TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: B 2612 – Elektrotechnika a informatika
Studijní obor: 1802R022 – Informatika a logistika

Efektivní internacionalizace webových projektů v JSP a PHP

Effective internationalization web projects with JSP and PHP

Bakalářská práce

Autor: Luboš Remplík
Vedoucí BP: Ing. Igor Kopetschke
Konzultant: Mgr. Zuzana Fenclová

V Liberci 18. 5. 2006

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Katedra aplikované informatiky Akademický rok: 2006/2007

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jméno a příjmení: Luboš Remplík

studijní program: B 2612 – Elektrotechnika a informatika

obor: 1802R022 - Informatika a logistika

Vedoucí katedry Vám ve smyslu zákona o vysokých školách č.111/1998 Sb. určuje tuto bakalářskou práci:

Název tématu: **Efektivní internacionalizace webových projektů v JSP a PHP**

Zásady pro vypracování:

1. Teoretický rozbor problematiky internacionalizace aplikací na webu.
2. Analýza metod tvorby vícejazyčných webů z pohledu technologií PHP a JSP.
3. Návrh struktury a implementace vícejazyčného webu.
4. Praktické řešení za použití obou výše zmíněných technologií

Rozsah grafických prací: dle potřeby dokumentace
Rozsah průvodní zprávy: cca 40 stran

Seznam odborné literatury:

[1] Rosebrock, Eric; Filson, Eric. Linux, Apache, MySQL a PHP : instalace a konfigurace prostředí pro pokročilé webové aplikace . Praha : Grada, 2005.

[2] Bráza, Jiří. PHP 4 : praktické příklady. Praha . Grada, 2003.

[3] Welling, Luke; Thomson, Laura. PHP a MySQL - rozvoj webových aplikací. Praha : SoftPress, c2002.

[4] Bollinger, Gary; Natarajan, Bharathi. JSP - Java Server Pages : podrobný průvodce začínajícího tvůrce webu. Praha : Grada, 2003.

[5] Burd, Barry A. JSP - JavaServer Pages : tvorba WWW stránek. Praha : Computer Press, 2003.

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

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

[8] http://java.sun.com

Vedoucí bakalářské práce: ing. Igor Kopetschke

Konzultant: Mgr. Zuzana Fenclová

Zadání bakalářské práce: **22.10.2006**

Termín odevzdání bakalářské práce: **18. 5. 2007**

L.S.

....

Vedoucí katedry Děkan

V Liberci dne 22.10.2006

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum 18. 5. 2007

Podpis

Poděkování

Děkuji tímto vedoucímu své bakalářské práce, panu Ing. Igoru Kopetschkemu, za vedení, cenné rady a podnětné připomínky při tvorbě této práce.

Abstrakt

Práce se zabývá efektivními metodami internacionalizace webových projektů a problémy s tímto spojené. Jsou zde uvedené praktické řešení v programovacím jazyku JSP a PHP.

Zaměření je několik. Rozebírá druhy znakových sad, ukazuje funkci pro jejich konverzi a ospravedlňuje výběr kódování UTF-8. Navrhuje několik metod internacionalizace a ukazuje na praktickém řešení – pomocí třídy ResourceBundle, pomocí kombinace skriptovacího jazyka JSP s databází MySQL a použitím knihovny gettext v PHP.

Důležitým cílem práce je porovnání jednotlivých způsobů řešení, vystižení jejich výhod, respektive nevýhod a doporučení oblasti použití.

Klíčová slova – Internacionalizace, lokalizace, ResourceBundle, gettext, iconv.

Abstract

The work introduces into effective methods of internationalization web projects and problems close of that. Here are some practical examples makes by JSP and PHP.

The several aims of work are mentioned. Project describes character encoding, shows function for converting and makes reason of UTF-8 choose. Proposes a few methods of internationalization with practical examples - with ResourceBundle class, with JSP and MySQL and with gettext library.

The important aim of work is to compare diferent kinds of solution methods, describe advantages, disadvantages and makes references for further use.

Keywords - internationalization, localization, ResourceBundle, gettext, iconv.

Obsah

Úvod

Stále více lidí různých národností, z mnoha koutů světa, se seznamují s osobními počítači a s výhodami získávání kvanta informací z internetu, zvláště pak z webových stránek. Bohužel bariéra není v rychlosti či způsobu předání informací, ale v jejich porozumění. Problém komunikace mezi jednotlivými národnostmi spočívá v dorozumění slovem, písmem (kódováním) a také například ve formátu času, data a podobně. V oblasti PC, respektive internetu se tento problém snaží řešit různé metody internacionalizace programů, respektive webu.

Existuje několik metod provedení internacionalizace. Některé jsou však nevhodné (v některých případech nepoužitelné) nebo málo používané. Práce se proto zabývá jednak způsoby, tak i efektivností internacionalizace.

Struktura práce je následující. V několika kapitolách se budeme seznamovat s jednotlivými metodami internacionalizace a způsobem konverze znakových sad. V první kapitole vymezím důležité pojmy a v sedmé kapitole způsoby internacionalizace zhodnotím.

Předpokládá se, že čtenář je seznámen se základy jazyka PHP, Java a tvorbou stránek v HTML.

1 Vymezení pojmů

Jelikož oblast pojmů považuji za velmi důležitou, uvádím ji zde na začátku a není úplně stručná. Významnější pojmy jsou nadepsány nadpisy druhé úrovně a podrobnější, nadpisy třetí úrovně. Pro odbornější čtenáře doporučuji tuto kapitolu přeskočit a v případě nejasností se k problematice pojmů vrátit.

V průběhu textu už nebudu čtenáře odkazovat na první kapitolu, berte to jako samozřejmost. Vymezení se především týká pojmů jako JSP, PHP a Internacionalizace.

1.1 Internacionalizace

Internacionalizace (značená i18n) je schopnost webu přizpůsobit se jazykovému prostředí a být lokalizován. Přizpůsobení jazykovému prostředí se netýká pouze jazyka, ale také dalších aspektů jako je měna, formát času a data, oddělovač desetinných míst a jiné vlastnosti jazyka.

1.2 Lokalizace

Lokalizace (l10n) spočívá v kompletním přeložení uživatelského rozhraní webu do dalších jazyků.

1.3 Apache HTTP server

Apache HTTP Server je softwarový webový server s otevřeným kódem pro Linux, BSD, Microsoft Windows a další platformy. V současné době dodává prohlížečům na celém světě většinu internetových stránek.

1.4 Apache Tomcat

Tomcat je oficiální referenční implementace technologií Java Servlet a Java Server Pages (JSP), obojí vyvíjené pod záštitou společnosti SUN. Apache Tomcat je hybridní instalace WWW serveru Apache a JSP serveru Tomcat. Apache servíruje statické dokumenty a JSP stránky a servlety předává Tomcatu.

1.5 Java Server Pages (JSP)

Java Server Pages je technologie s volně dostupným zdrojovým kódem (open source), která nám umožňuje směšovat statické HTML stránky s dynamicky tvořeným obsahem. JSP může vytvářet jak statickou, tak i dynamickou část v jednom programu. Tento program se pak skládá z normálního HTML kódu a z částí, které definují dynamiku stránky pomocí speciálních značek.

1.6 Java Development Kit (JDK)

Java Development Kit (JDK) je soubor základních nástrojů pro vývoj aplikací pro platformu Java. Někdy bývá označován jako Java SDK, od verze 1.2 do verze 1.4.x byl označován jako J2SE SDK nebo Java 2 SDK. Ke kompilování servletů je nutné mít nainstalované JDK.

