MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY



Správa obsahu www.muni.cz

BAKALÁŘSKÁ PRÁCE

Jana Galajdová

Brno, jaro 2008

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracovala samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používala nebo z nich čerpala, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: Mgr. Jaromír Ocelka

Poděkování

Chtěla bych poděkovat především vedoucímu této práce, Mgr. Jaromíru Ocelkovi, za jeho cenné rady a ochotu při řešení problémů, které se během práve objevily. Velký dík patří rovněž kolegům z ÚVT, kteří mi v případě potřeby vždy podali pomocnou ruku. V neposlední řadě chci poděkovat rodině a známým za podporu a zázemí.

Shrnutí

Náplní práce je zpřístupnit administrativním pracovníkum správu dokumentů, textů a dat celouniverzitní prezentace www.muni.cz. Webová prezentace obsahuje velké množství dat přebíraných z jiných databází (personalistika, studijní systém, ...) avšak prezentuje také dokumenty, texty a data speciálně pořízené pro prezentaci na www.muni.cz. Součástí práce je nejen výběr vhodných technologií a aplikací, ale i jejich propojení do vhodně fungujícího celku.

Klíčová slova

Systém pro správu obsahu, WEBDAV, SERVLET, XML, XML SCHEMA, CSS styly

Obsah

1 **Úvod** . . . 1
1.1 *Analýza stávající správy dat* . . . 1
1.1.1 Počátky a vývoj www.muni.cz . . . 1
1.1.2 Princip budování aktuální verze prezentace . . . 2
1.1.2.1 Číselník dlouhých textů . . . 3
1.1.2.2 Číselník externích odkazů . . . 3
1.1.2.3 Číselník součástí pracovišť . . . 4
1.1.3 Současný způsob správy dat . . . 4
1.1.4 Cíl práce . . . 4
2 **Jazyk XML a související technologie** . . . 6
2.1 *XML – univerzální formát pro zpracování informací* . . . 6
2.1.1 Vzhled dokumentů . . . 7
2.1.1.1 CSS styly . . . 7
2.1.1.2 XSLT transformace . . . 7
2.1.2 Jazyky pro popis struktury XML dokumentů . . . 7
2.1.2.1 DTD . . . 7
2.1.2.2 XML Schéma . . . 8
3 **Technologie zpřístupnění dat** . . . 9
3.1 *Protokoly pro přenos a správu dat* . . . 9
3.1.1 FTP . . . 9
3.1.2 SCP . . . 10
3.1.3 SFTP . . . 10
3.1.4 WebDAV . . . 10
3.2 *Porovnání vlastností protokolů* . . . 11
4 **Způsob zpřístupnění** . . . 13
4.1 *Výběr protokolu* . . . 13
4.2 *Implementace protokolu* . . . 14
5 **Výběr nástroje pro zobrazení vytvořených souborů** . . . 16
5.1 *Webdav klienti* . . . 17
5.1.1 Novell NetDrive . . . 17
5.1.2 WebDrive . . . 17
5.1.3 Další klienti . . . 18
6 **Nástroje na úpravu XML souborů** . . . 19
6.1 *Požadavky na možnosti editoru* . . . 19
6.2 *Vhodné XML editory* . . . 20
6.2.1 XMLmind . . . 20
6.2.2 Oxygen . . . 21
6.2.3 Serna . . . 22
6.2.4 XML Spy . . . 22
6.2.5 Další vhodné XML editory pro úpravu XML dat . . . 22

6.3 *Shrnutí* . . . 22
7 **Implementace systému** . . . 23
7.1 *Datová vrstva* . . . 24
7.1.1 Nová podoba tabulek . . . 24
7.2 *Aplikační vrstva systému* . . . 25
7.2.1 Princip mapování tříd . . . 26
7.2.2 Konfigurační soubor pro popis struktury . . . 26
7.2.3 Rozhraní IModule . . . 27
7.2.4 ModuleText – třída určená pro zpřístupnění textů . . . 28
7.2.4.1 Verzování . . . 29
7.2.5 ModuleList – třída určená k editaci číselníků . . . 30
7.2.5.1 Vytvoření nového virtuálního souboru z číselníku . . . 31
7.2.6 Servlet – metody implementující příkazy protokolu WebDAV . . . 31
7.2.7 Další použité metody a mechanismy . . . 31
7.2.7.1 Určování velikosti souborů . . . 32
7.2.8 Autentizace . . . 32
8 **Závěr** . . . 33
8.1 *Zhodnocení práce* . . . 33
8.2 *Možnosti rozšíření* . . . 34
Literatura . . . 36
Rejstřík . . . 37
A **Ukázka dokumentů vygenerovaných pro editaci číselníku Videa** . . . 38
B **Šablona generování CSS stylů pro jednotlivé číselníky** . . . 40
C **Text v podobě XML dokumentu** . . . 41
D **Obsah přiloženého média** . . . 43

Kapitola 1

Úvod

Internet z pohledu zdroje dat představuje nejrobustnější nástroj, který kdy měl člověk k dispozici. Díky dynamickému vývoji a zvyšujícím se nárokům na pohodlnost a dostupnost dat se během relativně krátké doby stal alfou i omegou při hledání, získávání nebo výměně informací. Být na internetu dohledatelný a zpřístupňovat data považuje většina z nás za samozřejmé. Vznikají osobní, vzdělávací nebo jiné webové prezentace, a to jak za účelem poskytování cenných informací, tak i pro pouhé pobavení bez větší informační hodnoty. Cenu získaných dat primárně udává informační charakter jejich obsahu. Zde obzvlášť platí, že kvalitní obsah zvyšuje užitnou hodnotu celého informačního systému a naopak – pokud uvedeme na stránkách zavádějící nebo zastaralé informace, vytváříme tak iluzi nedůvěryhodného zdroje, což může mít neblahé následky.

Pokud chceme provozovat kvalitní a aktualizovanou internetovou prezentaci, musíme toto všechno vzít v úvahu. V případě, že se jedná o robustní systém obsahující velké množství dat, je nutné zajistit bezproblémovou údržbu. Musíme být schopni poskytnout možnost správy aktuálních informací lidem, kteří jsou za obsah zodpovědní, a uvědomit si, že v drtivé většině rozsáhlejších projektů se nejedná o zaměstnance z řad programátorů nebo technických pracovníků. Podle toho by celý systém správy měl být navržen, aby data mohla být aktualizována nezávisle na použitých programovacích technikách. Vyhneme se tak řadě problémů a zajistíme, aby každý člen týmu mohl vykonávat tu část zodpovědnosti, která mu náleží, a nemusel suplovat práci někoho jiného.

Vývoj webové prezentace Masarykovy univerzity dospěl do fáze, kdy samotná správa dat a textů zobrazovaných na www.muni.cz není triviální. Vzhledem k charakteru informací poskytovaných prostřednictvím prezentace je nutné přiblížit editaci pracovníkům odpovědným za vlastní obsah zobrazovaných informací, tím zefektivnit aktualizaci dat a přenechat práci dříve vykonávanou techniky kompetentním osobám.

1.1 Analýza stávající správy dat

1.1.1 Počátky a vývoj www.muni.cz

Webové stránky Masarykovy univerzity prošly od data svého vzniku nejednou změnou. Na začátku roku 1997 byly v prezentačním systému zveřejneny základní informace o fakultách a celouniverzitních pracovištích zahrnující přehledy zaměstnanců, studentů, studijních oborů, dílčích pracovišť a stručný popis činností[8]. Každému zaměstnanci a studentu byla

vytvořena stránka s kontaktními údaji. Postupně byly do systému dodány vyhledávací mechanismy, telefonní a e-mailové seznamy, výroční zprávy a informace o různých aktivitách univerzity.

Data přibývala, nároky na jejich zpracování se stupňovaly. Spolu s vývojem nových technologií a postupů, případně s potřebou integrace vzhledu s novým vizuálním stylem univerzity tak prezentace dostávala novější tvář. Ruku v ruce s grafickými úpravami probíhala modernizace struktury a obsahu, čímž z původní verze projektu vznikl opravdu robustní webový informační systém [13]. S nárůstem složitosti a objemu dat pak začal být kladen větší důraz i na budoucí celkovou systematickou údržbu.

1.1.2 Princip budování aktuální verze prezentace

Posledních větších úprav se prezentace dočkala v roce 2006 (více informací naleznete ve Zpravodaji ÚVT MU[16]). Tehdy byla spolu s novým grafickým pojetím vytvořena i nová vnitřní architektura. Rovněž byl navržen specializovaný aplikační server, k jehož realizaci byla využita technologie Microsoft .NET. Zde bylo nutné ošetřit skutečnost, že předešlé verze využívaly technologii ASP. Jelikož většina obsahu verze předchozí byla použita i nyní a jediné, co stránky od sebe odlišovalo, bylo výsledné grafické znázornění, nabízelo se jako schůdné řešení ponechat možnost zobrazování stránek vygenerovaných oběma způsoby. Přepisování kódu z jednoho programovacího jazyka do druhého prakticky beze změny se v toto případe jevilo jako neefektivní. A tak byl navržen řídící dispečer, který na požadání o poskytnutí stránky zajišťuje na úrovni aplikačního serveru její zpracování odpovídající technologií. Pro generování vlastního obsahu webové stránky je tak obecně možné použít téměř libovolný programovací jazyk a technologii. Dispečer má i další funkce, stará se například o vykonávání *fragů*, speciálních programových kódů, jež jsou součástí XML dokumentů, avšak mají poněkud odlišný charakter než ostatní značky; starají se například o výpis odkazů v závislosti na zvoleném jazyce a podobně[17].

Pro potřeby správy dat bylo jedním z hlavních přínosů nového webu oddělení zdrojů od programového kódu. Texty již nejsou vpisovány přímo mezi řádky kódu, ale jsou uloženy v databázi. Nad takto uloženými daty lze pak budovat nejrůznější aplikace pro jejich systematickou údržbu, což je pro tak rozsahlý systém, budovaný navíc dvojjazyčně, nezbytné. Kromě samotných textů, jejichž počet s vybudováním nové verze významně vzrostl, se nově v databázi evidují i externí odkazy nebo fráze, vše za účelem lepší dostupnosti a uchopitelnosti dat. V neposlední řadě jsou zde i dokumenty, na které je pouze ze stránek odkazováno, tj. neobsahují vlastní text (doprovodné dokumenty formátu pdf, word, ...). Mimo výše zmíněných dat využívá prezenace v současné době i mnoho dalších informací, které jsou přejímány z ostatních informačních systémů univerzity (Inet, IS MU, personálně-mzdový systém Magion a další). Jak je vidět, potřebných dat je opravdu hodně, navíc jejich vlastnosti a požadavky na uchování se od sebe mohou výrazně lišit. Otázku importů datových podkladů z jiných systémů teď opomeňme, zaměřme se na vlastní texty a jejich možnosti správy, což je jeden z primárních cílů této práce. Zde jsou některé příklady tabulek:

1.1.2.1 Číselník dlouhých textů

Většina statických textů, která je zobrazována na webových stránkách univerzity, je uložena v tabulce *číselník dlouhých textů*. V této tabulce je pro každý text vygenerováno jednoznačné ID a jsou připojeny i další informace důležité pro zpracování textů. Pro uložení vlastního textu je použit formát XML, univerzální prostředek k uchovávání a dalšímu zpracování dat. Jeho hlavní předností je nezávislost na určitém programovém vybavení nebo technologii, kromě toho umožňuje budovat logickou strukturu, díky níž můžeme s daty dále pracovat. K uchování takto strukturovaného textu je použit datový typ *blob*, což je obecný datový typ používaný pro ukládání blíže nespecifikovaných binárních dat v databázi. Používá se v případě, kdy databáze nepotřebuje informace o uložených datech, protože se o jejich interpretaci postará až klientská aplikace, která data přijme.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="W3muXml2Html.xsl"?>
<text>
<h2 class="bold">Snímek ze spisu notářského</h2>
<p>
Přede mnou Janem Stichou, c.k. notářem ve Slavkově, dnes
v mé c.k.notářské kanceláři ve Slavkově ujednali
a uzavřeli:<br/>
a) vysokorodý pán D<sup>or</sup> Václav
hrabě z Kauniců,držitel svěřenského velkostatku přítomně
sídlící ve Slavkově, mně osobně známý,<br/>
b) pan Václav Řehořovský, profesor c.k. české vysoké školy
technické Brně jako místopředseda spolku "Kaunicovy
studentské koleje českých vysokých škol v Brně", jehož
osobní totožnost mi potvrdili svědci mně osobně známí
pánové: František Novák, ústřední ředitel velkostatku
a Emerich Valníček, měšťanosta - obá ve Slavkově,
následující
</p>

