Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Návrh a implementace rozhraní modulu "Projekty"

Plzeň, 2012 Kohout Josef

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 1.5.2012, Kohout Josef

….....

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Pavlu Královi, Ph.D. za cenné rady, připomínky a metodické vedení práce. Dále pak Ing. Pavlu Grigarovi a Ing. Petru Jirouškovi za konzultace ohledně získávání dat ze sytémů STAG a OBD.

Abstract

Design and Implementation of Interface of "Projects" Module

The main goal of this bachelor thesis is to develop the plugin *Projects* into the content management system OpenCMS in the programming language Java.

This module will be deployed as a part of the web pages at the Department of Computer Science and Engineering. It will be used for project administration and for presentation of these projects to public using the department web pages. This module can be also used to gather and to import data from other information systems used at the University of West Bohemia in Pilsen.

Obsah

1 Úvod

Cílem této bakalářské práce je vytvořit webové rozhraní pro modul na správu projektů/grantů do redakčního systému OpenCms použitého na Katedře informatiky a výpočetní techniky Fakulty aplikovaných věd na Západočeské univerzitě.

OpenCms je rozsáhlý redakční systém napsaný v jazyce Java. Jeho velkou předností je možnost rozšiřování základní funcionality pomocí modulů a je používán na KIV pro katedrální web.

Hlavním důvodem pro vznik nového modulu je usnadnění evidence projektů KIVu, a jejich následného zobrazení návštěvníkům webových stránek. Mezi podstatné usnadnění patří i získávání informací o projektech z informačního systému IS-STAG (IS/STAG je informační systém pro evidenci studijní agendy vysoké školy nebo univerzity) a OBD (Osobní bibliografická databáze) a implementace importu těchto dat. Snahou je celý tento proces pokud možno automatizovat a také mít projekty uloženy v jedné struktuře pro jednotné zobrazování.

Další snahou této práce je optimalizace vývoje modulů a zpřehlednění kódu v JSP stránkách.

Tato práce navazuje na Projekt 5, který jsme vytvářeli s kolegou Alešem Pivničkou a jehož hlavním obsahem bylo prozkoumat možnosti použití relačně objektového mapovaní pro moduly KIVu v prostředí OpenCms.

2 Teoretická část

2.1 Způsob evidence grantů na KIV

Návrh projektu – navržené projekty se na sekretariátu KIV zakládají kopií a do okamžiku přijetí projektu se dále nic neděje.

Po přijetí projektu dodá řešitel na sekretariát jednu kopii smlouvy a v ten okamžik se na sekretariátu zakládají desky, do kterých se vkládají kopie všech dokumentů souvisejících s projektem po celou dobu řešení a to včetně čerpání projektu. Na sekretariátu jsou tedy jen kopie dokumentů v papírové podobě. Všechny originály jsou uloženy na oddělení vědy na rektorátu, kde se všechny potřebné údaje zadávají do systému INIS. Z tohoto systému se poté dle potřeby provádí export a z něho získaná data se ručně vkládají na stránku o projektech na katedrálním webu KIV.

2.2 Technologie pro tvorbu internetových aplikací

2.2.1 Internetová aplikace

Internetová aplikace poskytuje data po síti - Internetu. Jsou to aplikace, které jako aplikační rozhraní používají v drtivé většině webovou prezentaci (prezentační vrstva). V dřívějších dobách stály „klasické“ aplikace (síťové) a web na odlišných stranách. Web sloužil pro zobrazování/sdílení dat a informací, jakými jsou např. prezentace firmy či zpravodajské servery, kdežto síťové aplikace se psaly jako serverová část a k ní klient, který se musel instalovat na každý počítač. Poslední dobou se ale trend psaní aplikací radikálně změnil. Není už třeba složitého klienta, který se musí instalovat na každý počítač, když lze využít pro zobrazení všeobecně dostupné a kvalitní webové prohlížeče, které již neslouží jen pro jednoduché zobrazovaní statických textů. Koncept webové aplikace přináší obrovské výhody, jako například:

- uživatel nemusí instalovat žádný software na svém počítači,
- k ovládání postačuje pouze internetový prohlížeč,
- snadnější aktualizace,
- rychlejší servis.

Mezi internetové aplikace lze zařadit

- publikační systémy pro správu obsahu (content managment systém (CMS)),
- objednávkové a rezervační systémy,
- e-shopy,
- rozličné webové služby,

což jsou klasické aplikace jak je známe z webu. Dnes se ale v tomto konceptu vyvíjejí i mnohem větší a komplexnější systémy (informační systémy) pro řízení a správu firem. Jsou to sice aplikace, které jsou nasazovány majoritně v rámci firemního intranetu, ale ten funguje na stejném principu a protokolech jako síť Internet.

Mezi nejčastěji používané technologie se dnes řadí:

- Groovy & GSP (Groovy Server Pages),
- PHP (PHP Hypertext Preprocessor, dříve Personal Home Pages),
- ASP (Active Server Pages),
- Java servlet & JSP (Java Server Pages),
- CGI (Common Gateway Interface).

2.2.2 Groovy

Groovy je alternativa k programovacímu jazyku Java. Je postaven nad platformou Java a jde o syntakticky rozšířený a s Javou zpětně kompatibilní jazyk. Je inspirován spoustou programovacích a skriptovacích jazyků – Perl, Python, Ruby a další. Je překládán do JVM bytecode a spolupracuje s javovými knihovnami. Asi nejzajímavější pro vývoj internetové aplikace s využitím Groovy je projekt Grails[1] (dříve známé jako Groovy on Rails). Grails je webový framework založený na jazyku Groovy, frameworku Spring a ORM frameworku Hibernate. Je to docela mladý framework, který se rychle vyvíjí. Mezi jeho hlavní přednosti patří:

- GORM – (Grails object relational mapping), vlastní objektově relační

1 http://www.grails.org/

framework,

- GSP – (Groovy server pages), rozšíření klasických JSP (Java Server Pages) tagů (značek) o Grails tagy a lepší práci s Grails frameworkem (jedná se o rozšíření klasických JSTL (Java Server Pages Standard Tag Library) tagů o nové tagy pro lepší práci s frameworkem, například zjednodušení práce s formuláři),
- Scaffolding – schopnost frameworku vytvářet automaticky CRUD(Create, Read, Update, Delete) operace k doménovým třídám (třídy které se mapují na databázi),
- integrované snadné a rychlé testování,
- rychlý vývoj aplikace, silná příkazová řádka – příkazy z příkazového řádku lze jednoduše generovat kostru aplikace či její části, např. obrazovky pro CRUD operace a metody pro jejich obsluhu,
- jednoduché výtváření vlastních tagů pro často se opakující části stránek,
- možnost psát některé pomocné třídy v čisté Javě,
- použití Springu – možnost využít Spring MVC (implementace konceptu Model-View-Controller ve frameworku Spring), Spring IoC (implementace návrhového vzoru Inversion of Control ve frameworku Spring), napojení na Hibernate (framework pro objektově relační mapování).

2.2.3 Java servlety & JSP (Java server pages)

2.2.3.1 Java servlety

Servlety jsou programy napsané v jazyce Java a slouží ke generování dynamickych webových stránek[2]. Implementují princip požadavek – odpověd, což je typické pro architekturu klient – server. Java Servlet API (Application Programming Interface) je množina tříd, které definují standardní rozhraní pro obsluhu požadavků a odpovědí mezi klientem a serverem:

- server obdrží od klienta požadavek,

2 Dynamické webové stránky jsou stránky generované ná základě určitého požadavku/podle určitých proměnných.

- server pošle požadavek na konkrétní servlet,
- servlet vytvoří odpověď (dynamicky podle požadavků klienta) a předá ji serveru,
- server pošle odpověď klientovi.