1.7 PHP

PHP (rekurzivní akronym pro "PHP: Hypertext Preprocessor") je rozšířený univerzální skriptovací jazyk s volně dostupným zdrojovým kódem (open source), který je obzvláště vhodný pro vývoj webových aplikací a lze jej zapouzdřit do HTML. PHP se liší od jazyků, jako je Javascript tím, že je vykonáván na straně serveru. Syntaxe se lehce podobá jazykům jako je Perl nebo C.

1.8 MySQL

MySQL je open source relační databázový systém typu DBMS (database managment system). Každá databáze v MySQL je tvořena z jedné nebo více tabulek, které mají řádky (řádek = záznam) a sloupce (sloupec = pole), které mají jméno a uvozují datový typ jednotlivých polí záznamu. Práce s databázemi, tabulkami a daty se provádí pomocí příkazů, respektive dotazů vycházejících z deklarativního programovacího jazyka SQL (Structured Query Language). Systém MySQL je využitelný v C, C++, Java, JSP, Perl, PHP, Python, Tcl, Visual Basic, .NET a další.

1.9 API

API je zkratka anglických slov application programming interface, což znamená rozhraní pro programování aplikací. Tento termín používá softwarové inženýrství v programování. Jde o sbírku procedur, funkcí či tříd nějaké knihovny, které může využívat programátor, který knihovnu využívá.

1.10 JDBC

Java Database Connectivity (známé spíše jako JDBC) je API pro programátory v programovacím jazyku Java, které definuje jednotné rozhraní pro přístup k relačním databázím. JDBC je součástí Javy SE (Standard Edition) od JDK 1.1. Pro přístup ke konkrétnímu databázovému serveru je potřeba JDBC ovladač, který poskytuje tvůrce databázového serveru.

1.11 W3C

World Wide Web Consortium (W3C) je mezinárodní konsorcium, jehož členové společně s veřejností vyvíjejí webové standardy pro World Wide Web. Cílem konsorcia je „Rozvíjet World Wide Web do jeho plného potenciálu vývojem protokolů a směrnic, které zajistí dlouhodobý růst Webu“. W3C se také zabývá vzděláním a přístupností.

1.12 Pojmy použité v JSP

Zde uvedu a popíšu pojmy, které používám v kapitolách 3 a 4.

1.12.1 Direktivy

JSP direktivy definují globální nastavení JSP. Existují 3 direktivy – page, import a taglib. Použijeme-li direktivu, musí se objevit na začátku souboru a má formát `<@direktiva atribut="hodnota">`.

1.12.2 Výrazy

Výrazy se vkládají ve tvaru `<%= výraz %>`. Používá se tedy pokud potřebujete něco vložit přímo do stránky (např.: obsah proměnné, datum, čas).

1.12.3 Deklarace

Deklarace metod nebo proměnných můžeme uvádět mezi tagy `<%! kód %>`

1.12.4 Skripty

Skripty se uvádějí mezi tagy `<% %>`. Jsou zde již složitější věci jako cykly, podmínky apod. Můžou se použít jakéhokoli třídy a metody používané v programovacím jazyku Java.

1.12.5 Komentáře

Ve scriptech můžete samozřejmě používat komentáře. Vkládají se mezi tagy `<%-- JSP komentář --%>`.

2 Kódování

Z hlediska definice lze kódování chápat jako reprezentaci nějaké konečné množiny znaků pomocí nějaké jiné konečné množiny (zpravidla čísel).

Pro internacionalizaci je udržování stejného kódování a výběr standardu pro konverzi mezi jednotlivými jazykovými mutacemi velmi důležité.

Můžeme mít více zdrojů informací a také více pohledů na tyto informace. Pokud budeme mít všechny tyto data ve stejném kódování usnadníme si práci nejen při jejich zpracování, ale také při jejich publikaci. Nemusíme používat nástroje pro konverzi mezi jednotlivým kódováním.

2.1 Unicode

Unicode je 16 bitové kódovací schéma s neměnnou šířkou, určené pro zápis znaků v textu. Toto mezinárodní kódování obsahuje znaky hlavních světových abeced a také často používané technické znaky. Kódování Unicode zachází se znaky abeced i různými jinými symboly stejným způsobem, takže mohou být používány společně. Unicode vychází z ASCII, používá ale 16 bitů pro identifikaci znaků, aby bylo možné podporovat vícejazyčné texty. Pro žádný znak z kteréhokoli jazyka nejsou třeba žádné escape sekvence nebo jiný kontrolní kód.

2.2 UTF-8

UTF-8 je zkratka pro „UCS Transformation Format“. Jedná se o doporučený způsob zápisu ISO/EIC 10646 znaků pro UCS-2 i UCS-4. Může tak posloužit i pro zápis Unicode. Je to způsob kódování řetězců znaků do sekvencí bajtů.

2.2.1 Cíle UTF-8

Cíle jsou zejména:

- Kompatibilita se staršími souborovými systémy. Souborové systémy zpravidla nepovolují v názvech souborů nulový byte ani (zpětné) lomítko.
- Kompatibilita s existujícími programy. Zápis jakéhokoli znaku by neměl obsahovat ASCII, pokud znak původně v ASCII nebyl.
- Snadnost konverzí z/do UCS.

2.2.2 Unicode v UTF-8

S Unicode textem zapsaným v UTF-8 lze zacházet stejně jako s obyčejným osmibitovým, není třeba nic speciálně ošetřovat ani psát žádný speciální programovací kód pro zacházení s Unicode. Není divu, že se UTF-8 prosazuje jako univerzální formát pro výměnu dokumentů v Unicode.

3 JSP a ResourceBundle

V JSP pro internacionalizaci použijeme třídu `ResourceBundle`, kde v návrhu řešení zmíním dva její potomky a v realizaci se jedním z nich budu zabývat.

`ResourceBundle` obsahuje objekty charakteristické pro danou lokalitu. Když program potřebuje lokální zdroj, například nějaký řetězec, může ho načíst z `ResourceBundle`, který odpovídá aktuální uživatelské lokalitě. V tom případě můžeme psát program, který je spíše nezávislý na určitých lokalizacích.

To nám umožní psát program který:

- je jednoduše lokalizován nebo přeložen do rozdílných jazyků
- umí zacházet s více lokalizacemi
- je později upravitelný nebo rozšířitelný do více lokalizací

Každý zdroj obsahuje stejné klíče, které jsou přeloženy do lokalizace odpovídající tomuto zdroji. Např. oba zdroje `MyResources` a `MyResources_en` smějí mít řetězec použitý pro tlačítko stornování. V `MyResources` se může `klíč = zrušit` a v `MyResource_en` se může `klíč = cancel`.

3.1 Návrh řešení

Třída `ResourceBundle` má dva potomky, které mají odlišný způsob zápisu zdrojových textů lokalizací. Liší se tím jestli jsou zapsány jako třída ve zdrojovém kódu aplikace nebo jako souboru uložený ve zvláštním adresáři.

3.1.1 ListResourceBundle

`ListResourceBundle` je potomek třídy `ResourceBundle`, který uspořádává zdroje pro lokalizaci v přehledných seznamech jednoduchých pro použití napsané jako třída ve zdrojovém kódu.

Vytvoříme podtřídu, která musí přepisovat metodu `getContents()` a poskytovat dvourozměrné pole, kde první rozměr je x-hodnot polí a druhý rozměr jsou dvě hodnoty. Kde první hodnota je klíč shodný pro všechny lokalizace, který musí být typu `String` a druhá hodnota je jazyková mutace.

