaci. Informaci, do které aplikace soubor patří, lze získat minimálně z jeho umístění.

Dále je požadováno uložení všech překládaných textů i s jejich hodnotami v databázi, aby k nim byl kdykoliv možný přístup. Aby byly jednotlivé samostatné soubory pro jednotlivé aplikace neustále aktuální, je potřeba, aby se po každém vydání nové verze zdrojových kódů – po akci commit nástroje SVN provedla kontrola změn v daných souborech. Pokud v souboru dojde ke změně volání překladatelských funkcí, musí být také změněn soubor patřící k dané aplikaci.

Nové texty budou ještě před vlastním vydáním nové verze uloženy v databázi. K tomuto bude sloužit webová aplikace která tuto funkci vývojářům umožní. Tato aplikace jim také umožní měnit již existující texty a to jak českou, tak i anglickou verzi. Vývojář tedy může zároveň i překládat, aby pro jednoduché překlady nebyl potřeba odborný překladatel, stejně tak jako tomu bylo doposud.

Jakmile dojde ke změně některého textu je potřeba opět v případě nutnosti provést změnu datových souborů pro jednotlivé aplikace.

Je také potřeba vytvořit webovou aplikaci pro překladatele, kteří po přihlášení obdrží seznam textů, které je potřeba přeložit. Aplikace jim musí umožnit vyžádat kontext daného textu od vývojáře. Po aplikaci se také požaduje, aby místo překladatele automaticky počítala počet přeložených znaků, což je potřeba pro vyplacení odměny.

Veškeré podstatné změny v databázi je potřeba zaznamenávat pro případ kontroly.

5.2 Aplikace pro vývojáře a překladatele

5.2.1 Neformální specifikace

Neformální specifikace definuje prvotní pohled na nový systém, jak by měl a neměl pracovat. Zahrnul jsem zde požadavky od jednotlivých stran, které budou aplikaci vyžívat. Tyto požadavky byly získány na základě analýzy z předchozí kapitoly.

Po aplikaci jsou ze strany vývojářů požadovány minimálně tyto služby:

- Možnost zobrazení souborů patřící do jednotlivé aplikace či všech souborů portálu
- Zobrazení všech textů vyskytující se ve zvoleném souboru
- Možnost vybrání konkrétního textu z textů zvoleného souboru a zobrazení jeho informací, mezi než patří:
    - Jméno
    - Česká varianta textu
    - Anglická varianta textu
    - Kdy a kým byl text vytvořen
    - Zda, kdy a kým byl text přeložen
- Možnost vyhledání textu určitého jména a zobrazení jeho informací stejně jako v minulém bodě
- Možnost změny vybraného textu a to tak, že vývojář uloží buď pouze českou variantu, kdy je potom po překladateli požadováno doplnit anglickou verzi textu, nebo uložit obě varianty. Potom vývojář vystupuje zároveň jako překladatel. Tato funkce je vyžadována pro potřeby malých změn, kdy není potřeba služby překladatele.
- Možnost vložení nového textu, přičemž aplikace musí zkontrolovat jestli již tento text neexistuje. V případě že ne, je vývojáři umožněno doplnit českou a anglickou verzi textu a opět uložit obě nebo pouze českou verzi.

Ze strany překladatelů nejsou jsou požadavky tak rozsáhlé, dáno je to tím, že překladatel texty „pouze“ překládá a nestará se o jejich management, jako vývojáři. Proto je pro překladatele potřeba zajistit následující:

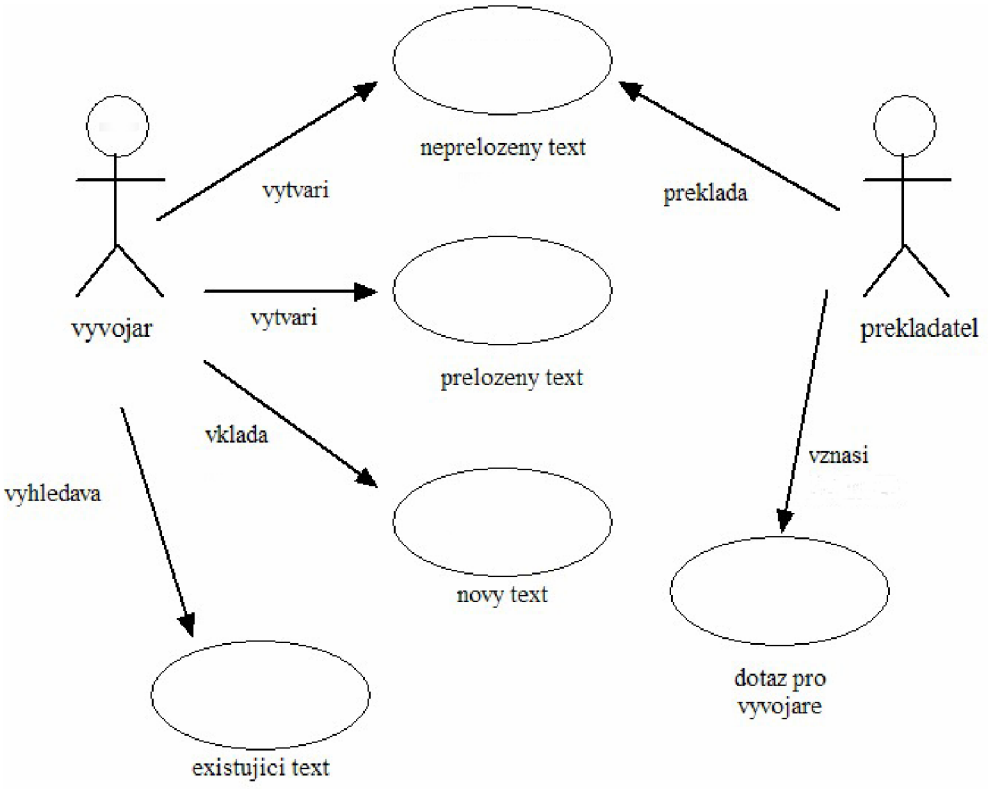
- Po přihlášení zobrazit seznam textů, které je potřeba přeložit.
- Překladatel vybere daný text, zobrazí se mu jeho česká varianta a další informace o textu:
    - Kdy byl text vytvořen/změněn
    - Kdo jej vytvořil – tato funkce je pro možnost vznesení dotazu na kontext, ve kterém se text vyskytuje, pokud jej překladatel potřebuje vědět.

- Dále je překladateli umožněno doplnit anglickou verzi textu a záznam uložit.
- Po aplikaci se požaduje, aby sama automaticky spočítala počet znaků, které překladatel za toto sezení přeložil.

Po aplikaci je obecně požadováno, aby zaznamenávala do logovacího souboru veškeré změny, ke kterým v databázi došlo spolu s identifikací uživatele a času změny pro možnost budoucího dohledání. Je také vhodné zaznamenávat, kdo se kdy přihlásil a odhlásil od aplikace, u překladatelů při odhlášení zapsat počet přeložených znaků. Dále je obecně požadováno, aby se po odhlášení uživatele v případě nutnosti spustilo generování nových verzí datových souborů jednotlivých aplikací.

5.2.2 Diagram případů použití aplikace

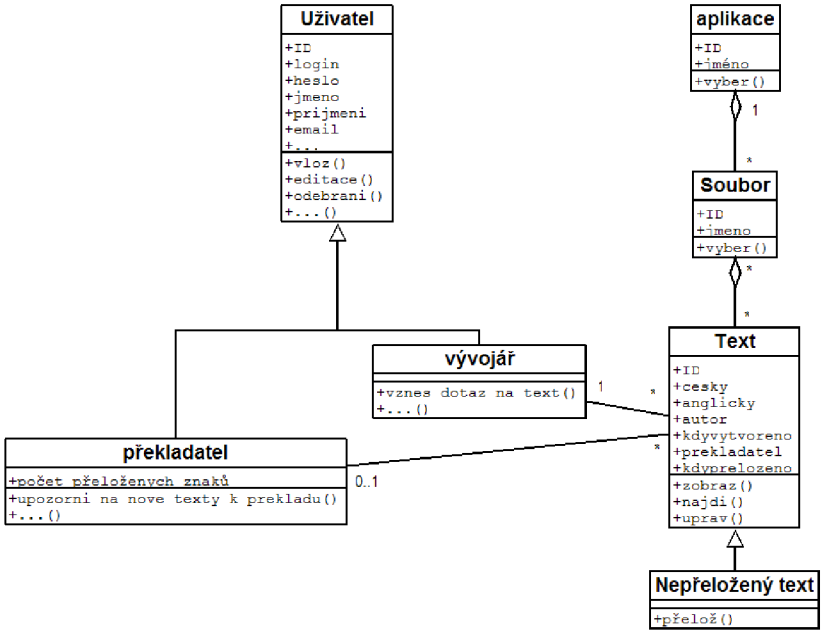
Daný model vycházející s jazyka UML slouží především k analýze modelu nového systému. Provádí zobrazení možností, které budou v systému nabízeny jednotlivým skupinám uživatelům. Use-case diagram, jak je tento diagram také označován, popisuje interakci uživatele se systémem pomocí tzv. scénářů. Na obrázku jsou vidět dva aktéři aplikace: vývojář a překladatel, jsou to jednotlivé skupiny uživatelů, které budou v systému pracovat.



