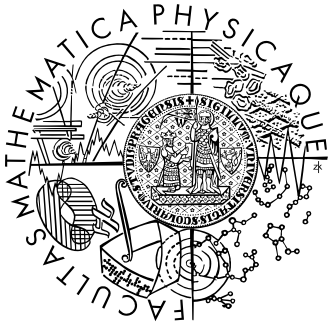
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Petr Vávro

Šablonovací systém pro generování textových dokumentů

Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Michal Kopecký, Ph.D.

Studijní program: Informatika, Programování

2010

Rád bych vyjádřil své poděkování vedoucímu bakalářské práce RNDr. Michalu Kopeckému, Ph.D za rady, inspiraci a trpělivost. Firmě Triada, spol. s r. o děkuji za možnost pracovat na tomto projektu. Díky patří i kolegům ze jmenované firmy za jejich podporu, rady a návrhy. Rovněž bych rád poděkoval své rodině a přítelkyni za podporu při studiu a trpělivost.

Prohlašuji, že jsem svou bakalářskou práci napsal(a) samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 5.8.2010 Petr Vávro

Obsah

Název práce: Šablonovací systém pro generování textových dokumentů

Autor: *Petr Vávro*

Katedra (ústav): *Katedra softwarového inženýrství*

Vedoucí bakalářské práce: *RNDr. Michal Kopecký, Ph.D.*

e-mail vedoucího: *michal.kopecky@mff.cuni.cz*

Abstrakt: *Tato práce popisuje návrh a implementaci třívrstvého systému generování textových dokumentů ze šablon a dodaných dat. Systém bude zapojen do informačního systému Munis společnosti Triada. Práce výraznou měrou rozšiřuje možnosti, poskytované komponentou Aspose.Words společnosti Aspose. Šablony mohou obsahovat výrazy vyhodnocované během vkládaní dat, konstrukce master-detail i více jak jedné úrovně a podmíněně zobrazené sekce.*

Klíčová slova: *šablony, třívrstvý systém, generování textových dokumentů*

Title: *Template system for generating text documents*

Author: *Petr Vávro*

Department: *Departement of Software Engineering*

Supervisor: *RNDr. Michal Kopecký, Ph.D.*

Supervisor's e-mail address: *michal.kopecky@mff.cuni.cz*

Abstract: *This thesis aims on design and implementation of a three-tier system for generating text documents from templates and supplied data. The system is going to be a part of the information system Munis developed by Triada. The thesis significantly extends the possibilities, provided by the component Aspose.Words by Aspose, Templates may contain expressions evaluated at time of data insertion, master-detail constructs of arbitrary level as well as conditionally shown text blocks.*

Keywords: *templates, three-tier system, generating text documents*

1 Úvod

V dnešní době je úspěch velkých společností a vyspělých zemí založen nejen na hmotných výrobcích, ale také na informacích a znalostech, obecněji na datech. Takováto data jsou obvykle rozsáhlá, mají složité vazby a jednotlivé entity mají obvykle jasný význam. Pro organizaci těchto dat a jejich zpracování se obvykle používají informační systémy. Informační systémy mají za úkol tato data uchovávat, zpracovávat případně z nich vyvozovat důsledky či doporučení. Potřebná data musí rovněž ve vhodném okamžiku a ve vhodné podobě prezentovat a pomáhat tak mj. s řízením firemních procesů.

Aby byl takovýto informační systém co nejdynamičtější, je často složen z mnoha navzájem komunikujících modulů. Při podpoře firemních procesů se tak nelze vyhnout potřebě sdílet sebraná data jak mezi moduly informačního systému, tak s dalšími informačními systémy obchodních partnerů, případně je prezentovat dalším lidem a organizacím. Data musejí být sdílena a prezentována ve srozumitelné formě, jako je tištěný, či elektronicky předaný textový dokument nebo strukturovaný XML dokument určený pro další strojové zpracování.

Práce se věnuje návrhu a implementaci modulu informačního systému, umožňujícího tvorbu textových dokumentů (sestav) z datových zdrojů na základě šablon. Druhá kapitola práce se zabývá analýzou stávajících řešení a požadavků zákazníka a existujícími řešeními na trhu. Ve třetí kapitole jsou shrnuty požadavky na návrh a implementaci tvorby textových dokumentů. Kapitola 4 popisuje návrh aplikace. Následující pátá kapitola obsahuje popis členění programu z hlediska implementace, nejdůležitější metody a popis, jak aplikaci případně dále upravovat a rozšiřovat. Šestá kapitola obsahuje uživatelskou příručku – popis instalace, tvorby šablon a uživatelského prostředí.

2 Analýza

Implementace generátoru sestav je určena pro pilotní nasazení u společnosti Triada[1], jež je dodavatelem řešení ucelených informačních systémů pro subjekty veřejné správy. Šablonovací systém pro generování dokumentů bude použit jako modul do nově navrhované architektury informačního systému *Munis*. V nynější dvouvrstvé architektuře tohoto systému sdílejí aplikace, implementované ve vývojovém prostředí Delphi[2], společnou databázovou vrstvu. Další vývojovou iterací architektury má být třívrstvý systém, který bude obsahovat pouze tenké klientské aplikace, napojené na centrální aplikační server, který zprostředkuje veškeré důležité funkce. Tento aplikační server bude vyvíjen na platformě .NET.

2.1 Stávající řešení

Stávající řešení tvorby dokumentů není v informačním systému jednotné. V některých komponentách je založeno na (OLE) Automation[3] s aplikací Microsoft Word, jinde je využito parsování textového souborového formátu RTF[4]. Další používanou variantou je použití generátoru tiskových sestav Sinea 2002[5].

2.1.1 (OLE) Automation s aplikací Microsoft Word

Automation (dříve známé jako *OLE Automation*) umožňuje aplikaci používat objekty implementované v jiné aplikaci a zároveň umožňuje aplikaci „vystavovat“ (angl. expose) implementované objekty. *Automation server* (typ COM[6] serveru) vystavuje ostatním aplikacím, nazývaných *Automation klienti*, svojí funkcionalitu skrze COM rozhraní. Vystavením objektů umožňuje server Automation klientům automatizovat jisté funkce skrze přímý přístup k těmto objektům a využití jimi nabízených služeb.

Pro tvorbu dokumentů je skrze (OLE) Automation využívána vlastnost hromadné korespondence, poskytovaná aplikací *Microsoft Word*. Hromadná korespondence umožňuje nahrazování speciálních „MergeField“ polí hodnotami z datového zdroje.

[1] http://www.triada.cz/
[2] Delphi http://www.embarcadero.com/products/delphi
[3] (OLE) Automation http://msdn.microsoft.com/en-us/library/dt80be78.aspx,
[4] Rich Text Format (RTF) Version 1.5 Specification http://www.biblioscape.com/rtf15_spec.htm
[5] Generátor tiskových sestav Sinea http://www.sinea.cz/default.asp?page=20
[6] COM: Component Object Model Technologies http://www.microsoft.com/com/default.mspx

Tímto zdrojem dat může však být pouze plochý DataSet (tedy tabulka, pohled, či výsledek dotazu) a neumožňuje realizovat vazbu otec-syn v datech, díky kterým by mohli být definovány pohledy typu master detail.

Automation servery i klienti využívají pro vzájemnou komunikaci COM rozhraní, které jsou vždy zděděná od rozhraní *IDispatch* a přijímají a vracejí specifickou sadu datových typů, nazývaných Automation typy. Bližší podrobnosti lze nalézt v [1]

Zdroj [2] uvádí, že: „*Vývojáři mohou používat funkci automatizace v systému Microsoft Office k vytvoření vlastních řešení, která využívají možnosti a funkce vestavěné v produktu Office. Takovýto programový vývoj může být poměrně snadno implementován v klientském systému. Pokud má ale dojít k automatizaci prostřednictvím kódu na straně aplikačního serveru, například prostřednictvím skriptů ASP (Active Server Pages), rozhraní DCOM nebo služby NT, mohou nastat různé komplikace.*“

Shrnu-li potenciální problémy při použití (OLE) Automation s Microsoft Office na aplikačním serveru podle [2], pak jimi jsou:

- **Interakce s desktopovou aplikací** – Microsoft Office je klientskou aplikací a jako taková vyžaduje zobrazení okna ke správnému chodu. Pokud dojde k chybě, může dojít k zobrazení chybové zprávy, které vyžaduje interakci uživatele
- **Škálovatelnost** – Microsoft Office jsou navrženy, tak aby nabízely funkcionalitu jedinému klientovi. Není doporučováno automatizovat Microsoft Office na serverových aplikacích, protože není škálovatelná
- **Stabilita** – Použití Microsoft Office jako komponenty serverové služby může snižovat stabilitu počítače, a tím způsobovat nestabilitu celé sítě.
- **Bezpečnost** – Programy Microsoft Office nebyly nikdy navrhovány pro serverový běh. Proto neberou v potaz bezpečnostní problémy, kterým čelí distribuované komponenty.

Ze zkušeností s tímto provozem dále vyplývají nevýhody v podobě závislosti na nainstalované verzi Microsoft Office Word a problémy v kompatibilitě zprostředkovávaných funkcí mezi jednotlivými verzemi, které někdy vedou až k nefunkčnosti řešení u mnoha instalací s Microsoft Office 2007.

Při použití rozhraní (OLE) Automation je Microsoft Office Word vždy nahrán do paměťového prostoru aplikace, což přináší časté běhové a paměťové problémy.

2.1.1.1 Výhody řešení

- Uživatelská upravitelnost šablony ve známé aplikaci Microsoft Office Word

2.1.1.2 Nevýhody řešení

- Nevhodné pro serverový provoz
- Závislost na aplikaci Microsoft Office Word a její verzi

2.1.2 Parsování textového souborového formátu RTF

Druhé řešení tvorby dokumentů je ve stávajícím systému založené na parsování textového souborového formátu Rich Text Format (RTF).

Rich Text Format je univerzálním, na platformě nezávislým, formátem pro kódování formátovaného textu a grafiky, vyvíjeným společností Microsoft. RTF používá k záznamu formátování dokumentu jak pro zobrazení, tak pro tisk znakové sady ANSI, PC-8, Macintosh nebo IBM PC. RTF tak může být považován za most mezi jednotlivými operačními systémy a aplikacemi.

RTF soubor se skládá z neformátovaného textu, řídících slov, řídících symbolů a skupin. Pro snadnost přenosu mezi platformami je standardní RTF soubor uložen v sedmi bitovém kódování pomocí ASCII[7] znaků.

Díky textovému charakteru RTF je generování sestav založeno na full-textovém vyhledávání polí, uzavřených ve speciálních posloupnostech znaků „<?“ a „?>“, které jsou vepsány do šablony. Prostor mezi těmito znaky určuje, kam mají být vložena data, a text mezi těmito znaky určuje mapování na jméno položky v datech.

[7] ASCII http://en.wikipedia.org/wiki/ASCII

2.1.2.1 Výhody řešení

- Nezávislost na dalším softwaru a na platformě, na které jsou šablony editovány
- Velký rozsah a flexibilita formátování
- Možnost nasadit i pro serverový běh

2.1.2.2 Nevýhody řešení

- Při upravení šablony, dojde u některých editorů k vložení dalších řídících symbolů nebo slov mezi speciální posloupnost znaků, se kterými vyhledávání nepočítá
- Nepodporuje master-detail vazby mezi daty

1. Specifikace RTF je rozšiřována s každou verzí Microsoft Office, což ovlivňuje zápis nových řídících znaků. Kvůli těmto změnám je potřeba vyhledávání a nahrazování dat v šablonách upravovat

2.1.3 Sinea 2002

Posledním používaným řešení je generátor tiskových sestav Sinea 2002. Jde o soubor uživatelských komponent určených k instalaci do programátorského prostředí Delphi a Kylix firmy Borland. Jak je uvedeno v manuálu [3], generátor umožňuje prostřednictvím zabudovaných komponent programátorovi i uživateli hotového programu velmi snadným způsobem vytvářet, upravovat šablony a generovat či tisknout výsledné sestavy. Jako zdroje dat jsou využity komponenty typu *Table* a *Query*, které se s komponentou *Sestava* propojí pomocí komponenty *DataSource*.

Knihovna *Sestavy* je především grafickým a textovým editorem pro snadné vytváření tiskových formulářů (předloh), které se ukládají do souboru nebo i na jiná zařízení. Předlohy se načítají za běhu programu před vlastním tiskem. Tiskové sestavy tak zbytečně nezabírají paměť, nezvětšují velikost výsledných EXE souborů a mohou být upravovány uživatelsky i za běhu programu.

Nejdůležitějšími vlastnostmi, kvůli kterým je využíván ve stávajícím informačním systému namísto reportovacího nástroje QuickReport[8], který je dodáván společně s vývojovým prostředím *Delphi*, jsou:

- Uživatelsky přívětivý návrhář tiskových sestav
- Generátor sestav je kompletně český, a to jak grafický návrhář, tak i ukázka před tiskem a všechny uživatelské dialogy.

Komponenta umožňuje výstup přímo na tiskárnu nebo do souborového formátu RTF a HTML. V sestavách lze definovat výrazy, používající základní aritmetické operace a obsahuje rovněž rozsáhlou nabídku vestavěných funkcí.

Ze zkušeností s použitím Sinea 2002 však vyplynulo, že návrh master-detail tiskových sestav je velice problematický. Taktéž výstup komponenty pouze do RTF a HTML není ideální. Proto je využívána především pro tisk úředních a dalších dokumentů, které již po vložení dat není potřeba dále upravovat. Pro získání elektronické podoby tisku je využívána PDF tiskárna třetí strany.

2.1.3.1 Výhody řešení

- Přesný tisk
- Možnost uživatelské editace
- Mnoho funkcí pro použití v tisknutých výrazech

2.1.3.2 Nevýhody řešení

- Obtížná definice master-detail tiskových sestav
- Použitelnost pouze z Delphi/Kylix

2.1.4 Shrnutí stávajících řešení

Všechna stávající řešení tisku a přípravy dokumentů v informačním systému jsou implementována v Delphi, které – jak bylo v úvodu analýzy napsáno – nezapadají do připravované serverové koncepce příští generace systému.

Dále všechna řešení pracují ideálně pouze s plochými daty a neumožňují tak vytváření pohledů master-detail. Ty jsou přitom často potřeba, například pro tisk faktur se všemi

[8] QuickReport http://www.qusoft.com/

položkami. Master-detail šablony jsou zatím řešeny komponentou Sinea 2002, ale – jak bylo napsáno výše – vytvoření a použití takových tiskových sestav je složité. Prakticky nemožným úkolem se pak stává tisk všech faktur se všemi položkami tříděných nebo seskupených podle určitého kritéria.

Ve stávajících řešení není dále například možné definovat univerzální šablonu pro různé typy předmětů zpráv, které jsou vyžadovány pro tisk různých poučení, či odkazů na paragrafy zákonů. Pro takovéto šablony by bylo vhodné mít možnost vypustit nebo naopak začlenit podmíněně určitou část dokumentu.

2.2 Požadavky zákazníka

Cílem této práce je proto navrhnout a implementovat systém pro generování dokumentů z předem připravených šablon, který by výše uvedeným a dalším podobným požadavkům vyhověl.

Systém proto musí umožnit uživateli šablony přidávat a dále editovat. Systém musí být navržen – jak již bylo řečeno v úvodu analýzy na str. 7 – jako třívrstvý na platformě *.NET* pro práci v intranetu, s oddělenými součástmi pro generování dokumentů, správu šablon, databázový stroj a webovou aplikaci.

Součásti aplikačního serveru a jejich klienti spolu musí komunikovat pomocí standardního rozhraní *Web Services*[9], které je v dnešní době používáno při návrhu informačních systémů státní správy, jakými jsou například *Datové schránky*[10] nebo *CSÚIS*[11].

Webová aplikace bude napojená na správu šablon, která m.j. umožní odesílat dokumenty ke konverzi. Jednotlivé součásti mohou být přitom instalovány na různých stanicích.

Součást generující dokumenty by měla být oddělená z důvodu předpokládaného vyššího vytížení stanice při generování dokumentů, dále nabízí takové řešení v budoucnosti lehkou rozšiřitelnost v podobě přidání dalšího serveru pro generování dokumentů a implementaci distribuce zatížení mezi všechny servery.

[9] Web services http://en.wikipedia.org/wiki/Web_service
[10] Datové schránky http://datoveschranky.info/
[11] CSÚIS http://www.mfcr.cz/cps/rde/xchg/mfcr/xsl/vf_sp_csuis.html

Šablony také musejí obsahovat konstrukce pro vytváření pohledů master-detail, a to do libovolné hloubky. Dále musí obsahovat možnost vkládat výrazy, které budou vyhodnoceny až během generování dokumentu a budou umožňovat vypuštění sekce dokumentu.