Příklad použití ukážu v následujících dvou třídách, jejichž základ jména je `MyResources`. Třída jež je tvořená pouze základem, tzn. `MyResources`, je volaná defaultně. Třída `MyResource_en` je lokalizací angličtiny.

- třída pro českou lokalizaci:

```
public class MyResources extends ListResourceBundle {
    public Object[][] getContents() {
        return contents;
    }
    static final Object[][] contents = {
    // LOCALIZE THIS
        {"s1", "žádný soubor"},//first ChoiceFormat choice
        {"s2", "jeden soubor"},//second ChoiceFormat choice
        {"s3", "{number} souborů"},//third ChoiceFormat
                                   //choice
        {"s4", "26. června 1985"},//sample date
    // END OF MATERIAL TO LOCALIZE
    };
}
```

- třída pro anglickou lokalizaci:

```
public class MyResources_en extends ListResourceBundle {
    public Object[][] getContents() {
        return contents;
    }
    static final Object[][] contents = {
    // LOCALIZE THIS
        {"s1", "no files"},//first ChoiceFormat choice
        {"s2", "one file"},//second ChoiceFormat choice
        {"s3", "{number} files"},//third ChoiceFormat choice
        {"s4", "06/26/1985"},// sample date
    // END OF MATERIAL TO LOCALIZE
    };
}
```

3.1.2 PropertyResourceBundle

`PropertyResourceBundle` je potomek třídy `ResourceBundle`, který jako zdroj pro lokalizaci používá množinu statických řetězců uložených v `property` souborech.

Na rozdíl od `ListResourceBundle` nevytváříme potomka `PropertyResourceBundle`, ale zprostředkováváme `properties` soubory obsahující jako prvek množiny klíčové slovo a jazykovou mutaci. Metoda `getBundle()` bude automaticky hledat odpovídající `property` soubor, referující požadovanou lokalizaci.

Následující příklad ukazuje použití Německé jazykové mutace. Bude uložené jako obyčejný textový soubor ve tvaru `MyResource_de.properties` v adresáři s relativní cestou */tomcat/work/Catalina/localhost/www/ /(jmeno_projektu)/*.

- soubor pro německou lokalizaci:

```
# LOCALIZE THIS

# first ChoiceFormat choice
s4=keine Dateien

# second ChoiceFormat choice
s5=eine Datei

# third ChoiceFormat choice
s6={number} Dateien

# sample date
s7=3. März 1996

# END OF MATERIAL TO LOCALIZE
```

3.2 Vlastní realizace

K vlastní realizaci jsem si vybral druhý způsob. To znamená lokalizaci pomocí třídy `PropertyResourceBundle`.

Vytvořil jsem si projekt s názvem `bundle_properties` ,kde `*.jsp` zdrojové soubory jsou uloženy v adresáři `tomcat/webapps/www/bundle_properties` a *`*.properties`* soubory s českou, anglickou a německou lokalizací jsou uloženy v adresáři *`tomcat/work/Catalina/localhost/www/bundle_properties`*.

3.2.1 HTML formulář jako index.jsp

Mám soubor s názvem `index.jsp`, který je napsán jako HTML stránka s formulářem. Také obsahuje JSP definice, kde `contentType` určuje typ `MIME`, který v našem případě je `text/html` a schéma kódování UTF-8.

```
<%@ page contentType = "text/html;charset=utf-8" %>
<%@ page pageEncoding="UTF-8" %>
```

Pro správné kódování nastavíme také `Content-Type` a `charset` v meta tagu HTML stránky.

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=utf-8">
```

Podstatný kód HTML stránky tvoří formulář. Ten je posílán metodou `post` a má nastavený parametr `action` na název souboru se scriptem, který bude formulářová data zpracovávat.

```
<form method="post" action="result.jsp">
```

Dále jsou ve formuláři 3 input tagy typu radio, kteří určují jaký jazyk bude lokalizován.

```
<input type="radio" name="lang" value="cz" checked>Čeština
<!-- atribut checked určuje primárně určený jazyk. -->
```

A následně `input` tag typu `submit` pro odeslání formuláře.

```
<input type="submit" name="Submit" value="Submit">
```

3.2.2 Zpracování výsledku v result.jsp

Formulář je odeslán a vyhodnocen souborem `result.jsp`. V tomto souboru je definice kódování stejná jako v předchozím případě, tzn.: JSP definicí a meta tagem. Je zde skript pro předání `lang` parametru (cz, en, de) do proměnné, která je zbavena bílých a jiných nepotřených znaků. Dále lokalizace a načtení property souboru.

Nejdříve musíme získat informaci o jazyku (státu), do kterého budeme stránku lokalizovat. Tuto informaci máme uloženou v datech odeslaných formulářem. Informaci uložíme do proměnné `parametr`, kterou získáme z `lang` proměnné formuláře pomocí metody `getParametr`.

```
String parametr = request.getParameter("lang");
```

Připravíme si proměnné `lang` a `country`, které později budou určovat jazyk a zemi, do které budeme lokalizovat.

```
String lang = "";
String country = "";
```

Otestujeme jestli proměnná `parametr` vůbec obsahuje nějaký řetězec. Metodou `substring()` vybereme první dva znaky, které by měli obsahovat kód jazyka, to kterého chceme lokalizovat. `Trim()` metoda nám zajistí ořezání tohoto krátkého řetězce od bílých znaků (mezera, konec řádku apod.) a tento výsledek uložíme do proměnné `lang`.

```
if (parametr.length() != 0) {
lang = parametr.substring(0, 2).trim();
```

Pokračujeme podmínkou, která nám pomocí metody `length()` zjistí, zda je řetězec delší nebo roven třem znakům, což by znamenalo, že je v parametru uložen nejen kód jazyka, ale také kód země, do které lokalizujeme. Upravíme řetězec obdobně jako u proměnné `lang` a uložíme do proměnné `country`.

```
if (parametr.length() >= 3) {
    country = parametr.substring(3, 5).trim();
}
```

*3.2.3 Soubory *.properties s lokalizacemi*

Vytvoříme proměnnou `soubor`, která je instancí třídy `ResourceBundle` a uložíme do ní informace, které získáme metodou `getBundle()`. Ta má dva parametry. Jako první parametr má cestu k souborům s lokalizacemi, plus základ jména souborů. A druhý parametr objekt, určující specifickou část jména lokalizačního souboru, zpracovaný metodou `Locale(lang,country)`, která převezme naše parametry `lang` a `country`.

```
ResourceBundle soubor =
ResourceBundle.getBundle("bundle_properties/PropertyResource",
                         new Locale(lang, country));
```

Pokud hodnoty `lang` a `country` jsou prázdné, načte se soubor s `PropertyResource.properties`, kde máme uložené texty našeho primárně lokalizovaného jazyka. Tedy soubor se jménem, které je tvořené základem jména souborů. Když není soubor nalezen, server nám pošle výjimku `MissingResourceException`.

3.2.4 Načtení klíčových slov

Konečně máme vše připravené, abychom mohli kdekoli v HTML stránce načítat lokalizované texty pomocí klíčových slov, která jsou ve všech souborech s lokalizacemi stejné.

K načtení textu (řetězce) potřebujeme metodu `getString()`, která vyhledá řádek a poté hodnotu klíčového slova a vrátí jí jako `String` hodnotu.