Obr.12: Diagram případů použití aplikace

5.2.3 E-R diagram

Na Obr. 13 je zobrazen Entity – Relation diagram, tedy diagram tříd jejichž informace chceme mít uloženy v databázi a vztahů mezi těmito entitními množinami. Na základě tohoto diagramu a předchozí neformální specifikace a diagramu případu užití je navrhnuta struktura databáze.



Obr. 13: E-R diagram aplikace

5.3 Návrh databáze

Databáze je tvořena řadou tabulek umožňujících uchovávat informace o všech aplikacích, jím příslušejících souborů. Dále informace o textech a uživatelích. Pro názvy jednotlivých tabulek jsem použil české pojmenování bez diakritiky. Záměrem bylo, aby byly jednoduché, ale výstižné. Ten samý postup jsem využíval i u atributů jednotlivých tabulek. Držel jsem se danými konvencemi pro pojmenovávání databázových objektů na CVIS. Primární klíč tabulky je tvořen spojením názvu tabulky a prefixem PK, cizí klíče prefixem FK, názvem referované tabulky a názvem referující tabulky. V případě více cizích klíčů – tabulka text je přidán ještě postfix charakterizující klíč.

5.3.1 Tabulka uživatel

V tabulce uživatel jsou uvedeny záznamy o všech uživatelích systému, tedy jak vývojářů, tak překladatelů. Data do této tabulky se musí dát ručně přes terminál SQL. Webová aplikace totiž neumožňuje editaci studentů a jelikož jsou tito uživatelé stálí, není toto ani potřeba. Údaje z této tabulky se používají ve webové aplikaci, při zobrazování, kdo provedl jakou změnu textů atd., ale

také v logovacích souborech. Speciální je uživatel parser s identifikačním číslem jedna. Tento uživatel je označován jako autor a překladatel u textů přidaných při prvotním sestavování systému.

Atribut	Význam
Pk_uzivatel	Identifikacni cislo
Login	Přihlašovací jméno uživatele
Heslo	Heslo uživatele
Jmeno	Křestní jmén uživatele
Prijmeni	Příjmení uživatele
Email	Emailová adresa uživatele
Role	Nabývá hodnot • 1 pro vývojáře • 2 pro překladatele (v budoucnu možno rozšířit o administrátora)

5.3.2 Tabulka aplikace

V tabulce aplikace jsou uloženy všechny systémem zpracovávané aplikace, data jsou tam daná při prvotním spuštění systému a v případě zavedení nové aplikace na portálu.

Atribut	Význam
pk_aplikace	Primární klíč záznamu
Jmeno	Jméno aplikace

5.3.3 Tabulka soubor

V tabulce soubor jsou uloženy veškeré systémem použité soubory, u kterých kdykoliv byly volány překladatelské funkce. V případě, že po vydání nové revize jsou ze souboru odebrány všechny volání těchto překladatelských funkcí, soubor v tabulce zůstává.

Atribut	Význam
pk_soubor	Identifikátor souboru
Jmeno	Jméno souboru i s nadřazenými adresáři až do hlavního adresáře s celým portálem.
fk_aplikace_soubor	Cizí klíč do tabulky aplikace

5.3.4 Tabulka text

V tabulce text jsou uloženy veškeré texty spolu s jejich vlastnostmi. Záznamy jsou do této tabulky dávány automatickými skripty spuštěnými při sestavování systému či operaci commit. Ale možno je také přidat nové texty v aplikaci pro vývojáře. Data jsou zde měněna pouze aplikacemi pro vývojáře a překladatele, a to při změně české nebo současně i anglické varianty textu vývojářem nebo překladem nepřeloženého textu. V případě, že již text není použit v žádném souboru, není tento text odstraněn, aby jej bylo možno využít v budoucnu. Jelikož potom tento text není zapsán v datovém souboru s překlady žádné aplikace, nezpomaluje běh portálu.

Atribut	Význam
Pk_text	Jednoznačný identifikátor textu, jde o stejný řetězec, jako je index v asociativním poli. Ve zdrojových kódech jsou právě překladatelské funkce volány s těmito parametry. Tyto texty jsou jednoznačné, proto mohou být uloženy pod stejným řetězcem.
Cs	Česká veze textu
En	Anglická verze textu
Fk_uzivatel_text_vyvojar	Cizí klíč referující do tabulky uživatel. Tento údaj nám říká, který uživatel položku vytvořil. V případě prvotního sestavování systému je vložena hodnota 1, což odpovídá uživatel parser. Tato položka je změněna vždy když je změněna česká verze textu, zapsána je hodnota odpovídající uživateli, který změnu provedl. Nebo vložení nového textu.
Fk_uzivatel_text_prekladatel	Cizí klíč referující do tabulky uživatel. V tomto sloupci tabulky jsou pro jednotlivé záznamy uvedeni hodnotu referující na uživatele, který provedl poslední změnu anglické verze tohoto textu. V případě prvotního sestavování systému je stejně jako u předcházejícího atributu uvedena hodnota 1 tedy parser. Hodnota je změněna v případě překladu nepřeloženého textu v aplikaci pro uživatele či pokud vývojář odsouhlasí změnu anglické verze. Pokud vývojář

	uloží pouze českou verzi textu, je položka naplněna hodnotou NULL, která nám udává, že soubor není přeložený a texty s takovou hodnotou jsou při přihlášení jakéhokoliv překladatele zobrazeny a nabídnuty na překlad.
Vytvořeno	Hodnota udávající datum a čas poslední změny české verze textu. Při sestavování systému je naplněna datem a časem vložení do databáze.
Přeloženo	Hodnota nám říká, kdy byla naposledy změněna anglická verze textu. Hodnota je zapsána v případě, kdy vývojář změní obě jazykové verze textu či vloží nový text v obou jazykový verzích. V opačném případě je hodnota nastavena na NULL, což aplikaci pro překladatele říká, že text není přeložen do angličtiny, nebo jeho znění není aktuální. V případě uložení překladu překladatelem je tato hodnota opět zaktualizována.

5.3.5 Tabulka textvsouboru

Jediná vazební tabulka v databázi zabezpečuje relaci mezi souborem a textem. Je totiž možné, že jeden konkrétní text je použit ve více souborech a zároveň že každý soubor může používat více textů. Tato tabulka nemá vlastní primární klíč určující tento záznam, ale pouze atributy odkazující na primární klíč v tabulce soubor a primární klíč v tabulce text. Položky této tabulky jsou využívány aplikací, kdy si vývojář prohlíží, jaké texty jsou použity v daném souboru a dále pro sestavování aktuálních datových souborů s překlady pro jednotlivé aplikace. V případě, že skript spuštěný po operaci commit zjistí, že se změnilo volání překladatelské funkce v daném souboru pro daný text, buď záznam přidá, nebo jej odebere. Skripty jsou napsány tak, aby i pro více volání jednoho textu v jednom souboru byl záznam v této tabulce pouze jednou. Stejně tak, aby byl záznam vymazán, musí být odstraněny všechna volání překladatelských funkcí pro daný text v určitém souboru.

Atribut	Význam
Fk_soubor_textvsouboru	Cizí klíč do tabulky soubor
Fk_text_textvsouboru	Cizí klíč do tabulky text

5.4 Sestavovací skript

Pro zavedení celého systému je potřeba nejprve vytvořit sestavovací skript, který projde všechny soubory v adresáři Portálu za účelem nalezení překladatelských funkcí. Je vhodné, aby neprocházel úplně všechny soubory, protože již ze struktury portálu víme, že v některých adresářích nejsou naše překladatelské funkce volány. Jde o adresáře kde jsou knihovny z cizích zdrojů, jako jsou knihovny pro zpracování pdf, tvorbu grafů atd., dále se jedná o adresáře s obrázky.

Skript prochází jednotlivé soubory a hledá v nich výskyty volání překladatelských funkcí. Pokud je argument překladatelské funkce přeložen v globálním datovém souboru s překlady, je text vložen do tabulky text a vytvořen záznam v tabulce textvsouboru pro danou dvojici soubor-text. Po skriptu se požaduje aby nás informoval o nalezených voláních překladatelských funkcí pro jednotlivé soubory spolu s informováním o tom, zda je text v globálním datovém souboru nalezen či nikoliv.