<div class="dashed"></div>
<p>
<image id="2806" align="left" zoom="true" href_size="h"/>
<image id="2807" align="left" zoom="true" href_size="h"/>
<image id="2808" align="left" zoom="true" href_size="h"/>
</p>
</text>
```



Obrázek 1.1: Ukázka statického textu uloženého ve formátu XML a následné zobrazení na stránkách prezentace

1.1.2.2 Číselník externích odkazů

Tento číselník vznikl z potřeby kontroly a správy externích odkazů, které byly dříve vpisovány přímo do textu. Pro jejich změnu bylo tehdy nutné projít všechny stránky, odkud bylo na příslušnou stránku odkazováno, což se vzrůstajícím objemem dat začalo být obtížné. Proto jsou nyní všechny externí odkazy umístěny v jedné tabulce a jejich modifikace je mnohem přehlednější. Výpis konktrétního textu z číselníku je realizován fragem.

1.1.2.3 Číselník součástí pracovišť

V této tabulce jsou podrobné informace pro potřeby strukturalizace podle fakult a podle ostatních součástí univerzity. Tyto součásti jsou zobrazovány v levé části stránek a slouží mimo jiné k lepší navigaci a k rychlejšímu pohybu mezi jednotlivými fakultami nebo pracovišti. Narozdíl od předchozích tabulek se data v podstatě nemění a je patrné, že i možnost měnit uvedené údaje přísluší jen povolaným.

1.1.3 Současný způsob správy dat

Tabulek potřebných pro uložení všech dat je samozřejmně mnohem více. Pro pochopení, s jakými typy tabulek musí správci pracovat, nám však postačí výše zmíněné.

Editovat externí odkazy, případně součásti pracovišť přímo v tabulce by z pohledu složitosti nebyl zásadní problém. Podíváme-li se však na číselník dlouhých textů, uložených navíc ve formátu, který nedává moc možností kontroly správnosti dat, je zřejmé, že se bez nástroje sofistikovanejšího způsobu úpravy neobejdeme. A nabízí se další otázka. Komu vlastně náleží právo editovat jednotlivá data, proč by se například statickými texty měl zabývat člověk, který se stará o komplexní správu prezentace a logickou strukturu? Zatímco aktuální podobu textů by tak ideálně měl mít možnost měnit administrativní pracovník zodpovědný za jejich obsah, do dalších tabulek, jako například součástí pracovišť, by měl mít přístup jen člověk spravující prezentaci komplexně.

Podívejme se nejprve na první tabulku. Ta slouží k ukládání textů, které se při načtení stránky zobrazí v její hlavní části. Jedná se tedy o vlastní obsah, který čtenáře při navštívení stránky zajímá. Možnost editovat tyto texty má v současné době jen úzká skupina lidí pracující na vývoji a údržbě celé prezentace, což vzhledem k charakteru obsahu textu není optimální. Samotné texty jsou tvořeny administrativními pracovníky různých pracovišť, takže editací rozumíme v podstatě zkopírování textu nebo jeho úpravu podle předem zaslaného vzoru. A to je neefektivní a pro obě strany zdlouhavé. Na druhou stranu aktuální forma editace vyžaduje minimálně základní znalosti jazyka XML, který je pro uložení textu použit, protože porušení struktury dokumentu nebo nesprávé uložení způsobí celkové nenačtení stránky. Díky tomu není vhodné editovat text přímo v intranetu za použití nějakého HTML browseru, ale je nutné najít propracovanější způsob. V současné době existuje provizorní zjednodušená verze editace pomocí protokou FTP, která se ukázala být dobrým krokem vpřed, především díky možnosti editovat text v nástrojích na to určených. Jako konečné řešení však není úplně optimální.

1.1.4 Cíl práce

Cílem této bakalářské práce je najít vhodnější způsob správy dat webové prezentace. V případě editace statických textů to znamená umožnit editaci pracovníkům, kteří texty vytváří, efektivní formou, jež bude dobře uchopitelná a znemožní porušení struktury celé stránky. Konkrétně to znamená:

- najít způsob, jak texty zpřístupnit. Použití FTP protokolu je sice funkční, avšak prozkoumaním dalších existujících protokolů bychom mohli najít protokol vhodnější;
- zvolený protokol implementovat;
- zajistit uložení textu správným způsobem, tzn. znemožnit uložení textu, který by při načtení stránky způsoboval její výpadek;
- zajistit pohodlnou editaci textu, využít vlastnosti formátu XML, vybrat vhodný XML editor a zobrazit text tak, aby i člověk neznalý syntaxe jazyka XML byl schopný text upravit;
- navrhnout mechanizmus plnohodného náhledu editovaného textu –uceleného zobrazení textu ještě před konečným zveřejněním na stránkách prezentace.

Při pohledu na současný způsob uložení dat prezentace se nabízí ještě další myšlenka. V případě, že vytvoříme nástroj, který bude schopen zobrazovat XML soubory pro uživatele příjemně a zároveň bude schopen hlídat určitá pravidla editace, bylo by dobré zastřešit správu pomocí tohoto nástroje i nad ostatními daty. Tím by správa měla jednotnou formu a řešení by bylo komplexnější. Konkrétně to tedy znamená najít obecnější způsob implementace zvoleného protokolu. Ten by umožnoval editovat formou XML souboru nejen texty již jako XML uložené, ale i celé tabulky, že kterých by se XML soubor nejdříve vytvořil.

Kapitola 2

Jazyk XML a související technologie

Jak už bylo několikrát zmíněno, současný způsob uchovávání statických textů využívá jazyk XML (eXtensible Markup Language). Jedná se o standard (přesněji doporučení konsorcia W3C) jak vytvářet značkovací jazyky[1]. XML je tedy obecný značkovací jazyk(metajazyk), který konceptuálně můžeme chápat jako podmnožinu staršího standardu SGML (Structure Generalized Markup Language)[19]. I když se jedná o relativně nový jazyk, díky svým výjimečným vlastnostem se brzy dočkal mnoha rozšíření a dalších doplňkových technologií. Proto je v současné době nespočet možností, jak XML jazyk využívat. Abychom vybrali ten nejvhodnější způsob, podívejme se na některé z jeho vlastností a technologií, které bychom mohli pro naše účely použít.

2.1 XML – univerzální formát pro zpracování informací

Formát XML vznikl z potřeby uchovávat data způsobem, který by nebyl přímo závislý na určité platformě nebo technologii, zároveň však řešil i způsob, jak uchovávat kromě dat samotných i jejich metainformace. Narozdíl od většiny ostatních značkovacích jazyků XML umožňuje popsat logickou strukturou dokumentu použitím vlastních tagů (značek), které označují jednotlivé elementy dokumentu obsahující data a dokáží tak mnohem přesněji označit význam prezentovaných informací[18]. V logické struktuře je kladen hlavní důraz na charakter jednotlivých informací a na jejich vzájemné vztahy, což dovoluje jednotlivé informace snadno spravovat, označovat data stejné povahy a umožnit jejich další zpracování. Kromě elemetů se pro dotvoření logické struktury v XML dokumentu setkáme ještě s atributy nesoucími dodatečné informace k elementům, dále s textovými uzly v elementech obsahujících vlastní texty, instrukcemi pro zpracování, případně s notacemi a komentáři.

Logická struktura dat se promítá i v syntaxi jazyka. Aby byl XML dokument dle obsažených dat správně strukturovaný („well-formed“), musí mít právě jeden kořenový element (root), všechny použité značky musí být párové (s výjimkou speciálního případu elementu bez obsahu) a všechny hodnoty atributů musí být uzavřeny v uvozovkách. Elementy mohou být vnořeny, ale nemohou se překrývat, jejich pojmenování je navíc citlivé na velikost písmen.

2.1.1 Vzhled dokumentů

Jazyk XML se sám o sobě vzhledem dokumentu nezabývá. Vizuální podoba se definuje až připojeným stylem nebo pomocí transformací do jiného typu dokumentu, případně do jiné struktury XML. Mezi nejznámější jazyky určující výsledný vzhled XML dokumentu patří následující.

2.1.1.1 CSS styly

CSS (Cascading Style Sheets) je jazyk pro popis způsobu zobrazení HTML, XHTML nebo XML dokumentů[2]. Umožňuje dynamicky formátovat vzhled dokumentu, aniž by jakýmkoli způsobem ovlivnil jeho obsah nebo strukturu. K XML dokumentu je připojen externě jako samostatný soubor, což s sebou přináší možnost připojovat týž styl několika XML dokumentům, a tak sjednocovat jejich vzhled.

2.1.1.2 XSLT transformace

Transformace XSLT (eXtensible Stylesheet Language Transformations) je součástí specifikace XSL (Extensible Stylesheet Language), která slouží k určení přesného vzhledu dokumentu [20]. Původně byla koncipována jako vylepšení kaskádových stylů CSS, ale postupem času se XSL vyprofilovala ke dvěma hlavním účelům. První, tzv. transformační část (XSLT), slouží k transformaci dokumentu a druhá, formátovací část (XSL-FO), která sledovala původní myšlenku CSS, slouží k definici jejich výsledného vzhledu. Základem je připravená XSLT šablona společně se zdrojovými XML daty. To slouží jako vstup pro XSLT procesor, který na výstupu produkuje šablonou předepsaný formát. Síla XSLT spočívá v tom, že z jednoho zdrojového XML dokumentu můžeme generovat několik různých druhů dokumentů jiných.

2.1.2 Jazyky pro popis struktury XML dokumentů

Skutečnost, že XML jazyk umožňuje vytvořit dokumenty s libovolnou strukturou za použití jakýchkoli značek, patří mezi jeho nejsilnější stránky. Jak ovšem ukazuje praxe, v mnoha případech právě tato volnost s sebou může přinášet řadu problémů. Často totiž potřebujeme pracovat s předem definovanými daty a ne s nahodilou skupinou XML elementů. Jako jeden příklad za všechny uveďme práci s daty databází, která se nás přímo dotýká.

V těchto případech potřebujeme nástroj, který by dokázal definovat, jaké elementy a atributy lze v XML dokumentu používat, jak je můžeme vzájemně kombinovat a co mohou obsahovat. Mezi tyto nástroje patří i následující dvě technologie:

2.1.2.1 DTD

DTD (Document Type Definition) je jazyk pro popis struktury XML/SGML dokumentu[3]. Omezuje množinu přípustných dokumentů spadajících do daného typu nebo třídy. Struk-

tura třídy nebo typu dokumentu je v DTD popsána popisem povolených značek (nebo též elementů) a atributů. Popisuje, jak mohou být značky navzájem uspořádány a vnořeny. Vymezuje atributy pro každou značku a typ těchto atributů. DTD je poměrně starý jazyk, jeho další nevýhoda je to, že není XML soubor.

2.1.2.2 XML Schéma

XML Schema Definition (XSD) popisuje strukturu XML dokumentu. Jeho možnosti definování pravidel jsou mnohem bohatší a propracovanější než u DTD. Jedná se v podstatě o dokument XML, jež používá speciální elementy, pro která platí následující pravidla. [9]

Celé schéma musí být vždy uzavřeno v elementu s názvem *schema* obsahujícím definice elementů, které lze použít jako kořenové elementy, tj. XML dokument jimi začíná. Definice elementu se zapisuje pomocí elementu <element/>. Pro každý element musí schéma určit jeho typ. Rozlišovány jsou přitom dva druhy typů – jednoduché a komplexní. Jednoduché typy se používají pro skalární hodnoty jako řetězec, číslo, datum apod. Obsahuje-li však element další elementy nebo atributy, musíme použít komplexní typ (complexType). Atributy se deklarují až za vnořenými elementy pomocí elementu <attribute/>. U atributu rovněž musíme určit jeho název a datový typ.

Takto vypadá základní náčrt vlastností, které technologie využívá, její skutečná struktura může být ještě mnohem složitější. Použitím XML schématu k definování pravidel pro tvorbu XML dokumentů vznikne opravdu robustní kontrola, která, je-li správně definovavá, pomůže k ošetření rozmanitých požadavků na dokument.

Kapitola 3

Technologie zpřístupnění dat

Aby texty mohly být k dispozici příslušným administrativním pracovníkům MU, musíme umožnit jejich správu. Z našeho pohledu to znamená zajistit přístup k datům, která jsou uložena v databázi na určitém databázovém stroji, data zpracovat a změněná je pak opět uložit do databáze. Koncový uživatel by ale měl být podobných technických faktů ušetřen, správa by pro něho měla být co nejpřirozenější a nejpohodlnější. Pro něj potřebujeme vytvořit iluzi, že zmíněné soubory vypadají podobně jako ostatní soubory uložené lokálně na jeho počítači, a proto i práce s nimi bude v podstatě stejná. Fakt, že editovaný text je uložen fyzicky někde jinde než na jeho počítači, by v ideálním případě pracovník vůbec řešit neměl.

Webová prezentace představuje pro informační systémy MU jen nepatrný střípek z mozaiky činností, které musí zajišťovat. Proto otázka přístupu k datům a principů jejich správy všeobecně byla v podobných souvislostech vyřčena už několikrát a jejím řešením vznikly různé informační podsystémy pro podporu řízení, správy a prezentace univerzity[11]. Pro nás to znamená, že se můžeme zaměřit na konkrétní způsob, jakým bude možné co nejefektivněji provést zpřístupnění textů; o zajištění zázemí se postará již vybudovaný nástroj poskytující prostředky pro správu a zpřístupňování dat, konkrétně celouniverzitní intranetový systém INET[4]. K samotnému přenosu tedy musíme vybrat některý z aplikačních protokolů, který bude nejvíce vyhovovat našim potřebám.

3.1 Protokoly pro přenos a správu dat

Protokol je specifikace, která definuje postupy a parametry potřebné ke kominukaci a výměně dat. My se zaměříme na protokly aplikační, které zajišťují přenos dat pro konkrétní služby aplikací.

3.1.1 FTP