Servlet tedy není samostatný běžící proces, ale beží v rámci nějakého webového serveru, v tomto případě se též nazývá servletový kontejner. Webový server má za úkol inicializovat, spoustět, ukončovat a uvolňovat servlet z paměti. Běžně v paměti serveru běží jedna instance servletu pro každé uživatelské sezení a je zavedena i po dobu své nečinnosti.

Java Servlet API se skládá ze dvou balíčků:

- `javax.servlet`,
- `javax.servlet.http`.

Pokud chceme vytvářet http servlet, stačí vytvořit třídu oddědenou od abstraktní třídy `HttpServlet` a pak překrýt některou z metod:

- `doGet()` pro podporu HTTP GET dotazu,
- `doPost()` pro HTTP POST,
- `doPut()` pro HTTP PUT,
- `doDelete()` pro HTTP DELETE,
- `init()` a `destroy()` pro specifickou správu prostředků při inicializaci a ukončování servletu (pokud nejsou implementovány, zavolá se metoda předka),
- `getServletInfo()` pro poskytování informací o servletu.

Nejčastěji se implementují pouze metody `doGet()` a `doPost()`. Pokud se implementují obě zároveň, je vhodné v jedné z nich zavolat druhou, aby byla vracená odpověď identická (většinou nechceme, aby se odpověď lišila podle metody dotazu).

Každý dotaz je objekt s rozhraním `HttpServletRequest` a daná odpověď objekt s rozhraním `HttpServletResponse`. Objekt `response` má `writer` (zapisovač),

do kterého je možno zapisovat výstup posílaný zpět ke klientovi podle parametrů v dotazu (`requestu`). Výstupu posílanému zpět klientovi lze nastavit např. i content type.

Konfigurace výsledné aplikace/servletu se provádí v souboru `web.xml` (konfigurace ale není bezpodmínečně nutná, k servletu se dá přistupovat i jeho plným jménem v URL, tzn. balicek.MujServlet). Jedná se o klasický XML soubor, konfigurující nastavení webového kontejneru, ve kterém můžeme např. uvést relativní URL, pod kterým bude `servlet` identifikován, nebo můžeme definovat bezpečnostní omezení. V konfiguračním souboru lze nadefinovat dále například inicializační parametry, spojení do databáze apod.