Dále je potřeba pro jednotlivé aplikace vygenerovat samostatné datové soubory. Je také vhodné informovat vývojáře, který tento skript spustil, o tomto generování souborů a nakonec je potřeba zobrazit naleznuté nepřeložené volání funkcí pro dané soubory.

5.5 Skript spouštěný po SVN operaci commit

Po každé operaci commit nástroje SVN je potřeba zajistit zaktualizování dat v databázi. Tento skript funguje podobně jako sestavovací skript, rozdíl je ale v tom, že nesmí procházet všechny soubory, což by zbytečně prodlužovalo proces aktualizace. Aktualizace repository je uskutečněna několikrát denně (obecně do 10 aktualizací/den). Avšak i při tak relativně malém počtu spuštění skriptu musíme hledět na jeho efektivnost..

V tomto případě tedy skript prochází jednotlivé soubory, a pokud najde nějakou změnu volání překladatelských funkcí, musí tomu adekvátně přizpůsobit databázi. Pokud je to potřeba, je na konci skriptu vygenerován jeden nebo více změněných datových souborů pro dané aplikace.

6 Realizace

Pro psaní kódu v jazyce PHP jsem použil editor PSPad [25], který je volně přístupný a umožňuje zvýraznění syntaxe v široké řadě programovacích jazyků. Jde o vlastnost která programátorovi značně usnadňuje orientaci ve vyvíjeném kódu.

Systém jsem vyvíjel na lokálním počítači, kde jsem měl zkopírované všechny aktuální zdrojové kódy portálu. Použil jsem balík WAMP, který používám standardně a mám s ním zkušenosti. Tento balík obsahuje Apache Server, PHP ve verzi 5 a databázi MySQL. Jeho instalace a nastavení je velmi jednoduché, protože téměř vše se nainstaluje a nastaví automaticky.

Aby byla zaručena přenositelnost do prostředí Portálu, používal jsem v PHP funkce pro práci s datovým zdrojem ODBC. Díky tomu je přenositelnost zaručena. U Oracle databáze, která je v Portálu využívaná je ODBC zdroj také definován a samotní vývojáři jej ve svých skriptech využívají.

6.1 Regulární výrazy v PHP

Jak pro samotné parsery zdrojových kódu tak i ve Webové aplikaci jsem velmi často používal funkce pro práci s regulárními výrazy. Tyto funkce jsou v PHP k dispozici jako součást standardního modulu, který je vždy dostupný. Každá funkce pro práci s regulárními výrazy, ať už má za úkol cokoliv vždy pracuje s dodaným řetězcem a vzorem regulárního výrazu. Tato dvojice je vždy parametry dané funkce. Vzor je následně aplikován na dodaný řetězec.

Vzory regulárních výrazů v PHP vycházejí z programovacího jazyku PERL, zde uvádím základní konstrukce:

- Každý regulární výraz musí být z obou stran označen omezovacím znakem (anglicky delimiter), jako tento omezovač může být použit jakýkoliv znak mimo číslici , písmene a zpětného lomítka. Pokud chceme tento zvolení znak použít uvnitř vzoru musíme mu předřadit zpětné lomítko, tím bude jeho význam oddělovače pro tento výskyt zrušen.
- Skupině znaků v uzavřené v hranatých závorkách např. [abc] odpovídá výskyt právě jednoho z těchto znaků.
- Pokud je znaků více a navíc spolu sousedí, můžeme je nahradit intervalem. Například pro všechny číslice použijeme zápis [0-9]
- Výčet znaků můžeme také negovat a to předřazením znaku „ ^ “. Libovolný znak mimo číslice bude tedy zapsán [^0-9].
- Jsou připraveny zkratky, které zastupují často využívané množiny znaků:
    - \d – číslice

    - \D – znak mimo číslice
    - \w – alfanumerický znak
    - \W – znak mimo alfanumerický znak
    - \s – prázdný znak
    - \S – neprázdný znak
- Tečka („ . “) odpovídá jakémukoliv znaku kromě znaku konce řádku
- Pokud potřebujeme zapsat výskyt speciálního znaku přeřadíme mu zpětné lomítko, například tedy pokud chceme aby se v řetězci vyskytla tečka použijeme zápis „ \. ”
- Opakovací konstrukce umožňují akceptování řetězce u kterého předem nevíme jeho délku. Jejich umístění je právě za daný požadovaný znak či množinu znaků. Konstrukce jsou tyto:
    - * – libovolný počet opakování
    - + – alespoň jeden výskyt
    - ? – nula až jeden výskyt
    - {2,8} – alespoň dva a nejvíce pět výskytů
    - {,7} – nejvíce sedm výskytů
    - {8,} – nejméně osm výskytů
    - {6} – požadujeme přesně šest výskytů
- Pomocí kulatých závorek se vytvářejí celku který se může opakovat. Pro určení počtu opakování se opět použije opakovací konstrukce.

Příklad PHP funkce pro práci s regulárním výrazem:

```
int preg_match_all(string $pattern, string $subject, array &$matches )
```

Funkce prohledává povinný parametr subject a hledá výskyty vzorů dodané v proměnné pattern, do té si před vlastním voláním zapíšeme vzor. Funkce vrací počet shod s celým vzorem v řetězci. Hodnota může být tedy také 0 nebo FALSE v případě výskytu chyby. Do dalšího parametru – pole matches jsou uloženy shody vzoru a předmětu prohledáváni. V položce matches[0] je první množina shod a v položce matches[0][0] je shoda s celým vzorem. V položce matches[0][1] je shoda s první skupinou (určená pomocí kulatých závorek) atd. Stejně tak pole $matches[1] obsahuje druhou množinu shod, opět prvně s celým vzorem, dále pak s jednotlivými podvzory.

Při tvorbě vzorů si musíme dávat pozor, nejenom na zrušení speciálních významů znaků pro zapsání regulárních výrazů, ale také na to, že sám jazyk PHP používá zpětné lomítko jako tento prostředek, obecně označován jako escape sekvence. Proto musíme zpětné lomítko v některých případech zdvojit. Více o použití regulárních výrazů v PHP je možné zjistit ve zdroji [1].

6.2 Předávání hodnot mezi skripty

Aplikace napsané v PHP jsou složeny z skriptů, které se vzájemné mohou volat mezi sebou. K tomu používají proměnné, které si předávají jako parametry funkcí nebo pomocí globálních proměnných. Pokud však skript generující na svůj výstup stránku HTML ukončí svůj běh, proměnné obvykle zanikají. Uživatel dostane vygenerovaný výstup – stránku, která může opět volat nějaký skript. A to odesláním formuláře, nebo kliknutím na odkaz. Potom je potřeba volanému skriptu předat nějaké hodnoty.

Pro odesílání z formuláře se většinou používá metoda POST, kdy potom PHP skript má proměnné k dispozici poli `$_POST[]` s indexem jména formulářového prvku. Tyto proměnné nejsou běžně uživateli viditelné, pouze v případě, že si zobrazí zdrojový kód stránky.
Druhá metoda, se označuje jako GET, data jsou přijímajícímu PHP skriptu opět přístupná v poli `$_GET[]`. Tyto hodnoty jsou však zobrazeny přímo v URL adrese. Není tedy vhodné touto cestou posílat citlivé údaje. V obou případech proměnné zanikají.

Pro zajištění přetrvání proměnné i po skončení skriptu se využívá tzv. super globální proměnná SESSION. Data uložená v SESSION mají trvání po celou dobu relace a jsou viditelná ze všech skriptů. Princip těchto proměnných je založen na identifikaci prohlížeče. Jakmile se konkrétní internetový prohlížeč přihlásí k webu, který využívá SESSION proměnné, server identifikuje prohlížeč a pokud jsou pro něj uložené proměnné SESSION jsou automaticky přístupné pro všechny skripty. SESSION může vzniknout automaticky po návštěvě určitého serveru (pokud zjistí, že pro daný prohlížeč SESSION není založena), nebo teprve po přihlášení. To záleží na charakteru aplikace. Toho je možné využít například pro informování uživatele o jeho poslední návštěvě, jeho uživatelského jména atd.

V konfiguraci serveru je možno nastavit, zda se mají SESSION zakládat automaticky při návštěvě jakékoliv stránky, potom není nutné volat v každém skriptu, který může být navštíven PHP funkci `session_start`. Právě protože jsou data uložen na serveru, je na něm potřeba nastavit dobu platnosti SESSION proměnných.