FTP (File Transfer Protocol) je protokol aplikační vrstvy z rodiny TCP/IP určený pro přenos souborů mezi počítači nezávisle na operačním systému, nad kterým aplikace daného počítače běží. Patří mezi nejstarší protokoly. Mezi jeho dovednosti patří řízení přístupu (přihlašování login/heslo), specifikace formátu přenášeného souboru nebo například výpis vzdáleného adresáře. Samotný přenos v jeho pojetí představuje službu fungující na principu klient/server. Role serveru náleží v této aplikaci počítači, který vlastní soubory, klientem je ten, kdo tyto soubory požaduje. Zatímco FTP server nabízí svá data ke stažení nebo naopak

nahrání na počítačový uzel, na kterém běží, klient je jen jakýsi prostředník, skrze kterého jsou tyto data uložená na vzdáleném počítači viditelná a uživatel si na ně může sáhnout, uložit na svůj počítač a nahrát je zpět na server.

3.1.2 SCP

Protokol SCP (Secure Copy) rovněž umožňuje přenos souborů a průchod jejich adresářovou strukturou. Pro bezpečnou komunikaci mezi dvěma počítači využívá protokol SSH (Secure Shell). Ten pokrývá tři základní oblasti bezpečné komunikace: autentizaci obou účastníků komunikace, šifrování přenášených dat a integritu dat. SCP se běžně používá na unixových serverech, ale existují i implementace pro Windows.

3.1.3 SFTP

Dalším protokolem je SFTP. Ten kromě samotného přenosu nabízí široké možnosti pro doplňující operace se soubory. Umožňuje pokračovat v přerušených přenosech, vypisovat adresáře i odstraňovat soubory na vzdáleném počítači. Sám o sobě ovšem nezajišťuje ani autentizaci ani zabezpečení přenášených dat, k tomu obvykle využívá protokol SSH-2. Je ale navržen tak, aby jej bylo možné používat i nad jiným protokolem.

3.1.4 WebDAV

WebDAV (World Wide Web Distributed Authoring and Versioning) je rozšíření protokolu HTTP/1.1 o možnost vzdáleného sdílení a správy souborů. Protokol umožňuje například zamykání souborů, správu verzí nebo řízení přístupu. Práce se zdroji je pomocí tohoto protokolu stejně snadná jako s lokálními soubory. Díky tomu, že WebDAV běží nad HTTP, lze snadno použít šifrovanou verzi HTTPS. Další z nesporných výhod je možnost využití WebDAV klienta, který dokáže soubory zpřístupnit formou virtuálního disku, což je z pohledu práce s daty velmi příjemné.

Síla WebDAV protokolu tkví ve spojení dvou propracovaných technologií: HTTP (Hypertext Transfer Protocol) a XML (Extensible Markup Language). Protokolem HTTP se ošetřuje způsob komunikace. Kromě standardních příkazů využívaných protokolem HTTP (PUT, GET, COPY, ...) zavádí WebDAV pro své potřeby některé nové. Technologii XML WebDAV využívá jako nositele strukturovaných dat, které doplňují informace jak na dotaz klienta, tak i na výsledek zasílaný serverem. To znamená, že v XML části komunikace mohou být odesílány informace o souborech a adresářích, výsledky zpracování příkazů pro jednotlivé soubory a podobně. V hlavním DTD (pojmenovaném DAV) popisující XML část protokolu jsou pak definovány všechny potřebné elementy a atributy nutné pro základní použití WebDAVu.

3.2 Porovnání vlastností protokolů

Abychom si z protokolů, které jsou v současné době k dispozici, vybrali pro účel správy textů ten správný, nabízí se jejich srovnání. Každý z nich má své silné i slabé stránky. Než se ale rozhodneme, co je pro nás důležité a co může zůstat v pozadí, podívejme se, čím se jednotlivé protokoly od sebe mohou lišit.

- Protokol FTP je populární a efektivní nástroj pro přenos dat mezi serverem a klientským počítačem. Dokáže za použití vhodného klienta zobrazit obsah uživateli jako další disk. Pro potřeby správy dat uložených na vzdáleném počítači však úplně ideálním řešením není, a to především v souvislosti s otázkou zajištění bezpečnosti. Při běžném připojování pomocí protokolu FTP jsou totiž přihlašovací údaje přenášeny nešifrovaně v textové podobě, což může mít za následek zachycení těchto údajů třetí stranou a ohrožení bezpečnosti přenosu. V dnešní době je bezpečnost přenosu citlivých dat řešena různým rozšířením protokolu (FTPS, který zjednodušeně přidá šifrování), i tak ale chybí metody a vlastnosti, které by z pohledu správy souborů byly velice přínosné (možnost zamykání souborů, poskytnutí metainformací, ...).

- SCP je protokol, který sice sám o sobě šifrování ani autentizaci nezajišťuje, ale využívá toho, že obojí poskytuje protokol SSH. V porovnáním s FTP přichází navíc s vylepšením při kopírování souborů na vzdálený počítač. Umožňuje totiž předat i další informace jako oprávnění, časy, modifikace a podobně. Ale i tady zjistíme, že bezpečnost přenosu může trochu pokulhávat. Protokol je navíc dost chudý, v podstatě jediné, co umí, je kopírování souboru z místa na místo. Ve většině případů je proto nahrazován komplexnějším protokolem SFTP.

- SFTP, neboli SSH File Transfer Protocol, je protokol pro přenos souborů pomocí SSH. S protokolem FTP má kromě podobného označení společné snad jen to, že řeší podobné problémy. Vychází ze zastaralého protokolu SCP, jehož funkcionalita byla silně omezená, a nabízí široké možnosti rozšířené práce se soubory. Lze ho tedy považovat za jeden ze způsobů vzdáleného přístupu pro správu dat. SFTP klient též umožňuje pokračovat v přerušených přenosech, vypisovat adresáře i odstraňovat soubory na vzdáleném počítači.

- V souvislosti se zpřístupněním a správou vzdálených dat je stále častěji skloňovám i protokol WebDAV [14]. Funkce, které tento protokol umožňujě, jej právem řadí mezi velmi slibné nástroje pro podporu vzdálené správy dat. Kromě základních operací se soubory a adresáři jako kopírování, mazání, přesouvání totiž umožňuje protokol mnohem více. Může libovolně připojovat k jednotlivým souborům informace o nich samotných, později je upravovat, číst nebo mazat. Navíc nejsou omezeny standardní vlastnosti, můžeme si je podle potřeb vytvářet a spravovat. Další ze zajímavých možností protokolu je existence zámků proti manipulaci a nechtěné změně souboru. Tím

se dá vyhnout konfliktu, který může vzniknout, když dva různí lidé pracují s jediným souborem. V neposlední řadě je dobré zmínit možnost tvořit kolekce. Tváří a chovají se jako obyčejné adresáře, ale nejsou určeny umístěním na pevném disku serveru, jak je obvyklé. Jejich identifikátory jsou URI, takže se může stát, že jedna kolekce může obsahovat soubory uložené na různých místech, uživatel ale nabude dojmu, že pracuje s daty jedné složky.

Kapitola 4

Způsob zpřístupnění

Po prostudování možností, které se nám v souvislosti s požadovanou správou textů a případně i dalších souborů nabízejí, je na řadě výběr vhodného řešení, jak zpřístupnění provést. To znamená najít způsob, který dovolí na jedné straně bezpečný a efektivní přístup k datům, na straně druhé jejich pohodlnou a plnohodnotnou editaci. Pro uživatele, který není zrovna zdatným programátorem a problém pro něj může být i důsledná kontrola správné syntaxe XML jazyka, musíme zajistit, aby veškeré změny, které nad daty provádí, mohly být hlídány samotným systémem. V neposlední řadě je dobré si uvědomit, že v případě slibně se rozvíjejícího systému na vzdálenou správu dat by se časem mohly vyskytnout nové možnosti jeho použití. Proto by způsob navržení měl být postaven na dostatečně obecném a v případě potřeby snadno rozšiřitelném základu.

4.1 Výběr protokolu

Předchozí kapitola byla věnovaná srovnání některých protokolů, které bychom mohli pro zpřístupnění dat použít. Po podrobném prozkoumání výše uvedených požadavků na navrhovaný systém i protokolů samotných stojí za zmínku dva, které by v našem případě mohly vyhovovat. Jedná se o protokol SFTP (SSH File Transfer Protocol) a WebDAV (World Wide Web Distributed Authoring and Versioning). Z hlediska bezpečnosti přenášených dat můžeme říci, že vyhovují protokoly oba. WebDAV pro zajištění bezpečnosi může využít místo HTTP protokol HTTPS, který je v podstatě identický jako HTTP, jen přidává možnost šifrovat data pomocí SSL/TSL. SSL je pro HTTP vhodné, protože umožňuje poskytnout ochranu přenosu i v případě, kdy je otevřená pouze jedna strana komunikace (typicky je ověřen pouze server, uživatel potvrdí certifikát). Naproti tomu SFTP využívá prokol SSH, který umožňuje transparentní šifrování přenášených dat při komunikaci mezi dvěma počítači a zabezpečuje autentizaci komunikujících, šifrování přenášených dat i jejich integritu.

Podívejme se tedy na oba protokoly z hlediska možností správy dat. I když základní operace, jako například kopírování, mazání souborů nebo výpis adresáře mají SFTP i WebDAV společné, v popisu dovedností jednoho z nich objevíme i mnohem pokročilejší a pro potřeby správy přinejmenším velmi zajímavé možnosti. Protokol WebDAV jde totiž ve výčtu svých výhod dál. Nabízí možnost zámků nad dokumenty, což je velmi užitečné v případě přístupů více uživatelů k jednomu dokumentu, dále pak verzování, za zmínku jistě stojí i možnost pokročilého definování vlastností, které je možno k dokumentu připojit. Jak již jeho název vypovídá, byl navržen k spolupráci a distribuovanému sdílení souborů, a tak i

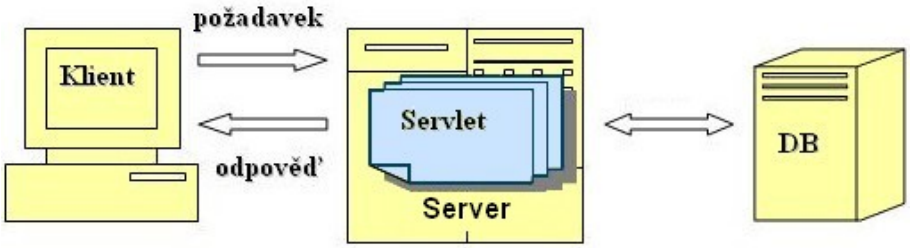
když v první fázi implementace všech jeho výhod nevyužijeme, případné rozšíření funkčnosti může zůstat otevřené. Protokol WebDAV je poměrně novou technologií, přesto je jeho podpora stále častěji přidávána do nejrůznějších aplikací, o čemž svědčí i fakt, že o integraci podpory protokolu do svých systémů se pokouší i firmy jako Microsoft nebo Apple. Některé jeho implementace v podobě WebDAV klientů poskytují ještě jednu velmi užitečnou vlastnost – umožňují pracovat s daty jako s virtuálními soubory a uživateli je dávat k dispozici, jako by se tyto soubory nacházely na nějakém běžném síťovém disku. To umožňuje nejen editovat data v libovolném editoru, ale především s nimi pracovat jako s běžnými soubory.

Na základě výše zmíněných informací jsme se proto přiklonili k vybudování systému zabezpečující zpřístupnění a vzdálenou správu dat pomocí protokolu WebDAV, neboť nejvíce vyhovuje vysloveným požadavkům. V první fázi využijeme zřejmě jen zlomek možností, které protokol nabízí. Ostatní vlastnosti a funkce nám však budou v případě rozříření k dispozici, takže systém bude stát na dobrém obecném základu.

4.2 Implementace protokolu

K implementaci vybraného protokolu použijeme technologii Java Servlets, která umožňuje obsluhovat protokoly fungující na principu požadavek/odpověď. Jedná se o programy napsané v jazyce Java a vykonávané na straně serveru. Mezi výhody patří platformová nezávislost, objektový přístup, popřípadně modularita. Je vystaven tak, že se zkompilovaný uloží do paměti při prvním spuštění, při dalším požadavku je již volaný odsud, a tím šetří systémové nástroje. Servlety jsou podle specifikace nezávislé komponenty, které můžeme použít na různé úlohy (mezi které patří i čtení a zápis do databáze) pro generování dynamického obsahu. Standardní rozhraní pro obsluhu požadavků a odpovědí mezi klientem a serverem definuje Java Servlet API.

Princip komunikace klienta a serveru pomocí servletu:



Obrázek 4.1: Ukázka komunikace klienta a serveru pomocí servletu

- Klient zašle serveru požadavek.
- Server přepošle požadavek příslušnému servletu pomocí Java Servlet API.
- Servlet vytvoří odpoveď a předá ji serveru. (Odpověď je vytvořená dynamicky a její

obsah závisí od požadavku, který poslal klient. Servlet může využít různé externí zdroje.)

- Nakonec pošle server klientovi odpověď.

Abychom mohli zpracovávat požadavky a odpovědi, které se budou generovat, podívejme se nejdřív, jaké způsoby WebDAV protokol při této komunikaci používá.

Základem komunikace je spojení dvou technologií – HTTP (HyperText Transfer Protocol) pro zajištění komunikace a XML (Extensible Markup Language) pro upřesnění posílaných informací (v XML části odpovědi mohou být posílány informace o souborech, případně parametry příkazů. Struktura informací je definována v hlavním DTD s názvem DAV, můžeme však definovat i atributy vlastní.

Vše pracuje na principu, kdy klient zašle prostřednictvím protokolu příkaz, který může být více specifikován v připojeném XML dokumentu. Server dotaz zpracuje a pošle výsledek opět ve formátu XML. Jak bylo řečeno již v úvodu, WebDAV je rozšířením protokolu HTTP 1.1, a proto k zajištění komunikace využívá jeho konkrétní metody. Mohou to být:

- GET – používá se pro získání dokumentu ze serveru
- POST – jako GET, za hlavičkami lze odeslat data
- HEAD – jako GET, ale získáme pouze hlavičky, ne tělo
- OPTIONS – používá se k zjištění možností serveru
- PUT – jako GET, k uchování těla požadavku na místě daném URL
- DELETE – používá se k odstranění dokumentu ze serveru
- TRACE – slouží k sledování cesty požadavku

Kromě těchto standardních metod si protokol WebDAV definuje další, konkrétně:

- PROPFIND – slouží k získání vlastnosti (hodnoty nebo názvu) vzdáleného dokumentu
- PROPPATCH – slouží k vytvoření nové vlastnosti, změně nebo odebrání stávající vlastnosti
- MKCOL – používá se k vytvoření nové kolekce