```
<p><%= soubor.getString("coffee")%></p>
<p><%= soubor.getString("tea")%></p>
```

Klíčová slova `coffee` a `tea` se přeloží do lokalizovaného jazyka (země). V případě, kdy rozlišujeme třeba angličtinu na Americkou a Britskou, tak se v případě americké angličtiny přeloží klíčové slovo `coffee` jako `java` (V Americe používaný jako slangový výraz pro slovíčko káva, jinak se samozřejmě používá `coffee`. Ale pro demonstraci příkladu, tento překlad poslouží dobře a v případě britské angličtiny se přeloží jako `coffee`. Jak je vidět toto byl dobrý příklad lokalizace nejen podle jazyka ale také podle oblasti. Mimo angličtiny je v programu připravená také lokalizace do Němčiny. Takže klíčové slovo `coffee` je přeloženo jako `kaffee`. Obdobně se lokalizuje klíčové slovo `tea`.

4 JSP a MySQL

Řešení internacionalizace webové stránky pomocí databáze a skriptovacího jazyka je v praxi oblíbené a rozšířené. Doslechl jsem se, že tento způsob je jedním z lepších řešení. Pravdu uvidíme při hodnocení této a ostatních metod v sedmé kapitole.

4.1 Návrh řešení

Přístup k databázi není potřeba zde v návrhu nějak rozebírat, bude ve všech možných realizacích stejný. Řešení připojení k databázi pomocí JSP popíšu v kapitole 4.2. Odlišné způsoby řešení budou hlavně ve struktuře databáze.

4.1.1 Struktura tabulky se třemi sloupci

První sloupec `klic` typu `varchar(15)` je nositel klíčových slov, které vystihují nadpis, větu, článek nebo cokoli jiného co bude překládáno. Druhý sloupec `lang` typu `char(2)` je identifikátor jazyka, do kterého je lokalizováno. A třetí sloupec `txt` typu `varchar(500)` je přeložený text daného klíče a jazyka. Sloupec `klic` i `lang` mohou mít opakující se hodnoty, avšak jako celek jsou jednoznačným identifikátorem. Počet sloupců tabulky se i při mnohojazyčných lokalizacích nemění, proto není potřeba zasahovat do struktury databáze. Všechny tři sloupce jsou ve stejném kódováním `utf8_czech_ci`. Výhodou je, že hodnota ve sloupci `txt` nikdy nebude `NULL`. Pokud neznáme překlad, nebude v tabulce ani kombinace klíče a jazyka tohoto textu.

Sloupec	Typ	Porovnávání	Vlastnosti	Nulový	Výchozí
klic	varchar(15)	utf8_czech_ci		Ne	
lang	char(2)	utf8_czech_ci		Ne	
txt	varchar(500)	utf8_czech_ci		Ano	*NULL*

Obr. 1: Tabulka s třemi sloupci

4.1.2 *Struktura tabulky se dvěma a více sloupci*

První sloupec tabulky je stejný jako u předchozí příkladu – `klic` typu `varchar(15)`. Další jeden nebo více sloupců je přeložený text, kde lokalizaci určuje název sloupce (například pro český text – `txt_CZ`). V případě tří jazyků budou 4 sloupce. Každý další jazyk je roven dalšímu sloupci. Jednoznačným identifikátorem je v tomto případě pouze sloupec `klic`. Hodnota textu lokalizace, pro kterou neznáme překlad, nabývá hodnoty `NULL`. Nevýhoda je, že při potřebě přidání dalšího jazyku do aplikace se musí měnit struktura databáze, respektive přidat sloupec.

Sloupec	Typ	Porovnávání	Vlastnosti	Nulový	Výchozí
klic	varchar(15)	utf8_czech_ci		Ne	
txt_CZ	varchar(500)	utf8_czech_ci		Ano	*NULL*
txt_EN	varchar(500)	utf8_czech_ci		Ano	*NULL*
txt_DE	varchar(500)	utf8_czech_ci		Ano	*NULL*

Obr. 2: Tabulka se dvěma a vícejazyčné sloupci

4.2 Vlastní realizace

Vybral jsem si první z uvedených způsobů struktury databáze z důvodu sympatie, jakým způsobem se databáze v budoucnu lehce upravuje. Není potřeba zásah do dat, jen přidávám překlady jakéhokoli dalšího jazyka dle klíčového slova spolu s kódem jazyka. Také zde nevznikají `NULL` hodnoty.

4.2.1 *Výběr jazyka a vytvoření proměnné lang*

Vytvoříme hypertextové odkazy, kde každý odkaz reprezentuje jazyk země. Kód země se předává pomocí parametru `lang` v URL adrese. V mém případě jsem si vytvořil dva odkazy – jeden pro Český jazyk a druhý pro Anglický jazyk.

```
<a href="index.jsp?lang=cs">česky</a>
<a href="index.jsp?lang=en">english</a>
```

Následně tvořím proměnnou `lang`, která je typu `String`. A pomocí metody `getParameter()` třídy `request` uložím do proměnné kód jazyka dle daného parametru.

```
String lang = request.getParameter("lang");
```

V případě url adresy `http://localhost/index.jsp`, kterou voláme úvodní webovou stránku, je parametr `lang` prázdný, musíme tedy do proměnné `lang` uložit nějaký defaultní kód jazyka – konkrétně `cs` pro češtinu.

```
if (lang == null || ( !lang.equals("en")&&!lang.equals("cs") ) )
{
  lang ="cs";
}
```

Je-li `lang` proměnná prázdná (`NULL`) nebo není-li v proměnné kód `en`, respektive `cs`, pak do `lang` přiřadíme hodnotu `cs`.

4.2.2 Připojení k databázi pomocí konektoru JDBC

Klíčová slova pro jazyky máme uložené v databázi. Proto se nejdříve připojíme k databázi a k tomu nám poslouží konektor JDBC

Předtím než se budeme připojovat k databázi je nejdříve potřeba importovat balíček `java.sql` pomocí direktivy `<%@ page import="java.sql.*" %>`

Pro připojení k Databázi musíme udělat následující kroky:

a) Načtení ovladače databáze

V tomto kroku vytváříme instanci třídy implementující JDBC driver příslušného databázového serveru

```
new com.mysql.jdbc.Driver();
```

b) Vytvoření mysql jdbc připojení

Třída `JDBC DriverManager` definuje objekty, které mohou připojit Java aplikaci k JDBC ovladači, který je nainstalován na našem systému. Metoda `getConnection()` je používána k zahájení připojení k databázi. Předáváme jí

data, pomocí kterých se připojujeme k databázi – jméno, heslo, server a jméno databáze. A vrací nám objekt s připojením.

```
Connection conn =
DriverManager.getConnection
      ("jdbc:mysql://localhost/bp?user=root");
```

c) Tvorba statement objektu

Objekt používaný pro spuštění SQL syntaxe.

```
Statement stm = conn.createStatement();
```

d) Vytvoření a zpracování dotazu

Mám dotaz na databázi, který mě vrátí vše z tabulky `jsp`, ve které mám uložený lokalizované texty. Dotaz provedu pomocí metody `executeQuery()` třídy `ResultSet`. Vrací nám tabulku, čili dvourozměrné pole.

```
ResultSet res1 = stm.executeQuery("SELECT * FROM jsp");
```

Nastavíme ukazatel na začátek, před první řádek pomocí metody `beforeFirst()`

```
res1.beforeFirst();
```

Nyní můžeme zpracovat dotaz například pomocí cyklu `While`. Posouvání na další řádek v tabulce nám dělá metoda `next()`

```
while (res1.next()) {
. . . . .
}
```

e) Ukončení práce se spojením

Po ukončení zpracování výsledku musíme uzavřít statement, uzavřít a vynulovat spojení. K tomu nám poslouží metoda `close()`.