Já jsem používal proměnné SESSION k uložení informací o přihlášeném uživateli a také dalších informací. Například o načtených databázích, souborech nebo textech. Na výstup potom byly poslány jenom hodnoty které uživatel chtěl zobrazit. A zase při posílání dat směrem zpět od prohlížeče k serveru bylo posíláno jenom minimální množství dat, například informace o stisknutí tlačítka, či vybrání konkrétního textu ze seznamu textů. Jelikož byly hodnoty uložené v SESSION, nemuseli se znovu data dolovat z databáze. Všechny proměnné SESSION jsem zrušil po stisknutí tlačítka odhlásit, tím dojde ke korektnímu ukončení aplikace.

6.3 Webová aplikace

Webová aplikace je tvořena pomocí dvou souborů, jeden je využit pro přihlášení uživatele a druhý funguje jako samotná aplikace. Data jsou mezi skripty předávána pomocí proměnných SESSION a přesměrování probíhá pomocí odesílání HTTP hlavičky funkcí `header()`. Volání této funkce musí být provedeno vždy před prvním výstupem ze skriptu. V hlavičce je uvedena adresa druhého souboru, čímž dojde k automatickému přechodu na daný skript.

Dále je pro chod aplikace potřeba mít nastavenou dostupnou databázi, pro což u všech skriptů slouží konfigurační soubor `db.conf`. V tomto souboru se definují konstanty pro daný ODBC zdroj.

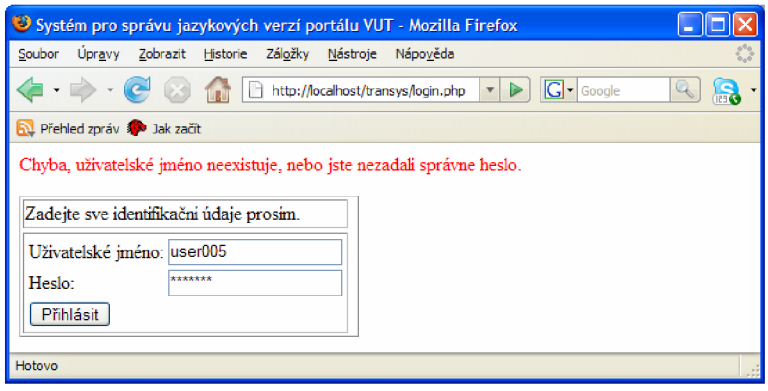
6.3.1 Soubor login.php

Jedná se o skript, který nejprve zkontroluje přístupnost databáze. Pokud není dostupná, zobrazí varovnou hlášku. Pokud je připojení navázáno, skript zkontroluje, zda mu nebyly metodou POST poslány hodnoty uživatelského hesla a jména. Tyto hodnoty jsou dostupné pouze v případě, že skript byl volán nějakým jiným skriptem, který mu tyto proměnné posla uvedenou metodou. Obvykle mu mohou být poslány sebou samým. Tyto hodnoty bývají poslány z formuláře, který je zobrazen právě v případě neexistence těchto proměnných v tomto běhu skriptu.

V případě existence proměnných se skript podívá do databáze, zda je tam uživatel jemuž tyto údaje odpovídají. Pokud tam takový uživatel není, je zobrazena varovná hláška spolu s přihlašovacím formulářem, toto je zobrazeno na Obr. 14 .

Pokud je uživatel úspěšně autorizován, jsou skriptem nastaveny SESSION proměnné identifikujícího daného uživatele. Z nejdůležitějších proměnných je jeho login, jeho ID, email a role. Právě role, určuje která varianta aplikace je uživateli zpřístupněna. To je však již úkol souboru index.php. Dále je na konec logovacího souboru učeného proměnnou `$logFile` zapsán záznam o přihlášení daného uživatele do aplikace.

```
2007-07-12 10:15:18 xpavli29  : prihlaseni k aplikaci.
```



Obr. 14 :Neúspěšné přihlášení

6.3.2 Soubor index.php

V tomto souboru nejprve proběhne nastavení proměnných určující jméno logovacího souboru a adresáře, kde jsou uloženy stránky Portálu.

Následně je provedena kontrola existence SESSION proměnné s hodnotou uživatelova loginu. Pokud taková hodnota není nastavena, je provedeno automatické ukončení tohoto skriptu přesměrováním na skript přihlašovací.

Pokud tomu tak není, nahraje se konfigurační soubor databáze a zkontroluje se její dostupnost. V případě neúspěšného připojení je zapsán záznam do logovacího souboru a uživatel odhlášen. Také zde je jméno logovacího souboru určeno proměnnou `$logFile`. Ta je deklarována na začátku souboru pro možnost její jednoduché změny.

```
2007-07-14 13:37:09 xpavli29  : Nepodařilo se připojit k DB, nastavte ji
prosím.
2007-07-14 13:37:09 xpavli29  : odhlaseni z aplikace.
```

V případě, že všechny testy proběhli úspěšně, je spuštěna vlastní aplikace, vzhled a chování aplikace je jak pro vývojáře, tak překladatele rozdílný. Vlastnímu popisu aplikace se věnuji v následující kapitole.

Soubor index.php generuje dynamicky aplikaci na základě SESSION, normálních a také POST proměnných. Pokud například bylo stisknuto tlačítko pro změnu české varianty texty, je přítomna proměnná `$_POST[change_czech]`, poté se zkontroluje přítomnost proměnné `$_SESSION[selected_text]`, což nám říká, že chceme změnit pouze českou variantu textu, který byl vybrán pomocí průchodu stromovou strukturou aplikace –> soubor –> text. Pokud by byla přítomna proměnná `$_SESSION[new_text]`, znamená to, že měněný text je nový text. Na základě těchto poznatků jsou provedeny záznamy v databázi a záznamy v logovacím souboru.

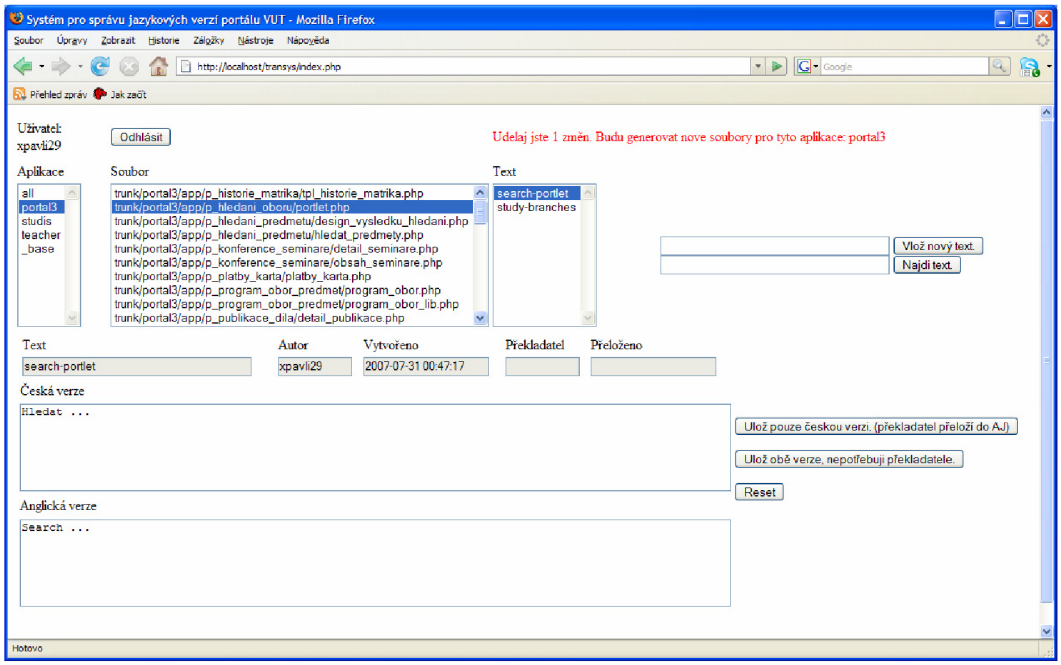
Při stisknutí tlačítka odhlásit, skript zkontroluje přítomnost proměnné $_SESSION['changed_applications'], ve které jsou v případě změny dat v databázi zaznamenány aplikace, kterých se tato změna dotkla. Proměnná je asociativní pole, takže opakovaným uložením hodnoty pod stejným klíčem dojde jenom k přepsání hodnoty. Při odhlášení projdeme toto pole a pro každý index – aplikaci spustíme generování datového souboru. Do losovacího souboru je zapsáno odhlášení daného uživatele. V případě překladatele je zaznamenaný celkový počet změněných znaků. Toto může sloužit pro pozdější kontrolu a stanovení odměn. Je také zapsáno generování nových datových souborů pro změněné aplikace:

```
2007-07-18 14:15:09 galova    : prihlaseni k aplikaci.
2007-07-18 14:15:12 galova     : Příručka oboru : preklad do AJ( 12 zn.):
Branch manual
```

```
2007-07-18 14:15:14 galova     : odhlaseni prekladatele z aplikace, celkovy
pocet znaku: 12(0.01 NS).
2007-07-18 14:15:14 : generovani noveho datoveho souboru pro aplikaci:
_base, soubor: trunk/_base/func_tr_data.php
```

6.3.3 Popis vlastní aplikace

Na Obr. 15 je zobrazení aplikace tak, jak se generuje pro vývojáře:



Obr. 15 : Aplikace pro vývojáře

Aplikace pro vývojáře je rozdělena do třech hlavních částí.