Systém jako celek bude tvořit doplněk informačního systému *Munis*. Informační systém *Munis* je modulární systém pro veřejnou správu, který bude v příští verzi třívrstevný. Centrálně, pomocí aplikačního serveru, bude v informačním systému *Munis* spravováno přihlašování uživatelů, ověřovány jejich práva a provozovány další zatím nedefinované služby pro stávající desktopové klienty, jež jsou implementovány v prostředí *Delphi,* a vznikající tenké webové klienty. Šablonovací systém musí umožňovat pozdější napojení na tyto aplikace a služby. Jednou ze služeb bude určitě ověření přihlašovacích údajů a práv celého informačního systému. V systému je dále přítomno negarantované datové úložiště (výraz zmiňovaný v některých zákonech, který ale nemá přesnou definici)[12] nazvané *DUL*, jež je centrálním prvkem pro dlouhodobé uchování dokumentů. Šablony, ze kterých bude tvořit dokumenty a vytvořené dokumenty tedy musí ukládat právě do tohoto úložiště.

Šablonovací systém nahradí některá ze stávajících – již nevyhovujících – řešení tvorby dokumentů v informačním systému. Musí proto nabízet řešení, využívající data, produkovaná v předzpracované podobě stávajícími klientskými aplikacemi. Data tedy systém nemá k dispozici přímo, prostřednictvím napojení na databázovou vrstvu informačního systému *Munis*, ale pouze zprostředkovaně. Data jsou zasílaná jednotlivými moduly spolu s žádostí o tvorbu konkrétního dokumentu.

Databázová vrstva šablonovacího systému by měla být kompatibilní primárně s databázovým systémem *Microsoft SQL*, včetně jeho volně šiřitelné verze *Microsoft SQL Express*. Pro svoji rozšířenost je však důležitá i podpora databáze *Oracle* verze 9 a vyšší.

Šablony musejí obsahovat syntax nejen pro přímé vkládání datových polí, ale i pro základní aritmetické výrazy, založené na vkládaných datech. Důležitá je podpora

[12] Garantované vs. Negarantované úložiště
http://oraclecze20.blogspot.com/2009/11/garantovane-vs-negarantovane-uloziste.html

vytváření dokumentů typu master-detail a také podpora podmíněných sekcí, které budou do výsledného dokumentu vloženy pouze po splnění určité podmínky, založené na datech, respektive na výsledcích výpočtů funkcí a aritmetických výrazů.

Dále je potřeba zajistit možnost převodu šablon ze stávajících systémů, založených na (OLE) Automation a RTF do nového systému. Tedy nějaký způsob konverze stávajících šablon ve formátech DOC a RTF do formátu nového. Řešení konverze již není předmětem bakalářské práce.

Vzhledem k zaměření systému na subjekty veřejné správy je podstatné mít možnost tvořit standardní dokumentové formáty Microsoft DOC, OOXML (.docx), WordML a RTF a rovněž standardizovaný formát OpenOffice OpenDocument. Nejdůležitějším z těchto formátů je pro zákazníka formát *OOXML*, který by v budoucnu díky své standardizaci mohl získat povolení pro uzákoněné užití ve státní správě pro archivaci apod. Pro přenos dokumentů, u kterých se nepředpokládají následné změny, je vhodná podpora formátu PDF.

2.3 Analýza trhu

Na trhu je dostupných několik řešení, zabývajících se podobnou problematikou. Tato řešení se liší formáty vstupů, výstupů a možnostmi šablon. Výše jsou popsaná stávající využívaná řešení (OLE) Automation s aplikací Microsoft Word (str. 7) a Generátor tiskových sestav Sinea (str. 10). Řešení, do kterých jsem dále nahlížel, něčím se z nich inspiroval nebo mne z nějakého hlediska zaujala, jsou OpenOffice a FastReport.Net.

2.3.1 OpenOffice[13]

OpenOffice je konkurenčním kancelářským balíkem k produktu *Microsoft Office*. Momentálně, po akvizici firmy *Sun Microsystems*, je jeho vývoj řízen společností *Oracle*. Balík obsahuje aplikaci *OpenOffice Writer*, která je srovnatelná s *Microsoft Office Word*. Pro tvorbu dokumentů ze šablon je i v tomto balíku k dispozici podpora pro hromadnou poštu, umožňující do předem určených polí vkládat data z datového zdroje. Datový zdroj přitom musí mít již v okamžiku tvorby šablony jasně definované názvy datových polí a jejich typy. Největší výhodou tohoto řešení je

[13] OpenOffice http://www.openoffice.cz/

kompletní lokalizace, možnost upravovat šablony i výsledné dokumenty, běh na více platformách a také fakt, že nástroj je poskytován zcela zdarma.

2.3.1.1 Omezení

Nevýhody tohoto řešení se do značné míry shodují s nevýhodami použití *Microsoft Office Word*. Není tak m.j. možné vytvořit šablony typu master-detail. K automatizaci procesu je opět možné využít pouze (OLE) Automation, které, jak je rozebráno v sekci 2.1.1 (OLE) Automation s aplikací Microsoft Word na str. 6, s sebou přináší mnohé problémy. Největším z nich je nevhodnost nasazení na síťovém serveru.

2.3.1.2 Výhody

Díky při návrhu známé definici datového zdroje může *OpenOffice Writer* nabízet kontrolu jmen vkládaných polí během návrhu šablony.

2.3.2 FastReport.Net[14]

FastRepost.Net je tzv. „pruhově orientovaný“ generátor tiskových sestav. Pruhově orientované generátory tiskových sestav jsou založeny na definování horizontálních pruhů určité výšky, ze kterých se tisková sestava skládá. Při generování výsledné tiskového dokumentu se pruhy opakují pro každý záznam určité skupiny dat.

Tvorba tiskových sestav typu master-detail je umožněna pomocí „podpruhů“. Dokumentace uvádí, že *„FastReport.Net je komplexní řešení pro práci s tiskovými sestavami pro Windows Forms a ASP.NET. Může být použit v Microsoft Visual Studio 2005, 2008 a Microsoft Visual Studio 2010. FastReport.Net nepoužívá k vytváření a úpravám sestav MS Visual Studio IDE, místo toho používá vizuální návrhář sestav. Návrhář sestav může být použit přímo za běhu vaší aplikace, takže můžete dát svým uživatelům možnost modifikovat existující sestavy a vytvářet nové.“*

2.3.2.1 Omezení

Nevyhovující je m.j. nedostatečná lokalizace návrháře sestav, který tak nemůže být nabídnut koncovým uživatelům.

[14] FastReport.NET http://ns.fast-report.com/cz/products/FastReport.Net.html

2.3.2.2 Výhody

Na FastReport.Net je dobře vyřešena možnost vkládání výpočetních funkcí a vkládání nejrůznějších skriptů do šablon. Dále je k dispozici přesná editace pozice prvků a možnost provádět nejrůznější dynamické výběry vkládaných obrázků a grafů na základě dat.

2.3.3 Shrnutí existujících řešení

Jak je uvedeno výše v sekcích Omezení na str. 15, ani jedno z nalezených řešení nevyhovuje plně požadavkům zákazníka. Nejvíce se požadavkům blíží řešení FastReport.Net. Zákazník se však rozhodl upřednostnit vlastní řešení, které půjde upravit pro případné budoucí požadavky, před případnou lokalizací produktu třetí strany.

Z nalezených a již používaných řešení se mi líbil zejména rozsah funkcí, obsažený v produktu Sinea 2002. Ten se také stal inspirací pro nabídku funkcí, zahrnutých v navrhovaném systému.

3 Specifikace

Z požadavků zákazníka na str. 12 plyne nutnost implementace třívrstvého systému, rozděleného na komponenty aplikačního serveru „Generátor dokumentů“ a „Správa šablon“, databázovou vrstvu a klientskou webovou aplikaci či aplikace. Hlavní komponenty musí být odděleny z důvodů předpokládaného budoucího vytížení pracovní stanice v průběhu generování dokumentů, a tedy případné nutnosti nasazení další stanice na generování dokumentů s přidáním podpory distribuce zatížení.

Komponenty aplikačního serveru a klientské aplikace spolu vzájemně komunikují prostřednictvím *Web Services.*

Systém nebude přímo napojen na databázovou vrstvu celého informačního systému *Munis*, ale musí data přijímat od ostatních modulů informačního systému *Munis*.

3.1 Generátor dokumentů

Generátor dokumentů je síťovou službou, která vytvoří z šablony a dat výsledný textový dokument. Generátor musí být, vzhledem k nasazení na serveru, *thread-safe*, aby nevznikaly konflikty při několika konverzích, probíhajících najednou.

3.1.1 Vstup

Na vstupu Generátoru šablon jsou zapotřebí pojmenovaná data z určitého zdroje dat a šablona dokumentu.

3.1.1.1 Zdroj dat

Data, tak jak je poskytnou jednotlivé zdroje – aplikace a služby informačního systému *Munis* – mohou obsahovat vazby otec-syn, díky kterým je možné definovat šablony typu master-detail.

Zdroj dat může vystavovat i více typů dat. Příkladem může být modul *Výkaznictví*, který poskytuje data pro výkazy Rozvaha, Výkaz zisku a ztráty, Přehled o peněžních tocích aj.

Na druhou stranu mohou různé zdroje dat generovat data stejného významu v různých strukturách. Pro účely sloučení dat se šablonou by ale tato data měla svoji strukturou odpovídat požadavku na typ dat, deklarovaný v konkrétní šabloně. Deklaraci typu dat

lze reprezentovat stromem, respektive lesem, kde každý uzel popisuje strukturu datového záznamu s názvy a typy jednotlivých polí a vazby otec-syn odpovídají vazbám master-detail. Data samotná by měla být předána v jednotném formátu. Data v požadovaném formátu a s požadovanou strukturou budeme nadále v práci nazývat *nativními daty* šablony.

Pokud zdroj neposkytuje přímo nativní data, bude nutné zajistit konverzi dat do jejich nativní podoby.

Deklarace jednotlivých typů záznamů v rámci deklarace nativního datového typu šablony by měla obsahovat hlavičku, určující jméno typu, seznam položek a seznamů, které typ obsahuje. Položka je definována jménem a typem a vyskytuje v instanci dat právě jednou. Seznam je definován jménem a typem a určuje položku, která se v typu nevyskytuje nebo se vyskytuje alespoň jednou.

```
Hlavička – Položka
   Položka – ID – číslo
   Položka – Název - řetězec

Hlavička – Objednávka
   Položka – ID – řetězec
   Položka – Počet položek – číslo
   Seznam – Položka - Položka

Hlavička – Zákazník
   Položka – ID – číslo
   Položka – Jméno – řetězec
   Položka – Příjmení – řetězec
   Položka – Kontakt Ulice– řetězec
   Položka – Kontakt Číslo popisné – číslo
   Položka – Kontakt Město – řetězec
   Položka – Kontakt PSČ - číslo
   Položka – Telefon – číslo
   Seznam – Objednávka - Objednávka
```

Příklad 1 Deklarace datového typu „seznam objednávek zákazníka“

```
Zákazník
   ID – 1
   Jméno – Adam
   Příjmení – Kozák
   Kontakt Ulice – Masarykova
   Kontakt Číslo – 15
   Kontakt Město – Olomouc
   Kontakt PSČ – 12345
   Telefon – 456123123
   Objednávka
      ID – 1
      Počet Položek – 2
      Položka
         ID – 1
         Název – CPU
      Položka
         ID – 2
         Název – Základní deska
   Objednávka
      ...
```

Příklad 2 Instance dat typu „seznam objednávek zákazníka“

3.1.1.2 Šablona dokumentu

Šablona musí obsahovat elementy pro vkládání výrazů (například pro dopočítání DPH jednotlivých položek faktury), možnost vytvářet podmíněně vynechané či zobrazené sekce a sekce opakované pro každou položku dat. Šablona je určena pro právě jeden nativní typ dat. Typů dat, které jsou vkládány do jedné šablony, může být více za předpokladu, že jsou na nativní typ převoditelné. Pokud má dva a více typů podobné struktury a význam položek při tisku, pak můžeme definovat univerzální šablonu zobrazující tyto data a odpovídající konverze na nativní datový typ. Konkrétním příkladem může být tisk výkazů za období. V informačním systému *Munis* mají například roční a čtvrtletní výkaz jiný význam, ale při tisku jsou strukturovaná stejně. Při vyhledávání dokumentů pak mohu chtít prohledat pouze ty, které obsahují roční výkaz.

```
Hlavička – Roční výkaz
   Položka – Období od – datum
   Položka – Období do – datum
   Položka – Roční bilance - číslo
```

Příklad 3 Deklarace typu „roční výkaz“

```
Hlavička – Čtvrtletní výkaz
   Položka – Období od – datum
   Položka – Období do – datum
   Položka – Čtvrtletní bilance - číslo
```

Příklad 4 Deklarace typu „čtvrtlení výkaz“

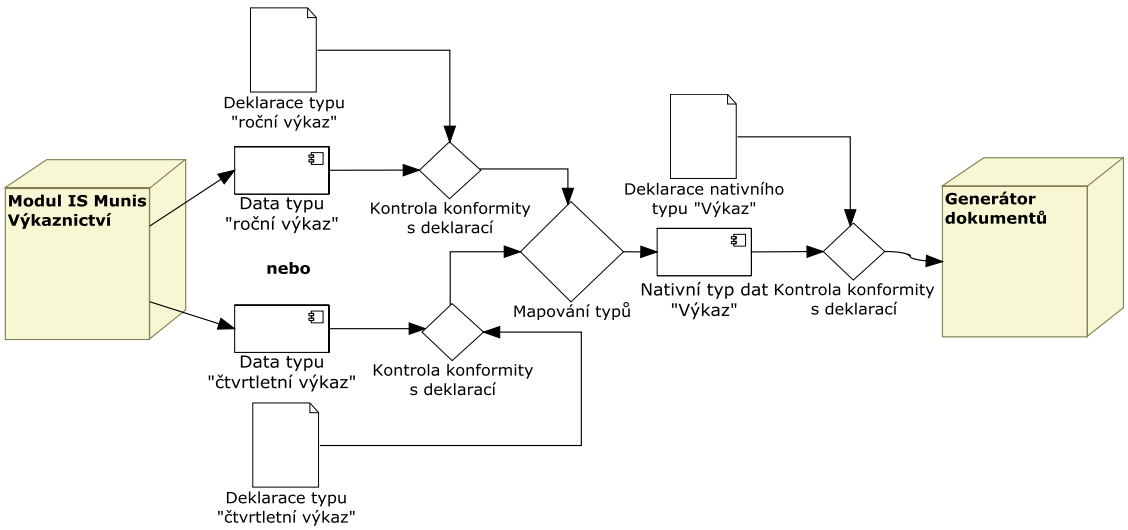
```
Hlavička – Výkaz
   Položka – Název - řetězec
   Položka – Období od – datum
   Položka – Období do – datum
   Položka – Bilance - Bilance
```

Příklad 5 Deklarace nativního typu „Výkaz“ vkládaného do šablony

Pro takové typy dat musí být zavedeno mapování mezi typem primárních dat, zasílaným zdrojem dat, a nativním typem dat, vkládaných do šablony. Toto mapování je jednosměrné, jelikož typ dat vkládaný do šablony již nemusí zachovávat význam všech položek.

```
Mapování – Roční výkaz -> Výkaz
   Položka – Název = „Roční výkaz“
   Položka – Období od –> Období od
   Položka – Období do –> Období do
   Položka – Roční bilance -> Bilance
```

Příklad 6 Mapování mezi datovými typy



Obr. 1 Schéma práce s datovými typy

Obr. 1 znázorňuje výše popsanou situaci.

3.1.2 Výstup

Výstupem Generátoru šablon je textový dokument s vyplněnými daty.

3.2 Správa šablon

Je síťovou službou, která nabízí rozhraní pro správu šablon, umožňuje tedy číst, přidávat, odebírat a upravovat záznamy šablon. Je napojena na databázovou vrstvu, obsahující informace o šablonách a jejich datových typech. Šablony ukládá do datového úložiště *DUL* systému *Munis*. Práva pro jednotlivé operace se záznamy šablon získává z centrální přihlašovací služby informačního systému *Munis*. Napojení na centrální přihlašovací službu a úložiště musejí být navržena tak, aby je bylo snadné v budoucnu měnit, protože se jejich rozhraní stále vyvíjí.