- COPY – používá se ke kopírování kolekcí a souborů
- MOVE – slouží k přesunu kolekcí a souborů
- LOCK – slouží k uzamčení souboru nebo kolekce
- UNLOCK – používá se k odstranění zámku uzamčené kolekce nebo souboru

Kapitola 5

Výběr nástroje pro zobrazení vytvořených souborů

Informace, které pomocí protokolu dostaneme, musíme nějakým způsobem zobrazit. Jak již bylo řečeno dříve, data určená k editaci a správě je nejvhodnější zobrazovat strukturovaně, pomocí XML souborů, nad kterými budeme moci dále pracovat. Tyto soubory by proto bylo vhodné nějakým způsobem setřídit a rozčlenit do složek, aby práce a vyhledávání mezi nimi bylo co nejintuitivnější. Vzhledem k tomu, že se předpokládá postupné rozšiřování systému, bude souborů k editaci přibývat a mělo by být předem jasné, jakým způsobem je bude možné organizovat.

Nejprve je potřeba vytvořit určitou statickou strukturu, která bude reprezentovat logické třídění zobrazovaných dat. Bylo by vhodné, aby uživatel měl soubory logicky rozčleněné do adresářů pro snažší orientaci, a to jak členění podle zdrojů, které zpřístupníme (číselníky, texty, ...), tak i textů samotných. Vzhledem k jejich počtu by totiž i vyhledání konkrétního souboru mohlo stát značné úsilí. Podobným způsobem by se mohly členit a seskupovat i další objekty, které postupem času do systému přibudou.

Vlastní provedení navrhované struktury zajistí servlet, který bude vytvořen; WebDAV, jakožto protokol navržený pro správu dat, základní operace se soubory a adresáři podporuje. Dalším úkolem je proto najít způsob, který dovolí nejen data v podobě XML souborů pohodlně upravovat, ale který zároveň umožní nahlížet a procházet virtuální adresářovou strukturou.

V dnešní době sice není podpora protokolu WebDAV samozřejmostí, přesto existuje celá řada nástrojů, která tento nadějný prokol přijala. Můžeme tak kromě klasických klientů, které známe například v souvislosti s protokolem FTP, najít i konkrétní XML editory, které umožnují připojení právě pomocí tohoto protokolu. Tato varianta mi zpočátku přišla jako ideální řešení, protože uživatel by k editaci potřeboval pouze daný editor a práce by tak pro něj byla celkem snadná. V nejhorším případě by se musel naučit používat nový software, v případě, že by ho již znal, by dokonce nebylo potřeba ani to. Začala jsem tedy pátrat po vhodném nástroji, ale brzy se ukázalo, že situace nebude tak růžová, jak zpočátku vypadala.

Editorů podporující protokol WebDAV není málo, ovšem podmínky, které jsou pro tento kontrétní případ potřeba, byly rozhodující (více o nich bude řeč později, vzpomeňme zatím pouze hlídání validity nebo například rozumné grafické zobrazení). Těmto podmínkám určitá skupina editorů vyhovovala, ovšem další požadavek, procházení výše zmíněné struktury, se ukázal být tím největším oříškem. V podstatě každý z vybraných editorů podporoval protokol WebDAV, ale způsobem, kdy uživatel musel zadat celou cestu k souboru, tzn. včetně složek, které byly vytvořeny za účelem přehlednosti. O editorech, které toto umož-

ňovaly, bude řeč dále, je jistě užitečné se s nimi seznámit. Zároveň však je zapotřebí najít způsob, který by i požadavek na procházení adresářové struktury splňoval.

5.1 Webdav klienti

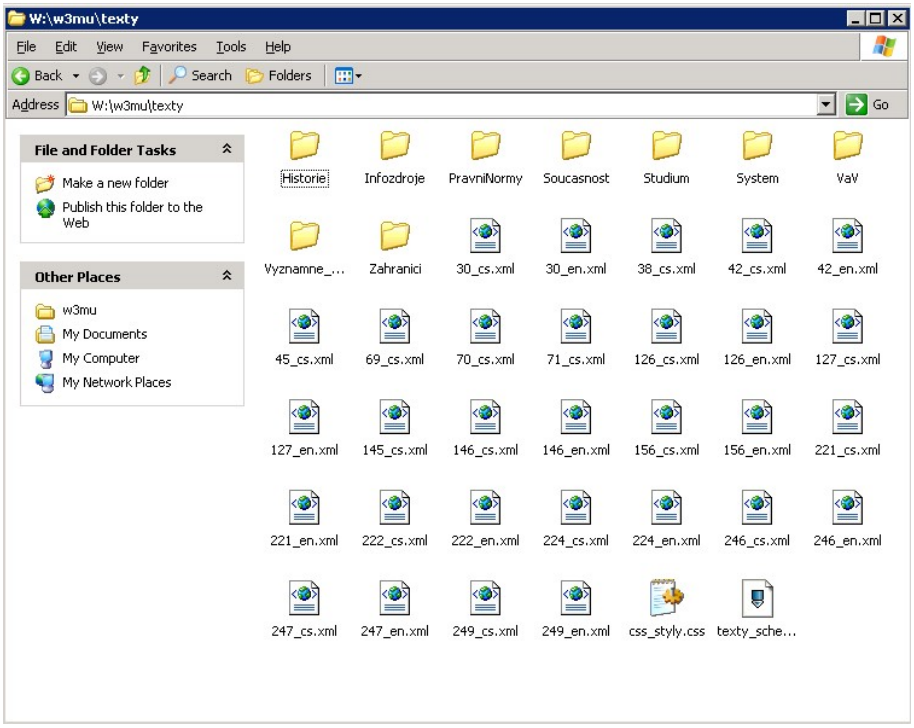
Klient je program, který přistupuje k určité službě, jež běží vzdáleně na straně serveru. Nejmasivnější použití objevíme ve spojitosti s protokolem FTP, popřípadně SCP, ale i WebDAV protokol se dočkal nejednoho povedeného provedení. Kromě klasických podob se můžeme setkat i s graficky velmi povedenými nástroji, které uživateli věrohodně připomínají práci s klasickými průzkumníkem, popř. simulují princip přídání dalšího disku. Je sice zřejmé, že uživatel bude muset kromě editoru samotného používat navíc WebDAV klienta, ale práce s ním se dá snadno zakomponovat do klasického používání systému Windows a zpřístupnit virtualní souborový systém jako sítový disk, takže po úspěšném nainstalování uživateli prakticky práce navíc nepřibyde. Zároveň s použitím klienta se rozšíří možnost výběru vhodného XML editoru, protože odpadne požadavek podpory samotného protokolu. Po průzkumu existujících editorů si myslím, že tato výhoda nutnost používání dalšího programu bohatě vynahradí. Podporu protokolu formou WebDAV klienta v sobě obsahuje i samotný operační systém Windows, avšak s určitými nedostatky, které nám nedovolují se na tento způsob plně spolehnout. Je tedy nutné podívat se i na další možnosti.

5.1.1 Novell NetDrive

Novell NetDrive je program, který umožňuje zpřístupnění FTP či WebDAV připojení jako klasické síťové jednotky [6]. Po instalaci jsou v programu již nastaveny vzorové účty, takže je snadné nastavit účet vlastní – stačí napsat adresu serveru, zadat jméno, případně heslo a zvolit písmeno, pod kterým chceme na disk přistupovat. Je přehledný, uživatelsky příjemný, jednoduchý na ovládání. Mezi další jeho vlastnosti patří také nastavení výchozího adresáře, udržování spojení, definice práv na soubory, atd. Pomocí namapování tak skutečně můžeme přistupovat k obsahu běžnými programy, které jinak umí ukládat jen na lokální disky. Možnost bezplatného používání tak kazí pouze fakt, že není plně kompatibilní s antivirovými programy F-Secure Anti Virus and KasperSky Anti Virus.

5.1.2 WebDrive

Tento program umožnuje rovněž namapovat prostor na serveru jako síťový disk. S tímto diskem se pak může pracovat v každé aplikaci běžící pod Windows stejně, jako by se jednalo o běžný sítový disk. Podporuje všechny standardní technologie jako *drag and drop* apod. Narozdíl od předchozího programu však není k dispozici zdarma, ale za určitý poplatek v závislosti na počtu uživatelů[15].



Obrázek 5.1: Ukázka obsahu síťového disku zpřístupněného pomocí WebDrive (v klasickém windows exploreru)

5.1.3 Další klienti

Mezi další klienty podporující protokol WebDAV patří například Total Commader nebo DAV client. Uživatelsky již nejsou tak příjemní ani nenabízí tak široké možnosti jako klienti výše zmínění, ale i oni jsou možností, jak se pomocí protokolu WebDAV připojit.

Kapitola 6

Nástroje na úpravu XML souborů

Po vyřešení otázky zobrazení dat se zaměříme na jejich vlastní editaci. XML dokument je v podstatě standardní textový soubor, pouze doplněný o speciální značky – tagy. Proto jejich úprava by klidně mohla být prováděna pomocí libovolného textového editoru. Pro člověka znalého syntaxe XML jazyka je to často velmi rychlý způsob, jak data měnit, vyžaduje to ovšem jistou dávku pozornosti, aby se úpravami neporušila samotná struktura dokumentu. Proto vzniklo nepřeberné množství editorů, které se snaží uživateli data podat v různých, více či méně podrobných pohledech. Navíc nabízí i další podpůrné funkce, které práci se soubory velmi usnadní. Jako hlavní jmenujme kontrolu struktury XML dokumentů, kontrolu validity oproti požadovanému DTD nebo schématu, našeptávání vhodných elementů k vložení nebo třeba elegantní grafické zobrazení.

V současné době existuje několik skupin XML editorů, v závislosti na tom, jak jsou data editorem zobrazována a do jaké míry musí uživatel syntaxi XML jazyka znát. První skupinu tvoří nástroje, které zobrazují přímo zdrojový kód jazyka, ovšem s možností barevného rozlišení jednotlivých tagů od prostého textu. Další skupinou jsou editory, které sice zobrazují dokument strukturovaně pomocí tagů, mají však další funkce jako automatickou kontrolu struktury, našeptávání elementů a atributů atd. Následují programy, které již umožňují pracovat v určitém WYSIWYG režimu. To je způsob zobrazení dokumentu, kdy uživatel vidí zformátovaný text dokumentu, jeho logickou strukturu řešenou pomocí XML elementů však ne. Pokročilé nástroje již syntaxi XML jazyka uživateli úplně skryjí a pomocí různých prostředků navodí dojem velmi podobný robustním textovým editorům (MS Word, Open Office, ...). Samozřejmě však ponechávají funkce, které jsou pro úpravu XML souboru nezbytné.

6.1 Požadavky na možnosti editoru

Otázkou, zda požadovat podporu protokolu WebDAV, jsme se již zabývali. Jistě není na škodu, když tuto funkci bude editor ovládat, avšak bez přehledného způsobu procházení souborů může tato vlastnost ztrácet na efektivitě. Pamatovat si celou cestu k jednotlivým souborům se nezdá být zrovna pohodlné, navíc chceme-li umožnit uživateli například kopírování souborů, bez WebDAV klienta se stejně neobejdeme. Proto tato funkcionalita může být vítaná a využívaná, není však bezpodmínečně požadována.

Je zřejmé, že struktura XML dokumentů bude záviset na konkrétních datech, ze kterých virtuální soubor vytvoříme. Jelikož je ale většina těchto dat brána z databáze, bude nutné

kontrolovat jejich správnost a kompatibilitu s danou databázovou tabulkou. K tomu je ideální využít možnosti jazyků pro popis struktury XML dokumentů. Do úvahy přichází DTD nebo XML schéma. Po porovnání jejich dovedností dojdeme k závěru, že oproti DTD je XML schéma daleko propracovanější, a tudíž nabízí více možností použití. Proto by i hledaný editor měl podporu XML schémat nabízet.

Dalším aspektem, který může hrát roli při výběru vhodného editoru, je grafické znázornění. Cílem této práce je zpřístupnit editaci dat a textů lidem, kteří odpovídají za jejich obsah, nikoli formu a způsob uložení. Neočekává se obecně jejich pokročilá znalost XML jazyka, a to by se při výběru nástroje mělo zohlednit. Pokud jde o editaci textů i nejrůznějších číselníků, bylo by ideální najít editor, který strukturu XML dokumentu a jeho elementy uživateli skryje. Ten tak ztrácí možnost případného porušení správnosti dokumentu a navíc bude ušetřen zmatení při pohledu na požadovaný text.

V dnešní době existuje několik principů, které editory využívají. Některé definují vlastní způsoby, pomocí níž text formátují, některé využívají standardní styly, jako například CSS styl. Výhodou tohoto způsobu je snadná přenositelnost mezi editory a také částečná nezávislost. Vytvoříme-li pro daný XML soubor CSS styl, můžeme ho použít v několika editorech, které tuto funkci umožňují. CSS podporují také webové prohlížeče, takže není problém si i bez editoru dokument s existujícím stylem zobrazit.

6.2 Vhodné XML editory

Jak již bylo řečeno, pro naše potřeby je vhodné najít editor, který umožňuje co možno nejpohodlnější způsob editace. Proto jsem se zaměřila na WYSIWYG editory (editory pracující způsobem, při kterém je verze zobrazená na obrazovce vzhledově totožná s výslednou verzí dokumentu) a porovnala jejich vlastnosti a funkcionalitu. Tyto editory samozřejmě nabízí nejen plně zformátovaný pohled na text, ale i další možnosti jako hierarchické zobrazení nebo jiné pohledy, které zobrazují i samotné tagy. Při srovnání hrály svou roli požadavky uvedené výše, ostatní možnosti, které editory nabízí, a v neposlední řadě také cena programu, která se editor od editoru občas výrazně liší.

6.2.1 XMLmind

XMLmind je jedním z nejznámějších WYSIWYG nástrojů pro úpravu XML dokumentů. Je napsán v Javě a mohou ho používat jak uživatelé operačního systému Windows, tak i Linux nebo např. Mac OS. Editor umožňuje několik druhů pohledů, a to jak klasický stromový, tak zformátovaný podle přiloženého stylu. Je-li připojených stylů víc, pak si uživatel může mezi nimi libovolně přepínat. Program existuje ve dvou verzích. Standardní, která je i pro komerční účely zdarma, a pak profesionální, s rozšířenými možnostmi, ovšem již za poplatek. Z funkcí, které by nás u profesionální verze mohly zajímat, stojí rozhodně za zmínku podpora protokolu WebDAV, kterou základní výbava neobsahuje, a pak kompletní podpora XML schémat. Když pomineme tyto nevýhody, můžeme i standardní verzi editoru považovat za velmi silný a pro naše potřeby slibný nástroj.

- Jednou z nejdůležitějších vlastností je podpora XML schémat. Základní verze má sice trochu omezené podmínky použití, pro naše potřeby jsou však dostačující. Editor umožňuje kontrolovat dokument oproti dodanému schématu, našeptávat možnosti při editaci elementů a ukazovat chyby, které schématu neodpovídají.

- Zamezit uložení nevalidního dokumentu na této úrovni nejde, ovšem tento úkol bude mít na starost servlet, který bude požadavky obsluhovat. Pro nás je ale velmi výhodné to, že editor nedá možnost uživateli zasahovat do struktury dokumentu přímou editací elementů. Je k dispozici řada náhledů, kde jsou nebo nejsou elementy vidět, nelze je však měnit standardním způsobem mazáním a přepisováním částí tagu. Nestane se tak, že by uživatel omylem smazal například koncový zobáček měněného tagu, a tím by dokument narušil, neboť ke změně či přidání XML značky dochází pouze na principu stisknutí tlačítka. Uživatel elementy přímo nepíše. Jediné, co může ručně přepisovat, je vlastní obsah elementů, popřípadně hodnoty jeho atributů. Navíc editor špatně vystavěný dokument ani nenačte, takže uživateli se vždy dostane do ruky jen jeho správná varianta.