V první jsou zobrazeny informace o přihlášeném vývojáři, tedy jeho login. Dále zde má uživatel tlačítko pro odhlášení. Dalším prvkem této části je informační pole zobrazující, co bylo právě uděláno, a jestli budou generovány nové datové soubory a pokud ano, pro které aplikace.

Druhá část (vybírací) umožňuje vývojáři vybírat texty a to pomocí průchodu stromovou strukturou. Kdy nejprve musí vybrat aplikaci, je také možno vybrat všechny současně. Dále si vybere konkrétní soubor a nakonec jeden z textů, který je v tomto souboru volán nějakou překladatelskou funkcí. Jakmile si vybere tento text, jsou v další části aplikace zobrazeny informace o tomto textu.

Všechna tato vybírání položek z objektů select formuláře jsou odesílány pomocí funkce JavaScriptu při výběru položky. Není tedy potřeba po výběru stisknout nějaké tlačítko, které by formulář odesílalo. Tento způsob vybíraní je pro živitele aplikace mnohem přívětivější.

Dále je možno v této části aplikace vyhledat konkrétní text podle jeho jména, pokud je takový text v databázi, jsou jeho informace nahrány do informační/ editační části aplikace.

Poslední funkcí této části aplikace je vložení nového textu do databáze textů. Po zadání jeho jména a odeslání, nyní již stejně jako při vyhledávání přes standardní tlačítko, dojde ke kontrole, zda již v databázi takový text neexistuje a v případě že ne, je automaticky do informační/ editační části aplikace předán nový text.

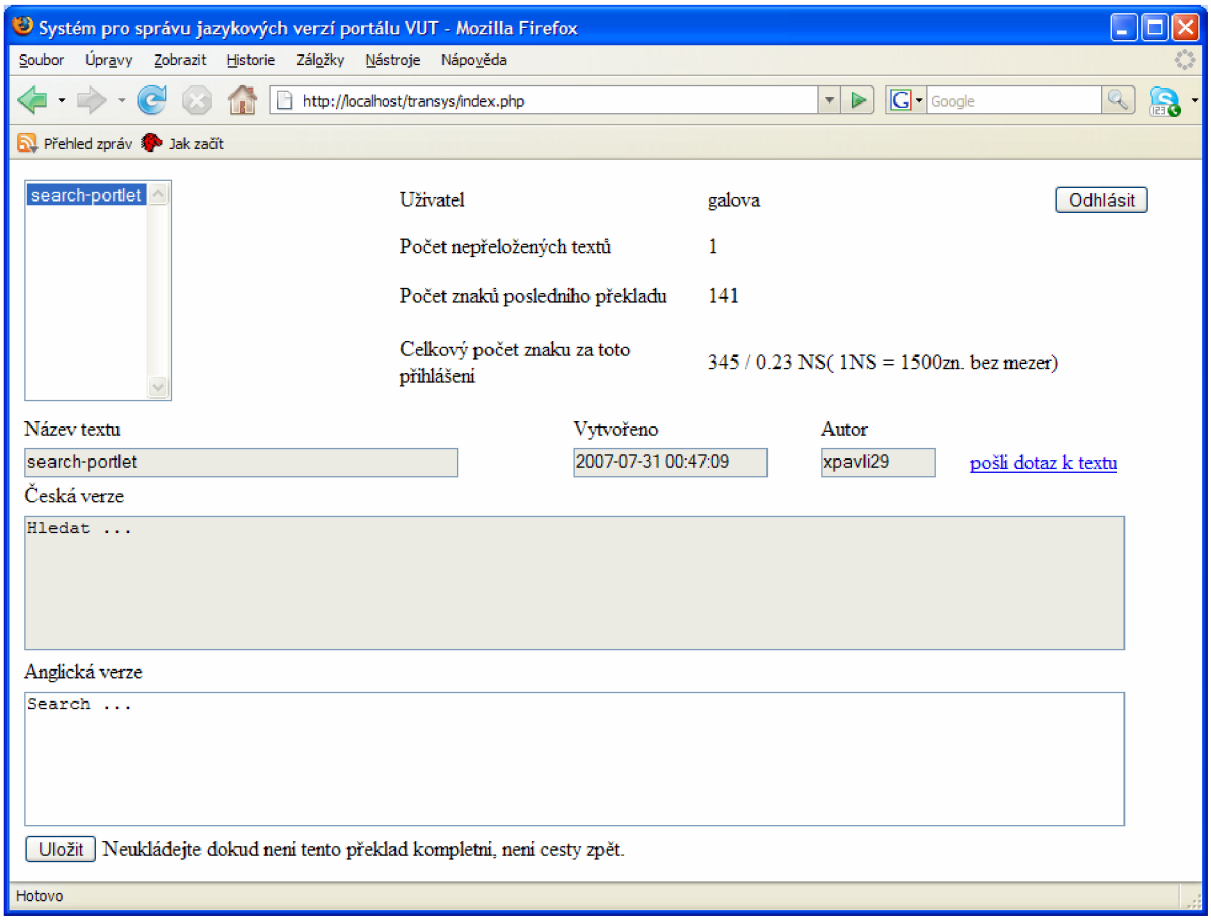
V poslední části, kterou jsem v předchozím textu označoval jako informačně/ editační část aplikace jsou primárně zobrazeny informace o vybraném, nalezeném nebo nově vloženém textu. Z těchto informací vývojář může zjistit, kdo a kdy naposledy změnil českou nebo anglickou verzi textu. Pokud je položka překladatel a přeloženo prázdná, znamená to, že je text nepřeložený a čeká na překlad od překladatele.

V této části aplikace uživatel nemusí data jenom prohlížet, ale má také možnost jejich úpravy. Pokud chce může změnit pouze českou verzi, v tom případě je text označen (sloupce překladatel a přeloženo) jako nepřeložený a čeká na překlad od překladatele. V databázi se aktualizuje pouze česká verze textu, jako autor se nastaví aktuální uživatel, datum a čas se nastaví pomocí SQL funkce `NOW()`, položky překladatel a přeloženy se nastaví jako NULL. Zároveň se do logovacího souboru provede záznam, který odpovídá, tomu, jestli byl text již před přihlášením v databázi, nebo jej uživatel přidal nový, příklady záznamů můžete vidět zde:

```
2007-07-12 14:11:49 xpavli29  : kniha : vlozen novy text, pouze CS verze : kniha
2007-07-12 14:11:05 xpavli29  : study-branches : zmena CS verze : Studijní programy a obory
```

Pokud uživatel vybere tlačítko pro uložení obou jazykových verzí, jsou při updatu nebo insertu do tabulky vyplněny i položky určující překladatele a datum a čas přeložení. Opět je do logovacího souboru zapsán záznam. Příklady záznamů:

2007-07-12 14:55:03 xpavli29 : kuchyn : vlozen novy text, CS i AJ verze : kuchyň : kitchen
2007-07-12 14:56:33 xpavli29 : faculties : zmena CS i AJ verze : Fakulta : Faculty