```
stm.close();
conn.close();
conn = null;
```

4.2.3 Zpracování klíčového slova

Konečně víme do jakého jazyka chceme lokalizovat a jsme připojený k databázi, kde máme uložené lokalizované texty, dle klíčového slova a jazyka. Teď musíme vytvořit podmínku, která nám zajistí načtení textu daného jazyka dle klíčového slova, které chceme.

Pomocí metody `getString()` nám již známe třídy `ResultSet`, vyhledáme hodnotu námi požadovaného sloupce právě zpracovávaného řádku. Porovnáme tedy hodnotu sloupce `klíč` s klíčovým slovem `history`, které zastupuje text s lokalizovanou historii a porovnáme hodnotu sloupce `lang` s proměnou `lang`, kterou již máme zpracovanou.

```
If(res1.getString("klic").equals("history")&&
res1.getString("lang").equals(lang))
```

Pokud se nám splní tato podmínka, už nám nic nebrání tomu, abychom načetli lokalizovaný text s historií pomocí metody `getString()` a zformátovali ho do odstavce pomocí tagu `<p></p>`

```
<p>
<%= res1.getString("txt")%>
</p>
```

5 PHP a funkce iconv

5.1 Možnosti využití funkce iconv

Při lokalizaci nás nezajímá pouze samotný překlad z primárního jazyka do jiného, ale také převod znakové sady (středoevropská, západní, atd.). K tomuto účelu nám slouží velice užitečná funkce v jazyku PHP, funkce iconv. Nemusí se jednat pouze o převod HTML stránek, ale také konverze kódování databáze, souborů a jiné.

Pro lepší představu uvádím pár situaci:

- Nastane situace kdy máme webový projekt psaný v kódování UTF-8, ale načítáme jiné texty ze souborů, které jsou uloženy ve středoevropském kódováním ISO-8859-2.
- Mám webovou stránku, která má dva typy zobrazení. Jednou jako HTML stránka s kódováním ISO-8859-2 a podruhé využívá k vizualizaci informací Macromedia Flash. Flash ovšem neumí správně zobrazit znakovou sadu ISO-8859-2 a proto můžeme udržovat všechny zdrojové informace v UTF-8 a pro zobrazení jako HTML stránku použijeme funkci iconv.
- Samotné převody HTML stránek z jednoho kódování do druhého
- Stránku HTML mám v kódování ISO-8859-2 a data v databázi uložené v UTF-8

5.2 Příklad použití funkce iconv

Jako názorný příklad ukážu webovou stránku, jejímž obsahem je jakýsi text ohledně historie Technické univerzity v Liberci. Tento text je v souboru, který je uložen s kódováním ISO-8859-2 (známe také jako Latin-2 či středoevropské kódování). Stránku však máme v kódování UTF-8, výpis tohoto textu by tedy nebyl správný.

5.2.1 *Nastavení kódování stránky*

Zpočátku nastavím kódování HTML stránky. Protože se snažím psát dle standardů W3C, je zde podle jednoho ze standardu nastaveno také kódování. Problém je v použití tagů `<?xml ?>`, které jsou zde použity. Server by si totiž mohl myslet, že jde o kód PHP. Řešením je vypsání těchto tagů a kódu v PHP kódu.

```
<?php
print '<?xml version="1.0" encoding="UTF-8"?>';
?>
```

Zde již je druhá část nastavení kódování spolu s nastavením jazyka a určení verze (X)HTML. Určení verze (X)HTML je důležité pro validátor kódu.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xml:lang="cs" lang="cs">
```

Ještě je potřeba definice jazyka a kódování pomocí meta tagů

```
<meta http-equiv="content-language" content="cs" />
<meta http-equiv="content-type" content="text/html;
charset=UTF-8" />
```

5.2.2 *Načtení a zpracování souboru*

Načtení textu v ISO-8859-2 je provedeno ze souboru. Pro tuto akci je potřeba znát funkce v PHP pro otevření souboru a jeho čtení.

```
$fp = fopen("./soubor.txt", "r");
$string_iso = fread($fp,2500);
```

Funkce `fopen()` otevře soubor z serverového filesystému nebo url a vrátí deskriptor. První parametr je cesta k souboru a druhý parametr je použit k volbě způsobu zpracování (čtení, zápis, nastavení ukazatele). Možnost `"r"` otevře soubor pro čtení a nastaví ukazatel na začátek.

Funkce `fread()` přečte soubor specifikovaného deskriptorem, který je uveden v prvním parametru. Druhý parametr určuje kolik bytů souboru se přečte. Čtení končí, pokud je přečteno daný počet bytů nebo je dosažen konec souboru (podle toho, co přijde dřív).

5.2.3 Konverze textu do UTF-8

Máme načtený text v ISO-8859-2 a potřebujeme ho v kódování stránky(UTF-8). Zde konečně použijeme funkci `iconv()`.

```
$string_utf8 = iconv( 'ISO-8859-2', 'UTF-8', $string_iso);
```

První parametr určuje výchozí kódování řetězce, druhý požadované kódování a poslední je zpracovávaný řetězec.

5.2.4 Výsledné zobrazení HTML stránky

Jako print screen zde uvádím HTML stránku, objasňující výsledek. Jak vidíme při nesprávném použití kódování se v textu místo neznámých znaků vypíše otazník.

text s kódováním ISO-8859-2, stránka však má kódování UTF-8

Technick? univerzita v Liberci se od roku 1989 stala z ?adov? vysok? ?koly s ryze technick?m zam??en?m a dv?ma fakultami uzn?vanou vysokou ?kolou s respektem doma i v zahrani??, ?kolou, kter? m? dnes ?est fakult a propojuje vzd?l?v?n? technick? se vzd?l?v?n?m humanitn?m.

text po převedení pomocí fce iconv z ISO-8859-2 do UTF-8

Technická univerzita v Liberci se od roku 1989 stala z řadové vysoké školy s ryze technickým zaměřením a dvěma fakultami uznávanou vysokou školou s respektem doma i v zahraničí, školou, která má dnes šest fakult a propojuje vzdělávání technické se vzděláváním humanitním.

Obr. 3: webová stránka s použitím funkce iconv

6 Knihovna gettext v PHP

Profesionálním řešením internacionalizace je použití knihovny GNU gettext, která je dostupná pro několik programovacích jazyků, a také i pro PHP. Knihovna nabízí funkci `gettext()`, kterou obalíme všechny řetězce, jež se mají lokalizovat.

Pro použití knihovny gettext musíme v PHP aktivovat příslušný modul. Většinou stačí v `php.ini` odkomentovat řádek s `extension=php_gettext.dll`.

6.1 Uvedení knihovny gettext

Jak jsme zvyklí vypisovat HTML kód a s tím text pomocí funkcí `echo` nebo `print`. Např.:

```
echo "Welcome!";
```

Pro použití knihovny gettext musíme vypisovat text takto:

```
echo gettext("Welcome!");
```

Nebo zkráceným aliasem této funkce je `_()`, takže píšeme:

```
echo _("Welcome!");
```

Vidíme tedy, že oproti původnímu zápisu stačilo přidat tři znaky navíc.

Funkce `gettext()` podle aktuálně nastaveného jazyka vrátí překlad odpovídající fráze. Pokud překlad neexistuje nebo jej nelze nalézt, vrací funkce původní argument – tedy text v angličtině nebo v jiném jazyce, ve kterém je aplikace primárně vyvíjena.