Ukázka jednoduchého konfiguračního souboru:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
 <!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
 <servlet>
  <servlet-name>Hello</servlet-name>
  <servlet-class>package.MujServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>Hello</servlet-name>
  <url-pattern>/myHello</url-pattern>
 </servlet-mapping>
</web-app>
```

- `DOCTYPE` říká, jakou verzi verzi servlet API chceme použít,
- `servlet` – párový tag, uvnitř tagu se nachází popis servletu,
  - `servlet-name` – jméno servletu,
  - `servlet-class` – vlastní třída servletu,
- `servlet-mapping` – část pro nastavení mapování servletu,
  - `servlet-name` – jméno servletu, který se má mapovat,
  - `url-pattern` – vzor cest přiřazených danému servletu.

2.2.3.2 JSP

Java Server Pages **[1]** (JSP) jsou vyšší úroveň servletů. Jedná se o HTML stránky, do kterých se vkládají speciální formátovací znaky, ve kterých se nachází klasický kód psaný v Javě (většinou `<% Java kód %>`). Tento kód vytváří dynamicky generovaný obsah. Ve webovém serveru dochází k překladu JSP na servlet, proto vyšší úroveň servletu. Celý proces vypadá přibližně následovně:

1. klient pošle dotaz na JSP stránku,
2. server obdrží dotaz a zjistí, že se jedná o JSP a přesměruje soubor do JSP servlet Engine,
3. pokud je stránka volána poprvé, dojde k parsování a kontrole JSP souboru, pokud už byl zavolán někdy dříve, pokračuje se rovnou bodem 6,
4. dojde k vygenerování zdrojového kódu servletu na základě JSP a zdrojový kód se převede do bytecode,
5. vytvoří se instance servletu,
6. dojde k vygenerování odpovědi na základě uživatelem poslaného dotazu.

Pomocí JSP není možno generovat něco, co by nešlo vytvořit servlety, to ostatně vyplývá z toho, že se JSP překládá na servlet. Hlavní výhoda JSP oproti servletům je v tom, že oddělují prezentaci od obsahu (aplikační logiky).

Kromě čistého HTML se v JSP používají tři základní konstrukce:

- skriptovací značky,
- direktivy,
- akce.

Skriptovací značky jsou čtyři a to:

- **Výrazy** `<%= vyraz %>` - výsledek výrazu se vloží přímo na výstup servletu.
- **Skriptlety** `<% kód %>` - obsahují složitější kód nežli výrazy, mají přístup k implicitním objektům `request`, `response` (`HttpServletRequest` a `HttpServletResponse`) a pododobně. Kód napsaný ve skriptletu se v nezměněné podobě vkládá do servletu. Pokud se v něm má něco vypsat

na výstup, je t5eba použít proměnnou `out`(u výrazu je použitá implicitně).

- **Deklarace** `<%! kód %>` - slouží pro definování metod a proměnných a negenerují žádný výstup.
- **Komentáře** `<%-- komentář --%>` - jedná se o komentáře, které ale nejsou vkládány do stránky generované a posílané klientovi

JSP direktivy (nerozšiřují možnosti generování stránky) se zapisují ve formátu `<%@direktiva atribut="hodnota">` a musí se vyskytovat na začátku souboru. JSP direktivy jsou:

- **page** – definuje nastavení stránky. Atributy jsou například **language** nebo **contentType**,
- **import** – tato direktiva se liší od ostatních tím, že se může vyskytovat kdekoliv na stránce. Má atribut **file** definující soubor, který se importuje před překladem stránky,
- **taglib** – umožňuje definovat vlastní značky a akce. Zapisuje se ve formátu `<%@taglib uri="TLD" prefix="prefix" %>` kde TLD je soubor descriptoru knihovny značek a prefix určuje, jak budou jednotlivé tagy přístupné (`<prefix:tag />`). Jedná se o takzvaný namespace.

Ohledně JSP ještě stojí za zmínku skriptovací jazyk EL (expression language), který se objevil v JSP verzi 2.0. Usnadňuje především přístup k proměnným v modelu, ke kterým se dříve muselo přistupovat přes výrazy. Místo výrazu lze tedy použít zápis `${proměnná nebo výraz}`, který je podstatně přehlednější.

V JSP můžeme dále používat sadu již připravených knihoven tagů, které řeší nejčastější akce při vývoji. Tato sada se souhrně jmenuje JSTL a rozděluje se na čtyři skupiny:

- Core – tagy pro podmínky, cykly, zpracování chyb a práce s URL,
- Formatting – podpora internacionalizace a lokalizace,
- XML – tagy pro zpřístupnění XML struktur, podpora pro transformace XSLT (eXtensible Stylesheet Language Transformations – transformace XML

do libovolného jiného formátu),

- SQL – podpora práce s relačními databázemi.

2.2.4 Další technologie

2.2.4.1 PHP

PHP[3] (PHP Hypertext Preprocessor) je skriptovací jazyk pro tvorbu webových aplikací. Skripty se provádějí na straně serveru (interpretem PHP skriptu) a uživateli se zpět posílá pouze čisté HTML. PHP je platformově nezávislé, má podporu pro přístup a práci s databází a je volně k dispozici obrovské množství knihoven např. pro práci s obrázky, textem atd. Dnes je to pravděpodobně nejrozšířenější jazyk pro tvorbu dynamických webů.

2.2.4.2 ASP.NET

ASP.NET[4] je technologie od firmy Microsoft pro vývoj webových aplikací. Základem je .NET framework. Jde o alternativu k jazyku Java a servletům, ale na rozdíl od Javy je tato technologie primárně určena pro platformu Windows.

2.2.4.3 CGI

CGI (Common Gateway Interface) **[2]** je protokol na komunikaci mezi webovým serverem a externí aplikací/skriptem. Webový server předeleguje dotaz skriptu, ten dotaz zpracuje a vrátí serveru zpět výsledné HTML. To ovšem znamená, že se pro každý dotaz spouští nový proces. Navíc musí mít skript nastavená práva tak, aby ho webový server vůbec mohl spouštět. V dnešní době se jedná již o docela zastaralý způsob vytváření dynamických stránek, je spíše vhodný pro výstupy různých statistik a informací ze systému (především Linux).

2.3 OpenCms

OpenCms **[3][4]** je open-source content managment system, tedy systém pro správu obsahu. Je vyvíjen firmou Alkacon Software, první verze byla vydána již v roce 2000 a je uvolněn pod licencí LGPL verze 2.1, která umožňuje upravit OpenCms a tuto

3 http://php.net/
4 http://www.asp.net/

úpravu dále šířit pod jinou licencí (tato nová licence ale musí např. umožnit modifikace pro uživatelovu vlastní potřebu). OpenCms je založen na technologiích Java a XML a jde o webovou aplikaci, která se instaluje na webový server. Přistupuje se k ní pomocí webového prohlížeče – primárně prohlížečem Firefox nebo Internet Explorer. Jako každý lepší CMS systém, poskytuje OpenCms tuto funkcionalitu:

- správa uživatelů a uživatelských skupin a oprávnění,
- workflow pro správu obsahu,
- sledování verzí dokumentů, jejich změn,
- publikování,
- má integrovaný WYSIWYG (What You See Is What You Get) editor,
- podpora vícejazyčných verzí dokumentů (prezentace),
- šablonování stránek,
- modulární systém pro snadnou rozšiřitelnost,
- integrovaný engine pro fulltextové vyhledávání (a to včetně v souborech typu pdf, doc a xls).

2.4 ERA model webu KIV

Celé schéma ERA modelu KIV je znázorněno v příloze 1A.

Potřebám modulu Projekty stačí pouze sedm tabulek z celého modelu. (příloha 1B). Proto se zabývám především jimi a několika dalšími, které jsou podstatné pro celý web .

Pro verzování celého modelu slouží tabulka `KIV_DB_VERSION`, která má dva sloupce:

- **autor** – datový typ `VARCHAR2`
- **verze** – u tohoto sloupce je překvapivé, že je datového typu `VARCHAR2`, ale ukládá se do něj datum ve formátu `YYYY-MM-DD`

Podstatnou tabulkou celého modelu je KIV_PERS_OSOBY. Slouží pro ukládání informací o členech katedry. Na tuto tabulku má každá tabulka (vyjma vazebních

a číselníků) nestandardní vazbu, která není řešena obvyklým způsobem, to jest pomocí integritního omezení `FOREIGN KEY`, ale pouze sloupcem typu `INT`, kam se programově ukládá primární klíč osoby, která záznam vytvořila (v tabulkách sloupec s názvem `ZAZNAM_EDITOR`).

Pro modul Projekty byly již ve schématu vytvořeny tabulky a to:

- **KIV_VYZK_GRANTY** – tabulka pro uložení informací o grantu,
- **KIV_VYZK_SKUPINY** – tabulka pro výzkumné skupiny,
- **KIV_VYZK_SKUPINY_OSOB** – vazební tabulka přiřazující osobu do výzkumné skupiny. V této tabulce lze nadefinovat, jestli je přiřazená osoba zároveň vedoucím skupiny. Zde je také jedna specialita – přestože se evidentně jedná o logickou hodnotu, která má pouze říct jestli je daná osoba vedoucím, sloupec je nadefinován jako `tinyint` a může tedy nabývat hodnot od - 126 do 127,
- **KIV_VYZK_GRANTY_OSOB** – vazební tabulka přiřazující osoby přímo ke grantu, stejná jako předchozí zmíněná, jen místo vazby na skupinu má vazbu přímo na grant,
- **KIV_VYZK_GRANTY_TYPY** – číselník pro typy grantů,
- **KIV_VYZK_SKUPINY_TYPY** – číselník typů výzkumných skupin.

2.5 OpenCms moduly webu KIV

Jedná se o moduly vytvořené speciálně pro účely webu Katerdy informatiky a výpočetní techiky. Jsou to moduly **[5]**:

- **Common** – společný základ všech modulů, v tomto modulu je důležitá knihovna **webkiv-db**, která poskytuje operace nad databází,
- **Témata** - evidence témat PRJx, BP, DP pro studenty a supervisory KIV,
- **Publikace** - zobrazování publikací členů KIV, import dat z OBD přes webové služby,
- **Předměty** - zobrazování předmětů vyučovaných členy KIV,

- **Produkty** – modul pro správu výzkumné činnosti katedry a produktů vytvořených v rámci výzkumných aktivit,
- **Osoby** – modul pro správu členů katedry a jejich činností,
- **Login** – modul pro přihlašování, integrace na Kerberos,
- **Aktuality** - správa aktualit pro web KIV.

2.6 Projekty, IS-STAG a OBD

Součástí zadání je i prozkoumání IS-STAG a OBD, kvůli možnosti získávání informací o projektech z těchto systémů. Informační systém STAG se projekty (granty) vůbec nezaobírá. Je to je informační systém pro evidenci studijní agendy vysoké školy nebo univerzity. Systém OBD slouží pouze pro správu publikační činnosti. V tomto systému je pouze číselník grantů, ze kterého se přiřazují hodnoty (čísla grantů) k určitým publikacím. Takže ani z jednoho z uvedených systémů nelze získávat informace o projektech. Při prozkoumávání informačního systému STAG se ale zjistilo, že existuje aplikace "WebServices Web Access" - Webovský přístup ke službám **[6]**, která poskytuje seznam všech aktuálně dostupných služeb v systému STAG. V seznamu aktuálních služeb jsou i služby, které získávají data ze systému INIS (Integrovaný informační systém) a v tomto systému se projekty evidují (evidence všech projektů na univerzitě). Data lze získávat přes dva standardy a to SOAP a REST.

2.7 Webové služby

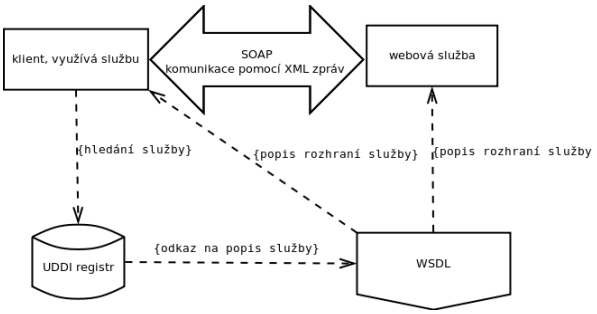
Je to část bussines logiky, která je přístupná přes klasické internetové protokoly a ke komunikaci používá formát XML. Díky těmto skutečnostem je webová služba nezávislá na programovacím jazyce a platformě. Jsou podporované Javou, .NET a teoreticky jakýmkoliv jiným programovacím jazykem. Webová služba nemá nic společného s HTML a webovým prohlížečem.

Je to standardizovaný formát, na kterém se shodli všichni. Dříve byla spousta různých způsobů – CORBA (Common Object Request Broker Architecture), DCOM (Distributed Component Object Model) , Java RMI (Java Remote Method Invocation). Ty ovšem byly nejednotné. K webovým službám existuje WSDL (Web Services Description Language), což je XML popis webové služby. WSDL může být statický

nebo generovaný automaticky samotnou službou. Obsahuje definice:

- **datových typů**,
- **typů zpráv**,
- **typů portů** (rozhraní) a jejich **operací**,
- **vazeb** – napojení portů na transportní protokol,
- **služeb** – konkrétní URL + port s vazbou.

Základními prvky architektury jsou:



Ilustrace 2.1: Architektura SOAP

- **klient** – využívá službu,
- **poskytovatel** – poskytuje službu,
- **registry** – uchovávají informace o umístění služeb, jsou nepovinné, používají se jen zřídka.

2.7.1 REST

REST (Representational State Transfer) je na rozdíl od SOAP (Simple Object Access Protocol) orientován datově, nikoliv procedurálně. REST určuje, jak se přistupuje ke zdrojům. Zdrojem mohou být libovolná data, ale i stavy aplikace, pokud se dají popsat konkrétními daty. Každý zdroj má vlastní identifikátor URI. REST definuje čtyři základní metody pro přístup (Create, Read, Update, Delete). Pojem REST představil v roce 2000 Roy Fielding, což je jeden ze spoluautorů protokolu HTTP a proto jsou tyto čtyři metody implementovány odpovídajícími metodami z HTTP. Jsou to metody

- GET – získání zdroje,
- POST – vytvoření dat,
- DELETE – mazání - tuto možnost většinou není možno volat a proto se například používá metoda POST s nějakým parametrem, který určuje,

že se má mazat,

- PUT – metoda pro update - volání této metody mívá stejné omezení jako metoda DELETE.

2.7.2 SOAP

SOAP (Simple Object Access Protocol) **[7]** je univerzální a na platformě nezávislý způsob přístupu k metodám a službám vzdálených aplikací. Ostatní standardy jako WSDL (Web Services Description Language) a UDDI (Universal Description, Discovery and Integration) vznikly až později, a jen rozšiřují jeho možnosti a snadnost použití. SOAP umožňuje zaslání XML zprávy mezi dvěma aplikacemi. Zpráva je jednosměrný přenos informace od odesílatele k příjemci. Jedna aplikace pošle v XML zprávě požadavek druhé aplikaci, ta požadavek obslouží a výsledek zašle jako druhou zprávu zpět. V tomto případě bývá webová služba vyvolána webovým serverem, který čeká na požadavky klientů a v okamžiku, kdy přes přenosový protokol, např. HTTP, přijde SOAP zpráva, spustí webovou službu a předá jí požadavek. Zpráva v SOAPu je jednoduchý XML dokument, který má kořenový element `Envelope`. V této obálce jsou pak uzavřeny dva elementy `Header` (hlavička) a `Body` (tělo). Všechny tyto tři elementy patří do stejného jmenného prostoru `http://schemas.xmlsoap.org/soap/envelope/`. Element `envelope` identifikuje XML jako SOAP zprávu a musí být kořenovým elementem. Element header umožňuje libovolným způsobem rozšířit protokol SOAP. Element body obsahuje vlastní zprávu. Kostra SOAP zprávy tedy vypadá přibližně takto:

```
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'
soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
  <soap:Header>
    <!-- rozšiřující elementy -->
  </soap:Header>
  <soap:Body>
    <!-- vlastní zpráva -->
  </soap:Body>
</soap:Envelope>
```

Element body – při požadavku je obsahem žádost o provedení nějaké služby, při odpovědi je obsahem výsledek předchozího požadavku nebo chyba kódovaná v elementu `<soap:Fault>` **[8]**.

Jmenný prostor – `http://schemas.xmlsoap.org/soap/encoding/`

identifikuje způsob kódování přenášených dat do XML. Se SOAP můžeme použít libovolný způsob serializace (převod datových typů/objektů na XML schéma). V praxi se nejčastěji používají běžné skalární datové typy (jakými jsou čísla, řetězce a podobně), které definují XML schémata. Navíc SOAP kódování definuje způsob serializace složených datových typů – seznamů skalárních hodnot (polí) a struktur. Lze serializovat i reference na objekty.

2.7.3 WSDL

WSDL (Web Service Definition Language) je jazyk pro popis webové služby. WSDL soubor s definicí rozhraní služby je XML dokument. Skládá se zejména z následujících elementů, které tvoří základní části každého WSDL popisu.

- **Types** - obsahuje definici datových struktur používaných ve zprávách. K definici lze použít teoreticky libovolný typový systém, ale nejčastěji se používají XML schémata. Nástroje pro webové služby se starají o mapování datových typů podle XML schémat na nativní datové typy použitého jazyka.
- **Message** - definuje formát předávaných zpráv pomocí dříve definovaných datových typů. Zprávy fungují jako vstupní nebo výstupní struktury pro operace. Každá zpráva se může skládat z několika logických částí s vlastním datovým typem. Při použití SOAP pro RPC (Remote procedure call) odpovídá jedna část zprávy jednomu parametru vzdálené metody.
- **Operation** - abstraktní definice operací, které jsou službou podporovány. U operace se definuje jaké má vstupy a výstupy. Vstup a výstup je popsán již existující zprávou (message). V SOAP RPC modelu odpovídá operace metodě.
- **PortType** - sdružuje dohromady několik operací.
- **Binding** - slouží pro navázání určitého typu portu (portType) na konkrétní protokol a formát přenosu zpráv.
- **Port** - jeden koncový bod služby definovaný jako kombinace síťové adresy a dříve definované vazby (binding).
- **Service** - sdružuje několik koncových bodů (portů) do jedné služby.

2.7.4 UDDI

UDDI (Universal Description, Discovery and Integration) nabízí mechanismy pro registrování, kategorizování a vyhledávání webových služeb. UDDI funguje jako velký adresář, který obsahuje informace o subjektech (firmách) a jimi poskytovaných službách. Samotný registr pracuje rovněž jako webová služba a komunikace s ní tedy opět probíhá pomocí SOAP.

2.7.5 Java a webové služby

V Javě je výborná podpora obou zmiňovaných formátů webových služeb. Pro REST máme specifikaci RESTful webových služeb „JSR-000311 JAX-RS: The Java API for RESTful Web Services“[5] a pro SOAP „JSR 224: JavaTM API for XML-Based Web Services (JAX-WS) 2.0“[6] a samozřejmě existuje i dostatečné množství implementací. Čistě pro REST například Jersey[7], pro SOAP Metro[8] či Apache Axis, obě webové služby pak podporuje implementace Apache Axis2 nebo Apache CXF. Pro modul Projekty je použit Apache CXF a SOAP a to především kvůli dobrým zkušenostem s touto kombinací.

5 http://jcp.org/en/jsr/detail?id=311
6 http://jcp.org/en/jsr/detail?id=224
7 https://jersey.dev.java.net/
8 https://metro.dev.java.net/

3 Realizační část

3.1 Návrh OpenCms rozhraní pro projekty

Pro práci s projekty (granty) je zapotřebí vytvořit obrazovky (screeny) pro

- seznam typů výzkumných skupin,
- seznam výzkumných skupin,
- seznam typů grantů,
- seznam grantů.

Samozřejmě jsou zapotřebí, pro všechny zmíněné objekty, také obrazovky pro vytvoření a editaci.

Dále obrazovka s přehledem projektů získaných z webových služeb z INIS, detail jednotlivých projektů a jejich import.

Pro přiřazování osob do výzkumné skupiny byl první návrh na samostatnou obrazovku, tedy že až po vytvoření skupiny by se přiřazovaly osoby. Druhá varianta byla použít javaScript pro dynamické přidávaní dalších osob přímo na obrazovce vytváření skupiny. Nakonec jsme se rozhodli pro druhé řešení, které je sice náročnější na vytvoření stránky a ukládání, ale z uživatelského hlediska je tento způsob podstatně příjemnější. Stejné je to i s přidáváním osob k projektu.

Vzhledem k tomu, že již vytvořené schéma (tabulky určené pro granty) pro web KIVu strukturou zcela neodpovídá struktuře získané z INIS, import nemůže být úplně automatický. Problémem je například přiřazení osob pracujících na grantu. Momentálně není jednoznačný identifikátor osoby společný jak datům z INIS tak datům v databázi webu KIV. Navíc osoba pracující na grantu v datech z INIS může být i člověk, který na KIVu není veden. Také poskytovatel projektu nemá stejnou strukturu. V databázi webu kiv je pro tuto infomaci vyhražen jeden textový sloupec, v datech z INIS jsou však informace o poskytovateli hned v několika polích (jméno, adresa, webové stránky, telefon atd.)

3.2 Modul Projekty

Modul Projekty je závislý na modulu Common, který obsahuje knihovnu `webkiv-db`. V této knihovně došlo k velkému rozšíření – použití frameworku Hibernate. Tato část je popsána v bakalářské práci kolegy Aleše Pivničky **[9]**.

Jádrem celého modulu Projekty je knihovna `webkiv-granty`. Ta obsahuje java třídy pro obsluhu požadavků z JSP stránek**[10]**. Jednotlivé balíčky a struktura modulu je popsána v následujících kapitolách. Schéma struktury modulu(včetně závislosti na knihovně `webkiv-db`) lze nalézt v práci kolegy Aleše Pivničky. Logování v celé knihovně je realizováno pomocí frameworku Log4J, který umožňuje logovat zprávy na různých úrovních a s pohodlně definovatelným výstupem.

3.3 DTO (Data Transfer Object)

Balíček `cz.zcu.kiv.granty.dto` obsahuje DTO objekty, které slouží pro přenost dat na stránky (prezenční vrstvu aplikace). Každý objekt implementuje rozhraní `Serializable` a odpovídají mapovaným objektům z `webkiv-db` (DTO odstiňují prezenční vrstvu od business a datové vrstvy. Zajišťují tak větší modulárnost a bezpečnost řešení. V použitém řešení tyto objekty kopírují mapované objekty z `webkiv-db` až na Hibernate anotace určující ORM mapování). Používat mapované objekty přímo by bylo také možné, ale někdy stačí z mapované entity jen několik atributů a bylo by zbytečné předávat vyšší vrstvě aplikace celé entity. Podstatnější ale je, že pokud dojde ke změně datového zdroje, stačí pouze upravit převod ze zdrojových entit na tyto DTO a není tím pádem potřeba přepisovat polovinu kódu.

Momentálně je implementován jen jeden datový zdroj a to databáze. Navenek je datový zdroj prezentován třídou manager, který je popsán dále. O převod entit se stará tzv. mapper.

3.4 Maper

V balíčku `cz.zcu.kiv.granty` se nachází třída mapper, která obsahuje statické metody pro převod mapovaných entit(objekty z databáze) na DTO a obráceně. Při převodu z mapovaných entit na DTO zároveň řeší lokalizaci (u objektů, které mají v databázi vícejazyčné údaje).

3.5 Manager

Manager je třída, která poskytuje operace nad datovým zdrojem. V balíčku `cz.zcu.kiv.granty` se nachází rozhraní `GrantyManager` a v něm jsou definovány metody, které každá implementace musí obsahovat. Obecně jsou to metody pro ukládání (save nebo update), mazání záznamů, získávání všech záznamů, získávaní záznamů podle určitého atributu a kontrola záznamů (unikátnosti hodnot) oproti stávajícím záznamům.

Modul Projekty používá pouze jednu implementaci a to `GrantyManagerImpl`, která má jako datový zdroj relační databázi. Má zaregistrovány Data access objekty (DAO) z webkiv-db pomocí IoC kontejneru z frameworku Spring a to podle návrhového vzoru (Dependency Injection pattern) setter injection. DAO objekty již fyzicky doopravdy provádějí operace nad databází (jsou v nich metody pro práci s databází). Manager volá statické metody z mapperu pro převod entit, nastavuje atributy, které nejsou přímo viditelné uživateli (například `zaznam_editor` u nečíselníkových záznamů) a využívá funcionality z Data access objektů. Co se týče ověřování unikátnosti hodnot, je tato funkcionalita zabudována přímo v DAO objektech a manager tedy pouze volá poskytnuté metody.

Inversion of Control, dependency injection

Inversion of Control (IoC) je malinko jiný pohled na programování. Aplikace se nepíše jako čistý program, ale jako sada komponent. Tyto komponenty se poté konkrétně spojí pomocí kontejneru (skript popisující, jak se mají komponenty složit dohromady). Každá komponenta má svojí úlohu a pokud nám její funkcionalita přestane vyhovovat, lze ji jednoduše nahradit za jinou implementaci. Aby bylo možné komponenty takto měnit, musí každá komponenta splňovat určité náležitosti – implementovat rozhraní (pouze pro tento návrhový vzor, jinak nemusí).

Všechny komponenty mezi sebou skrze tato rozhraní komunikují, posílají si data, využívají jednotlivé služby a samy o sobě nevědí, kterou konkrétní komponentu (implementaci) ale za daným rozhraním hledat. O to se stará IoC kontejner.

IoC kontejner je jakýmsi mozkem aplikace. Je jediný který ví, jaké komponenty se aktuálně nachází za daným rozhraním. Při startu aplikace do kontejneru zaregistrujeme všechny komponenty a on je v případě potřeby vrací (korektně

nainstancované). Samozřejmě pokud nějaká komponenta chybí (není zaregistrována), dochází k chybě.

Kontejner se tak stará o dependency injection. To ale není jediná služba, kterou kontejner může poskytovat. Může i řídit životnost komponent, například místo ruční obsluhy singletonu stačí komponentu definovat jako singleton kontejneru a ten se postará o to, že je komponenta vytvořena pouze v jedné instanci.

Spring framework umožňuje registraci buď anotací `@Repository(klicek)` nebo přes záznam v konfiguračním XML souboru a to přes