- Jak je již uvedeno, XML dokumenty je možné v tomto editoru zobrazovat pomocí nejrůznějších pohledů, od prostého stromového zobrazení až po graficky náročně vystavěného. K těmto pohledům editor používá technologii CSS stylů. Pro jeden dokument jich může být k dispozici hned několik. Protože se jedná o standardní způsob připojování stylů k XML dokumentu, svůj styl si může vytvořit každý přesně na míru. To je výhodné v případě, kdy máme specifické nároky na zobrazení souboru. V našem případě se jedná především o to, aby zobrazení textu určeného k editaci bylo pokud možno podobné výslednému zobrazení prezentace. Pro člověka měnícího údaje je tak práce s XML dokumentem mnohem intuitivnější. V případě CSS stylů se jedná o celkem rozšířenou a čím dál více editory podporovanou technologii, není na škodu ani fakt, že při změně editoru si vytvořené styly můžeme ponechat a znovu použít.

- Po vyzkoušení několika nástrojů na úpravu XML dat stojí za zmínku ještě jedna poznámka. Tento editor vcelku slušně pracuje s češtinou, a to dokonce i v případě CSS stylů. Nutno říci, že i tato celkem základní vlastnost není u ostatních editorů samozřejmostí. Podrobné informace o produktu je možné nalézt na stránkách [5].

6.2.2 Oxygen

Oxygen je další slibný editor pro úpravu XML dokumentů. Podpora XML schémat i CSS stylů je samozřejmostí, ve skutečnosti se jedná o podobně silný nástroj jako editor zmíněný výše (včetně podpory protokolu WebDAV). Validace probíhá přímo při psaní, podtržením chybně napsaných elementů a atributů, ať už něco chybí, či přebývá. Za menší neduhy lze

považovat fakt, že umožňuje ruční editaci jednotlivých tagů, čímž dává možnost porušení struktury, při testování se dále objevil problém s češtinou v CSS stylech. Tento program není k dispozici zcela zdarma, pouze za poplatek, a to i při použití základní verze [7].

6.2.3 Serna

Tento WYSIWYG editor také splňuje vyslovené požadavky, i když ne sto procentně. Kontrola validity oproti XML schématu zde nechybí, zobrazení je také celkem zajímavé, ovšem styl, jakým XML data zobrazuje, není obecně rozšířená technologie jako u CSS stylů. Opět umožňuje porušení struktury dokumentu a navíc není zdarma, za zmínku snad ještě stojí podpora protokolu WebDAV [10].

6.2.4 XML Spy

XML Spy patří k nejznámějším profesionálním nástrojům. Podporuje validování oproti XML schématu, podporu CSS stylů a mnoho dalšího. Silnou stránkou nástroje je uživatelské rozhraní, kde uživatelé mohou editovat soubory v nejrůznějších pohledech – tabulková forma, textové zobrazení a podobně. Na první pohled se jedná o nástroj s velkým počtem funkcí a vylepšení, na druhou stranu opět umožňuje porušení XML struktury, není zdarma a navíc zde není možné připojení pomocí WebDAV protokolu [12].

6.2.5 Další vhodné XML editory pro úpravu XML dat

Editorů pro úpravu XML dokumentů je opravdu celá řada, kromě výše zmíněných můžeme vzpomenout ještě Vex, XMetal, XML Writer atd. Všechny tyto nástroje mají své silné stránky, příjemné uživatelské prostředí, ale také některé nedostatky, které by práci pro naše účely mohly v porovnání s ostatními trochu ztěžovat.

6.3 Shrnutí

Po té, co jsme si stanovili požadavky pro zobrazení XML dokumentů a seznámili se s několika editory, se nabízí jejich srovnání a výběr toho nejlepšího. Je jisté, že každý z výše zmíněných má své světlé i stinné stránky, a proto prohlásit o jediném, že je zaručeně nejsprávnější, by nebylo optimální. Pokud bychom uvažovali o nutné podmínce podporu WebDAV protokolu, pak editory, které toto splňují jsou Oxygen, XMLmind (profesionální verze), popř. editor Serna. Ani jeden není k dispozici zdarma, takže roli při případném rozhodování by určitě hrála i cena a s ní spojené další dovednosti programů.

Pokud otázku podpory WebDAV necháme nevyřčenou, možnosti výběru se podstatně rozšiřují. Z mého pohledu se jako nejlepší varianta jeví použití editoru XMLmind, který je volně dostupný, požadavkům plně vyhovuje a navíc jako jediný dokáže plně ohlídat správnou strukturu XML dokumentu. A to je v případě, kdy XML soubor přebíráme z databáze již vytvořený a nemáme moc možností hlídat jeho strukturu, velmi podstatné.

Kapitola 7

Implementace systému

Doposud jsme na základě požadavků vyřčených v úvodu hledali a prověřovali nejvhodnější způsoby zobrazení, zpracování a modifikace dat. Ačkoli nemůžeme jednoznačně určit, která technologie nebo způsob je v jednotlivých částech správy nejvhodnější, přesto se v každé oblasti objevilo několik zajímavých možností, jejichž zkombinováním bychom mohli vytvořit systém odpovídající zadání. Zopakujme si tedy jednotlivé výsledky, ke kterým jsme po menším průzkumu v jednotlivých oblastech přišli.

1. Nejprve výběr protokolu. Možností je několik, za celkem slibné však považuji použití protokolu WebDAV. Byl vytvořen za účelem vzdálené správy dat, jeho podpora v ostatních aplikacích pomalu, ale jistě roste a i když nevyužijeme všechny jeho možnosti, ponechává nám otevřená vrátka pro případné rozšíření.

2. Následuje zobrazení konkrétních dat přímo z databáze. Již nyní se pro prezentaci těchto informací používá jazyk XML, který díky svým přednostem dává celkem zajímavé možnosti následného zpracování. Mám na mysli technologie a rozšíření, jež nám dovolí sofistikovanou úpravu a elegantní zobrazení dat. Editory, které za tímto účelem vznikly, se toho samozřejmě snaží co nejvíce využívat. Proto jsem provedla menší průzkum dostupných nástrojů a na základě požadavků na systém a možností editorů došla k následujícím závěrům. Pro optimální kontrolu dat a zamezení konfliktu s databází je ideální ke každému vytvořenému XML dokumentu doplnit další soubor, a to XML schéma. Na jeho základě řada editorů dokáže kontrolovat a našeptávat způsoby editace, což je velmi přínosné. Pro následnou práci s daty pak lze využít další z možných technologií, konkrétně CSS styly. Je to opět obecný způsob, kterým řada editorů dokáže data zobrazit, a tím uživateli zpřehlednit práci.

Z obecného zadání, tedy zpřístupnění dat, vznikly konkrétní požadavky na vlastní implementaci. Cílem je navrhnout a implementovat nástroj, který dostane data ze surové formy v databázi do podoby XML dokumentů, navíc ke každému dokumentu dotvoří XML schéma a CSS styl a výsledné virtuální soubory zpřístupní uživateli pomocí WebDAV protokolu, a to nejenom pro čtení, ale i pro zpětný zápis s promítnutím do databáze. Tímto způsobem pak vhodným editorem bude možné data zpracovat a opět pomocí vybraného protokolu data modifikovat. Jedním z dalších požadavků je vhodné zobrazení struktury takto získaných dat, aby celá práce byla nejen pohodlná, ale také přehledná.

7.1 Datová vrstva

Veškerá data, která mají být zpřístupněná k editaci, jsou určitým způsobem uložena v databázi. Primárním účelem tohoto systému je editace tzv. dlouhých textů, které byly původně uloženy pomocí jazyka XML v tabulce s názvem *Stranka_Text*. Tato tabulka měla za svůj primární klíč atribut *ID*, kterým byl text jednoznačně určen, dále atributy *Text_cs* a *Text_en*, ve kterých byly uloženy konkrétní jazykové verze textů, doplňující atributy jako *Text_cs_No*, *Text_en_No*, jež poskytovaly informaci, zda v daném jazyce text nikdy nebude, a v neposlední řadě atributy typu *Zmenil_kdo*, *Datum_zmeny*, uchovávající další informace o textech. Při změně textu vždy došlo k nahrazení vlastního textu (v závislosti na použitém jazyku se tak modifikoval obsah atributů *Text_cs* nebo *Text_en*) a pozměnily se další údaje jako datum poslední změny a podobně. Tento princip byl jednoduchý, avšak náročný na hlídání chyb, které se mohly během editace objevit. Uložil-li někdo špatně strukturovaný text, došlo k porušení celé stránky prezentace zobrazující tento text. Když naopak někdo udělal změnu, která sice strukturu neporušila, ale nebyla korektní (například z informačního pohledu), neměl možnost se vrátit k původní verzi. Text byl jednoduše přepsán novým, a uživateli nezbylo nic jiného, než chybu najít a ručně odstranit. Vzhledem k tomu, že nyní se okruh uživatelů editující text značně rozšíří, nebyl by tento způsob právě ideální.

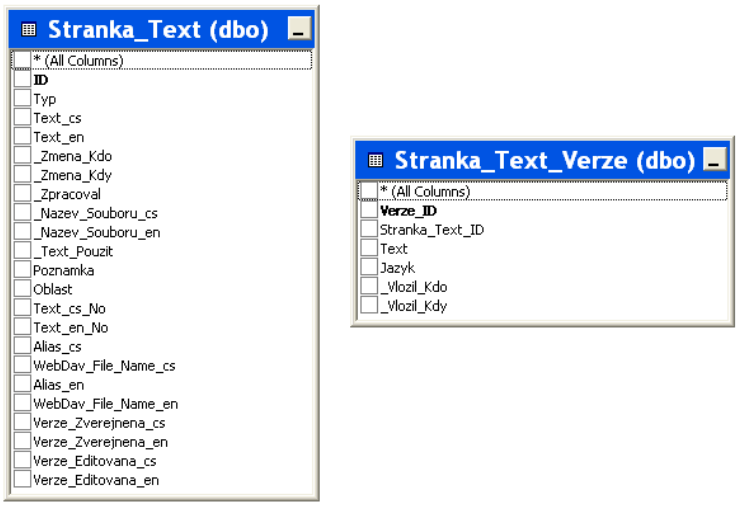
Ačkoli otázkou verzování se zabývá i protokol WebDAV, v rozsahu této práce nebudeme verzování skrze protokol WebDav implementovat. Postačíme si zatím jen s ukládáním verzí na straně serveru. Tím, že způsob ukládání textů pozměníme podle našich představ, snáze pohlídáme kompatibilitu v rámci databáze a zajistíme správný chod a přísun aktuálních dat pro další nástroje, které s touto tabulkou spolupracují.

7.1.1 Nová podoba tabulek

Kromě původní tabulky *Stranka_Text* budeme používat ještě tabulku *Stranka_Text_Verze*. Zde se budou ukládat editované texty – pod jedním ID bude vždy uložen pouze jeden text. Uživatel totiž při editaci skrze WebDAV v rámci virtuálního souboru pracuje jen s jednou jazykovou alternativou.

Do tabulky *Stranka_Text* přibudou atributy *Verze_Zverejnena_cs, Verze_Zverejnena_en* a dále *Verze_Editovana_cs, Verze_Editovana_en*. Při aktualizaci textu se neprovede skutečné nahrazení textu. Text se pouze uloží do tabulky *Stranka_Text_Verze* a vyplní se atributy *Verze_Editovana_cs*, případně *Verze_Editovana_en* (v závislosti na použitém jazyku textu). Tento krok byl nutný pro zajištění verzování, které doposud chybělo. O tom, který text je skutečně zobrazen na stránkách prezentace, vypovídá atribut *Verze_Zverejnena_en* a *Verze_Zverejnena_cs*, zbylé dva nově přidané atributy vypovídají o poslední verzi, která se editovala.

Protože textů určených k editaci je nepřeberné množství, bylo nutné zamyslet se nad jejich způsobem seskupení pro zajištění přehlednější správy dat. Jedním z nejintuitivnějších způsobů se nakonec ukázalo rozčlenění textů do příslušných oblastí. Jedná se o zažité členění, které uživatelé znají, a proto pro ně nebude problém se v této struktuře orientovat.



Obrázek 7.1: Tabulky určené k uchovávání dlouhých textů

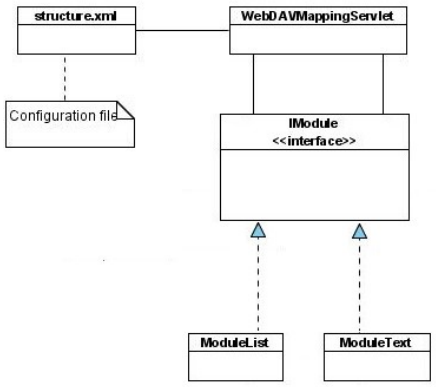
7.2 Aplikační vrstva systému

Zbývá popsat poslední část systému, jeho aplikační vrstvu. Tato vrstva zajišťuje funkcionalitu systému a provádí požadované operace nad daty. Pro komunikaci mezi serverem a klientem je použita již dříve zmíněná technologie *Java Servlets*. Kromě této komunikace je však zapotřebí zajistit i další, neméně důležité funkce systému.

Jednou ze základních operací, které musíme na této úrovni provést, je vytvoření virtuálních souborů z dat uložených v databázi. Je zřejmé, že způsob vytváření souborů obsahujících jednotlivé texty bude poněkud odlišný od budování samostatných souborů reprezentujících různé číselníky, přesto metody, které vzniknou, budou v principu podobné. Základní metody, které volá servlet k obsloužení požadavků, by se měly v obou případech lišit různými pohledy do databáze, výsledek poskytovaný servletu by však měl zůstat stejný. To znamená, že např. při zavolání metody *getChildrenNames(String folderUri)*, která má na základě URI dané složky vrátit její obsah (tzn. soupis jmen souborů, které obsahuje), by servlet vždy měl dostat seznam jmen virtuálních souborů, které se zde mají vytvořit. Podobných metod určených pro obsluhování požadavků protokolu WebDAV je několik. Pro tento případ bylo vybudováno rozhraní *IModule*, které všechny potřebné metody popisuje. Vzniklo několik tříd, které na základě účelu, pro který byly vytvořeny, tyto metody různým způsobem implementují. Třída *ModuleText* určená pro zpřístupnění dlouhých textů tak při zavolání výše zmíněné metody vrátí seznam virtuálních XML souborů (případně zastřešený složkami zajišťujícími jejich přehlednost), třída *ModuleList*, sloužící k zobrazení celé tabulky (typicky číselníků), vrátí jen jeden virtuální XML soubor s daty. Na řadu přichází další otázka, jakým způsobem zajistit mapování těchto tříd na příslušné metody protokolu, tj. jak servlet pozná, kterou třídu má v jakém okamžiku zavolat.

7.2.1 Princip mapování tříd

Protože stromová adresářová struktura se dá velmi elegantním způsobem simulovat pomocí XML dokumentu, byl za tímto účelem definován konfigurační XML soubor, v němž lze popsat statickou strukturu souborového systému včetně mapování modulů na konkrétní adresáře. Jejich obsah je pak již popisován dynamicky samotnými moduly (*getChildrenNames*, ...). WebDAV protokol nepracuje s adresáři úplně běžným způsobem. Jelikož soubory, které vznikají, jsou pouze virtuální, i složky, ve kterých se soubory nachází, nejsou nikde fakticky vytvořené. Zavádí se pojem kolekce, která k adresářům přistupuje pomocí jejich jednoznačných URI. Začátek této URI odpovídá stromové struktuře názvů elementů výše zmíněného XML dokumentu a může pokračovat dalšími názvy, které vzniknou pomocí metod příslušné namapované třídy.



Obrázek 7.2: Diagram znázorňující použité třídy a konfigurační soubor

7.2.2 Konfigurační soubor pro popis struktury

Konfigurační soubor **structure.xml** byl vytvořen pro vnitřní potřeby servletu. Je využíván při vytváření statické struktury, jednotlivým třídám navíc poskytuje další informace, které pro své metody potřebují.