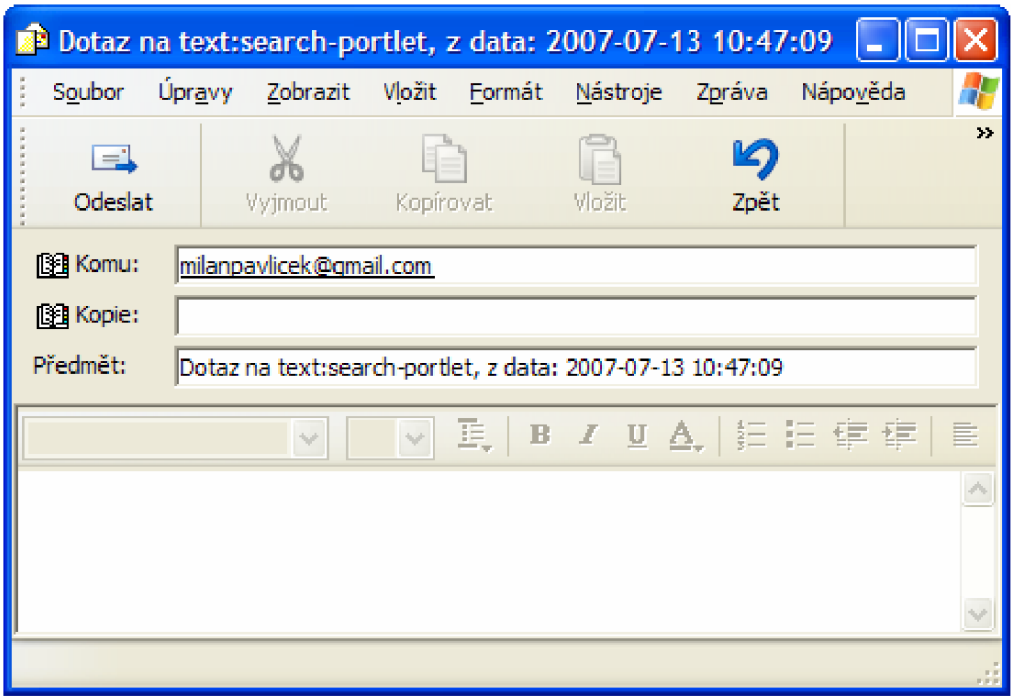


Obr. 16: Aplikace pro překladatele

Aplikace pro překladatele (Obr. 16) má podobné, rozložení, ale podle analýzy poskytuje jiné možnosti. Po přihlášení je uživateli zobrazen seznam textů, které je potřeba přeložit. Vybráním jednoho z textů dojde k jeho načtení do spodní části aplikace. Zde si překladatel může přečíst informace, kdy byl text vytvořen (změněn) a kým.

Pro možnost kontaktování vývojáře z důvodu potřeby získání podrobností o textu je vytvořen odkaz, který po kliknutí přímo umožňuje poslání emailu odpovědnému vývojáři. Email je již předpřipraven (Obr.17), takže překladatel může jenom doplnit otázku a odeslat. Tento email je poslán z emailového klienta který musí mít překladatel nainstalovaný. Tato podmínka je na u překladatelů z Ústavu jazyků, FSI VUT Brno, kteří překlady zajišťují splněna.

Překladateli je oproti vývojáři umožněna změna pouze v textovém poli určeného pro anglickou verzi textu. Po stisknutí tlačítka Uložit. Dojde k aktualizaci záznamu v databázi pro daný soubor. Obnoví se seznam nepřeložených textů. Dále se spočítá počet znaků posledního překladu a také se přepočítá celkový počet přeložených znaků překladatelem za toto přihlášení. Hodnota je uvedena jak ve znacích, tak také v normo stranách. Jak již bylo řečeno při odhlášení překladatele je zaznamenán celkový počet změněných textů při tomto připojení.



Obr 17. Připravená emailová zpráva pro vývojáře

6.4 Sestavovací skript

Sestavovací skript slouží k prvotnímu naplnění databáze a vytvoření samostatných datových souborů pro jednotlivé aplikace.

Soubor vyžaduje konfigurační soubor pro přístup k databázi – soubor db.conf. Dále je na začátku naplněn důležitými proměnnými:

- Jménem souboru, kde doposud byly uloženy všechny překlady.
- Startovacím adresářem, ve kterém jsou umístěny soubory portálu.
- Jménem pro nové jednotlivé soubory s překlady
- Polem, jehož položky jsou adresáře, které se mají při průchodu přeskočit.

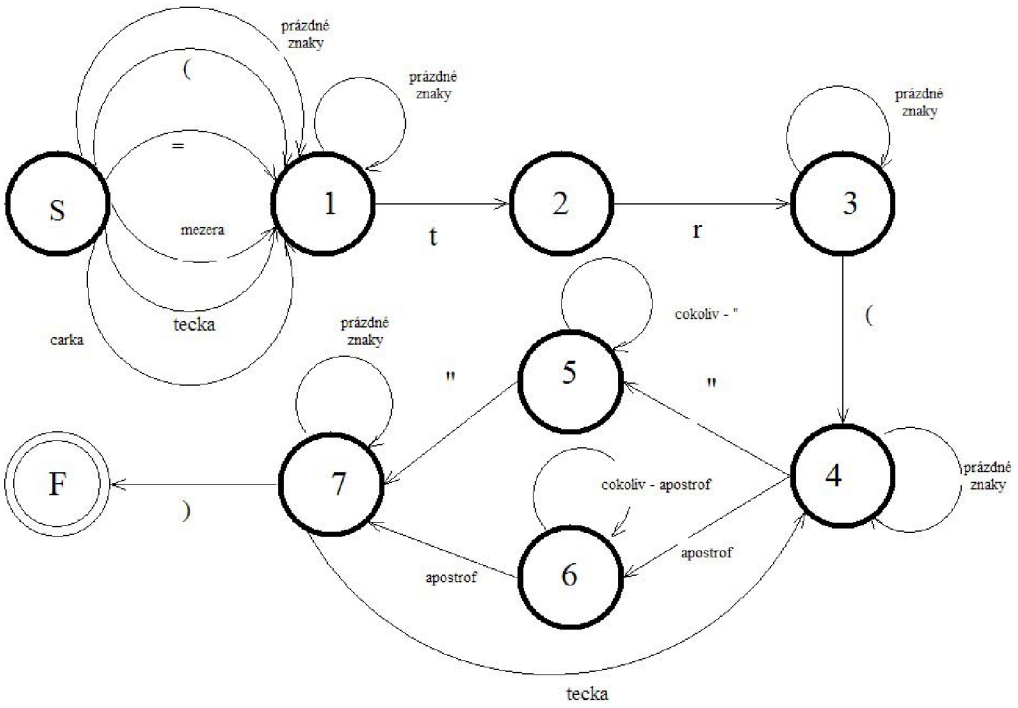
Skript nejprve zkontroluje existenci globálního datového souboru a možnosti připojení k databázi. Pokud s těmito operacemi nenastanou problémy, načte všechny položky globálního datového souboru, tyto hodnoty stejně jako překladové funkce načte do globálního asociativního pole.

Nejprve skript projde startovací adresář a do tabulky aplikace vloží jednotlivé aplikace. Tyto aplikace bude dále používat ve svém běhu.

Pro rekurzivní průchod všemi adresáři skript používá funkci `Dir_scan()`. Tu nejprve spustí na startovací adresář. Princip této funkce spočívá v tom, že prochází adresář, který je jejím

parametrem. Na všechny jeho podadresáře mimo položky ukazující na sebe sama a nadřazený adresář spustí opět funkci `DirScan`. Pouze vynechává adresáře určené na začátku skriptu.

Na všechny soubory je spuštěna funkce `Parser()`. Tato funkce zpracovává dodaný soubor. Nejprve vypíše na výstup o jaký soubor se jedná. Pomocí regulárních výrazů hledá výskyty překladatelských funkcí v tomto souboru. Na Obr. 18 je konečný automat odpovídající těmto voláním. Jde o konečný automat pro volání `funkce tr()`. Tato funkce se vyskytuje buď na pravé straně přiřazovacího operátoru, jako první, druhý nebo n-tý parametr jiné funkce, nebo je její výstup použit pro spojení s jiným řetězcem – operátor spojení řetězců „ . “ , který v PHP slouží ke spojování řetězců. Tato funkce se vyskytuje s parametrem řetězec či skupinu řetězců spojenými pomocí operátoru „ . “. Pokud jsou překladatelské funkce volány s doplňujícími parametry, jde vždy o číselnou hodnotu., proto posledním znakem který ukončuje konečný automat je pravá kulatá závorka. Prázdným znakem mám namysli prázdné znaky v používané v textových souborech.



Obr. 18: Graf konečného automatu přijímajícího volání funkce `tr()`

Pokud se v souboru vyskytují nějaké volání těchto překladatelských funkcí, jsou jejich argumenty postupně zpracovávány. Pokud je argumentem překladové funkce více spojených řetězců, je nutné tyto spojit dohromady. Pokud je argument nalezen v globálním datovém slovníku, je argument přeložený. Proto jej pokud ještě v databázi není, do ní vložíme. Dále pokud ještě není soubor vložený v databázi vložíme i jej. A nakonec, pokud ještě takový záznam neexistuje vložíme záznam do tabulky textvsouboru pro tento text a tento soubor. Pokud klíč v datovém poli nenajdeme, je

vložen do pole nepřeloženo spolu se jménem souboru, ve kterém byl volán.. Pro pozdější výpis na stdout.