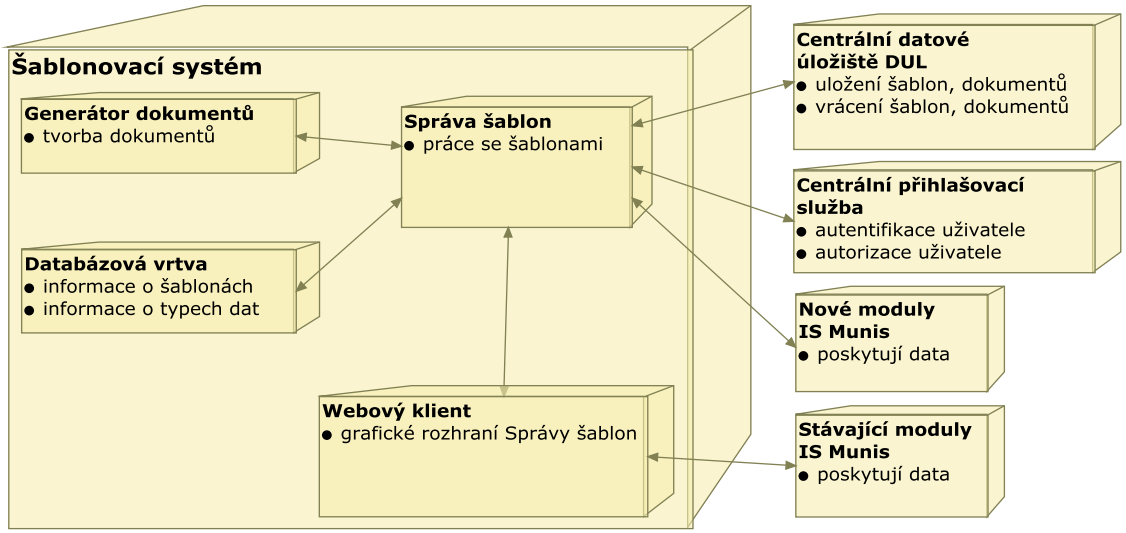
Správa šablon je jediným prvkem, který bude komunikovat s *Generátorem šablon*, webovou aplikací a databázovou vrstvou a nabízet rozhraní pro výběr vhodné šablony podle datového typu a odeslání této vybrané šablony do tvorby dokumentů. Je tedy v podstatě zprostředkovatelem komunikace jednotlivých částí šablonovacího systému s moduly informačního systému *Munis*, které budou do šablonovacího systému přistupovat přímo. V první verzi budou veškeré dokumenty tvořeny s pomocí webového klienta. Díky takovému centrálnímu prvku šablonovacího systému, lze do další verze implementovat zmiňovanou distribuci zátěže *Generátoru dokumentů*.

3.3 Databázová vrstva

Vrstva musí být navržena pro provoz na *Microsoft SQL*, *Microsoft SQL Express* a *Oracle* verze 9 a vyšší. Uchovává informace o šablonách, zdrojích dat a asociacích mezi nimi.

3.4 Webový klient

Vzhledem k chystané architektuře s tenkými klienty a potřebě zapojení i některých stávajících modulů informačního systému *Munis* k nové tvorbě dokumentů je nejvhodnější navrhnout uživatelské rozhraní správy šablon jako webovou aplikaci. Tato webová aplikace bude stěžejní pro integraci stávajících modulů informačního systému *Munis*, protože bude zprostředkovávat nabídku výběru vhodné šablony podle aktuálního zdroje dat. Webový klient by v příští verzi mohl nabídnout i jednoduchý editor šablon a výsledných textových dokumentů.



Obr. 2 Struktura šablonovacího systému

Obr. 2 popisuje strukturu šablonovacího systému spolu s napojeními na další komponenty informačního systému *Munis.*

4 Návrh

Návrh systému je rozdělen do dvou hlavních částí. První se zabývá návrhem šablon. Druhá část se věnuje návrhu systému, tedy platformou a jejím rozvrstvením a zodpovědnostmi jednotlivých součástí.

4.1 Návrh šablony

Tato sekce řeší následující témata:

- Formát šablon
- Formát dat
- Výrazy v šablonách
- Syntax elementů šablony
- Podporované výstupní formáty dokumentů
- Převod starých šablon – pouze diskuze řešení, není součástí BP

4.1.1 Formát šablony

Jak je uvedeno v požadavcích zákazníka na str. 12 je potřeba tvořit více formátů výstupních dokumentů. V těchto situacích existují dva základní přístupy. První možností je mít šablony uchovávané v jednom univerzálním formátu, z ní vytvořit výsledný textový dokument a z něj pak transformací vytvořit cílové dokumentové formáty. Druhým přístupem je mít šablony, uchované a vytvářené přímo ve výstupním formátu a jen do nich doplňovat požadovaná data.

Výhodou první metody je jednotný formát šablon, tedy jeden definovaný nástroj pro tvorbu, editaci a zobrazení šablon, ve které lze připravit všechny šablony uživatele. Jelikož je proces vkládání dat do šablony také závislý na formátu šablony, může být implementován pouze jednou.

Nevýhodou je nutnost definovat převod formátů pro tvorbu výsledných dokumentů v jiném formátu než v jakém byla šablona. Převod formátů je problematický, protože formáty neobsahují stejné definice a často ani stejnou logiku záznamu určité struktury textového dokumentu. Dále je nevýhodná závislost na konkrétním nástroji na úpravu šablon.

Výhodou druhého přístupu se šablonami, uchovávanými ve výstupním formátu, je jednoznačně jednoduchost vložení dat do šablony, při kterém nemůže dojít ke ztrátě formátování jako je tomu u první metody při konverzi formátů.

Nevýhodou je nutnost vytvářet několik sad šablon – pro každý podporovaný formát jednu. Dále je potřeba několika nástrojů pro tvorbu, editaci a zobrazení šablon, které musí uživatel, pokud chce upravovat a používat šablony více formátů, vlastnit. Nevýhodná je také nutnost implementovat a udržovat program pro manipulaci (vkládání dat) se šablonou pro každý formát zvlášť kvůli specifikám jednotlivých formátů. Problematická je také situace, kdy je cílový formát dokumentu binární.

Zákazník požaduje – na str. 12 – tvořit dokumenty ve více formátech. Bylo by časově náročné a neefektivní vytvářet šablony a aktualizovat aplikaci pro všechny formáty. Mnohem výhodnější je použít jeden univerzální formát šablony, pro který budou definovány převody pro všechny požadované výstupní formáty.

Formáty, nad kterými bylo uvažováno jako univerzální základ pro vlastní definici šablony, jsou PostScript[15], vlastní XML schéma popisující vzhled dokumentu, XSL-FO[16], DocBook[17] nebo některý z používaných dokumentových formátů jakými jsou OOXML či *ODT*.

PostScript jak je uvedeno na české wikipedii [4] je: „programovací jazyk určený ke grafickému popisu tisknutelných dokumentů vyvinutý v roce 1985 firmou *Adobe Systems Incorporated*. Jeho hlavní výhodou je, že je nezávislý na zařízení, na kterém se má dokument tisknout. Je považován za standard pro dražší tiskárny. Díky svým rozsáhlým možnostem se však brzy stal i formátem k ukládání obrázků.“ Je založen obdobně jako *RTF* na textových formátovacích znacích. V tomto formátu by však bylo obtížné psát vlastní parser a taktéž se ukázalo, že na trhu není nástroj, který by byl vhodný pro editaci šablon koncovými uživateli.

Vlastní *XML* schéma by obsahovalo elementy pro uspořádání textu v dokumentu. Parsování a vkládání dat do takového *XML* by nebylo náročné, obecně by vyznačení

[15] PostScript http://www.adobe.com/devnet/postscript/pdfs/5003.PPD_Spec_v4.3.pdf
[16] XSL-FO http://www.w3.org/TR/xsl/#fo-jc-intro
[17] DocBook http://docbook.org/

sekce k opakování či vypuštění v *XML* formátu bylo jednoznačné díky základní stromové struktuře *XML*. Definovat vlastní formát však samo o sobě překračuje rámec bakalářské práce a navíc *XML* schémata, určená k popisu grafické stránky dokumentů již existují.

XSL-FO neboli *Extensible Stylesheet Language Formatting Objects* je dle [5] jazyk navržený pro formátování *XML* dat pro výstup na obrazovku, papír a jiná média. Bohužel není jazyk příliš rozšířený a tím pádem neexistují žádné lokalizované editory, které by připadaly v úvahu pro použití koncovými uživateli.

DocBook jak uvádí [6] je: „značkovací jazyk[18] pro psaní dokumentace, původně byl zaměřený na dokumentaci k softwaru a hardwaru, ale může být použit na jakoukoliv dokumentaci. Lze jej však využít i pro psaní knih, článků, tvorbu prezentací nebo přípravu celých webů. *DocBook* je implementován jak pomocí XML, tak i SGML. Mezi výhody *DocBooku* patří možnost konverze dokumentů do dalších formátů jako HTML, PDF, RTF, HTML Help a dalších. Z jedné předlohy lze zcela automaticky – s využitím nástrojů, které jsou většinou zdarma – vytvořit několik výstupních podob původního dokumentu.“ Hlavní výhodou tohoto formátu je, že jde o otevřené řešení a tím pádem je - jak je výše zmíněno – většina nástrojů zdarma. Ovšem pro zákazníka je důležitější zaručená podpora i v budoucnosti, která se u otevřených řešení nedá předpokládat.

Všechny výše zmíněné formáty mají další pro zákazníka závažnou nevýhodu, neumožňují pro zákazníka přijatelnou konverzi do formátu *OOXML*. Přijatelnou konverzí byla zákazníkem definována konverze, kterou by bylo možné nasadit pro serverový provoz bez nutnosti platit za každou používanou instanci (tzv. *royalty free* licence) a zároveň by měla zaručenou podporu. Proto bylo rozhodnuto založit generování dokumentů nad nativním standardizovaným formátem OOXML aplikace *Microsoft* Word 2007 a vyšších, který je pro zákazníka důležitější než formát, využívaný balíkem *OpenOffice.*

Jelikož je formát *OOXML* velice obsáhlý, bylo zákazníkem stanoveno, že pro parsování a generování tohoto formátu bude zakoupena již hotová komponenta za předpokladu,

[18] Značkovací jazyk http://cs.wikipedia.org/wiki/Značkovací_jazyk

že bude *royalty free*. Pro platformu *:NET* byly nalezeny dvě takové komponenty a to *Aspose.Words*[19] a *TX Text Control*[20].

4.1.1.1 Aspose.Words

Jak uvádí [7] *Aspose.Words pro .NET* je: „pokročilá knihovna tříd, která umožňuje práci s dokumenty přímo z programovacího jazyka bez využití *(OLE) Automation* s *Microsft Office Word*."

Zprostředkovává Document Object Model nad textovými dokumenty. Tento Document Object Model umí načíst z formátů DOC, OOXML, RTF, HTML, OpenDocument, PDF, XPS, EPUB a dalších. Mezi všemi formáty *Microsoft Office Word* – DOC, OOXML, WordML, RTF – nabízí výbornou úroveň konverze. Při uložení jednoho dokumentu ve formátech DOC, OOXML, WordML a RTF se neztratí žádné formátování textu ani grafiky. Konverze do ostatních formátů probíhají rovněž velice uspokojivě – některé formátování textu a grafiky však není ve všech formátech podporováno. Například při převodu z formátu OOXML do ODT se ztrácejí některé horizontální čáry. Nástroj dále umožňuje – pomocí rozšířené syntaxe v *MergeField* polích *Microsoft Office Word* – tvorbu reportů nad dokumenty, včetně master-detail pohledů, ale pouze do první úrovně. V základu podporuje různé formáty datových zdrojů jako je ADO.NET, XML či vlastní zdroje. Neobsahuje žádný vlastní vizuální editor. O to lépe je však zpracována rozšiřitelnost celého systému vlastními třídami pomocí definovaných rozhraní.

4.1.1.2 TX Text Control

TX Text Control je – podobně jako *Aspose.Words* – komponenta, zprostředkovávající funkce pro zpracování textu pro platformu *.NET*. Umožňuje tak konverze a úpravy podporovaných typů dokumentů v aplikaci, bez využití *(OLE) Automation* s *Microsoft Office Word*. Oproti *Aspose.Words* nabízí navíc WYSIWYG editor jak pro klientskou aplikaci, vytvořeného ve *WinForms*, tak pro webovou aplikaci *ASP.NET* s WebForms. Editor Bohužel není úplně dokonalý, nepodporuje některé důležité formátovací funkce, a dokumenty vytvořené mimo něj často nezobrazuje korektně. Nenabízí tak rozsáhlé rozhraní pro vkládání dat do šablon jako řešení *Aspose.Word*. Pro

[19] Aspose.Words http://www.aspose.com/categories/.net-components/aspose.words-for-.net/default.aspx

[20] TX Text Control http://www.textcontrol.com/en_US/

dosažení ideálních výsledků musí být kód pro generování dokumentů navržen pro každou šablonu zvlášť.

4.1.1.3 Závěr

Jako nejvhodnější byla zákazníkem vybrána komponenta *Aspose.Words pro .NET*, která nabízí podporu formátu *OOXML* a navíc podporu formátu *ODT*. Také má implementovány jednoduché master-detail pohledy. Tvorba dokumentů tedy bude založena na této komponentě.

4.1.2 Formát vstupních dat

Šablonám je potřeba předávat data, která systém do šablon vloží. Aby mohla být automaticky zpracovávána, tato data musí mít předem určenou strukturu, která je dána jejich typem. Přenos dat mezi moduly systému nás nijak neomezuje. Je však důležité pokrýt nároky, kladené na data při vkládání do šablon.

V sekci Specifikace na str. 17 je uvedeno, že data musejí umožňovat realizaci vztahů otec-syn mezi daty. Takové vztahy jsou obvykle řešeny buď vlastní objektovou strukturou, tabulkami s definovanými referencemi, nebo pomocí *XML* stromu, popsaného pomocí *DTD*[21] nebo schématy XSD. *XML* v tomto pohledu nabízí flexibilitu a přidání dalšího datového typu ze zdroje dat nebo nativního typu dat tak obnáší pouze definování dalšího schématu či *DTD*. Nejčastěji budou data generována zdrojem přímo v nativním formátu. Pokud budou generována alespoň v podobě XML, mapování na nativní typ dat může být snadno realizováno pomocí *XSLT*.

Navíc nám toto řešení přináší výhodu v jednoduchém ověření, zda zaslaná data strukturálně odpovídají těm požadovaným. Pro popis *XML* struktur jsou použita *XSD* schémata, která jsou flexibilnější a představují oproti DTD modernější způsob popisu.

Dále musejí data popisovat i případ tvorby několika dokumentů, proto bude mít kořenový element XML dat vkládaných do šablony pouze jeden typ syna. Pro každého syna kořenového elementu v primárních datech bude vytvořen jeden samostatný textový dokument.

[21] DTD http://www.w3.org/XML/1998/06/xmlspec-report-19980910.htm

```xml
<?xml version="1.0" encoding="utf-8"?>
<ZakazniciSpolecnostiSObjednavkamiDleSekciDocuments
xmlns="http://www.triada.cz/ZakazniciSpolecnostiSObjednavkamiDleSekciDocuments">
   <ZakazniciSpolecnostiSObjednavkamiDleSekci>
      <JmenoSpolecnosti>Fiktivní společnost s.r.o.</JmenoSpolecnosti>
      <UliceSpolecnosti>Masarykova 15</UliceSpolecnosti>
      <MestoSpolecnosti>Praha</MestoSpolecnosti>
      <PSCSpolecnosti>120 00</PSCSpolecnosti>
      <Zakaznik>
         <Jmeno>Jan</Jmeno>
         <Prijmeni>Novák</Prijmeni>
         <Ulice>U Kováře 15</Ulice>
         <Mesto>Tábor</Mesto>
         <PSC>123 45</PSC>
         <Objednavka>
            <Cislo>1</Cislo>
            <Datum>1.1.1993</Datum>
            <Polozka>
               <Jmeno>Čaj</Jmeno>
               <Popis>Popis čaje</Popis>
               <PocetKusu>110</PocetKusu>
            </Polozka>
            ...
         </Objednavka>
         ...
      </Zakaznik>
      ...
   </ZakazniciSpolecnostiSObjednavkamiDleSekci>
   ...
</ZakazniciSpolecnostiSObjednavkamiDleSekciDocuments>
```

Příklad 7 Vkládaná data

Příklad 7 Vkládaná data znázorňuje možný nativní typ dat. Pro snadnější použití v informačním systému *Munis* by bylo vhodné využít jmenné prostory pro jednotlivé elementy, které by byly znovu použitelné, a definovat zdroje dat přehledněji pomocí nich.