Psaní stránky se může kombinovat jako HTML a generované HTML pomocí PHP. Pro překládaný text se musí použít generování pomocí PHP abychom mohli použít funkci `gettext()`. Píšeme tedy kombinaci PHP a HTML.

```
<p>
<?php echo _("Enter your password:")?>
<input type="password" name="pwd">
</p>
```

Zvýšená námaha spojená s úpravou kódu aplikace se nám však brzy vrátí. Přeložené texty se pak pomocí dalších nástrojů převedou do binárního tvaru, který zajišťuje rychlý chod aplikace. Knihovna si navíc překlady ukládá do vyrovnávací paměti, takže je opravdu velmi rychlá. Zpomalení lokalizované aplikace oproti nelokalizované verzi je prakticky neměřitelné.

6.2 Praktický příklad použití knihovny gettext

V této kapitole uvedu způsob, jak co nejlépe zpracovat jazyk pro lokalizaci. Nastavíme gettext, zpracujeme soubor `*.po` s originálními hodnotami pro lokalizaci a vypíšeme přeloženou stránku.

6.2.1 Výběr jazyka

Nabízí se otázka, jak nejlépe vybrat jazyk pro lokalizaci. Nejlepší volbou se zdá vybrat jazyk automaticky, dle toho, kde se PC nachází nebo není-li dostupná tato lokalizace, poté vybrat jazyk uživateli blízký. Může nastat však situace, kdy se ocitneme v internetové kavárně v cizině a asi bychom neradi kdyby se nám web s více lokalizacemi nabízel pouze v Čínštině. Proto by uživatel měl mít možnost lokalizaci změnit. Například pomocí vlajky země, která je odkaz na url adresu, kterou změnu provedeme. Po volbě uživatele, také chceme, aby si systém pamatoval jaká lokalizace byla zvolena.

Výběr jazyka volíme pomocí odkazů, které jsou reprezentováni vlajky zemí lokalizovaných jazyků.

```
<a href="index.php?lang=cs"><img src="./czech.gif" /></a>
<a href="index.php?lang=en"><img src="./english.gif" /></a>
<a href="index.php?lang=de"><img src="./deutch.gif" /></a>
```

Odkaz jsme vytvořili takový, abychom mohli vytvořit proměnou `$lang` pomocí globální proměnné `$_GET["lang"]` a předat jí kód jazyka. Funkce `setcookie()` nám umožní uložit do `cookie` jazyk, který si systém bude pamatovat. Parametry jsou `klíčové slovo`, `proměnná` a `datum` do kdy bude `cookie` platit.

```
if ($_GET["lang"]) {
```

```
        $lang = $_GET["lang"];
        setcookie("lang", $lang, time() + 365*24*60*60);
    }
```

Načítáme proměnou z cookie pokud není předána url adresou a je uložený klíč `lang` v globální proměnné `$_COOKIE["lang"]`.

```
    if ((!isset($lang)) && (isset($_COOKIE["lang"]))) {
        $lang = $_COOKIE["lang"];
    }
```

Při volbě jazyka automaticky využíváme globální proměnné `$_SERVER`, která obsahuje v `HTTP_ACCEPT_LANGUAGE` informace o nastavení prohlížeče uživatele. Například `cs-CZ,cs;q=0.9,en;q=0.8`, kde první část je jazyk a země, druhá a každá další je preferovaný jazyk s kvocientem přednosti. Čím vyšší kvocient, tím více preferovaný jazyk. Pro lepší práci nám funkce `explode()` převede řetězec do pole. První položku tohoto pole poté zpracujeme tak, abychom dostali kód jazyka.

```
    if (!isset($lang)) {
        $lang = explode(",",$_SERVER["HTTP_ACCEPT_LANGUAGE"]);
        $lang = StrToLower(Substr(chop($lang[0]),0,2));
    }
```

Jak jistě bylo z kódu poznat, aplikace primárně vybírá jazyk pomocí url, který si poté pamatuje. Pokud uživatel nezadá nebo nezmění jazyk odkazem, přichází výběr dle nastavení prohlížeče.

Teď již máme zpracovaný jazyk, který uživatel preferuje. Bohužel máme omezený počet lokalizací. Proto si pokud není lokalizace dostupná, zvolíme defaultní jazyk a ošetříme možnost výběru podobného jazyka.

```
    if (($lang <> "de") && ($lang <> "cs") && ($lang <> "en")) {
        $lang = $lang == "sk" ? "cs" : "en";
    }
```

6.2.2 *Nastavení gettextu*

Předpokládáme-li, že proměnná `$lang` obsahuje kód požadovaného jazyka, pak jeho výběr pro gettext provedeme pomocí příkazů.

```
putenv("LANG=$lang");
setlocale(LC_ALL, $lang);
```

Adresář, v němž se očekává struktura adresářů s překlady pro jednotlivé jazyky, nastavíme pomocí příkazů.

```
bindtextdomain("messages", realpath("../locale"));
textdomain("messages");
```

Text `messages` přitom určuje jméno pro soubory s překlady. Můžeme použít libovolný jiný text. V našem případě používáme implicitní jména `messages.po` a `messages.mo`. Funkce `bind_textdomain_codeset()` nastavuje, které kódování bude `messages` vracet.

```
bind_textdomain_codeset("messages", "utf-8");
```

6.2.3 *Soubor s lokalizací*

Již máme připravený PHP kód, včetně cest, pro generování slov překladu. Protože gettext je primárně vyvíjen pro operační systém Linux, je lepší s ním pracovat pod Linuxem. Je potřeba Shell. Pod operačními systémy Windows můžeme emulovat shell pomocí programu cygwin.

Vytáhneme všechny texty z kódu pomocí příkazu

```
xgettext --from-code=utf-8 index.php
```

Kde parametr `--from-code=utf-8` určuje kódování a `index.php` je soubor z kterého generujeme texty.

Vznikne soubor `messages.po`, který obsahuje všechny texty k přeložení. Hlavička obsahuje informace o autorských právech

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE
package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
```

Části označené jako `msgid` odpovídají jednotlivým textům v originálním skriptu. Do částí `msgstr` se pak píše překlad do jednotlivých jazyků.

```
msgid "Our multilanguage page"
msgstr ""
```

Knihovna gettext přitom předpokládá, že pro každý jazyk je překlad uložen v souboru, který je v adresáři `cesta/jazyk/LC_MESSAGES`. Vygenerovaný soubor `messages.po` proto přesuneme do adresáře `../locale/en/LC_MESSAGES`.

Přípona souboru `.po` je zkratkou PO (Portable Object – přenositelný objekt). Pro použití gettextem je potřeba soubor převést z formátu PO do MO (Machine Object – strojový objekt), který je binární a umožňuje rychlé nalezení a vrácení překladu. Převod do binární podoby provedeme příkazem.

```
msgfmt messages.po
```

Vznikne tak binární soubor `messages.mo`.

Pro každý jazyk, do kterého chceme aplikaci přeložit, vytvoříme rovněž odpovídající adresář (například pro češtinu `../locale/cs/LC_MESSAGES`) a nakopírujeme do něj soubor `messages.po` a jeho binární podobu `messages.mo`. Před převodem doplníme do souboru české překlady.

```
msgid "Our multilanguage page"
msgstr "Naše vícejazyčná stránka"
```

6.2.4 Vypsání lokalizovaného textu

Binární soubory s lokalizací jsou připraveny a uloženy v adresářích pro češtinu, angličtinu a němčinu. Vybraný jazyk také máme. Stačí tedy pouze použít funkci `gettext()` nebo její alias `_()` a texty se lokalizují.