```
<root>
<structure>
  <folder name="w3mu">

    <ref_folder name="texty"
       class="jfx.app.webdav.WebdavMappingServlet">
      <param name="module" value="jfx.app.webdav.ModuleText"/>
    </ref_folder>
```

```
    <ref_folder name="pracoviste_soucasti"
       class="jfx.app.webdav.WebdavMappingServlet">
      <param name="module" value="jfx.app.webdav.ModuleList"/>
      <param name="tableName"
       value="pracoviste_soucasti-Pracoviště součásti"/>
      <param name="table" value="people..Pracoviste_Soucasti"/>
      <param name="columns"
       value="Zkratka-1,LP_Nazev_cs-6,LP_Nazev_en-7,
       Zkratka_cs-9,Zkratka_en-10,ICO-11,DIC-12"/>
      <param name="key-columns" value="Zkratka-1"/>
      <param name="connection"
       value="jfx.security.DBConnection.getW3muConnection()"/>
    </ref_folder>

  </folder>
</structure>
</root>
```

- Jedním z hlavních úkolů konfiguračního souboru je simulace adresářové struktury. Servlet prochází postupně elementy na základě URI obdrženého z požadavku. Narazí-li na speciální element označený *ref_folder*, předá řízení třídě, jejíž název přečte z atributu, a dále ve výstavbě nepokračuje; o další generování struktury se již postará volaná třída. Pokud však na tento element nenarazí, přečte jména jeho potomků a ty prohlásí na složky nižší úrovně.

- Protože potomci elementu *ref_folder* již nejsou pro budování struktury použití, můžeme je naplnit parametry, které využije třída mající kontrolu. Jedná se především o názvy tabulek, popisků využitých při generování CSS stylů, klíčů tabulka nebo sloupců určených k editování.

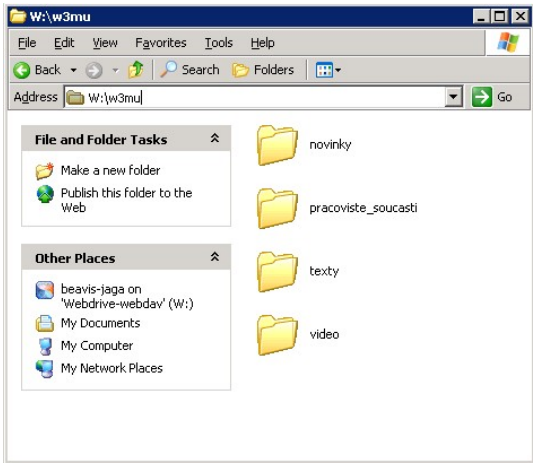
7.2.3 Rozhraní IModule

Toto rozhraní obsahuje základní metody, které jsou potřebné k obsloužení jednotlivých příkazů protokolu WebDAV. O konkrétní způsob jejich provedení se starají třídy *ModuleText* a *ModuleList*.

- **begin(Hashtable parameters)** – provádí načítání vstupních parametrů z konfiguračního souboru – názvu tabulky, klíčů, ...

- **isFolder(String uri)** – určuje, zda se jedná o adresář

- **isResource(String uri)** – určuje, zda se jedná o soubor



Obrázek 7.3: Statická struktura vytvořená na základě konfiguračního souboru

- **String returnKeys()**– vrací klíče tabulky
- **readOnly(final String uri)** – udává, zda je soubor určený k modifikaci, nebo jestli se jedná pouze o doplňkový soubor (např. XML schéma číselníku, ...)
- **setResourceContent(String resourceUri, InputStream content)** – provádí update tabulky a vrací informaci, jestli se jej povedlo provést
- **getChildrenNames(String folderUri)** – vrací seznam jmen souborů, které obsahuje kolekce identifikovaná parametrem folderUri
- **getResourceContent(String resourceUri)** – vrací obsah souboru
- **getResourceLength(String resourceUri)**– vrací délku souboru
- **getLastModified(String uri)** – vrací datum poslední modifikace souboru

7.2.4 ModuleText – třída určená pro zpřístupnění textů

Pomocí této třídy vznikají virtuální soubory obsahující jednotlivé texty.
Zde je přehled nejdůležitějších metod:

- **getChildrenNames(String uri)** – tato metoda zodpovídá za tvorbu adresářové struktury. Jelikož virtuálních souborů obsahující texty je nepřeberné množství, třídí jejich umístění do podsložek, které reprezentují tzv. oblasti. Kromě podsložek a názvů virtuálních souborů definuje i názvy dalších souborů (CSS styly, XML schémata, ...).

- **getResourceContent(String uri)** – pomocí této metody dostaneme obsah virtuálního souboru.

  – V případě vytváření virtuálních souborů pro editaci textů je použit select do databáze. Obdržený text je modifikován přidáním upravené hlavičky, ve které je definováno kódování, použití CSS stylu a XML schématu.

  – Obsah virtuálního souboru reprezentující CSS styl, popř. XML schéma je proveden načtením příslušného externě uloženého souboru, který pro tento účel vznikl.

- **setResourceContent(String uri, InputStream is)** – poslední stěžejní metoda odpovídá za update v případě modifikace textu následujícím způsobem:

  1. zajistí odstranění přidané hlavičky
  2. provede update/insert
  3. sdělí servletu, zda vše proběhlo korektně, nebo jestli nastaly komplikace a k updatu nedošlo

  Vytvořené virtuální soubory textů lze pak s využitím připojeného CSS stylu pohodlně zobrazovat ve vybraném editoru; pomoc při editaci zajistí XML schéma, které popisuje základní použití XML elementů určených pro tvorbu prezentace. Ukázka v PřílozeC

7.2.4.1 Verzování

Pro zajištění verzování bylo zapotřebí poupravit tabulky uchovávající texty. Vznikla samostatná tabulka pro editované texty, původní byla rozšířena atributy nesoucí informaci o textu skutečně zveřejněným a o textu naposledy editovaném. Při updatu textu se neprovede přímé nahrazení textu, pouze se nastaví hodnota jeho *ID* jako hodnota atributu *Verze_Editovana_cs*, případně *Verze_Editovana_en*. Opravdové zveřejnění textu se provede až v dalším kroku.

Pro zajištění uceleného zobrazení textu ještě před konečným zveřejněním na stránkách prezentace bylo vhodné navrhnout mechanismus, který by toto zajišťoval. Vznikl testovací web, který zobrazuje nikoli verze zveřejněné, ale editované. Na konci této zobrazované stránky je tlačítko, jehož stisknutím se provede zveřejnění zobrazovaného textu na stránkách prezentace.



Obrázek 7.4: Ukázka způsobu zobrazení nezveřejněného textu

Proces editace textu:

1. Pomocí vhodného klienta si text zobrazíme v příslušném editoru.

2. Provedeme změny a výsledek uložíme.

3. Změněný text si prohlédneme na testovacím webu a pokud se změnami souhlasíme, potvrdíme stisknutím tlačítka na konci stránky jejich zveřejnění.

4. Text je nahrazen a bude zobrazen při otevření příslušné stránky.

7.2.5 ModuleList – třída určená k editaci číselníků

Třída *ModuleList* buduje virtuální XML soubory obsahující celé číselníky. Díky parametrům uvedených v konfiguračním dokumentu nejprve určí název tabulky, její klíče, sloupce určené k editování, popř. připojení do databáze, jež má být použito, a na základě těchto informací provádí jednotlivé metody. Zároveň pro každý číselník automaticky vygeneruje jeho XML schéma za použití metainformací o tabulce a CSS styl vytvořený z předpřipravené šablony a doplněný o konkrétní názvy popisků (viz Příloha B).

7.2.5.1 Vytvoření nového virtuálního souboru z číselníku

Chceme-li vytvořit nový virtuální soubor sloužící k editaci některého z číselníků, zapíšeme do konfiguračního souboru název složky, ve které se bude dokument nacházet, a doplníme příslušné hodnoty parametrů. Virtuální XML dokument se pak automaticky vygeneruje při otevření dané složky spolu s jeho XML schématem a CSS stylem. Jednotlivými elementy XML souboru jsou řádky tabulky, ze které je dokument generován. Kromě těchto hodnot obsahuje každý element i speciálního potomka *<element_backup/>*, ve kterém se nachází původní hodnoty všech editovaných dat. Tento záložní element může sloužit k porovnání změn při editaci, zároveň je využit při updatu do databáze. Zajišťuje totiž to, aby byly aktualizovány pouze hodnoty, které se skutečně změnily. V současné době je možné nejen hodnoty editovat, ale i přidávat celé řádky tabulky (klasické vložení). Názorný příklad vygenerovaných souborů naleznete v Příloze A.

7.2.6 Servlet – metody implementující příkazy protokolu WebDAV

- **doPropfind** – Jak název napovídá, zavoláním této metody obsloužíme požadavek PROPFIND pro výpis vlastností. Na základě URI získané z parametru požadavku nejprve rozhodneme, zda bude pokračovat servlet sám, nebo předá řízení některému z modulů. V případě, že se má úkolu ujmout sám, prochází konfiguračním souborem popisujícím statickou strukturu a vypíše potomky elementu, jehož název se shoduje s poslední částí URI. Tyto potomky pošle spolu s dalšími parametry (typ, velikost) jako odpověď. V případě, že se má obrátit na některý modul, učiní to zavoláním příslušné metody. Obdrží opět seznam názvů a zpracuje ho stejným způsobem.

- **doGet** – Implementace příkazu GET pro získání souboru. Z proudu bytů obdrženého zavoláním příslušné metody modulu, které bylo předáno řízení, vytvoří virtuální soubor, opět nastaví hodnoty jeho parametrů a pošle je jako odpověď.

- **doPut** – Slouží jako reakce na příkaz PUT. Znovu se jedná o metodu pracující s konkrétními daty, tentokrát však data z databáze nedostáváme, ale formou updatu je tam ukládáme. Zavolá se tedy metoda, která v příslušném modulu provádí update/insert do dané tabulky a v závislosti úspěchu či neúspěchu nastaví požadované hodnoty vlastností.

- **doOption** – Tato metoda slouží k nastavení vlastností určených pro obsluhu (například seznam implemetovaných metod).

7.2.7 Další použité metody a mechanismy

- Hlavních metodou servletu je metoda **service()**, která ověří oprávněnost uživatele, v případě, že bude nutné předat řízení některému z odpovědných modulů, zajistí jeho inicializaci, rozpozná příkaz a zavolá konkrétní metodu pro jeho obsluhu.

- Inicializace třídy řeší metoda **initMap()**. Jak jsem uvedla již dříve, předání řízení znamená to, že při procházení konfiguračním dokumentem narazíme na speciální element *<ref_folder/>*. Tento element má parametr s uvedením třídy, které má být řízení předáno, avšak může obsahovat i další informace. Využijeme toho, že při hledání další adresářové úrovně už neprocházíme strukturou dál, ale voláme příslušnou metodu třídy. Potomky tohoto elementu tak můžeme naplnit libovolným způsobem a využít je právě při inicializaci. Takto můžeme třídě předat informace o tabulce, jejich klíčích, jmen sloupců, které chceme editovat, a podobně.