Uchování dat v XML a jejich asociace s vytvořeným dokumentem navíc přináší možnost v budoucích verzích nebo dalších modulech informačního systému *Munis* vyhledávat ve vytvořených dokumentech podle obsahu a významu vkládaných dat, který se při vložení do textového dokumentu ztrácí.

4.1.3 Výrazy v šablonách

Výraz je jednořádkovou formulí s jednoznačnou hodnotou, která se skládá z operandů a operátorů. Operandy by mohly být typované nebo netypované, jako je tomu u skriptovacích jazyků. Použití netypovaných členů, by však vzhledem k použité platformě *.NET* vyžadovalo neustálé převody mezi typy a vyhledávání možných operací nad výrazem by se tak stávalo obtížnější. Navíc pro běžného uživatele je podle mého názoru vhodnější mít jasně rozlišené typy spolu s definovanými operacemi. U netypovaných jazyků se často může uživateli stát, že výsledek je naprosto jiný, než jaký požadoval a hledání chyb v takových výrazech považuji za náročné.

Proto jsem zvolil možnost typovaných členů výrazů, kde bude jasně definované, které typy a jak lze používat.

Výrazy lze v šabloně použít dvěma způsoby. Buď jsou jejich výsledky tisknuty a výsledná hodnota je převáděna automaticky na její textový ekvivalent, nebo jde o logické výrazy, jejichž výsledkem je logická hodnota, které jsou používány v podmíněných sekcích.

Hodnoty výrazů mohou být čtyř typů:

- Číslo
- Datum
- Textový řetězec
- Logická hodnota *TRUE* a *FALSE*.

V první verzi není požadována možnost upravit formát tisku hodnot z vyhodnocených výrazů v šabloně, proto se řídí následujícími neparametrizovanými pravidly:

- Čísla – Pouze jako čísla.
- Datum – Ve tvaru *datum čas*. *Datum* ve tvaru *dd.mm.rrrr*, kde *dd* je dvoumístné číslo dne, *mm* dvoumístné číslo měsíc a *rrrr* čtyřmístné číslo roku. Čas ve tvaru *hh:mm:ss*, kde *hh* je dvoumístné číslo hodiny, *mm* je dvoumístné číslo minuty a *ss* je dvoumístné číslo sekund. (příklad 1.1.1993 12:45:00)
- Textové řetězce – Hodnota textového řetězce.
- Logické hodnoty – nejsou určeny k tisku

V příští verzi by mohlo být řešeno formátování tisknuté hodnoty čísel a dat pomocí masky.

Operandy jsou, co do významu při parsování výrazu, tří základních typů:

- Proměnné
- Funkce
- Konstantní hodnoty.

Proměnné jsou položky datového typu aktuálního záznamu (vzhledem k iteracím seznamy), či položkami jeho předků, takto je zajištěna jednoznačnost významu dosazované proměnné. Proměnné jsou rozpoznávány jako identifikátory, které musejí začínat písmenem a mohou obsahovat pouze písmena a čísla. Pro přístup do konkrétní položky aktuálního záznamu nebo jeho předka je použita posloupnost jmen seznamů od kořenového záznamu hierarchie, spojených znakem „.“.

Při použití dat z Příklad 7 tedy pro přístup na položku *Cislo* v seznamu *Objednavka*, pokud je právě zpracováván záznam objednávky nebo sekce, použijeme syntax *Zakaznik.Objednavka.Cislo*.

Funkce budou definovány v pevně předem daném rozšiřitelném rozsahu. Každá funkce přijímá určitý počet parametrů. Parametry jsou opět výrazy, které mají hodnotu určitého typu. Na první verzi nebyl kladen požadavek na implementaci funkcí jako je suma nebo průměr, které vyžadují celé sloupce hodnot, toto se může stát součástí verze příští. Zatím nebyl se zákazníkem dohodnut přesný rozsah funkcí pro projekt, proto funkce zatím nelze ve výrazech použít.

Jelikož jsou výrazy pouze jednořádkové a neobsahují žádné části deklarace, budou různé typy konstant rozpoznávány podle speciálních znaků:

- Číselné konstanty jen jako číslo se znakem „.“, oddělujícím desetinnou část
- Datum je uvezeno znakem ‚#‘ ve tvaru *#datum čas*. *Datum* ve tvaru *dd.mm.rrrr*, kde *dd* je dvoumístné číslo dne, *mm* dvoumístné číslo měsíce a *rrrr* čtyřmístné číslo roku. Čas ve tvaru *hh:mm:ss*, kde *hh* reprezentuje hodiny ve dvouciferném tvaru v rozmezí 00-23, *mm* je dvoumístné číslo označující minuty a *ss* je dvoumístné číslo pro sekundy. Části *datum* a *čas* jsou volitelné a přítomna musí

být alespoň jedna z nich. Validními daty tedy jsou: *#1.1.1999*, *#13:00:00* i *#1.1.1999 13:00:00*

- Řetězcové konstanty jsou obklopeny znakem ,"', výskytu znaku ,"' uvnitř řetězce proto musí předcházet znak ,\'. Příklad takového řetězce je: *„přezdívaný \"Hrozba\""*, který bude zobrazen jako: *přezdívaný „Hrozba"*. Aby byl znak ,\' tištěn, musí být zdvojen. Například řetězec *„C:\\Data"* bude zobrazen jako: *C:\Data*
- Logické hodnoty PRAVDA a NEPRAVDA, tak jak jsou zde uvedeny

4.1.3.1 Podporované operátory

Podporovány jsou základní binární aritmetické operátory sčítání: +, -, operátory násobení: *, /, % (modulo), operátory rovnosti: ==, !=, porovnávací operátory: <=, >=, <, >, operátory priority: (,), a logické operátory: & (and), | (or). Dále jsou podporovány také unární operátory - (minus) a ! (not).

Každý operátor má ve výrazu prioritu, podle které je výraz vyhodnocován. Priority jsou stanoveny podle obvyklých zvyklostí (shora operátory s nejvyšší prioritou):

1. Unární operátory
2. Operátory násobení
3. Operátory sčítání
4. Operátory porovnání
5. Operátory rovnosti
6. Operátor And
7. Operátor Or

Tato priorita může být změněna pomocí operátorů priority.

Operátory +, -, *, / a % lze použít mezi dvěma číselnými výrazy, operátor + lze navíc použít pro spojení dvou řetězců.

Porovnávací operátory jsou podporovány pro všechny typy výrazů za předpokladu, že oba operandy jsou stejného typu. Řetězce jsou porovnávány lexikograficky a používá národní abecedy podle aktuálního místního nastavení v prostředí *.NET*.

Logické operátory lze použít pouze mezi dvěma logickými operandy.

4.1.3.2 Vyhodnocení výrazu

Výrazy budou vyhodnocovány až v momentě dotazu na hodnotu daného výrazu, protože obsahují proměnné, které se mění s každým datovým záznamem. To nám navíc umožňuje mít funkce nahrazující ternární operátory, jejichž parametry nemusejí být vyhodnoceny vždy, příkladem takové funkce je *IF.*

Před vyhodnocením výrazu, ale nejdříve musí být sestaven vyhodnocovací strom, který je vhodné v tomto případě uchovat, vzhledem k potřebě výraz vyhodnotit znovu pouze se změněnými proměnnými. Ten může být stavěn přímo při čtení výrazu, nebo s mezistupněm přes tzv. *„tokeny“*, kdy je nejdřív výraz tzv. *„tokenizován“* a poté je z tokenů stavěn strom výrazu.

Byl zvolen přístup výraz nejdříve ztokenizovat, protože je tento přístup jednodušeji upravitelný a rozšiřitelný.

Komponenty, které by řešili tento problém na platformě *.NET*, existují. Příklad jedné z nich je dostupný na komunitní síti *Code Project*[22] v článku *The expression evaluator revisited*[23], který ale neobsahuje proměnné – jak jsou definovány v sekci Výrazy v šablonách. Zákazník se rozhodl raději implementovat vlastní vyhodnocení výrazů, které tak bude plně upravitelné a zdokumentované.

Jelikož budou výrazy vyhodnocovány mnohokrát při opakování nějaké sekce dokumentu, je strom výrazu postaven pouze jednou, uložen a při dalším vyhodnocení jsou pouze přečteny aktuální proměnné do uzlů stromu reprezentující proměnné.

4.1.3.3 Objektový návrh

Pro tokenizaci je výhodné mít pro každý rozpoznávaný znak vlastní typ token objektu, který tak může být při tokenizaci vytvářen pomocí návrhového vzoru *factory*[24]. Tuto „továrnu“ pak není složité rozšiřovat, či zaměnit, na rozdíl od procedurálního přístupu.

[22] Code Project http://www.codeproject.com/
[23] The expression evaluator revisited http://www.codeproject.com/KB/recipes/eval3.aspx
[24] Návrhový vzor *Factory* http://www.dofactory.com/Patterns/PatternFactory.aspx

Taktéž každý prvek výrazu je představen vlastním typem, který je zapojen do stromu výrazu, se společným předkem. To umožní jednoduše rozšiřovat možnosti výrazů a jednoduše vyhodnotit strom.

4.1.4 Syntax elementů šablony

Při návrhu elementů šablony byl brán ohled na XML strukturu vstupních dat, umožňující podporu vytváření pohledů master-detail do libovolné hloubky, stejně jako na nutnost podpory vkládání výrazů a podmíněné vypouštění sekci dokumentů.

V komponentě *Aspose.Words*, která je použita pro načtení a úpravy dokumentů, jsou definovány párové elementy pro označení části šablony, která bude opakována pro každý záznam. Tyto párové elementy využívají, stejně jako hromadná korespondence aplikace *Microsoft Office Word, „MergeField“* polí. Tato pole neobsahují název datové položky záznamu, jež má být vložena, ale název seznamu v datovém typu, pro jehož záznamy bude sekce dokumentu opakována. Název seznamu v datovém typu je uvezen prefixem „*TableStart:*“. V základu umožňuje komponenta pouze jednu úroveň vnoření. Neumožňuje tedy v šabloně definovat detail detailu.

Tato vlastnost komponenty *Aspose.Words* je v řešení rozšířena aby mohly být elementy vnořeny do sebe a definovat tak vnořené sekce.

Pro vynechání části šablony je zapotřebí definovat párový element, který ji bude ohraničovat. Nejvhodnější syntaxí pro takovéto elementy by z programátorského hlediska byla možnost IF, ElSE IF, ElSE, ENDIF. Vzhledem k cílové skupině uživatelů jsou definovány uživatelsky pochopitelnější možnosti VYNECHKDYŽ, ZOBRAZKDYŽ a jejich párové zakončovací elementy.

4.1.5 Podporované výstupní formáty dokumentů

Komponenta *Aspose.Words* umožňuje dokument načtený do jeho *Document Object Modelu* uložit do mnoha formátů, z nich jsou pro zákazníka důležité *OOXML*, *ODT* a *PDF*. Výstupní dokumenty ve formátu *OOXML* a *ODT* lze po vytvoření editovat, čímž je splněn jeden z požadavků zákazníka.

Dalším formátem, který by bylo dobré tvořit je univerzální otevřený formát *DocBook*, jehož základní vlastnosti jsou popsány v sekci Formát šablony na str. 23. Ten by bylo

možné získat pomocí návrhového vzoru *Visitor*[25], který je v komponentě *Aspose.Words* podporován.

4.1.6 Převod starých šablon

Díky vybranému formátu šablon, je převod starých šablon realizovatelný jednoduše. V případě šablon v dokumentovém formátu *DOC*, které používají funkci hromadné korespondence aplikace *Micrsoft Office Word,* stačí přidat tuto šablonu do Správy šablon a vytvořit pro ni odpovídající nativní typ dat.

Šablony v RTF – popsané v sekci Parsování textového souborového formátu RTF na str. 9 – budou muset být editovány a jejich speciální znaky nahrazeny odpovídajícími poli *MergeField*. Poté již obdobně jako pro šablony ve formátu *DOC* musejí být vloženy do Správy šablon a musí pro ně být vytvořen odpovídající datový zdroj.

4.2 Návrh systému

Podle požadavků zákazníka je potřeba systém navrhnout pro platformou *.NET*. Vzhledem k tomuto požadavku nebyla při návrhu uvažována jiná platforma, ani možnost tvorby hybridního řešení, využívající součásti implementované v jiném prostředí.

Návrh systému je rozdělen do několika částí:

- Vrstvy systému
- Databázová vrstva
- Platforma komunikace modulů aplikační vrstvy s klienty
- Generátor dokumentů
- Správa šablon
- Webový klient

4.2.1 Vrstvy systému

Podle specifikace jde o třívrstvý systém, s aplikační vrstvou – jak je zmíněno v sekci Specifikace na str. 17 – rozdělenou do dvou částí.

[25] Návrhový vzor Visitor http://www.dofactory.com/Patterns/PatternVisitor.aspx

4.2.2 Databázová vrstva

Vzhledem k požadavku na nezávislost na platformě, jsou všechny operace nad daty obsaženy v aplikační vrstvě. Databázová vrstva tedy uchovává pouze informace o šablonách, datových typech a asociacích mezi nimi.

Pro uchování popisu šablon je v databázové vrstvě možné uložit její jméno a popis. Dále je pro potřeby celého informačního systému uchovávána informace o jejím vlastníkovi a datu přidání. Tyto informace mohou být užitečné pro zákaz editace šablon dodávaných přímo s produktem. Primárním klíčem pro záznamy šablon je *Guid*, pod kterým je šablona vedena v negarantovaném datovém úložišti *DUL*. Protože negarantované datové úložiště nemaže uchované dokumenty, ale pouze je označuje za neaktivní, je i pro záznam o šabloně přítomno logické pole, které určuje, zda je šablona aktivní.

Pro popis typů dat je možné uložit jméno datového typu, jeho slovní popis a je vyžadováno *XSD* popisující tento typ dat. Vůči tomuto XSD jsou instance těchto typů dat validovány, aby nedocházelo k pokusu o vložení nesprávných dat do šablony. Primárním klíčem je tady zvoleno automaticky se zvyšující číslo.

Associace jsou vazby typu n:m a jsou tedy ošetřeny vazební tabulkou, která kromě dvou vzdálených klíčů obsahuje také případné XSLT pro mapování datového typu poskytovaného zdrojem dat na nativní datový typ.Pokud je pole XSLT prázdné, pak tento poskytovaný typ je přímo nativním typem šablony.

4.2.3 Platforma komunikace modulů aplikační vrstvy s klienty

Komunikace je založena na standardu *Web Services*, který definuje univerzální rozhraní pro libovolné programovací prostředí.

Na platformě *.NET* lze aplikace s rozhraním *Web Services vytvořit*, buď vytvořením aplikace *ASP.NET Web Services*[26], nebo lze použít rozhraní *Windows Communication Foundation*[27] (dále jen *WCF*).

[26] *ASP.NET Web Services* http://msdn.microsoft.com/en-us/library/yzbxwf53.aspx
[27] WCF http://msdn.microsoft.com/en-us/netframework/aa663324.aspx

Přestože *WCF* umožňuje vyvíjet aplikační servery jako služby a pouze k nim definovat rozhraní pro různé standardy síťové komunikace, bylo zákazníkem upřednostněno vyvinout systém přímo jako *Web Service*, protože s tímto vývojem již má zkušenosti.

I když lze v rozhraní *Web Services* použít libovolný typ platformy *.NET* musí být použity pouze základní typy, kterým bude rozumět libovolný klient zpracovávající webovou službu. Těmito základními typy se rozumí libovolný vlastní definovaný typ, který neimplementuje speciální rozhraní přítomná pouze v platformě *.NET* (příkladem je rozhraní *IQueryable*[28], jež umožňuje odložené vykonání dotazu), případně pole tohoto typu.

4.2.4 Generátor dokumentů

Generátor dokumentů pouze generuje dokumenty ze šablon a zaslaných dat. Jak bylo řečeno v sekci Generátor dokumentů na str. 17, musí být generátor schopen zpracovávat požadavky paralelně. Možnou paralelizaci dotazů na generátor dokumentů řeší webové služby automaticky, protože jsou bezkontextové a každý požadavek je zpracováván ve vlastním vlákně.

V případě že by docházelo k zahlcování generátoru dokumentů, může být do další verze systému implementována fronta na tvorbu dokumentů v součásti správa šablon.

Přes webové služby není možné vrátit najednou několik vytvořených dokumentů, proto obsahuje rozhraní generátoru dokumentů metody pro vyzvednutí jednotlivých vytvořených dokumentů. V případě delší neaktivity session, která žádala o vytvoření dokumentů, dojde ke smazání vytvořených dokumentů.