Nakonec skriptu generujeme jednotlivé datové slovníky pro aplikace a zobrazíme které klíče nebyly v globálním datovém souboru a tudíž ani nyní nejsou v jednotlivých samostatných datových souborech přeloženy.Ukázka výstupu tohoto skriptu je v příloze

6.5 Skript volaný po SVN operaci commit

Tento skript je spouštěn po SVN operaci commit. Je pro něj tedy potřeba v nastavení SVN vytvořit tzv. commit hook (háček), který zařídí jeho spuštění po každém vystavení nové verze zdrojových kódů.

Je vyžadováno, aby se již neprocházeli všechny zdrojové soubory, ale jenom ty, které byly modifikovány, proto součástí tohoto skriptu je kód pro načtení souboru ve kterém se zaznamenávají změny v repositury. Tento soubor má přesně danou strukturu, proto je možné jej opět zpracovat pomocí regulárních výrazů. Je zapotřebí aby byly načteny i svn statusy pro jednotlivé soubory.

Poté je na jeden každý soubor spuštěn parser, který opět vyhledává volání překladatelských funkcí podle uvedeného konečného automatu. Tentokrát je ale potřeba pro každý soubor nejprve zjistit, jaké texty v něm byly použity v minulosti. Pokud najdeme takový klíč, který v minulosti volaný nebyl a je přeložený, je do vazební tabulky vložen záznam pro tuto dvojici. Pokud v nové verzi souboru ubylo volání nějakého klíče, musí být z této vazební tabulky takový záznam odebrán. Navíc je nutné si evidovat, u kterých aplikací došlo ke změně, pro tyto aplikace je na konci skriptu vygenerována nová verze zdrojových souborů. Výstup tohoto skriptu je opět v příloze.

7 Integrace

Vlastní integrace do Portálu VUT se neuskutečnila, ale jelikož byl systém naprogramován pomocí prostředků, které jsou všechny dostupné i v prostředí CVIS, nemela by integrace činit větší problémy. Problémy k řešení se však vyskytují ve volání překladatelských funkcí a datovém souboru s překlady. Před vlastní integrací do Portálu VUT se nejprve musí vyřešit:

Eliminace opakovaných klíčů v globálním datovém souboru. Právě kvůli již tolikrát zmiňované velikosti datového souboru se v něm vyskytují opakovaně klíče s různými hodnotami pro českou a anglickou verzi. Tato přítomnost stejných klíčů s různými hodnotami může značit, že tam byli dány nevědomky a později přidaný záznam mohl ovlivnit dříve vytvořené stránky. Pro nalezení opakujících jsem vytvořil skript `hledej_opakované.php`.

Zjistit volání překladové funkce, která pro tvorbu argumentu využívá nějakou proměnnou.Tento argument pak může být pro každou hodnotu proměnné jiný, z čehož plyne také různá varianta klíče, kterému odpovídá jiná česká i anglická verze. Taková volání funkcí musí být zrušena.

Příklady takových funkcí:

```
$a_error_msg[]=tr('V individuálním plánu nemáte pro rok ' .
$modul_params['rok'].'/'.($modul_params['rok']+1).' žádné předměty.' );

<td class="b"><img title="<?= get_lang('sex_'.$person['sex']);?>" alt="<?=
get_lang('sex_'.$person['sex']);?>" class="select-all" src="<?=
sprintf('%ssex_%s.gif',$GLOBALS['img_ldir'],$person['sex']);?>" />

Insert_label(`jazyk_vyuky_`.$val[`jazyk_vyuky`],1,1)
```

Eliminace dlouhých argumentů překladatelských funkcí. Existují volání překladů, jejichž argument je delší jak 70 znaků, protože primární klíč pro záznamy textů jsem nastavil právě na tuto hodnotu. Delší záznamy jsou při uložení do DB zkráceny. Samotné texty zkráceny nejsou. Nejprve jsem uvažoval o délce 50 znaků, která se mi zdála jako dostatečná pro jednoznačnou identifikaci textu. Při podrobnější analýze jsem však objevil argumenty překladatelských funkcí, které délku 50 znaků překračují. Jedná se sice většinou o řetězce, které nejsou vůbec přeložené v datovém souboru, v tom případě většinou překladatelská funkce vrací hodnotu argumentu. Ale vyskytují se také volání, kdy je argument funkce přeložen a délku 50 znaků přesahuje. Jako například:

```
$tr_mem['Litujeme, dle zadaných kritérií nebyly nalezeny žádné osoby'] =
array('Litujeme, dle zadaných kritérií nebyly nalezeny žádné osoby',
'No results found');
```

Takové záznamy se ve slovníku vyskytují a zbytečně zabírají místo a zpomalují překlad. V nových datových souborech jsou již klíče zkráceny, texty samozřejmě ne. Pro urychlení integrace je možné prozatímně upravit překladatelské funkce, aby před hledáním hodnoty pro daný klíč argument zkrátili na 70 znaků. Tímto způsobem ale nemůžeme zaručit, že některý pro dva různé texty nevznikne jeden stejný klíč. Když pravděpodobnost je velmi malá, musely by totiž být shodných prvních 70 znaků obou textů.

Musí se přepsat kódy volání souboru s překladovými funkcemi pro načtení dat ze samostatného aplikaci odpovídajícího datového souboru. Protože překladatelské funkce již nebudou používat jeden globální soubor, ale pro každou aplikaci samostatný soubor, umístěný na určeném místě – například v kořenovém adresáři aplikace. v jeho kořenovém adresáři.

Jelikož již dnes je hlavní datový soubor rozdělen na sekce, pro jednotlivé aplikace (dotazem na globální proměnnou `$GLOBALS['app_name']`), je potřeba před spuštěním sestavovacího skriptu tyto sekce zrušit a skriptu tak zpřístupnit celý obsah datového souboru. Provedl jsem analýzu textů v těchto sekcích a není se čeho obávat, texty se neopakují. Jsou zde pouze texty specifické pro danou aplikaci. Právě proto mohou být v této sekci do nichž se skripty z jiných aplikací nedostanou.

8 Závěr

V rámci mé diplomové práce jsem se věnoval problematice vytváření vícejazyčných webových aplikací. Nejprve jsem popsal prostředky které se pro tvorbu webových aplikací používají. Následně jsem rozebral jak je možno implementovat vícejazyčné aplikace. Existuje mnoho cest, jak vícejazyčnost zajistit a žádná z nich však není univerzální. Při práci jsem se také seznámil s již existujícími nástroji, které jsou programátorům k dispozici.

V dalších částech práce jsem se věnoval Portálu VUT. Jak je zajištěn jeho provoz, jaké nástroje používá. Po této analýze a konzultace s vývojáři a překladatelem jsem navrhl řešení, které následně i implementoval. Rozdělení jednoho společného souboru, který má zhruba 9000 řádků mezi více vztahující se k jednotlivým aplikacím Portálu je velmi vhodný krok. Bohužel však neproběhla integrace tohoto řešení, což je úkol do budoucna. A nejenom integrace, samotnou implementaci je samozřejmě možné rozšiřovat a zlepšovat. Může jít například o zavedení slovníků překladateli často používaných slov přímo do jejich části aplikace.

Literatura a internetové zdroje

[1] http://www.php.net/

[2] http://www.vutbr.cz

[3] Jiří Bráza, PHP 5 Začínáme programovat. Grada 2005

[4] http://www.unicode.org/

[5] http://www.apache.org/

[6] http://www.mysql.com/

[7] http://www.w3.org/WAI/ER/IG/ert/iso639.htm

[8] http://www.gnu.org/software/gettext/

[9] http://interval.cz/serialy/gnu-gettext-snadna-lokalizace-webovych-aplikaci/

[10] http://kbabel.kde.org/

[11] http://yats.sourceforge.net/

[12] http://smarty.php.net/

[13] http://www.wikipedia.org

[14] http://www.pear.php.net/package/Translation2

[15] http://www.pear.php.net/

[16] http://wiki.ro.vutbr.cz

[17] http://www.subversion.org/

[18] http://merlin.fit.vutbr.cz/FITkit/?pg=navody&cl=20060209svn

[19] www.netcraft.com/

[20] Apache Server 2 - Kompletní příručka administrátora, Mohammed J. Kabir, ISBN:8025103196, Nakladatelství Computer Press, a.s.