- Další zajímavou metodou je metoda **xmlProperties()**, jež generuje data, jež jsou formou XML dokumentu připojená k odpovědi serveru.

7.2.7.1 Určování velikosti souborů

Jedním z oříšků, které se při implementaci protokolu objevily, bylo určení velikosti souborů. Určit velikost souboru již při pouhém procházení názvů se zpočátku zdálo zbytečné, znamenalo to nejen zjištění jména, ale i velikosti obsahu, který by se v případě zavolání příkazu GET načetl. Ukázalo se však, že bez této informace někteří klienti nebudou pracovat korektně. Z neznalosti nebo špatného udání velikosti pramenila řada problémů – špatné rozpoznávání druhu souboru (jedná-li se o složku či vytvořený dokument) nebo neúplné načítání virtuálního souboru. Jelikož chceme vytvořit systém, který bude moci využívat více způsobů zobrazení, což znamená i nezávislost na konkrétním druhu klienta nebo editoru, bylo nutné toto ošetřit. Vznikla keš, do které se postupně ukládají velikosti souborů. Zjištění velikosti musí být provedeno určením velikosti obsahu i v případě, že se jedná o pouhý výpis adresáře. Systém to nakonec nijak výrazně nezpomalilo, určení velikosti se v podstatě provede jen jednou, pak už je pouze čteno z keše, a případně se upraví při updatu souboru.

7.2.8 Autentizace

Jednou z věcí, o které zatím nebyla v práci zmínka, je autentizace. WebDAV protokol je nadstavba HTTP protokolu, tudíž v případě potřeby můžeme využít např. *HTTP basic autentizaci* pro zabezpečení přístupu. Ač je práce implementovaná v prostředí Inetu, který má vlastní autentizaci, není možné ji z bezpečnostního důvodu využít. Hrozí totiž riziko, že v případě mapovaní WebDAV struktury např. jako virtuální disk v OS Windows uživatelé použijí zapamatování hesla – což bývá u nezanedbatelného procenta uživatelů bohužel zvykem. Implementovala jsem tedy již zmíněnou HTTP basic autentizaci až na úrovni servletu – pro aplikační server se servlet jeví jako veřejný. Pro uložení přihlašovacích jmén a hesel byla vytvořena tabulka, do níž se ukládají hesla šifrovaně. Samotné zadávání hesla uživatelem bude v dohledné době zpřístupněno pomocí aplikace v systému Inet.

Kapitola 8

Závěr

8.1 Zhodnocení práce

Hlavním cílem této práce bylo vytvoření systému správy dat potřebných pro chod webové prezentace www.muni.cz. Požadavky, které byly v úvodu vyřčeny, byly splněny následovně:

- Na základě analýzy požadavků na zpřístupnění dat a průzkumu existujících protokolů byl k implementaci vybrán protokol WebDAV. Ten nejvíce vyhovoval všem kritériím, navíc se ukázal být dobrým základem i pro případné rozšíření systému.

- Vybraný protokol byl implementovaný pomocí technologie Java Servlets. Mimo to byly použity i mechanismy, které výsledný program obohatily o další funkcionalitu i grafický efekt.

- Po srovnání existujících editorů pro správu XML dokumentů byla vybrána skupina programů, která využívá příjemné grafické rozhraní, hlídá validnost dokumentu, popř. našeptává možnosti editace.

- Z těchto editorů byl doporučen XMLmind. Ten jako jeden z mála volně dostupných programů zabraňuje porušení struktury XML souborů. Uživateli nedává možnost zasahovat do konstrukce tagů, čímž znemožní umazání jeho částí nebo nekorektní uzavření.

- Tabulky uchovávající texty byly upraveny pro možnost verzování. Zároveň byl navržen způsob, který uživateli umožní pohled na ještě nezveřejněnou verzi tak, jak bude zobrazena po zveřejnění.

Implementovaný způsob zpřístupnění dat zahrnuje i další vlastnosti:

- Je možné editovat libovolnou tabulku. Po zadání jejího jména, názvů sloupců určených k editování, klíčů, popř. popisků a druhu připojení jsme schopni vytvořit virtuální XML soubor včetně jeho XML schéma a CSS stylu potřebného k pohodlnému zobrazení.

- Výběr WebDAV klientů, co by nástrojů k zpřístupnění dat, umožňuje mimo jiné intuitivní zacházení se soubory, na které jsme zvyklí při práci s externím diskem.
- Principy zpřístupnění i zobrazení dat jsou postaveny na technologiích používaných řadou programů a nástrojů plnící požadované úkoly. Proto je systém nezávislý na konkrétním výběru jednoho programu, v rámci práce je pouze několik doporučení na nástroje, které se nejvíce osvědčily.

8.2 Možnosti rozšíření

Základní verze systému se zabývala především vhodným zpřístupněním dat pro jejich editaci. Díky výběru vhodného protokolu poskytující celou škálu vlastností, jež jsme zatím nevyužili, i díky použitým technologiím, které nejsou vázány k omezenému typu editačních nástrojů, můžeme systém časem rozšířit o další funkce, například:

- Přístupová práva – bylo by dobré zajistit, aby různí uživatelé měli v závislosti na svých pravomocech přístup pouze k určitým datům. Tento autorizační mechanismus by měl být zajištěn na úrovni servletu například tím, že by uživatelům byly přiřazeny role. Na základě těchto rolí by pak byly generovány pouze složky s povolenými daty.

- Přejmenovaní souborů – v souvislosti se zpřístupněním velkého počtu textů určených k editaci se objevila otázka, jak tyto virtuální soubory co nejpohodlněji spravovat. Jednou z možností je právě přejmenování názvů. Uživatel by si tak místo standardního označení (ID textu + jazyk) mohl zvolit alias (což by byla jen implemetace příslušné metody WebDAVu pro přejmenování souborů) skutečně prostým přejmenováním souboru. Podpora v databázi byla pro tuto operaci zabudována už při její modifikaci pro umožnění verzování, takže nic nebrání toto vylepšení implementovat.

- Možnost editace rozsáhlejších číselníků rozdělením do více datových souborů po určitém počtu řádků. V případě, že by číselník obsahoval velké množství řádků, by toto vylepšení zajistilo lepší orientaci v datech.

- Podpůrné aplikace v Inetu – pro ucelení informací o zpřístupňovaných datech by bylo dobré přidat aplikace, které by sloužily například k zobrazování historie změn souboru, celkovému přehledu souborů i s metadaty a podobně.

Literatura

[1] Kosek, J.: *XML pro každého*, Grada Publishing, 2000, 80-7169-860-1, <http://www.grada.cz>. 2

[2] Cascading Style Sheets: , <http://cs.wikipedia.org/wiki/Cascading_Style_Sheets>. 2.1.1.1

[3] Document Type Definition: , <http://cs.wikipedia.org/wiki/DTD>. 2.1.2.1

[4] Kohoutková, J. a Měcháček, J.: *Intranetový server Informačního systému MU*, Zpravodaj ÚVT MU, 2000, <http://www.ics.muni.cz/zpravodaj/articles/194.html>. 3

[5] XMLmind: *stránky produktu*, <http://xmlmind.com/>. 6.2.1

[6] NetDrive: *stránky produktu*, <http://www.novell.com> <http://www.stahuj.centrum.cz/internet_a_site/ftp_klienti/novell-netdrive>. 5.1.1

[7] Oxygen: *stránky produktu*, <http://www.oxygensoftware.com>. 6.2.2

[8] Kohoutková, J.: *Masarykova univerzita na internetových WWW stránkách*, Zpravodaj ÚVT MU, 1997, <http://www.ics.muni.cz/zpravodaj/articles/90.html>. 1.1.1

[9] Kosek, J.: *XML schémata*, 2005, <www.kosek.cz/xml/schema/index.html>. 2.1.2.2

[10] Serna: *stránky produktu*, <http://www.syntext.com/>. 6.2.3

[11] Ocelka, J.: *Serverová infrastruktura informačních systémů MU*, Zpravodaj ÚVT MU, 2005, <http://www.ics.muni.cz/zpravodaj/articles/333.html>. 3

[12] XML Spy: *stránky produktu*, <http://www.altova.com/>. 6.2.4

[13] Kohoutková, J.: *Webové systémy*, 25 let ÚVT MU v Brně - Systémy, 2004, <http://www.ics.muni.cz/25let/systemy/weby.html>. 1.1.1

[14] wikipedia: *WebDAV*, <http://www.webdav.org>. 3.2

[15] WebDrive: *stránky produktu*, <http://www.novell.com>. 5.1.2

[16] Ocelka, J.: *www.muni.cz ve verzi 2006*, Zpravodaj ÚVT MU, 2006, <http://www.ics.muni.cz/zpravodaj/articles/533.html>. 1.1.2

[17] Ocelka, J.: *www.muni.cz ve verzi 2006(2)*, Zpravodaj ÚVT MU, 2006, <http://www.ics.muni.cz/zpravodaj/articles/539.html>. 1.1.2

[18] Vochozka, J.: *Značkovací jazyky a XML*, Zpravodaj ÚVT MU, 2000, <http://www.ics.muni.cz/zpravodaj/articles/201.html>. 2.1

[19] Pitner, T.: *Studijní materiály k předmětu Moderní značkovací jazyky*, <http://is.muni.cz/el/1433/podzim2005/PB162/>. 2

[20] wikipedia: *Transformace XSLT*, <http://cs.wikipedia.org/wiki/XSLT> <http://www.sovavsiti.cz/2003/xslt-1.html>. 2.1.1.2

Rejstřík

CSS, 7

DTD, 7

FTP, 9

HTTP, 15
HTTP 1.1, 10

Inicializace, 32

Klient, 17
Konfigurační soubor, 26

logická struktura, 6

SCP, 10
Servlet, 14
SFTP, 10
SGML, 6

tag, 6
Transformace XSLT, 7

Virtuální disk, 10

WebDAV, 10
WYSIWYG editory, 19

XML, 6
XML Schema Definition, 8
XSL, 7

značkovací jazyky, 6

Příloha A

Ukázka dokumentů vygenerovaných pro editaci číselníku Videa

XML dokumentobsahující data:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<?xml-stylesheet href="video.css" type="text/css"?>
<table xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
 xs:noNamespaceSchemaLocation="video.xsd">
  <element id_inter="1">
    <element_backup ID="1" Nazev_cs="Dies academicus 2006"
     Nazev_en="Dies Academicus 2006" />
    <ID>1</ID>
    <Nazev_cs>Dies academicus 2006</Nazev_cs>
    <Nazev_en>Dies Academicus 2006</Nazev_en>
  </element>
  <element id_inter="2">
    <element_backup ID="2" Nazev_cs="Slavnostní otevření budov ILBIT"
     Nazev_en="ILBIT Buildings Inauguration" />
    <ID>2</ID>
    <Nazev_cs>Slavnostní otevření budov ILBIT</Nazev_cs>
    <Nazev_en>ILBIT Buildings Inauguration</Nazev_en>
  </element>
  <element id_inter="3">
    <element_backup ID="3" Nazev_cs="Inagurace rektora Petra Fialy"
     Nazev_en="Petr Fiala Rector's Inauguration" />
    <ID>3</ID>
    <Nazev_cs>Inagurace rektora Petra Fialy</Nazev_cs>
    <Nazev_en>Petr Fiala Rector's Inauguration</Nazev_en>
  </element>
  </table>
```