```
<bean id="klicek" class="package.impl.class"autowire="byType"/>
```

Pokud daný objekt někde potřebujeme získávat ručně, je přístupný přes hodnotu klíček. Hodnota `byType` u atributu `autowire` znamená, že kontejner projde všechny `set` metody a podle typů pozná, jaké závislé beany má vložit. Pokud je registrace vytvářena anotací, přidá se k daným `set` metodám anotace `@Autowired`.

3.6 Mediator

Mediátory jsou třídy obsluhující dotazy z JSP stránek a jsou v balíčku `cz.zcu.kiv.granty.mediator`. Předkem všech mediátorů je třída `MainMediator`, která rozšiřuje třídu z OpenCms `CmsJspActionElement`. `MainMediator` si v sobě uchovává instanci aplikačního kontextu, kontextu stránky, instanci managera, vektory chybových a informačních zpráv, použité locale a ID aktuálně přihlášeného uživatele. Obecně tedy hodnoty potřebné na každé stránce (mediátoru). Všechny tyto hodnoty se nastavují v konstruktoru. V konstruktoru každého odděděného mediátoru se tedy volá konstruktor tohoto předka. Mediátory tímto způsobem mají přístup k funkcionalitě managera a dle potřeby získávají data pro zobrazení, nebo volají metody pro ukládání a podobně. Dále mediátory využívají služeb z validátorů, které jsou popsány v následující části. Mediator získává od validátoru DTO objekty vytvořené z parametrů requestu a dále pak vektor chybových zpráv. Pokud se tedy řádně nepovede vytvořit DTO objekt, mediátor vrátí zpět na JSP stránku korektně vyplněné hodnoty a vektor chybových zpráv (v jakém poli byla chyba a jaká).

3.7 Validator

Validátory se nachází v balíčku `cz.zcu.kiv.granty.validator`. Společným předkem pro všechny validátory je abstraktní třída `MainValidator`. Asi nejdůležitější metodou v této abstraktní třídě je metoda `validateAgainstOM()`, která umožňuje validaci libovolných atributů z mapovaných entit podle restrikcí uvedených v anotacích pro mapování. Podrobnější popis lze nalézt v již zmiňované bakalářské práci kolegy Aleše Pivničky.

Jedinou třídou, která je prozatím oddděděna od `MainValidator` je třída `GrantyValidator`. Té se v konstruktoru předává objekt typu `HttpServletRequest`, locale requestu a manager. Poté poskytuje metody, které z requestu získají poslané parametry, tyto parametry zvaliduje a pokusí se vytvořit daný DTO objekt. Pokud při těchto operacích dojde k chybě, lze z validátoru získat vektor chyb s lokalizovanými popisy chyb (což je využíváno mediátory).

3.8 Import dat

Podpora získávání dat ze systému INIS se nachází v balíčku `cz.zcu.kiv.granty.ws`. Obsah tohoto balíčku je automaticky vygenerován z WSDL popisu webové služby konzolovým programem `wsdl2java`, který je součástí Apache CXF. Podle WSDL popisu vygeneruje veškeré datové třídy (třídy se suffixem `Type`) a třídy pro volání jednotlivých služeb. Pro vytvoření takto jednoduchého klienta je tento způsob plně dostačující a plně funkční.

V mediátorech určených pro zobrazovaní dat z INISu stačí pouze vytvořit objekt služby, z něho získat port a nad ním už jen volat jednotlivé metody, které vrací objekty nebo kolekce objektů.

Při samotném importu určitého projektu máme konkrétní jeden objekt, který se převede na DTO grantu, respektive se převádějí odpovídající si hodnoty, např. číslo projektu. Poskytovatel projektu získaný z webové služby má kromě názvu i další doplňující informace, jakou je například adresa, webové stránky apod. Vzhledem k tomu, že ve schématu databáze webu KIV je pro poskytovatele vyčleněno jen jedno pole, převádí se data z INIS v jeden textový řetězec. Další poměrně nepříjemnou záležitostí se ukázal převod osob pracujících na grantu. Z webové služby se získává

atribut `ucitIdNo`, což je ID osoby ze systému Orion. Takový záznam ovšem v databázi KIV není. Z tohoto důvodu se tedy osoby ke grantu nepřiřazují automaticky, ale vypisují se pouze informativně na stránce pro uložení importu, a osoba která projekt importuje může podle této „nápovědy“ přiřadit pracovníky KIVu k danému projektu. Mezi osobami pracujícími na projektu může být i externí osoba. V takovémto případě je vhodné dopsat tyto osoby do doplňujících informací o grantech.

3.9 JSP stránky

Na začátku každé stránky je direktiva určující `content type` a kódování stránky, importy používaných tříd a definice knihoven tagů. Dále pak získání aplikačního kontextu, vytvoření odpovídajíciho mediátoru a v případě potřeby volání metod z mediátoru.

Na stránkách jsou použity standardní JSTL tagy pro lokalizace(prefix fmt), tagy pro podmínky, cykly (prefix c), tag link z knihovny OpenCms a několik nově vytvořených vlastních tagů.

Funkce přídávání, editování a mazání jsou zpřístupňovány na základě oprávnění právě přihlášeného uživatele.

3.10 Vlastní tagy

Na nově vytvořených stránkách se často vyskytují stále stejné části kódu, jako například výpis chybových/informačních zpráv, zobrazování obrázků pro logické hodnoty a především vytváření rozbalovacích seznamů. Z tohoto důvodu vznikla nová vlastní knihovna tagů pro zpřehlednění a zjednodušení samotných stránek.

Vznikl tedy soubor granty.tld, což je XML soubor pro popis jednotlivých tagů. Jako příklad bych uvedl tag pro zobrazení obrázku na základě logické hodnoty. Popis v *tld* souboru je následující:

```
<tag>
   <description>Description</description>
   <name>yesno</name>
   <tag-class>cz.zcu.kiv.jsp.KivJspTagYesNo</tag-class>
   <body-content>empty</body-content>
   <attribute>
       <name>value</name>
       <required>true</required>
   </attribute>
</tag>
```

XML značky jsou v podstatě samodokumentující. Zde jsme definovali tag `yesno` pro vytvoření HTML tagu `img` s obrázkem odpovídajícím logické hodnotě. Značka `body-content` říká, že tak je nepárový, tedy bez obsahu.

Java třída `KivJspTagYesNo` je odděděna od `SimpleTagSupport` z balíčku `javax.servlet`. Tato třída má jeden atribut, který musí odpovídat atributu uvedenému v tld souboru, a to atributu `value`. Dále je zapotřebí implementovat metodu `doTag()`. V ní se získává instance contextu stránky (obsahuje předek) a z této instance `writer` (`JspWriter`), do kterého již lze zapisovat samotný HTML výstup tohoto tagu. Samotná logika tagu je pouze zjištění, jaké logické hodnoty nabývá atribut `value` a podle toho se na výstup zapíše HTML kód pro obrázek křížku (hodnota false) nebo obrázek fajfky (pro hodnotu true). Tím vznikl jednoduchý tag, díky kterému ale není potřeba stále testovat nějakou logickou hodnotu přímo na stránce. Stačí jej jednoduše vyvolat