[21] http://www.w3.org/Style/CSS/

[22] Groff,R.,J, Weinberg, N.,P.: SQL - Kompletní průvodce, CP Books, Brno 2005,ISBN-80-251-0369-2

[23] http://www.kiv.zcu.cz/~brada/vyuka/aswi/navod-svn

[24] http://merlin.fit.vutbr.cz/wiki/index.php?title=SVN_tutori%C3%A1l

[25] http://www.pspad.com/

Seznam obrázků

Obr.1: SQL konzole
Obr. 2: Panel pro vybírání jazyka
Obr. 3: Schéma aktualizace překladů
Obr. 4: Použití jedné tabulky pro uložení více variant textu
Obr. 5: Použití tří tabulek pro uložení více variant textu
Obr. 6: Standardní struktura pro knihovnu Translation2
Obr. 7: Schéma webového clusteru VUT
Obr. 8: Správce zdrojů dat ODBC ve Windows XP
Obr. 9: MySQL ODBC Connector
Obr. 10: Integrace TortoiseSVN klienta do Windows XP
Obr. 11: Operace nástroje Subversion
Obr. 12: Diagram případů použití aplikace
Obr. 13: E-R diagram aplikace
Obr. 14: Neúspěšné přihlášení
Obr. 15: Aplikace pro vývojáře
Obr. 16: Aplikace pro překladatele
Obr. 17: Připravená emailová zpráva pro vývojáře
Obr. 18: Graf konečného automatu přijímajícího volání funkce `tr()`

Seznam tabulek

Tab. 1: Výběr některých jazykových kódů
Tab. 2: SVN statu
Tab. 3: Instance web aplikací

Seznam delších bloků kódů

Kód 1: Příklad dokumentu HTML
Kód 2: Ukázka datového souboru
Kód 3: ukázka položky datového souboru
Kód 4 : Funkce get_lang()

Seznam příloh

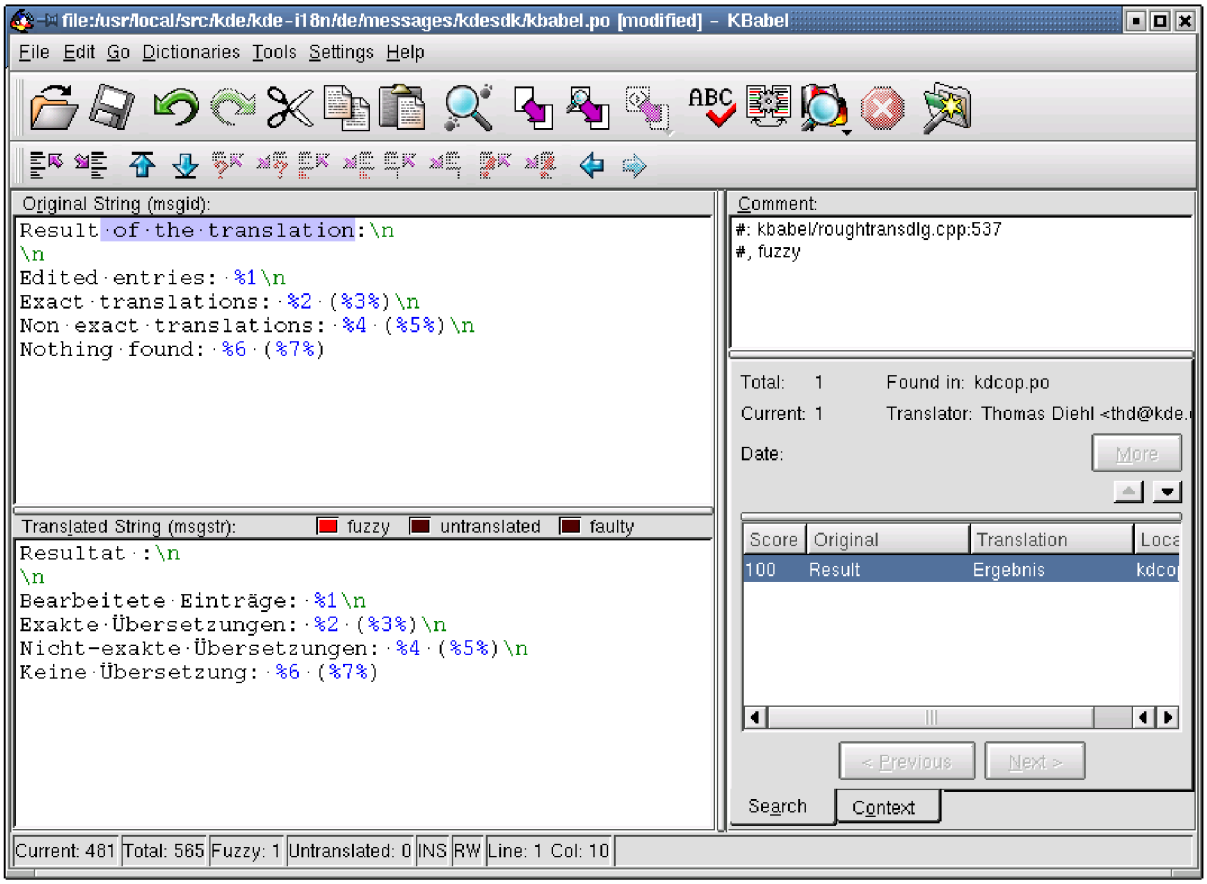
Příloha 1: Prostředí editoru KBabel

Příloha 2: Výstup sestavovacího skriptu

Příloha 2: Výstup skriptu volaného po SVN operaci commit

Příloha 3: CD

Příloha 1: Prostředí editoru KBabel



Příloha 2: Ukázka výstupu sestavovacího skriptu:

```
Vypis systemu jazykovych verzi, sestavovaci skript
------------------------------------------------------------------------
Zpracovavane aplikace:
Vkladam do tabulky aplikace polozku: _base
------------------------------------------------------------------------
Zpracovavam soubor: trunk/_base/.htaccess
Zpracovavam soubor: trunk/_base/site_lib/func_app.php
Klic Nemáte dostatek práv pro přístup. - neni prelozeno - neni vlozeno do
DB
Zpracovavam soubor: trunk/_base/site_lib/func_base.php
Zpracovavam soubor: trunk/_base/site_lib/func_init_lite.php
Zpracovavam soubor: trunk/_base/site_lib/func_login.php
Klic Nepodařilo se přejít zpět na stránku zobrazenou před přesměrování. -
neni prelozeno - neni vlozeno do DB
Klic Nepodařilo se přihlásit do LDAPu pro zjištění práv. - neni prelozeno
- neni vlozeno do DB
Klic Již jste přihlášen. Zadané přihlašovací údaje nebylo nutné ověřovat.
- neni prelozeno - neni vlozeno do DB
Klic Interní chyba. Pro per_id=%s nebyl nalezen záznam v bs_person. - neni
prelozeno - neni vlozeno do DB
Zpracovavam soubor: trunk/_base/www_base/priloha.php
Zpracovavam soubor: trunk/_base/www_base/zav_prace_soubor.php
------------------------------------------------------------------------
Generujeme datove soubory s preklady pro jednotlive aplikace:
Aplikace _base: soubor: trunk/_base/func_tr_data.php
------------------------------------------------------------------------
Neni prelozeno:
Nemáte dostatek práv pro přístup. ze souboru
trunk/_base/site_lib/func_app.php
Nepodařilo se přejít zpět na stránku zobrazenou před přesměrování. ze
souboru trunk/_base/site_lib/func_login.php
```

Příloha 3: Ukázka výstupu skriptu po SVN akci commit se zvýrazněnými důležitými údaji

```
Vypis systemu jazykovych verzi, volany po akci commit svn
------------------------------------------------------------------------
Cislo revize: 2914
Autor revize: jurosz
Datum revize: 2007-07-18
Cas revize: 10:58:49
------------------------------------------------------------------------
Soubory z revize (pocet:4):
M trunk/_base/templ/gm_vizitka_cv_templ.php
M trunk/_base/templ/gm_vizitka_hledat_templ.php
D trunk/_base/templ/gm_vizitka_pub_templ.php
M trunk/_base/templ/gm_vizitka_templ.php
------------------------------------------------------------------------
Soubor: trunk/_base/templ/gm_vizitka_cv_templ.php
V souboru se nezmenila volani prekladatelskych funkci
Soubor: trunk/_base/templ/gm_vizitka_hledat_templ.php
Ze souboru bylo odebrano volani pro klic Obsah
V souboru pribylo volani klice: Místnost
Soubor: trunk/_base/templ/gm_vizitka_pub_templ.php
Mazeme vsechny zaznamy pro tento soubor
Soubor: trunk/_base/templ/gm_vizitka_templ.php
V souboru se nezmenila volani prekladatelskych funkci
Soubor: trunk/_base/templ/gm_vizitka_vyuka_templ.php
------------------------------------------------------------------------
Generujeme datove soubory s preklady pro jednotlive aplikace:
Pro aplikaci _base generujeme novy datovy soubor
trunk/_base/func_tr_data.php
------------------------------------------------------------------------
Neni prelozeno:
jazyk_vyuky_'.$val['jazyk_vyuky ze souboru
trunk/_base/templ/gm_vizitka_vyuka_templ.php
téma disertační práce: ze souboru
trunk/_base/templ/gm_vizitka_vyuka_templ.php
```