4.2.4.1 Algoritmus generování dokumentu

Algoritmus pro vytvoření dokumentu ze šablony musí být rekurzivní, protože se v šablonách vyskytují vnořené sekce. Každý typ sekce vyžaduje jiné zpracování, proto musí být rekurzivní krok závislý na nalezené sekci. Aby bylo v další verzi možné přidat další typy sekcí jednoduše, bude použita obdoba návrhového vzoru *dynamická rekurze* dle [8].

[28] Rozhraní *IQueryable* http://msdn.microsoft.com/cs-cz/library/system.linq.iqueryable.aspx

Podle zdroje je definován objekt pro každý typ rekurzivního kroku. Tento objekt dědí předka společného pro všechny typy rekurzivního kroku, který obsahuje hlavní krok rekurze. V tomto hlavním kroku rekurze je pomocí obdoby návrhového vzoru *faktory* vrácen objekt dalšího kroku rekurze nebo je rekurze ukončena.

Dále musí algoritmus obsahovat vyhodnocení vkládaných výrazů. Vzhledem k možnému použití i v jiných projektech pro platformu *.NET* je vyhodnocení výrazů implementováno v samostatném projektu. Jelikož je definice proměnných ve výrazech závislá na konkrétním projektu, bude čtení proměnných abstrahováno použitím rozhraní. Konkrétní objekt implementující toto rozhraní pak bude poskytován projektem, který výrazy používá.

4.2.5 Správa šablon

Správa šablon je napojena na centrální negarantované datové úložiště *DUL* informačního systému *Munis*. Toto napojení je realizováno pomocí rozhraní, které je implementováno třídou komunikující s datovým úložištěm. Takto lze zaměnit úložiště šablon a vytvořených dokumentů lehkým zásahem do aplikace.

Implementace třídy komunikující s datovým úložištěm je oddělena do vlastního projektu, pro případné použití do dalších modulů pro platformu *.NET*.

Pro vznikající přihlašovací službu je také připraveno rozhraní, které pokrývá nároky na ověření uživatele a jeho práv kladené správou šablon. Jelikož přihlašovací služba ještě není hotová, implementuje toto rozhraní jednoduchá třída, která ověřuje pouze na základě konstant.

Hlavní operace, které správa šablon zprostředkovává, vyžadují určitá uživatelská práva. Těmito operacemi jsou:

- Přidat šablonu do systému
- Editovat šablonu
- Odebrat šablonu ze systému
- Odeslat šablonu do generátoru dokumentů

Přidání šablony do systému, vyžaduje na vstupu šablonu a seznam zdrojů dat, na které se šablona váže. Před zavedením do systému je šablona zkontrolována vůči těmto

zdrojům, aby výrazy v ní obsahovaly pouze přístupy na platné proměnné. Dále by bylo třeba v transakci provést uložení šablony do centrálního datového úložiště *DUL* a zapsání dat o šabloně do databázové vrstvy šablonovacího systému. Transakce ale nejsou centrálním datovým úložištěm *DUL* podporovány, proto bude v transakci proveden pouze zápis dat o šabloně do databázové vrstvy. Tato transakce může být vrácena, pokud se nepodaří uložit šablonu do centrálního datového úložiště *DUL* nebo mohou nastat problémy při komunikaci s databází. Pokud dojde k přerušení transakce po úspěšném uložení do centrálního datového úložiště *DUL,* zůstanou v úložišti uloženy šablony, které nejsou nikde odkazovány.

Jelikož jsou šablony úzce spjaty s datovými typy, na které jsou navázány, nedovoluje editace šablony změnit seznam datových typů, který je na ni napojen. Umožňuje pouze editaci souboru šablony. Slouží tak především k úpravě vzhledové stránky šablony (přebývající mezera, či změna formátu odstavce), než prostředek k úplně změně šablony, k tomu je určena metoda přidat šablonu. Navíc původní šablona není nahrazena novou, pouze je označena v systému jako neaktivní, tento přístup kopíruje chování centrálního datového úložiště *DUL*, protože toto uložené data nikdy nemaže, pouze je označuje za neaktivní. Aby mohla být šablona editována, musí být na rozhraní správy šablon metoda pro stažení konkrétní šablony ze serveru do klienta.

Při odebrání šablony dojde ke smazání záznamu o šabloně z databázové vrstvy šablonovacího systému, ačkoliv šablona zůstane uložena v centrálním datovém úložišti *DUL*. Toto chování bylo při návrhu upřesněno zákazníkem.

Pro odeslání šablony do generátoru dokumentů, musí být šablona nejdříve vybrána podle aktuálního zdroje dat. Proto rozhraní navíc musí obsahovat metodu pro vrácení všech záznamů šablon, které lze aktuálně použít.

Správa šablon neumožňuje uživatelům definovat datové typy, ani je spravovat, protože datové typy musejí být podporovány v modulech, které je poskytují.

V první verzi není podporována tvorba několika dokumentů najednou ve vrstvě Správa šablon, její implementace byla zákazníkem zadána do příští verze.

4.2.6 Webový klient

Webové aplikace nad platformou *.NET* je možné tvořit pomocí dvou *frameworků* a to *ASP.NET WebForms*[29] nebo *ASP.NET MVC*[30]. Tyto dva frameworky nabízejí různý přístup při tvorbě webových aplikací.

Framework *ASP.NET WebForms* se snaží o přiblížení vývoje webových aplikací k desktopovým aplikacemi ve *Windows Forms*[31]. A je tedy založena na psaní obsluh (handlerů) k vyvolaným událostem (eventům) a na uchovávání *ViewState*[32], které slouží k uchování změn dat ve webových formulářích napříč postbacky. Tyto data jsou uchovány pomocí speciálních skrytých polí. Uchování stavu formulářů způsobuje ve velkých aplikacích značné zpomalení a těžkopádnost, kvůli objemu přenášených dat, které ani nemusí být potřeba.

ASP.NET MVC je novějším frameworkem jak *ASP.NET WebForms*. Je založen na architektuře *model-view-controller*[33]. Na rozdíl od *ASP.NET WebForms* umožňuje lepší kontrolu nad prováděnými operacemi, přenášenými daty a zobrazovaným HTML kódem ve webové aplikaci. Tím je vhodný i pro rozsáhlé webové aplikace.

Pro webovou aplikaci je použit framework *ASP.NET MVC*, jelikož je flexibilnější a lépe spravovatelný, díky své modularitě.

5 Programátorská příručka

Programátorská příručka je rozdělena na tři sekce. První z nich – Členění programu – pojednává o tom, jak je systém členěn z hlediska implementace. Druhá sekce – Rozbor volání typických operací – popisuje sled volání nejdůležitějších funkcí systému. A poslední část Jak upravit systém projednává možnosti úprav díky členění systému a častému využití rozhraních. Detailní vygenerovaná programátorská dokumentace se nachází na přiloženém CD.

[29] ASP.NET WebForms http://www.asp.net/web-forms/what-is-web-forms
[30] ASP.NET MVC http://www.asp.net/mvc/whatisaspmvc
[31] Windows Forms http://msdn.microsoft.com/en-us/library/8bxxy49h(VS.80).aspx
[32] ViewState http://msdn.microsoft.com/en-us/library/ms972976.aspx
[33] Model-view-controller http://cs.wikipedia.org/wiki/Model-view-controller

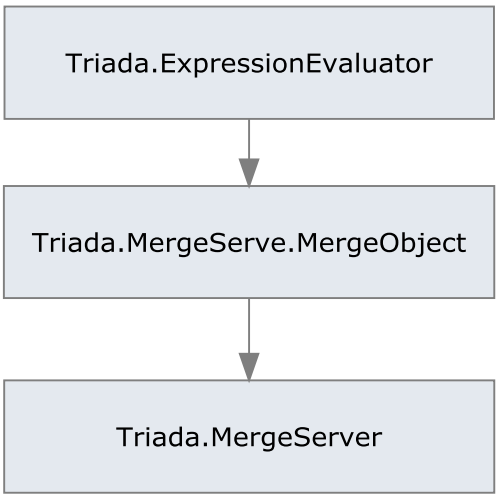
5.1 Členění programu

Sekce je rozdělena podle jednotlivých částí systému. U každé komponenty jsou uvedeny projekty, ve kterých je implementována. Pro každý projekt jsou uvedeny nejdůležitější objekty a jejich důležité funkce.

5.1.1 Generátor dokumentů

Aplikační server Generátoru dokumentů je rozdělen do tří projektů:

- *Triada.MergeServer* – Obsahuje rozhraní webové služby Generátor dokumentů a implementaci správy aktivních session
- *Triada.MergeServer.MergeObject* – Implementuje tvorbu dokumentů
- *Triada.ExpressionEvaluator* – Implementace vyhodnocení výrazů



Obr. 3 Schéma závislosti projektů části Generátor dokumentů

5.1.1.1 Triada.MergeServer

Projekt obsahuje dva hlavní objekty:

- *TriadaMergeService*
- *Instance*

TriadaMergeService

Třida implementující webovou službu a její rozhranní, spravuje všechny aktivní session pomocí instancí třídy *Instance*.

Nejdůležitější funkce:

- *CreateInstance ()* – vytvoří novou session, vrací *id* této nové session
- *UploadTemplate (sessionId, šablona)* – slouží k nahrání souboru šablony v parametru *šablona* k session identifikované parametrem *sessionId.*
- *UploadXMLData (sessionId, xmlData)* – slouží k nahrání xml souboru dat v parametru *xmlData* k session identifikované parametrem *sessionId.*
- *MergeDocuments (sessionId)* – vygeneruje dokumenty z nahrané šablony a dat v session identifikované parametrem *sessionId*, vrací počet vytvořených dokumentů.
- *DownloadDocument (sessionId, IdDokumentu)* – vrátí vytvořený dokument identifikovaný proměnnou *IdDokumentu* ze session identifikované parametrem *sessionId*.
- *DeleteInstance (sessionId)* – slouží k explicitnímu ukončení session

Instance

Třída pro uchování všech informací k jedné session.

Nejdůležitější veřejná pole třídy:

- *Template* – šablona, podle které budou tvořeny dokumenty.
- *DataXml* – data, která budou vložena do šablony
- *FinalDocuments* – po vygenerování dokumentů obsahuje všechny

Nejdůležitější funkce:

- *MergeIt ()* – vygeneruje dokumenty na základě vložených dat a vložené šablony pomocí třídy *TriadaMerge* v projektu *Triada.MergeServer.MergeObject*

5.1.1.2 Triada.MergeServer.MergeObject

Nejdůležitější třídy a rozhraní v projektu:

- Třída *TriadaMerge*
- Rozhraní *IMergeAction* a třídy jež jej implementují
- Rozhraní *IDataContext*

Veřejná Třída TriadaMerge

Je jediným veřejným členem, kterým je volána tvorba dokumentů.

Nejdůležitější veřejná pole třídy:

- *Template* – šablona, podle které budou tvořeny dokumenty.
- *DataXml* – data, která budou vložena do šablony
- *FinalDocuments* – po vygenerování dokumentů obsahuje všechny

Nejdůležítější funkce:

- *MergeIt ()* – Zahajuje generování dokumentů voláním soukromé funkce *createMergeDocument (RootChildData)* pro každého syna kořenového elementu dat
- *createMergeDocument (xmlData)* – Zahajuje rekurzivní algoritmus vytvořením objektu typu *MergeActionDefaultStep* nad kopií vložené šablony

Rozhraní IMergeAction

Je rozhraním pro implementaci rekurzivního algoritmu dle návrhového vzoru *dynamicka rekurze*.

Nejdůležitější funkce:

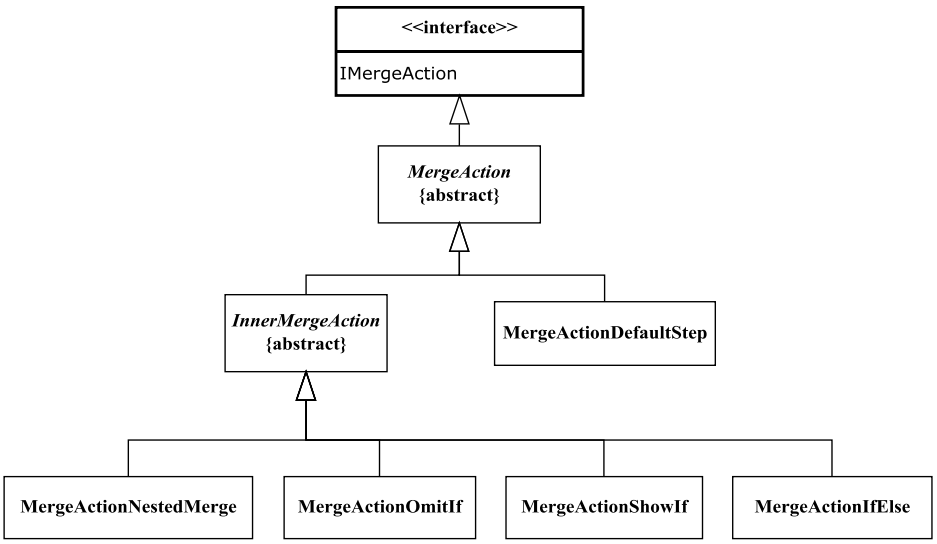
- *Execute (data)* – provede krok rekurze, vrátí část dokumentu s vyplněnými daty, které získává z parametru *data*

Abstraktní třídy implementující toto rozhraní:

- *MergeAction* – Implementuje společný rekurzivní krok v metodě *MergeStep*, je předkem pro všechny třídy implementující toto rozhraní
- *InnerMergeAction* – Společný předek pro akce nad vnitřní částí dokumentu definovanou mezi dvěma elementy

Třídy implementující toto rozhraní (za „–“ je uveden krok rekurze, kdy jsou použity):

- *MergeActionDefaultStep* – První krok rekurze
- *MergeActionNestedMerge* – Opakující se část šablony, která obsahuje další opakující se části
- *MergeActionOmitIf* – VYNECHKDYŽ část šablony
- *MergeActionIfElse* – IF část šablony
- *MergeActionShowIf* – ZOBRAZKDYŽ část šablony



Obr. 4 Schéma dědičnosti mezi objekty implementujícími *IMergeAction*

Rozhraní IDataContext

Je rozhraní pro abstrakci přístupu k datům.

Nejdůležitější funkce:

- *GetChildSectionsByName (jménoSekce)* – Vrátí všechny syny jména *jménoSekce* z aktuálního záznamu
- GetValue (*jménoPole*) – Vrátí hodnotu proměnné jména *jménoPole* dle definice v sekci Výrazy v šablonách na str. 29

Ve stávající verzi generování dokumentů je implementován pouze přístup do xml dat ve třídě *XmlDataContext.*

5.1.1.3 Triada.ExpressionEvaluator

Pro parsování výrazů je použit algoritmus tokenizace. Tokenizace probíhá pomocí dvou zásobníků. Jeden pro již přijaté tokeny a druhý pro odkládání operátorů z důvodu různých priorit. Rozpoznané tokeny operandů jsou okamžitě dávány na zásobník přijatých tokenů. Při rozpoznání tokenu operátoru je tento odložen na zásobník operátorů, ale před jeho vložením je zkontrolován vrchol zásobníku operátů, a pokud má operátor na vrcholu nižší prioritu jak přidávaný operátor, je vrchol zásobníku odebrán a vložen do zásobníku přijatých tokenů, tato operace je opakována, dokud na vrcholu zásobníku operátorů je operátorový token nižší priority. Po přečtení všech znaků výrazu jsou všechny operátory ze zásobníku operátorů postupně odebírány a

vkládány na zásobník přijatých tokenů. Na zásobníku přijatých tokenů nyní máme výraz v reverzní polské notaci (RPN), ze kterého již lehce sestavíme strom výrazu.

Po získání stromu výrazu stačí jen tento projít a vyhodnotit každý jeho uzel.

Nejdůležitější třídy a rozhraní v projektu:

- Třída Parser
- Rozhraní IVariableProvider
- Třída Tokenizer
- Rozhraní ITokenStack a jeho implemntace
- Třída BasicTokenFactory
- Rozhraní IToken, jeho rozšíření a implementace
- Rozhraní IFunctionFactory a jeho implementace
- Třída ExpressionTree
- Rozhraní *IExpressionNode*, jeho rozšíření a implementaceTřída Parser

Hlavní třída projektu. Nabízí rozhraní pro parsování a vyhodnocení výrazů. Spravuje již sestavené stromy vyhodnocení výrazu.