```
print "<h1>" . gettext("Welcome on my website") . "</h1>\n";
print "<p>" . _("some text about me") . "</p>\n";
print "<p>" . _("Thank you for your visit! See you soon.") .
"</p>\n";
```

6.2.5 *Množné číslo při intenacionalizaci*

Občas potřebujeme v aplikaci vypisovat texty, jejichž obsah a gramatický tvar se liší podle doplňované hodnoty. Například:

```
Máte 1 nový e-mail.
Máte 2 nové e-maily.
Máte 16 nových e-mailů.
```

K dispozici proto máme funkci `ngettext()`, která standardně počítá s angličtinou, kde je jeden jiný tvar množného čísla. Funkce má tři parametry - frázi pro jednotné číslo, frázi pro množné číslo a počet. Podle posledního parametru gettext vybere odpovídající ze dvou frází.

Některé jazyky však mají více tvarů množných čísel – například čeština tři. I to lze pomocí gettextu řešit. Do PO souboru překladu vložíme direktivu:

```
Plural-Forms:nplurals=3;plural=n==1 ? 0 : n>=2 && n<=4 ? 1 : 2;
```

Ta podle parametru `n` určí, která ze tří variant textu se použije. Tato varianta podmínky odpovídá progresivnějšímu způsobu skloňování (jeden email, 2-4 emaily, 5 a více emailů).

Do PO souboru s českým překladem pak umístíme několik tvarů množného čísla:

```
msgid "You have %s new e-mail."
msgid_plural "You have %s new e-mails."
msgstr[0] "Máte %s novou e-mailovou zprávu."
msgstr[1] "Máte %s nové e-mailové zprávy."
msgstr[2] "Máte %s nových e-mailových zpráv."
```

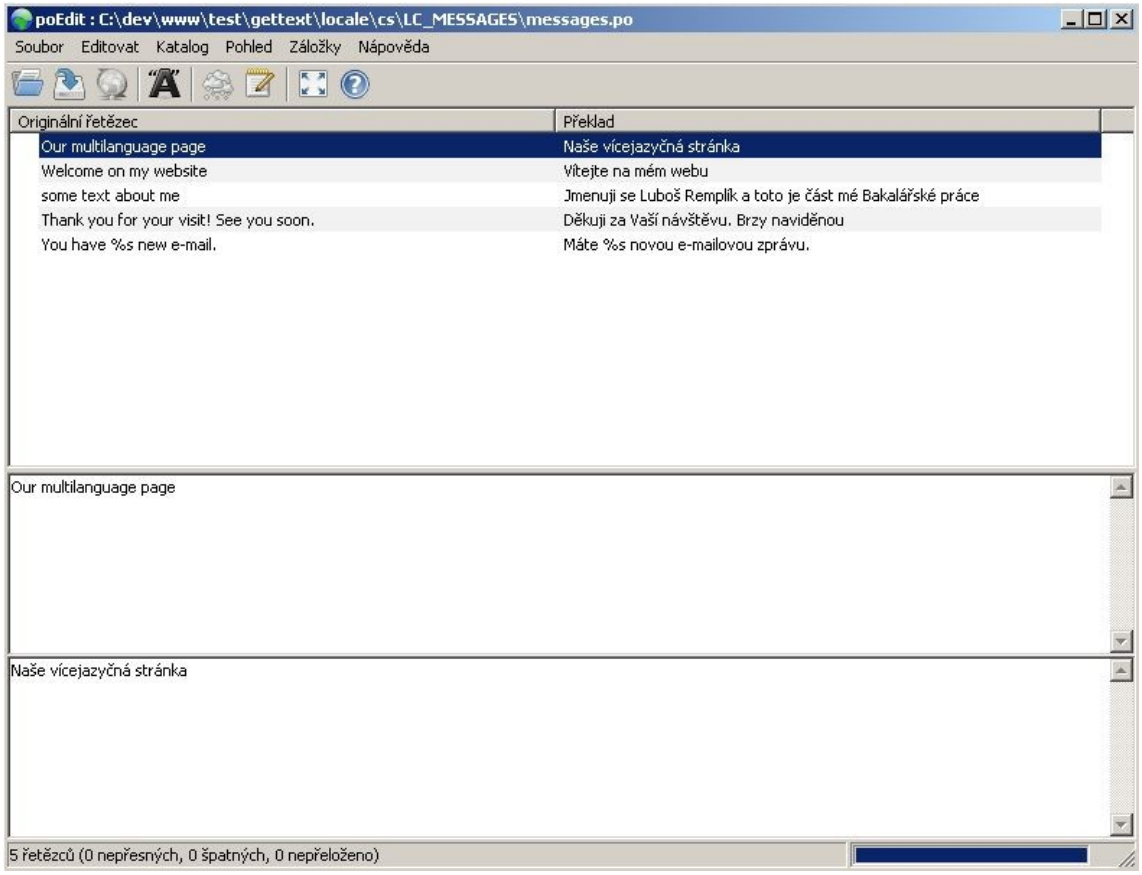
Různé tvary množného čísla se do výstupu nejčastěji vkládají pomocí funkce `printf()`, která umožňuje doplnit číslo do výstupu:

```
$num = 5;
for ($n = 1; $n <= $num; $n++) {
    printf(ngettext("You have %s new e-mail.", "You have %s new
           e-mails.",$n),$n);
}
```

6.2.6 Software pro překlad

U větších aplikací začne být ruční editování PO souborů poměrně nepohodlné. Naštěstí existuje několik programů, které nabízejí pohodlné grafické prostředí správy souborů s překlady. Nejznámější z nich jsou uvedeny v následujícím přehledu:

- poEdit - lze spustit jak v systému Linux, tak i v systému Windows
- gtranslator – určený pro systém Linux s desktopovým prostředí Gnome
- Kbabel – určený pro systém Linux s desktopovým prostředí KDE



*Obr. 4: poEdit - program pro editaci *.po souborů*

7 Zhodnocení jednotlivých řešení

Uvedu zde další způsoby internacionalizace, které jsou nevhodné nebo málo používané. A hlavně tuto kapitolu věnuji analýze výhod a nevýhod předchozích řešení, včetně doporučení, kde metodu použít či nepoužít.

7.1 Nevhodné metody internacionalizace

Na internetu, na různých webových portálech je k vidění lokalizace webu prostým přeložením jednotlivých HTML stránek pro různé jazyky. Tyto překlady tvoří dílčí weby, které jsou uložené v adresáři (co jazyk, to adresář). Tento způsob, jak jistě uznáte, je velice nevhodný a v rozsáhlejších projektech téměř nepoužitelný. Řešení má dvě velké nevýhody. Jednak se web těžko udržuje, protože kód aplikační logiky se vyskytuje opakovaně v několika jazykových mutacích. Druhým problémem je zbytečně náročný překlad, protože opakující se fráze v textech se musí překládat několikrát.

Další řešení spočívá v používaní konstant na místě textů. Tyto konstanty se pak za běhu programu nahradí odpovídajícím překladem z textového souboru nebo z databáze. Je to obdoba uvedeného řešení pomocí ResourceBundles, které nám však nabízí několik užitečných metod, které již nemusíme programovat a můžeme je rovnou využívat.

7.2 Analýza výhod a nevýhod uvedených řešení

Jistou výhodou, jak jsem během testování jednotlivých internacionalizací zjistil, je udržovat stejné kódování (webových stránek, databáze, souborů atd.). Existují sice užitečné funkce pro konverzi znakových sad, jako je uvedená funkce iconv() v PHP, ale každé jejich použití je práce navíc. Pří výběru kódování doporučuji znakovou sadu UTF-8, která je doporučeným zápisem Unicode a navíc odstraňuje všechny jeho nevýhody - zachovává kompatibilitu se stávajícím kódem a přitom umožňuje aplikacím použít všech výhod univerzální mezinárodní znakové sady.

Při použití ResourceBundle v JSP, nebo obecně jakékoli třídy v jazyce Java, je velkým přínosem kvalitní dokumentace a javaDoc pro její tvorbu. Jakoukoli vlastní používanou třídu tímto nástrojem jednoduše dokumentuji a pozdější použití nemusí znamenat nutné pochopení zdrojového kódu třídy.

Třída ResourceBundle je užitečná při použití nejen u lokalizace dle jazyka, ale může také lokalizovat dle kraje a dokonce podle varianty – druhu Operačního systému (Windows, Linux). Jestli je prospěšnější použití PropertyResourceBundle či ListResourceBundle je otázkou programátora a jeho způsobu psaní aplikací. Osobně preferuji první způsob, protože řetězce s lokalizacemi jsou uloženy přehledně v souboru jehož název určuje jazyk (oblast, varianta) a má také striktně dané místo v systému souborů na webovém serveru. Při překládání textů do různých jazyků snadno najdeme cestu souborů a jednoduše určíme jméno pro soubor – můžeme si vytvořit aplikaci pro překládání, která s těmito hodnotami počítá. Nevýhodou těchto metod je pomalejší načítání dat ze souboru oproti načítání z databáze či binárního souboru.

Při použití skriptovacího jazyka (JSP) a databáze (MySQL) pro internacionalizaci je největší výhodou rychlost databáze. Autoři MySQL se dokonce netají, že právě rychlost je jejich hlavní prioritou a že se kvůli ní ani nechystají implementovat některé funkce. Dříve se například dlouho bránili použití triggerů. Dalšími výhodami použití databáze je jednoduchost, přehlednost a spousta podpůrných nástrojů. Databázím se dá namítnout složitější, ale bezpečnější způsob připojení k datům – potřeba zavedení ovladače, tvorba statementu a poté až možnost použití dotazu.

Ve čtvrté kapitole jsem uvedl dvě možné struktury databáze – tabulka se třemi sloupci a tabulka s dvěma či více sloupci. Použití jedné nebo druhé struktury se liší dle druhu webu. Výhodnější se zdá tabulka s třemi sloupci, protože při potřebě přidání další lokalizace není potřeba zasahovat do struktury databáze a také zde nevznikají NULL hodnoty. Prázdné hodnoty jsou klíčovým parametrem výběru druhu tabulky. Pokud administrátor webu upravuje texty

lokalizace přímo v databázi je pro něj z důvodu přehlednosti výhodnější druhý způsob – NULL hodnoty jasně vymezují jazyk ve kterém nebyl přiřazen text ke klíčovému slovu. Pokud však administrátor používá k editaci textů nějakou aplikaci, je z důvodů výše uvedených lepší použít první způsob struktury tabulky.

Skriptovací jazyk pro tento způsob internacionalizace nemusí být jen JSP, ale také například PHP. Tento jazyk je v dnešní době velmi oblíbený a rozšířený. Existuje mnoho knihoven či frameworků, které usnadňují práci s databází (Pear, ZendFramework atd.). Jedním z ukazatelů oblíbenosti jsou servery s free webhostingem, které v mnoho případech PHP podporují. V Čechách jsem nenašel jediný webhosting poskytující JSP zadarmo.

Posledním způsob internacionalizace, kterým jsem se zabýval je použití knihovny gettext. Gettext je bezesporu kvalitní knihovna. Nabízí podporu u hodně programovacích jazyků, umí pracovat s množným číslem a nabízí značné množství programů k editaci překladů. Nevýhodou se může jevit ukládají do textového souboru a nikoli do databáze, což může znesnadňovat jejich editaci. Rozdíl rychlosti načtení lokalizované a nelokalizované webové stránky je neznatelný.

7.3 Doporučení použití uvedených řešení

Při výběr způsobu internacionalizace by programátor měl brát v úvahu, že existuje mnoho způsobů a rozhodnout se dle toho jaký skriptovací jazyk (JSP, PHP) používá. Další parametr výběru je budoucí způsob editace překladů. Edituje-li texty tvůrce webu je zbytečné hledat nástroje pro administraci. Při editaci správcem webu hledáme nástroje pro editaci (poEdit). Posledním činitelem, i když u uvedených způsobů je rozdíl zanedbatelný, je rychlost.

Pro každé řešení existuje situace, kdy jej nemůžeme použít. Nemáme podporu skriptovacího jazyka. Není na severu databáze – použijeme tedy soubory. V případě gettextu nemusí být na hostingu knihovna k dispozici a protože v PHP neexistují interní funkce pro vytvoření binární podoby souboru s překladem musíme volat externí příkaz, což také není možné vždy.

Obecně usuzuji, že pokud vytváříme menší web je lepší sáhnout po hotovém řešení – tzn. využít gettextu v případě PHP či ResourceBundle v JSP. V případě většího projektu bych použil řešení uvedené ve čtvrté kapitole pomocí JSP (PHP) a databáze s doplněním o ošetření množného čísla.

Závěr

V mojí bakalářské práci jsem měl za úkol seznámit čtenáře s efektivní podobou internacionalizace v programovacích jazycích JSP a PHP. Pomocí několika metod spolu s řešením souvisejících témat a zhodnocením, jsem splnil vytčené cíle práce v zadání.

Internacionalizace je jistě důležitou oblastí při tvorbě webových stránek tvořených jak v JSP, tak PHP. Může zvýšit podvědomí u lidí cizích národností o společnosti či organizaci. Existence různých znakových sad tento proces znesnadňují. Výběr univerzálního kódování a použití funkcí jako je iconv v PHP nám internacionalizaci ulehčí.

Jak je jistě vidět nejvíce prostoru je poskytnuto řešení pomocí knihovny gettext. Toto řešení má i přes své nevýhody mé sympatie. Pokud nám to okolnosti umožňují (PHP s gettextem, program pro převod souboru do binární podoby) doporučuji použít tuto metodu internacionalizace.

Uvedl jsme ukázky zdrojového kódu. Snažil jsem se poukázat na hlavní rozdíly mezi jednotlivými řešeními. Posouzení, zda se mi to povedlo, nechám na čtenáři. Děkuji za vynaloženou energii se čtením této práce.

Zdroje

[1] Burd, Barry: JSP. Brno, Computer Press, 2003

[2] Herout, Pavel: Učebnice jazyka Java. České Budějovice, Kopp, 2000

[3] http://java.sun.com dokumentace k programovacímu jazyku JAVA a JSP

[4] http://jakarta.apache.org dokumentace k serveru Tomcat

[5] http://php.net dokumentace k programovacímu jazyku PHP

[6] http://www.gnu.org/software/gettext/ dokumentace ke knihovně gettext

[7] http://www.mysql.com/ dokumentace k databázi MySQL

[8] http://interval.cz tutoriály a diskuzní fóra tykající se technologií PHP a MySQL

[9] http://www.jsptut.com/ tutoriál k programovacímu jazyku JSP

Příloha A

Tato příloha je CD obsahující všechny zdrojové kódy praktických příkladů a elektronickou formu bakalářské práce ve formátu *.odt a *.pdf. Také přikládám Acrobat Reader pro čtení PDF formátů.