```
<kiv:yesno value="${boolean_hodnota}" />
```

a získáme odpovídající fragment HTML kódu.

Jako další vznikly tagy pro výpis chybových a informačních zpráv. Opět část kódu, která se opakuje na každé stránce. Místo stávajícího testování existence atributu `errMsg` v kontextu stránky a jeho případného vypsání stačí na stránku uvést `<kiv:errors />`. Stejné je to i pro výpis informačních zpráv.

Asi nejzajímavějším vlastním tagem je tag pro vytváření rozbalovacího seznamu(výběru) - tag `select`. Tento tag má následující atributy:

- **name** – text, povinný, určuje HTML atribut name u HTML tagu select,
- **id** – text, nepovinný, určuje HTML atribut id u HTML tagu select,
- **size** – číslo, nepovinný, implicitně hodnota 1, hodnota pro HTML atribut size tagu select,
- **multiple** – logická hodnota, nepovinný, implicitně hodnota false, hodnota pro HTML atribut multiple tagu select,
- **items** – seznam libovolných objektů, povinné, kolekce objektů, z kterých

se skládají jednotlivé položky seznamu,

- **value** – libovolný objekt, nepovinné, říká, jaký objekt má být v seznamu vybrán. Položky v `items` a tato hodnota musejí být stejného datového typu,
- **optionKey** – text, podle kterého se získává atribut z objektů v `items` a objektu určeného v atributu `value` (pokud je objekt typu Osoba a osoba má atributy `name` a `id`, uvede se jako tato hodnota `name`, protože tu chceme zobrazovat v možnostech seznamu),
- **optionValue** – text, stejné jako atribut `optionKey` pouze s tím rozdílem, že se tato hodnota používá jako atribut value u HTML tagu option (v případě objektu Osoba by zde bylo uveden text `id`),
- **noSelectionKey** – text, nepovinné, hodnota která se zobrazí v seznamu pokud není vybrána žádná hodnota,
- **noSelectionValue** – text, nepovinné, obsah tohoto atributu se použije jako hodnota `value` u HTML tagu option, který odpovídá záznamu z `noSelectionKey`.

Opět tedy stačí na stránce, kde má být HTML tag pro výběr, místo složitého testování a psaní cyklů zavolat jeden tag,