Nejdůležitější funkce:

- *EvalExpr (výraz)* – vyhodnotí *výraz*, vrací hodnotu v typu *object*, který lze metodou *ToString()* převést na textový ekvivalent hodnoty
- *EvalNumberExpr (výraz)* – vyhodnotí *výraz,* pokusí se jeho hodnotu převést na typ *double*
- *EvalStringExpr (výraz)* – vyhodnotí *výraz,* pokusí se jeho hodnotu převést na typ *string*
- *EvalBoolExpr (výraz)* – vyhodnotí *výraz,* pokusí se jeho hodnotu převést na typ *bool*
- *EvalDateTimeExpr (výraz)* – vyhodnotí *výraz,* pokusí se jeho hodnotu převést na typ *DateTime*

Rozhraní IVariableProvider

Slouží k abstrakci práce s proměnnými ve výrazech.

Nejdůležitější funkce:

- *GetVariableType (jménoProměnné)* – vrací typ proměnné *jménoProměnné*
- Funkce pro vrácení hodnoty proměnné podle jejího typu

Implementace tohoto rozhraní musí být poskytována projektem, který používá vyhodnocení výrazů.

Třída Tokenizer

Třída figurující jako proxy třída návrhové vzoru *Proxy*[34]. Implementace jejích metod je ve třídě *BasicTokenFactory*, která je zde abstrahována rozhraním *ITokenFactory*.

Nejdůležitější metody:

- *NextToken ()* – vrátí další token z výrazu, který je tokenizován

Rozhraní ITokenStack a jeho implementace

Definuje rozhraní pro implementaci zásobníku pro parsování výrazů pomocí tokenů tak, jak je popsáno v úvodu tohoto projektu. Rozhraní je použito pro zastřešení obou zásobníků používaných v algoritmu. Je implementován ve třídě *TokenStack*. Zároveň toto rozhraní slouží k vytvoření uzlů stromu výrazu ve správném pořadí.

Třída TokenFactory

Implementace parsování výrazu na tokeny. Vrací postupně tokeny, které se vyskytují ve výrazu, voláním metody *NextToken* ().

Rozhraní IToken, jeho rozšíření a implementace

Rozhraní je implementováno všemi typy tokenů. Rozšířeními tohoto rozhraní jsou *IExprToken* a *IOperatorToken*. Pro tokeny implementující *IExprToken* existuje typ uzlu stromu výrazu. Tokeny implementující *IOperatorToken* jsou operátory, které mají navíc svoji prioritu. Detailní informace o existujících třídách implementujících toto rozhraní a jejich dědičnostech lze nalézt v programátorské dokumentaci.

Třída ExpressionTree

Reprezentuje strom vyhodnocení výrazu.

[34] Návrhový vzor *Proxy* http://www.dofactory.com/Patterns/PatternProxy.aspx

Nejdůležitější funkce:

- Statická *BuildexpressionTree (poskytovatelProměnných, výraz)* – postaví vyhodnocovací strom výrazu *výraz* a předá mu *IVariableProvider poskytovatelProměnných*
- Metody pro vyhodnocení výrazu a převedení na určitý typ pokud to lze.

Rozhraní IExpressionNode, jeho rozšíření a implementace

Rozhraní *IExpressionNode* je implementováno všemy typy uzlů ve vyhodnocovacím stromu výrazu.

Nejdůležitější pole:

- *Tree* – strom do kterého uzel patří
- *Parent* – Otec tohoto uzlu

Nejdůležitější funkce:

- *Evaluate ()* – Vyhodnotí uzel a vrátí hodnotu jako nový *IExpressionNode*
- Funkce předepisující možné operace mezi uzly stromu výrazů

Rozhraní je rozšířeno pro každý typ, který se vyskytuje ve stromu výrazů. Základní implementací rozhraní *IExpressionNode* je abstraktní třída *ExprNode*, která předepisuje přepsání funkce *ToString ()*.

Detailní informace o existujících třídách implementujících toto rozhraní a jejich dědičnostech lze nalézt v programátorské dokumentaci.

Rozhraní IFunctionFactory a jeho implementace

Rozhraní je použito k abstrakci třídy vracející *IExpressionNode,* která bude zapojena do stromu výrazu, podle jména funkce.

Nejdůležitější funkce:

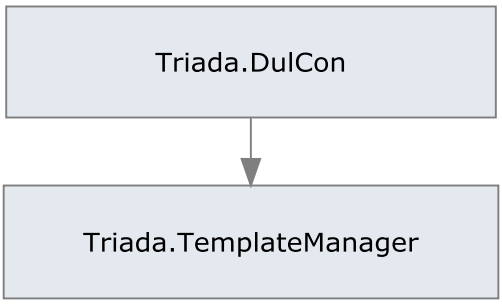
- *CreateFunctionNode (jménoFunkce, strom, rodič)* – Vrátí funkci jako uzel stromu výrazu podle *jménoFunkce* nastaví uzlu jeho *strom* a *rodiče*.

Tokeny typu funkce jsou aplikací rozpoznávány a toto rozhraní je používáno k vytvoření odpovídajících uzlů stromu výrazu, ale zatím není definovaný rozsah podporovaných funkcí.

5.1.2 Správa šablon

Aplikační server Správy šablon je rozdělen do dvou projektů:

- TemplateManager – Obsahuje rozhraní webové služby Správa šablon a jeho implementaci se všemi potřebnými typy.
- Triada.DulCon – Implementuje rozhraní pro přístup do centrálního úložiště



Obr. 5 Schéma závislosti projektů části Správa šablon

5.1.2.1 Triada.TemplateManager

Nejdůležitější třídy a rozhraní projektu:

- Třída *TriadaTemplateManager*
- Rozhraní *IAccessFiles*
- Rozhraní *IAccessToken*
- Třída *ConnectionToMergeServer*
- Třída *TriadaMergeService*
- Třída *SQLQuestioner*

Třída TriadaTemplateManager

Třida implementující webovou službu a její rozhranní, spravuje všechny aktivní session pomocí instancí třídy *Instance*.

Nejdůležitější funkce:

- *LogIn (jméno, heslo)* – provede přihlášení do přihlašovací služby, vrátí *identifikátor* přihlášeného uživatele
- *LogOut (identifikátor)* – provede odhlášení uživatele přihlašovací služby uživatelů, vrátí *true* pokud je odhlášení úspěšné
- *GetTemplateRecords ()* – vrátí informace o všech šablonách v systému
- *GetTemplateRecord (IDšablony)* – vrátí informace o šabloně *IDšablony*
- *AddTemplateRecord (šablona)* – přidá šablonu do systému
- *EditTemplateRecord (IDšablony, novaŠablona)* – upraví šablonu *IDšablony* na šablonu *novaŠablona*
- *DeleteTemplate (IDšablony)* – smaže šablonu *IDšablony*
- *DownloadTemplate (IDšablony)* – vrátí soubor šablony *IDšablony*
- *GetPossibleTemplateRecords (typDat)* – vrátí šablony použitelné s *typDat*
- *SendToConversion (sessionId)* – slouží k explicitnímu ukončení session

Rozhraní IAccessFiles

Definice rozhraní na datové úložiště.

Nejdůležitější funkce:

- *StoreDocument (dokument)* – uloží *dokument* do datového úložiště, vrátí nové *IDdokumentu*
- *GetDocument (IDdokumentu)* – vrátí dokument z datového úložiště dle parametru *IDdokumentu*

Implementováno třídou *DulAccessFiles*, která využívá projektu *DulCon* ke komunikaci s datovým úložištěm *DUL* informačního systému *Munis.*

Rozhraní IAccessToken

Definice rozhraní pro přihlašovací službu.

Nejdůležitější funkce:

- *Login (jméno, heslo)* – vrátí *identifikátor* uživatele, při úspěšném přihlášení
- *Logout (identifikátor)* – odhlásí uživatele dle *identifikátor* vrátí *true* při úspěchu
- Funkce pro ověření jednotlivých práv uživatele dle *identifikátoru*

Identifikátor, podle kterého je uživatel autentifikovat a autorizován, je předávám v hlavičkách zpráv požadavků na Správu šablon.

Třída ConnectionToMergeServer

Třída implementující postup pro volání webové služby Generování dokumentů.

Nejdůležitější funcke:

- *SendToConversion (šablona, vystupníFormát, xmlData)* – odešle *šablonu* s *xmldaty* do webové služby generování dokumentů a vrátí pole *byte* obsahující vygenerovaný dokument. Pokud *xmlData* vyžadují vytvoření několika dokumentů, jsou tyto na straně webové služby Generování dokumentů vytvořeny, ale funkce vrací první z nich.

Třída TriadaMergeService

Proxy třída pro komunikaci s webovou službou Generování dokumentů.

Třída SQLQuestioner

Obsahuje implementaci všech dotazů do databázové vrstvy na základě použití *ORM* mapování promocí *nHibernate*.

5.1.2.2 Triada.DulCon

Obsahuje funkce specifické pro uložení a přečtení souboru z centrálního datového úložiště *DUL* informačního systému *Munis.*

5.1.3 Webový klient

Je celý obsažen v jednom projektu *Triada.TemplateManager.Client*.

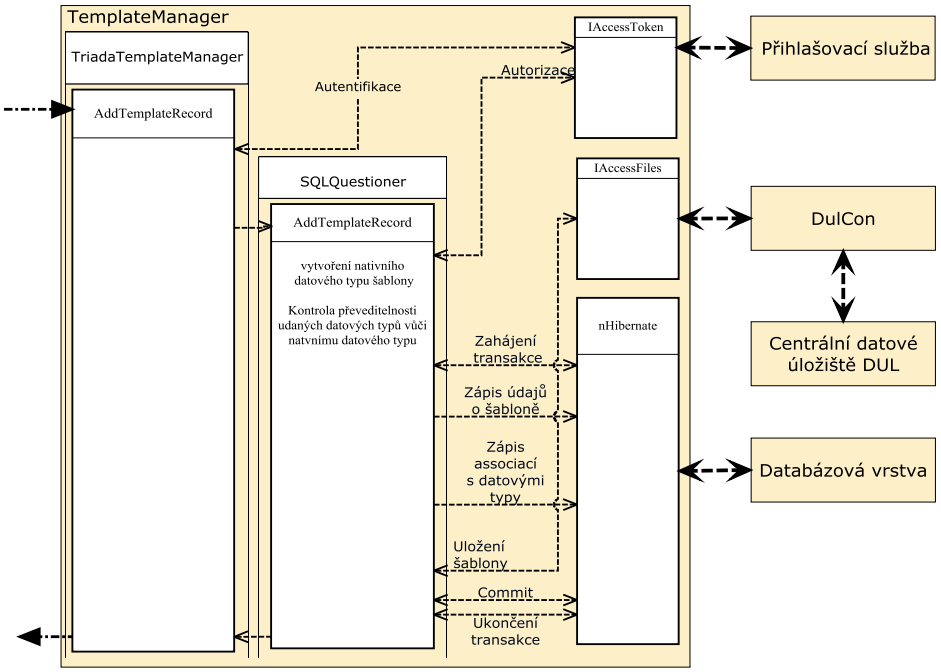
5.1.3.1 Trida.TemplateManager.Client

Aplikace typu *ASP.NET MVC*. Aplikace je grafickým rozhraním pro webovou službu Správa šablon.

5.2 Rozbor volání typických operací

Jelikož může být část webového klienta lehce nahrazena jiným grafickým rozhraním pro systém budou všechny rozbory volání začínat na rozhraní webové služby Správa šablon.

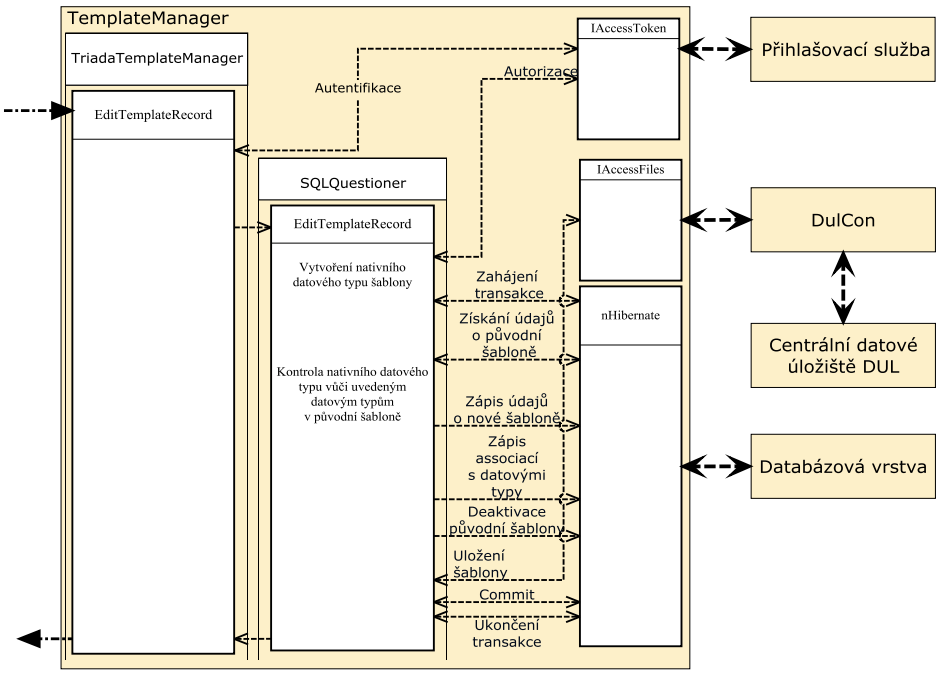
5.2.1 Přidání šablony



Obr. 6 Schéma volání během operace přidání šablony

Obr. 6 zobrazuje sled volání v systému během operace přidání šablony podle jeho definice v sekci Správa šablon na str. 37

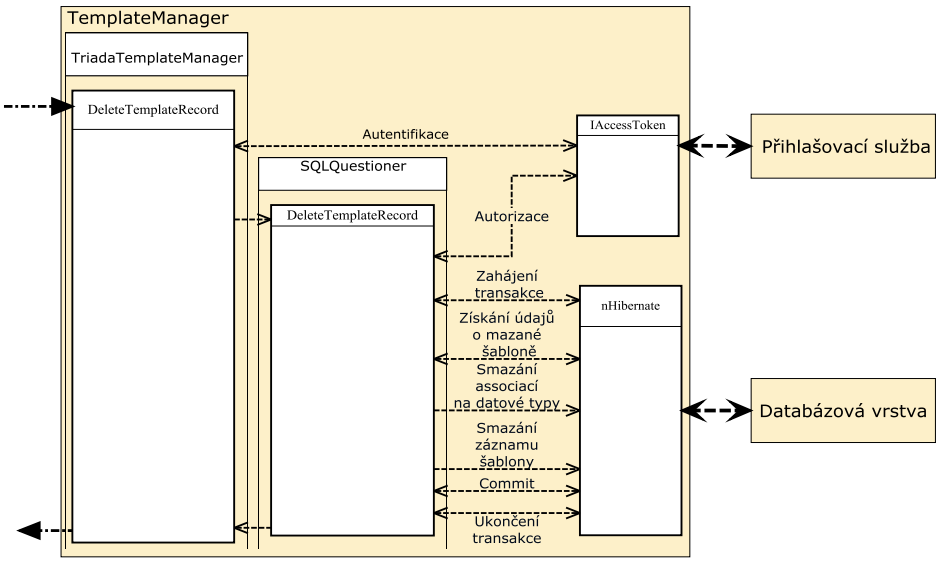
5.2.2 Editace šablony



Obr. 7 Schéma volání během operace editace šablony

Obr. 7 zobrazuje sled volání v systému během operace editace šablony podle jeho definice v sekci Správa šablon na str. 37