XML schéma vytvořené na základě metadat tabulky:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="element">
  <xs:complexType><xs:all>
    <xs:element name="element_backup" minOccurs="0">
    <xs:complexType>
      <xs:attribute name="ID" type="xs:string" />
```

```
      <xs:attribute name="Nazev_cs" type="xs:string" />
      <xs:attribute name="Nazev_en" type="xs:string" />
    </xs:complexType>
    </xs:element>
    <xs:element name="ID" minOccurs="0" />
    <xs:element name="Nazev_cs" minOccurs="0">
      <xs:simpleType><xs:restriction base="xs:string">
        <xs:maxLength value="100" />
      </xs:restriction></xs:simpleType>
    </xs:element>
    <xs:element name="Nazev_en" minOccurs="0">
      <xs:simpleType><xs:restriction base="xs:string">
        <xs:maxLength value="100" />
      </xs:restriction></xs:simpleType>
    </xs:element>
  </xs:all>
      <xs:attribute name="id_inter" type="xs:positiveInteger" />
  </xs:complexType></xs:element>
  <xs:element name="table">
    <xs:complexType><xs:sequence minOccurs="0" maxOccurs="unbounded"
    <xs:element ref="element" /></xs:sequence>
    </xs:complexType>
  </xs:element>
  </xs:schema>
```

CSS styl doplněný o konkrétní popisky:

```
@charset "iso-8859-2";
table{display:block;margin:1em;}
table:before{display:block;border: 0.1em solid rgb(37,41,84);padding: 0.3em;
font-size:large;font-weight:bold;color:rgb(37,41,84);background-color:inherit;
content:"Video" ;}
element{display:block;margin: 1em;border:0.1em solid #DDDDEE;
padding:0.5em;color:inherit;background-color: #EFEFEF;}
element:before{display:block;padding-top:0.3em;padding-bottom:0.3em;
text-decoration:underline;font-weight:bold;color:inherit;
background-color:#EEFFEE; content:"Položka" ;color: #404e6f;}
*:before{color:black;background-color:inherit;font-family:monospace;
font-style:normal;font-weight:bold;}
element_backup:before{content:"*";font-size: small;color: rgb(0,51,102);}
Nazev_cs{display:block;}Nazev_cs:before{font-weight:bold;content:"Nazev_cs:  ";}
Nazev_en{display:block;}Nazev_en:before{font-weight:bold;content:"Nazev_en:  ";}
ID{display:block;}ID:before{color: #404e6f;font-weight:bold;
font-style:italic;content:"(nelze nastavovat) ID: ";}
```

Příloha B

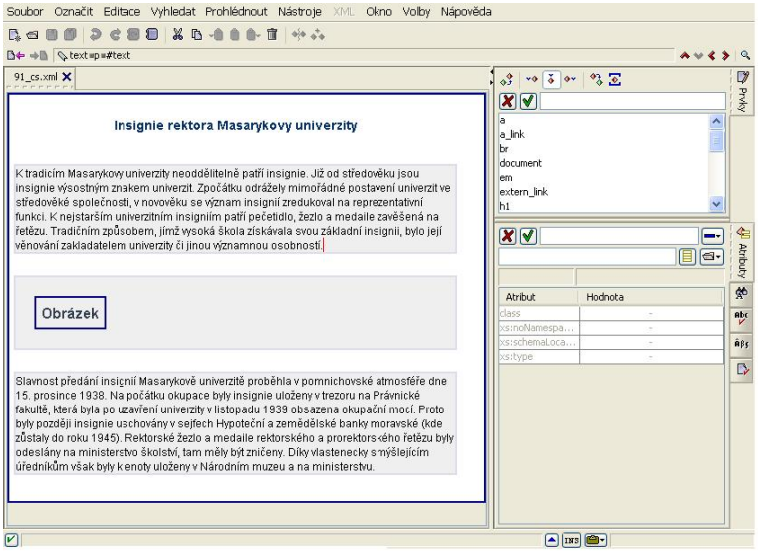
Šablona generování CSS stylů pro jednotlivé číselníky

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd>
  <table>table{display:block;margin:1em;}</table>
  <table_before>table:before{display:block;border: 0.1em solid rgb(37,41,84);
   padding: 0.3em;font-size:large;font-weight:bold;color:rgb(37,41,84);
   background-color:inherit;}
  </table_before>
  <element>element{display:block;margin: 1em;border:0.1em solid #DDDDEE;
   padding:0.5em;color:inherit;background-color: #EFEFEF;}
  </element>
  <element_before>element:before{display:block;padding-top:0.3em;
   padding-bottom:0.3em;text-decoration:underline;font-weight:bold;
   color:inherit;background-color:#EEFFEE;}</element_before>
  <all_before>*:before{color:black;background-color:inherit;
   font-family:monospace;font-style:normal;font-weight:bold;}
   element_backup:before{content:"*";font-size: small;
   color: rgb(0,51,102);}
  </all_before>
</xsd>
```

Příloha C

Text v podobě XML dokumentu

```
<?xml version="1.0" encoding="iso-8859-2"?>
<?xml-stylesheet type="text/css" href="css_styly.css"?>
  <text xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
     xs:noNamespaceSchemaLocation="texty_schema.xsd">
    <h2>Insignie rektora Masarykovy univerzity</h2>
    <p>K tradicím Masarykovy univerzity neoddělitelně patří insignie.
    Již od středověku  jsou insignie výsostným znakem univerzit.
    Zpočátku odrážely mimořádné postavení univerzit ve středověké společnosti,
    v novověku se význam insignií zredukoval na reprezentativní funkci.
    K nejstarším univerzitním insigniím patří pečetidlo, žezlo
    a medaile zavěšená na řetězu. Tradičním způsobem,
    jímž vysoká škola získávala svou základní insignii,
    bylo její věnování zakladatelem univerzity
    či jinou významnou osobností. </p>
    <p><image id="2752" db_size="n1" zoom="true" href_size="h"/></p>
    <p>Slavnost předání insignií Masarykově univerzitě proběhla
    v pomnichovské atmosféře dne 15. prosince 1938.
    Na počátku okupace byly insignie uloženy v trezoru na
    Právnické fakultě, která byla po uzavření univerzity
    v listopadu 1939 obsazena okupační mocí.
    Proto byly později insignie uschovány v sejfech Hypoteční
    a zemědělské banky moravské (kde zůstaly do roku 1945).
    Rektorské žezlo a medaile rektorského a prorektorského řetězu
    byly odeslány na ministerstvo školství, tam měly být zničeny.
    Díky vlastenecky smýšlejícím úředníkům však byly klenoty
    uloženy v Národním muzeu a na ministerstvu.</p>
  </text>
```



Obrázek C.1: Ukázka textu zobrazeného pomocí speciálního CSS stylu

Příloha D

Obsah přiloženého média

Součástí práce je i přiložené CD, kde jsou k dispozici následující soubory:

- Zdrojový kód této práce ve formátu XML, PDF a příslušné obrázky a přílohy
- Zdrojový kód aplikace včetně šablon a konfiguračních dokumentů
- Ukázka vygenerovaných souborů