```
<kiv:select name="nazev" id="nazev" items="${osoby}" value="${osoba}"
optionKey="name" optionValue="id" noSelectionKey="Vyberte ..." />
```

který celou problematiku vyřeší.

3.11 Sestavení knihoven

Pro správnou funkčnost modulu je zapotřebí vytvořit jar soubor jádra modulu (`webkiv-granty`) a dále mít k dispozici modul Common, který obsahuje třídu pro zjišťování práv právě přihlášeného uživatele a především obsahuje knihovnu `webkiv-db`, která zajišťuje operace nad databází. Již při práci na Projektu 5 jsme byli nemile překvapeni nad složitostí skládání všech jednotlivých částí. Z tohoto důvodu byl při práci na Projektu 5 vytvořen funkční popis projektu `webkiv-db` pro Maven2. Maven2 je build manager pro Java projekty založený na konceptu project object model

(POM). Základem je XML soubor popisující strukturu projektu, závislost projektu na určitých knihovnách apod. Popis je uložen v kořenovém adresáři projektu a je to soubor pom.xml. Pro práci s projektem tedy stačí mít nainstalovaný Maven2 a pro vytvoření projektu do Eclipse pouze spustit z příkazového řádku příkaz `mvn eclipse:eclipse`. Pro složení výsledného jar souboru pak `mvn clean compile jar:jar`. Tento příkaz smaže přeložené třídy, znovu je přeloží a sestaví jar soubor. Samozřejmě přibalí i potřebné knihovny, které jsou uvedeny v popisu projektu (přiloží i další, protože zná i knihovny knihoven, a tím zajistí, že žádná nechybí). Tyto knihovny se automaticky stahují z repositářů volně přístupných na Internetu a instalují se do lokálního repositáře. Ideálně by se měl výsledný jar soubor instalovat do repositáře (Maven2 repositáře jsou určeny pro přeložené knihovny, kdežto CVS/SVN jsou repositáře pro zdrojové kódy). Zde by bylo ovšem zapotřebí změnit způsob verzování knihoven které se momentálně používá. Tento stávající způsob totiž neodpovídá standardním zvyklostem. Maven totiž předpokládá formát `název-majoritní verze.minoritní verze.jar` kdežto systém používaný na katedře je `název.verze.jar` **[11][12]**. Stejný postup je i pro práci s jádrem modulu, tedy s projektem `webkiv-granty`. Výsledný modul Granty se poté složí přímo v prostředí OpenCms, kde se také provede jeho export do souboru zip, který je určen k distibuci.

4 Závěr

Cílem této práce bylo vytvořit webové rozhraní pro správu projektů a spojením s prací kolegy Aleše Pivničky vytvořit funkční modul do redakčního systému OpenCms, což se podařilo splnit. Po dohodě s vedoucím této práce bylo nasazení na ostrou verzi webu odloženo na pozdější dobu vzhledem k tomu, že jsou v současné době nasazovány moduly jiné a dále pak kvůli změnám v konfiguraci samotného webu, které vznikly použitím frameworku Hibernate v knihovně webkiv-db. Ač jde v podstatě o neinvazivní změny, bylo nutné v konfiguraci upravit definici datového zdroje. Vypracované řešení tedy bylo testováno lokálně a je plně funkční. Jako největší přínosy vidím použití frameworku Hibernate pro objektově relační mapování (část od kolegy Aleše Pivničky) a změny ve struktuře modulu.

Základní myšlenka stávajícího modulu zůstala zachována, ale atributy a části kódu, které jsou společné pro všechny mediátory (tzn. i moduly), byly přesunuty do předků. Vznikl univerzální validátor a několik nových JSP tagů. Díky těmto úpravám je vývoj modulu rychlejší, snazší a výsledný kód je přehlednější a tudíž snazší na údržbu. Aby všechny tyto úpravy mohly být použity i v ostatních modulech, bylo by vhodné je přesunout do společného modulu Common.

Vytvoření modulu do OpenCms byla poměrně zajímavá činnost, ale nabyl jsem dojmu, že tento způsob vytváření vlastního webu není příliš šťastný. OpenCms je sice výborný redakční systém, ale vývoj vlastních modulů je těžkopádný a zdlouhavý. Jako lepší řešení se mi zdá použít modernější technologie (například framework Grails) a na nich si postavit vlastní redakční systém, který bude také umožňovat přidávání vlastních modulů nebo pluginů. Čas strávený na vývoji modulů by se tak podstatně zkrátil.

Dále bych rád upozornil na několik nedostatků v databázi. Problémem se zde ukázaly složené primární klíče (složené z cizích klíčů), s kterými framework ve stávající verzi nepočítá. I to se ale povedlo vyřešit bez zásahu do schématu. Poněkud zvláštní jsou i některé nadefinované kaskády (smazání typu výzkumné skupiny smaže všechny skupiny s tímto typem) a dále pak `null` hodnoty v `boolean` sloupcích nebo sloupec pro `boolean` hodnotu nadefinovaný jako `tinyint`. Bylo by vhodné řádně prostudovat schéma a tyto drobné nedostatky upravit.

Po výsledném spojení mé práce s prací kolegy Aleše Pivničky jsem ale s výsledkem spokojen a těším se na jeho reálné nasazování.

Seznam použitých zkratek

- **CORBA** - Common Object Request Broker Architecture, standard pro tvorbu distribuovaných objektově orientovaných aplikací
- **CRUD** - Create, Read, Update, Delete
- **DAO** – Data Access Object
- **DCOM** - Distributed Component Object Model, protokol definující interakci mezi komponentami a jejich klienty
- **DTO** – Data Transfer Object
- **GSP** - Groovy Server Pages
- **IoC** - Inversion of Control, návrhový vzor
- **JSP** - Java Server Pages
- **JSTL** - označení knihovny standardních JSP tagů
- **KIV** – Katedra informatiky a výpočetní techniky
- **LGPL** - Lesser General Public License, licence svobodného softwaru, publikovaná Free Software Foundation
- **POJO** – Plain Old Java Object
- **REST** - Representational State Transfer, architektura rozhraní navržená pro distribuované prostředí
- **RMI** - Remote Method Invocation, volání vzdálených objektů
- **SOAP** – Simple Access Object Protocol, protokol pro posílání zpráv XML, je základem webových služeb
- **SQL** - Structured Query Language, standardizovaný dotazovací jazyk používaný v relačních databázích
- **UDDI** - Universal Description, Discovery and Integration, mechanismus umožňující registraci a vyhledávání webových služeb
- **URL** - Uniform Resource Locator („jednotný lokátor zdrojů“)
- **WSDL** - Web Services Description Language, standard pro popis webové služny
- **XML** - eXtensible Markup Language, značkovací jazyk
- **XSLT** - eXtensible Stylesheet Language Transformations, slouží pro převod dat z XML formátu na libovolný jiný formát

Seznam použité literatury

[1] BOLLNGER Gary, NATARAJAN Bharathi. *JSP Java Server Pages*. GRADA, 2003. ISBN 80-247-0340-8

[2] GUELICH Scott, GUNDAVARAM Shishir, BIRZNIEKS Gunther. *CGI Programming with Perl*, O'Reilly Media, 2000. 480s.

[3] EXNER, Miroslav. *OpenCms: Příručka prvních kroků*. Plzeň, 2009. 40s.

[4] BUTCHER, Matt. *Managing and Customizing OpenCms 6 Websites: A complete guide to set up, configuration and administration*. Packt Publishing, 2006. 256s. ISBN 1904811760

[5] PŘEMEK Brada a kolektiv. *Dokumentace a návody k vývoji KIV modulů do OpenCms*. <http://wiki.kiv.zcu.cz/OpenCMS/> (adresa platná k 1.5.2012)

[6] VALENTA Lukáš. *Webovský přístup ke službám*. <https://stag-ws.zcu.cz> (adresa platná k 1.5.2010)

[7] Soap version 1.2, W3C Recommendation 27 April 2007[online]. <http://www.w3.org/TR/soap12-part1>

[8] SKONNARD Aaron, GUDGIN Martin. *XML pohotová referenční příručka*. GRADA, 16.12.2005. 344 stran. ISBN 80-247-0972-4

[9] PIVNIČKA ALEŠ. BP – Návrh a implementace modulu Projekty.ZČU, Plzeň, 2010.

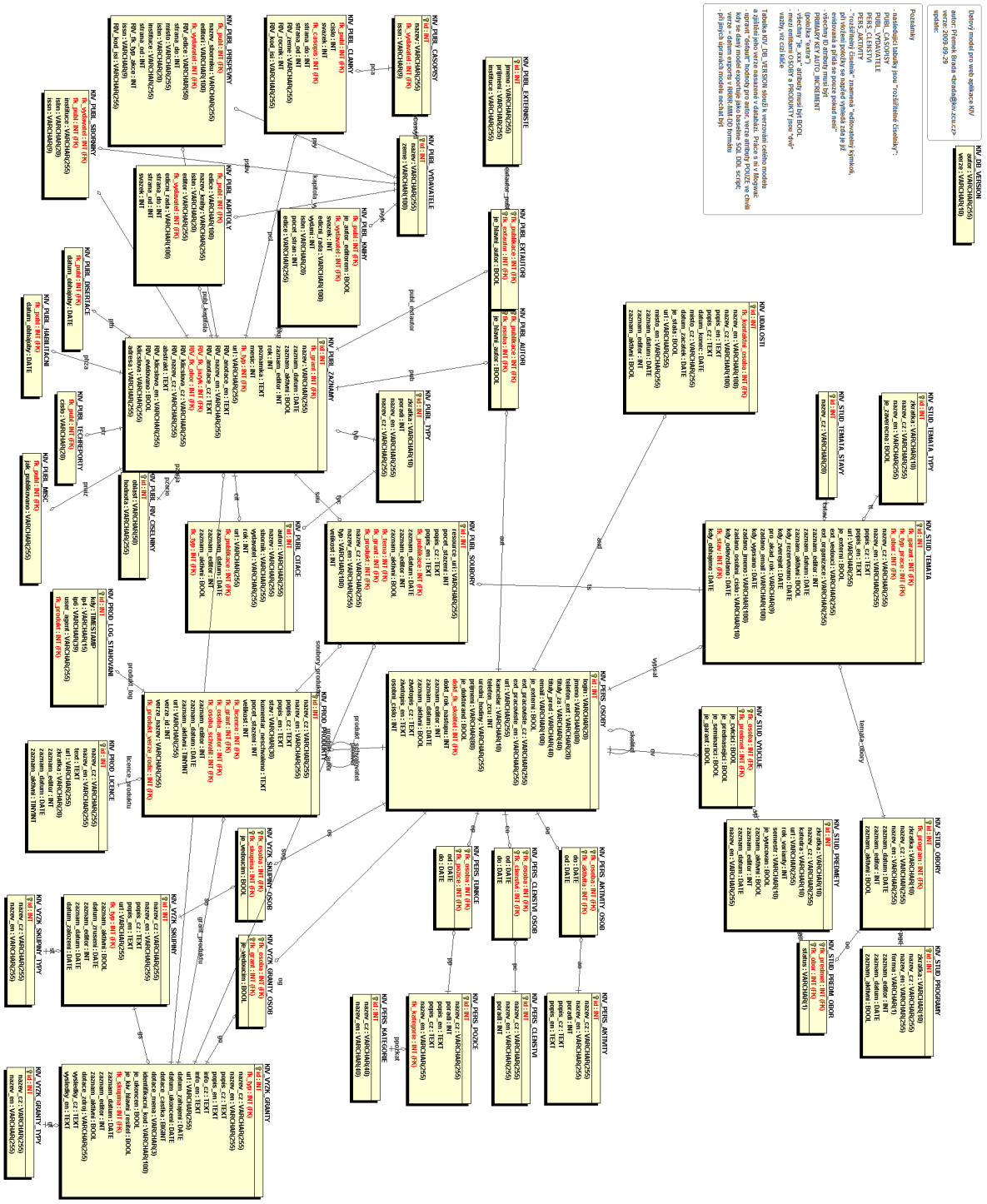
[10] LILIEDAHL, Dan. *OpenCms 7: Extending and customizing OpenCms through its Java API*. Packt Publishing, 2008. 292s. ISBN 1847191053

[11] The Apache Software Foundation. *Maven – Maven Getting Started Guide*. 2012. <http://maven.apache.org/guides/getting-started/index.html>

[12] The Apache Software Foundation. *Maven – POM reference*. 2012. <http://maven.apache.org/pom.html>

Příloha 1A

kompletní schéma DB KIV (převzato z http://wiki.kiv.zcu.cz)



Příloha 1B

Tabulky z DB KIV potřebné pro modul projekty (převzato z http://wiki.kiv.zcu.cz)