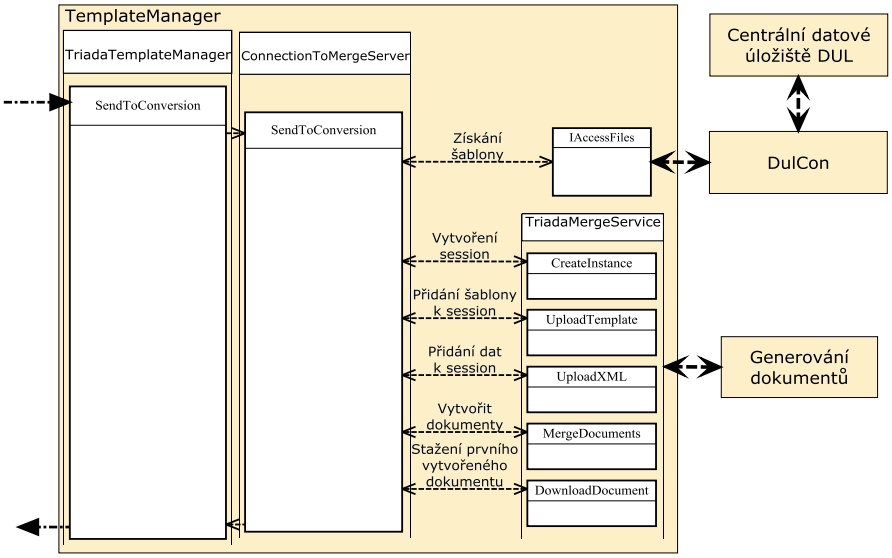
5.2.3 Odebrání šablony



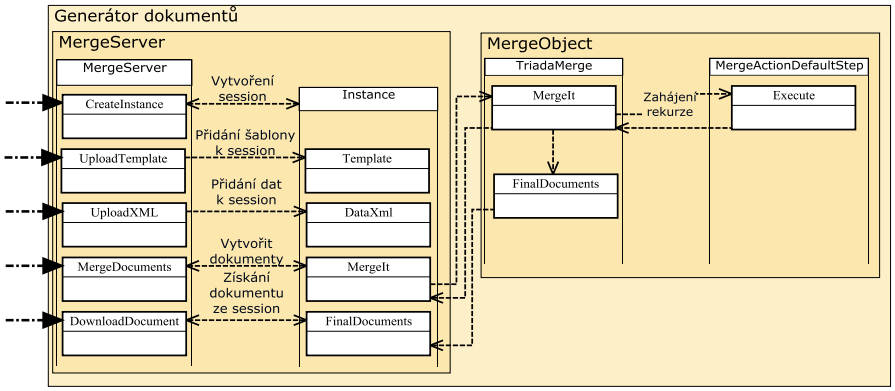
Obr. 8 Schéma volání během operace odebrání šablony

Obr. 8 Obr. 7zobrazuje sled volání v systému během operace editace šablony podle jeho definice v sekci Správa šablon na str. 37

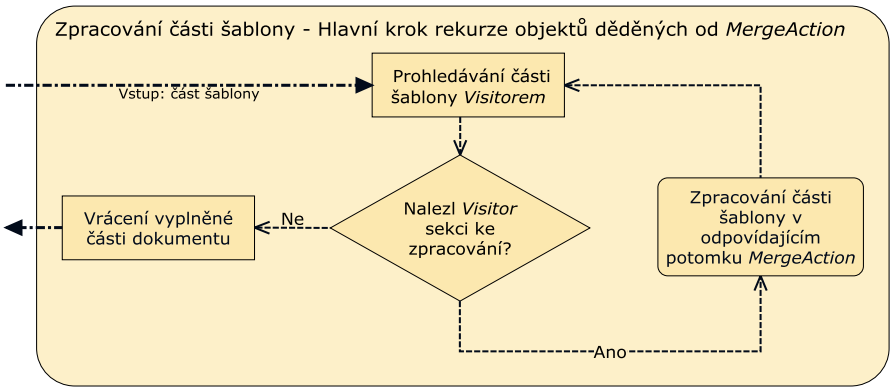
5.2.4 Vytvoření Dokumentu



Obr. 9 Schéma volání během operace vytvoření dokumentu v části Správa šablon



Obr. 10 Schéma volání během operace vytvoření dokumentu v části Generátor dokumentů



Obr. 11 Schéma hlavního rekurzivního kroku tvorby dokumentů

Obr. 9 zobrazuje sled volání v části správa šablon pro vytvoření dokumentu. Obr. 10 zachycuje jak jsou volání zpracovávána Generátorem dokumentů a Obr. 11 zobrazuje schéma hlavního rekurzivního kroku tvorby dokumentu.

5.3 Jak upravit systém

Díky častému využití rozhraní lze většinu základních chování systému lehce změnit. V této kapitole jsou rozepsány některé z těchto možných změn.

5.3.1 Změna poskytovatele proměnných

Poskytovatel proměnných je ve vyhodnocení výrazů v projektu *ExpressionEvaluator* používán pro zjištění hodnoty a typu proměnné určitého jména. V implementaci je užíván prostřednictvím rozhraní *IVariableProvider*. Toto rozhraní musí být předáno při vytváření objektu *Parser*, proto vlastní implementací tohoto rozhraní můžeme změnit, odkud jsou proměnné čteny a jak je vyhodnocováno jejich jméno.

5.3.2 Změna tokenizeru

Třida *Tokenizer* slouží k vytváření tokenů z výrazu, který má být vyhodnocen, a je implementován pouze jako proxy objekt k rozhraní typu *TokenFactory*. Do tohoto rozhraní je inicializován objekt *BasicTokenFactory* v konstruktoru třídy *Tokenizer*. Implementací rozhraní *ITokenFactory* v nové třídě a změnou inicializace v konstruktoru třídy *Tokenizer* můžeme nahradit stávající způsob vytváření tokenů.

5.3.3 Rozšíření rozpoznávaných výrazů

Výrazy mohou být rozšířeny dvěma způsoby. Prvním je rozpoznávání nového elementu ve výrazu, který ale nemá ekvivalent uzlu ve stromě vyhodnocení výrazu, ale pouze mohou měnit jeho strukturu atp. (příkladem mohou být znaky „(“ a „)“). Druhým pak je nový element, který má ekvivalent ve stromu výrazu.

Pro implementaci nového elementu výrazu, který není ve výsledném stromě, stačí v nové třídě implementovat rozhraní *IToken*, případně *IOperatorToken* pokud má mít token určitou prioritu při sestavování výrazů. Dále je potřeba rozšířit třídu *BasicTokenFactory* aby tento nový token vytvářela a poslední třída, která bude muset být rozšířena, je *TokenStack,* který zajišťuje správné řazení tokenů pro vyhodnocení v postfixu.

Pro implementaci druhého typu nového elementu je potřebné udělat vše popsané pro první element a navíc musí tato třída implementovat rozhraní *IExprToken*, který říká že daný token má ekvivalent v uzlech stromu výrazu. A vytvořit nový objekt implementující rozhraní *IExpressionNode* a případné další rozhraní podle typu objektu. Všechna rozhraní jsou popsány v programátorské dokumentaci v příloze.

5.3.4 Definování nové sekce v šabloně

Pro rozpoznání nové sekce v šabloně je potřebné, aby tato byla rozpoznávána v třídě *MergeSectionVisitor*, což je třída procházející dokument pomocí návrhového vzoru *Visitor*. Tento hledá v dokumentu sekce a při nalezení sekce se zastaví, tuto sekci vrátí a pokračuje v prohledávání šablony.

Třída implementující rekurzivní krok pro danou sekci šablony musí implementovat rozhraní *IMergeAction*. A ve své funkci *Execute* musí volat buď metodu *MergeStep*

třídy *MergeAction* (metoda je *protected*, proto by měl třídu dědit), nebo implementovat další krok rekurze sám.

5.3.5 Změna datového úložiště

Implementaci datového úložiště lze nahradit vlastní třídou implementující rozhraní *IAccessFiles*. Datové úložiště je sdíleno do celé aplikace Správa šablon z třídy *TriadaTemplateManager* ve statické proměnné *accessFile*. Pokud dosazení zde změníme, změníme tím chování v celé aplikaci.

5.3.6 Změna přihlašovací služby

Implemantaci přihlašovací služby lze nahradit vlastní třídou implementující rozhraní *IAccessToken*. Přihlašovací služba je sdílena do celé aplikace Správa šablon z třídy *TriadaTemplateManager* ve statické proměnné *loginManager*. Pokud dosazení zde změníme, změníme tím chování v celé aplikaci.

6 Uživatelská příručka

V kapitole uživatelská příručka je uveden postup instalace systému, popis tvorby šablon a popis uživatelského prostředí tvorby šablon.

6.1 Instalace systému

6.1.1 Systémové požadavky

Pro provoz systému je vyžadován operační systém *Microsoft Windows XP* a vyšší nebo serverový operační systém *Microsoft Windows Server 2003* a vyšší s nainstalovaným *.NET Framework* verze 3.5 SP1. Instalační soubor prostředí .NET je přiložen na CD ve složce Requirements. Dále je vyžadován webový server *IIS* nebo *Apache*. Pro provoz databázové vrstvy je vyžadován *Microsoft SQL Server 2005 Express* a vyšší nebo systém *Oracle* verze 9i a vyšší. Popis instalace se věnuje pouze instalaci na *Microsoft SQL Server 2005 Express*, který je nabízen zdarma a jeho instalátor je rovněž přiložen na CD, a webovém serveru *IIS* verze 7, který je součástí *Microsoft Windows 7 Professional* a vyšší.

6.1.2 Postup instalace

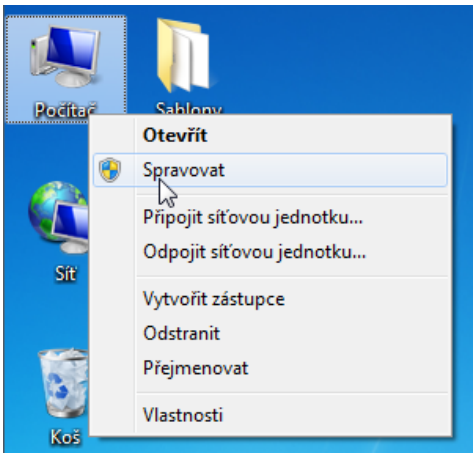
Postup instalace pokrývá pouze instalaci všech součástí systému centralizovaně na jeden stroj – webový server. Změny, potřebné pro instalaci na různé stroje pro využití jiného databázového serveru nebo vzdálené instance serveru jsou popsány v kapitole Rozšíření instalace.

Nejdříve zkopírujeme složky TriadaMergeServer, TriadaTemplateManager, MunisTDok ve složce Aplikace na přiloženém CD na místní disk do umístění ze kterého budeme provozovat webové služby (nejčastěji C:\inetpub\wwwroot). Poté musí být do webového serveru *IIS* přidány tři webové aplikace, jedna pro každou zkopírovanou složku.

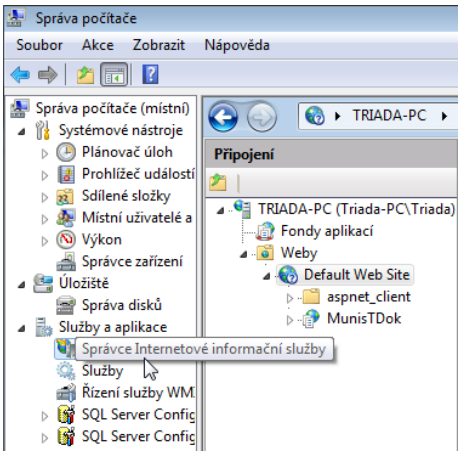
Založení webové aplikace ve webovém serveru *IIS* verze 7:

1. Spustíme *Správce Internetové informační služby* – například přes kontextové menu pro *Počítač* (kontextové menu vyvoláme klikem pravého tlačítka myši na

ikonu Počítač), položka *Spravovat*. V levé části obrazovky vybrat položku Služby *a aplikace -> Správce Internetové Informační Služby*.

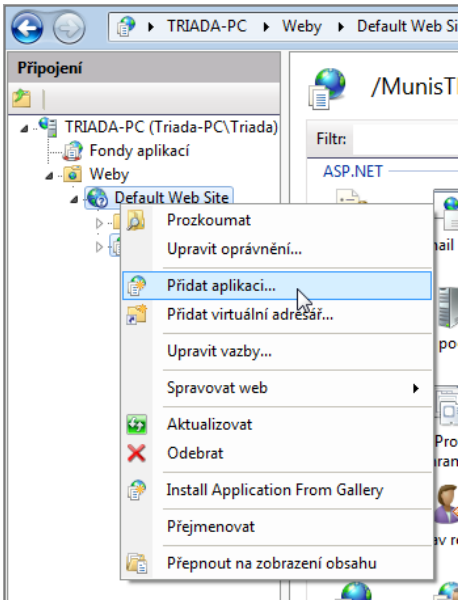


Obr. 12 Položka Spravovat kontextového menu Počítač



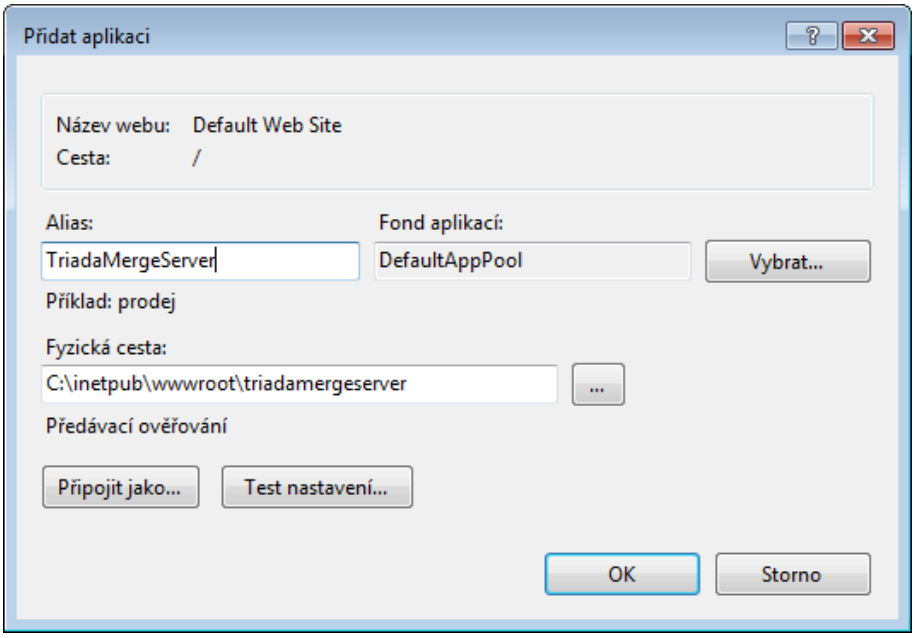
Obr. 13 Položka Správce Internetové informační služby

Pokud položka Správce Internetové informační služby není zobrazena, nejspíše není *IIS* nainstalována.



Obr. 14 Přidání aplikace do webového serveru IIS

2. V kontextovém menu položky *Default Web Service* vybereme položku Přidat aplikaci…



Obr. 15 Dialog Přidat aplikaci

3. V dialogu Přidat aplikaci vyplňte pole Alias: jménem zkopírované složky a do pole fyzická cesta napište nebo vyberte cestu k dané složce a potvrďte OK. Do pole alias může být uveden libovolný název, ale musí být změněno nastavení aplikace viz kapitola Rozšíření instalace.

Vzhledem k tomu, že bakalářská práce neobsahuje externí komponenty systému Munis, bylo připojení do datového úložiště nahrazeno jednoduchou třídou, jež čte a zapisuje šablony na místní disk. Pro tuto třídu je nutné nastavit v souboru *web.config* v součásti TriadaTemplateManager hodnotu value položky <appSettings><add key="CestaSoubory" value="C:/Sablony"> na absolutní cestu na místním disku, odkud budou šablony čteny a kam budou zapisovány. Této nastavené složce musí být přidělena práva pro čtení a zápis uživatele ASP.NET a musejí do ní být zkopírovány soubory šablon ze složky Aplikace\Příklady\Šablony na přiloženém CD.

Databázová vrstva se automaticky připojuje k instanci SQLEXPRESS. Proto pokud je nainstalován Microsoft SQL Server 2005 Express v základním nastavením s instancí sql serveru jménem SQLEXPRESS, mělo by být nyní vše připraveno pro provoz systému.

6.1.3 Rozšíření instalace

Pro instalaci součástí na různé webové servery je potřeba správně změnit adresu připojení dané součásti v souboru *web.config*, který se nachází u každé aplikace. Příklad 8 červený text zobrazuje, kterou položku změnit.

```
<applicationSettings>
   <Triada.TemplateManager.Properties.Settings>
      <setting name="TriadaTemplateManager_TriadaMergeService_TriadaMergeService"
         serializeAs="String">
         <value>http://localhost/triadamergeserver/triadamergeservice.asmx</value>
      </setting>
   </Triada.TemplateManager.Properties.Settings>
</applicationSettings>
```

Příklad 8 Adresa připojení v souboru *web.config*

Hodnotu změňte na *síťovaAdresaCíle/aliasWebovéAplikace/triadamergeservice.asmx*, kde *síťováAdresaCíle* je síťové jméno nebo IP adresa, *aliasWebovéAplikace* je alias zadávaný při přidání aplikace do *IIS*, poslední položka *triadamergeservice.asmx* je pevně daná pro každou součást systému následujícími hodnotami:

- TriadaMergeServer – triadamergeservice.asmx
- TriadaTemplateManager – triadatemplatemanager.asmx.

Je možné databázi provozovat na jiné instanci SQL Serveru. Pro změnu instance SQL Serveru musí být změněna položka *connection_string* v souboru *hibernate.cfg* v součásti TriadaTemplateManager.

```
<property name="connection.connection_string">
Data Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\TriadaTemplateManager.mdf;
Integrated Security=True;User Instance=False
</property>
```

Příklad 9 Položka *connection_string* souboru *hibernate.cfg*

Také je možné databázi připojit ke konkrétnímu databázovému serveru, soubory databáze se nachází v součásti TriadaTemplateManager ve složce App_Data.

Správný tvar nového *connection_string* zjistíte například na stránkách MSDN[35].

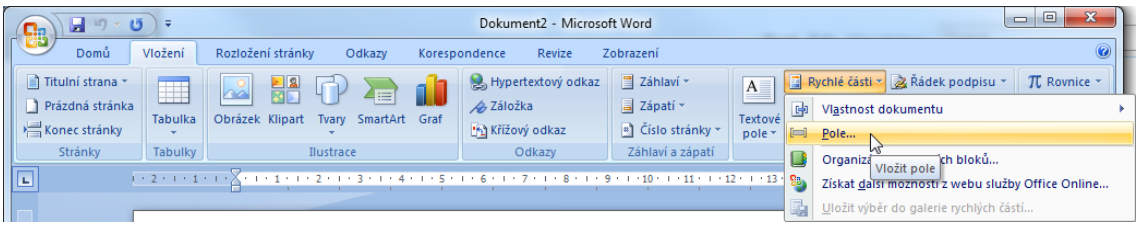
Pro provoz na jiném databázovém serveru než *Microsoft SQL Server 2005* je potřebné změnit další nastavení v souboru *hibernate.cfg*, bližší informace v dokumentaci projektu *nHibernate*.

6.2 Tvorba šablony

Všechny výrazy používané v šablonách se vkládají do tzv. *„MergeField“* aplikace *Microsoft Office Word*. Nejdříve bude ukázáno jak takové pole přidat. Dále již budou popsány možnosti obsahu těchto polí.

6.2.1 Vložení *MergeField* do šablony

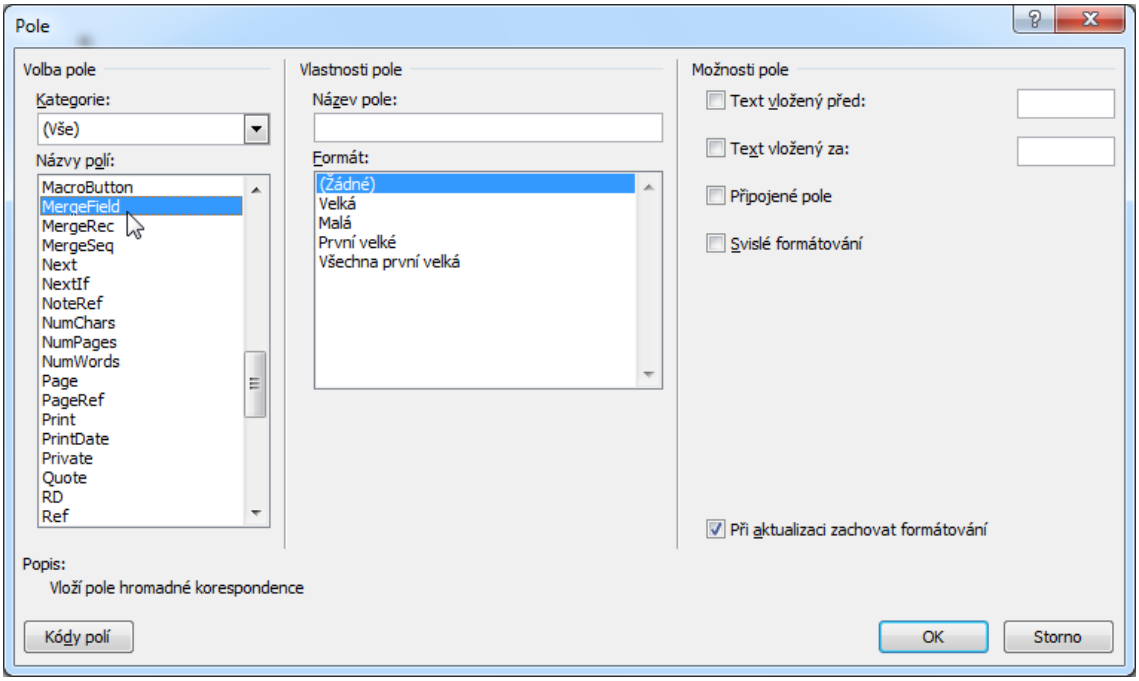
Abychom vložili *MergeField* do dokumentu v aplikaci *Microsoft Office Word 2007,* vybereme na pásu karet kartu *Vložení*. Zde v nástroji *Text* vybereme položku *Rychlé části* a dále její podpoložku *Pole.*



Obr. 16 Vložení pole do dokumentu

V zobrazeném dialogovém okně *Pole* v části *Volba pole* vyberte položku *MergeField*.

[35] http://msdn.microsoft.com/en-us/library/ms722656(VS.85).aspx



Obr. 17 Výběr typu pole a možnosti

Název pole je vlastnost pole, do které jsou vkládány výrazy popisované kapitole Podporované elementy v *MergeField*. U pole můžeme také zvolit jeho formátování, které bude pro vyhodnocené výrazy ve výsledném dokumentu zachováno.

6.2.2 Podporované elementy v *MergeField*

Šablony podporují vyhodnocení výrazů na základě dat, sekci opakující se pro každou položku určitého seznamu v datech, a dvě různé podmíněné sekce.

6.2.2.1 Výrazy

Výrazy se skládají z operandů a operátorů. Operandy jsou čtyř typů: číslo, řetězec, datum a logická hodnota.

Typy operandů mohou být konstanty nebo proměnné, které jsou ekvivalenty konkrétní položky v datech.

Zápis konstant do šablony se řídí následujícími pravidly:

- Číselné konstanty jen jako číslo se znakem „.“ oddělující desetinnou část
- Datum je uvezeno znakem ‚#‘ ve tvaru *#datum čas*. *Datum* ve tvaru *dd.mm.rrrr*, kde *dd* je dvoumístné číslo dne, *mm* dvoumístné číslo měsíce a *rrrr* čtyřmístné číslo roku. Čas ve tvaru *hh:mm:ss*, kde *hh* je dvoumístné číslo hodiny, *mm* je dvoumístné číslo minut a *ss* je dvoumístné číslo sekund. Části *datum* a *čas* jsou

volitelné a přítomna musí být alespoň jedna z nich. Platnými daty tedy jsou: *#1.1.1999*, *#13:00:00* i *#1.1.1999 13:00:00*

- Řetězcové konstanty jsou obklopeny znakem ,"', výskyt znaku ,"' uvnitř řetězce musí předcházet znak ,\'. Příklad takového řetězce je: *„přezdívaný \"Hrozba\""*, který bude zobrazen jako: *přezdívaný „Hrozba"*. Aby byl znak ,\' tištěn musí být zdvojen. Například řetězec *„C:\\Data"* bude zobrazen jako: *C:\Data*
- Logické hodnoty PRAVDA a NEPRAVDA, tak jak jsou zde uvedeny

«4.3» «#01.01.1993 12:45» «"řetězec"» «PRAVDA» «NEPRAVDA»

Příklad 10 Konstantní pole v dokumentu

Proměnná se zapisuje ve tvaru *CeléJménoZanoření.Položka*. *CeléJmenoZanoření* jsou jména seznamů zřetězené znakem „.", odpovídající struktuře dat od kořene až po předka, ve kterém se nachází položka, na kterou chceme přistoupit.

```xml
<?xml version="1.0" encoding="utf-8"?>
<ZakazniciSpolecnostiSObjednavkamiDleSekciDocuments
xmlns="http://www.triada.cz/ZakazniciSpolecnostiSObjednavkamiDleSekciDocuments">
   <ZakazniciSpolecnostiSObjednavkamiDleSekci>
      <JmenoSpolecnosti>Fiktivní společnost s.r.o.</JmenoSpolecnosti>
      <UliceSpolecnosti>Masarykova 15</UliceSpolecnosti>
      <MestoSpolecnosti>Praha</MestoSpolecnosti>
      <PSCSpolecnosti>120 00</PSCSpolecnosti>
      <Zakaznik>
         <Jmeno>Jan</Jmeno>
         <Prijmeni>Novák</Prijmeni>
         <Ulice>U Kováře 15</Ulice>
         <Mesto>Tábor</Mesto>
         <PSC>123 45</PSC>
         <Objednavka>
            <Cislo>1</Cislo>
            <Datum>1.1.1993</Datum>
            <Polozka>
               <Jmeno>Čaj</Jmeno>
               <Popis>Popis čaje</Popis>
               <PocetKusu>110</PocetKusu>
            </Polozka>
            ...
         </Objednavka>
         ...
      </Zakaznik>
      ...
   </ZakazniciSpolecnostiSObjednavkamiDleSekci>
   ...
</ZakazniciSpolecnostiSObjednavkamiDleSekciDocuments>
```

Příklad 11 Vkládaná data

Příklady používají jako data Příklad 11 Vkládaná data:

- Pro přístup na jméno zákazníka použijeme syntax *Zakaznik.Jmeno*
- Pro přístup na číselnou položku Číslo objednávky použijeme syntax *Zakaznik.Objednavka.Cislo*

Typ proměnné je odvozen od typu odkazované položky.

Podporovány jsou operátory sčítání: +, -, operátory násobení: *, /, % (modulo), operátory rovnosti: ==, !=, porovnávací operátory: <=, >=, <, >, operátory priority: (,), a logické operátory: & (and), | (or). Dále jsou podporovány také unární operátory - (minus) a ! (not). Operátory jsou vyhodnocovány v obvyklém pořadí.

S číselnými hodnotami můžeme použít operátory:

- +, -, *, / a % výsledkem je opět číselná hodnota
- ==, !=, <=, >=, <, > výsledkem je logická hodnota

Příklady výrazů s číselnými hodnotami:

- 5 + 4 - - 3 výsledkem je hodnota 12
- 9/3 výsledkem je hodnota 3
- 4 + 6/2 výsledkem je hodnota 7
- 4 <= 4 výsledkem je hodnota PRAVDA
- 5 > 1 výsledkem je hodnota NEPRAVDA

S řetězcovými hodnotami můžeme použít operátory:

- + pro zřetězení dvou hodnot výsledek je opět řetězec
- ==, !=, <=, >=, <, > výsledkem je logická hodnota, řetězce jsou porovnávány lexikograficky

Příklady výrazů s řetězcovými hodnotami:

- „ah“ + „oj“ výsledkem je hodnota „ahoj“
- „aa“ < „az“ výsledkem je hodnota PRAVDA
- „aa“ < „b“ výsledkem je hodnota PRAVDA
- „a“ < „A“ výsledkem je hodnota PRAVDA
- „aa“ < „A“ výsledkem je hodnota NEPRAVDA
- „aA“ < „A“ výsledkem je hodnota NEPRAVDA
- „a“ < „B“ výsledkem je hodnota PRAVDA

S logickými hodnotami můžeme použít operátory:

- ==, !=, &, |, ! výsledkem je opět logický hodnota

Příklady výrazů s logickými hodnotami:

- PRAVDA == PRAVDA výsledkem je PRAVDA
- PRAVDA != PRAVDA výsledkem je NEPRAVDA
- PRAVDA & NEPRAVDA výsledkem je NEPRAVDA
- PRAVDA | NEPRAVDA výsledkem je PRAVDA
- !PRAVDA výsledkem je NEPRAVDA

S hodnotami typu datum můžeme použít operátory:

- ==, !=, <=, >=, <, > výsledkem je logická hodnota

Příklady výrazů s hodnotami typu datum:

- #13.03.2010 14:55:00 == #13.03.2010 14:56:00 výsledkem je NEPRAVDA
- #13.03.2010 14:55:00 != #13.03.2010 14:56:00 výsledkem je PRAVDA
- #13.03.2010 14:55:00 <= #13.03.2010 14:56:00 výsledkem je PRAVDA
- #13.03.2010 14:55:00 >= #13.03.2010 14:56:00 výsledkem je NEPRAVDA
- #13.03.2010 14:55:00 < #13.03.2010 14:55:00 výsledkem je PRAVDA

6.2.2.2 Opakující se sekce

Šablonu, ve které se bude určitá sekce opakovat pro každou položku určitého seznamu v datech, vytvoříme tak, že tuto sekci obklopíme dvěma poli. Začátek této sekce je označen polem s hodnotou *TableStart:NázevSekce* a konec sekce je označen polem *TableEnd:NázevSekce*. *NázevSekce* je jméno seznamu v aktuálním záznamu.

6.2.2.3 Sekce VYNECHKDYŽ

Šablonu, ve které se bude určitá sekce vynechána za určité podmínky, vytvoříme tak, že tuto sekci obklopíme dvěma poli. Začátek této sekce je označen polem s hodnotou *VYNECHKDYŽ:Podmínka* a konec sekce je označen polem *ENDVYNECHKDYŽ:*. *Podmínka* je výraz, jehož výsledkem je logická hodnota. Pokud je hodnota *výrazu* rovna PRAVDA, bude tato sekce dokumentu vynechána.

6.2.2.4 Sekce ZOBRAZKDYŽ

Šablonu, ve které se bude určitá sekce zobrazena za určité podmínky, vytvoříme tak, že tuto sekci obklopíme dvěma poli. Začátek této sekce je označen polem s hodnotou *ZOBRAZKDYŽ:Podmínka* a konec sekce je označen polem *ENDZOBRAZKDYŽ:*. *Podmínka* je výraz, jehož výsledkem je logická hodnota. Pokud je hodnota *výrazu* rovna PRAVDA, bude tato sekce dokumentu zobrazena.

6.3 Uživatelské prostředí

Uživatelské prostředí – webová aplikace – je přístupné (pokud je systém nainstalován ve výchozí konfiguraci) přes webový prohlížeč na adrese http://localhost/MunisTDok/.

Pro účely bakalářské práce je k dispozici prostředí, umožňující správu šablon a jejich odeslání pro zpracování. Nejsou k dispozici asociace s jinými než nativními datovými typy a výběry šablon podle aktuálního datového typu. Rozhraní správy šablon je určeno pro odborně vyškolené pracovníky.

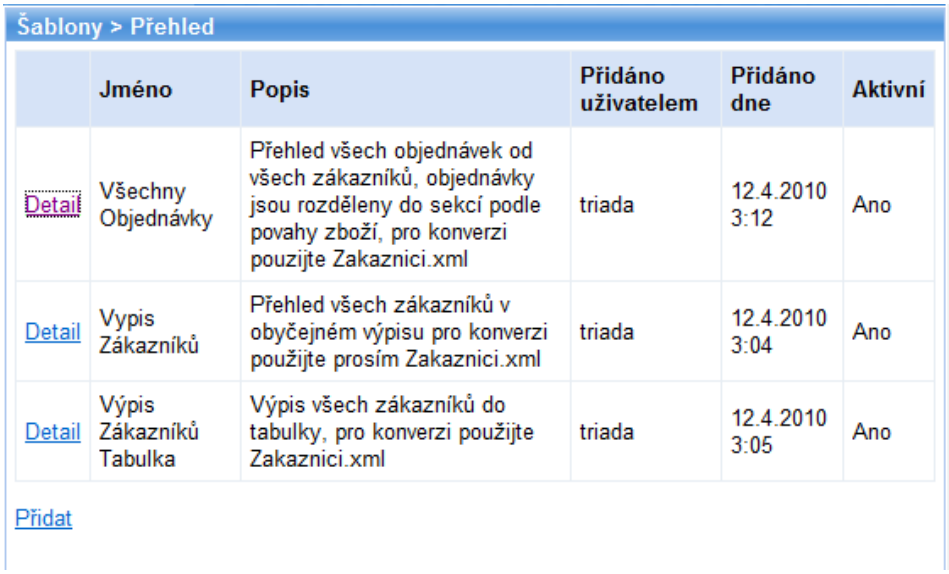
Obr. 18 Přihlašovací dialog

Před použitím je potřebné se do systému přihlásit, pro předvedení v bakalářské práci je k dispozici uživatel „a“ s heslem „a“ (bez uvozovek). Tento uživatel má přiděleny všechny práva pro práci se soubory šablony.

Obr. 19 Menu webového rozhraní

Na Obr. 19 je zobrazeno menu, které se nachází v levé části webové stránky, položky menu umožňují navigaci ve webové aplikaci

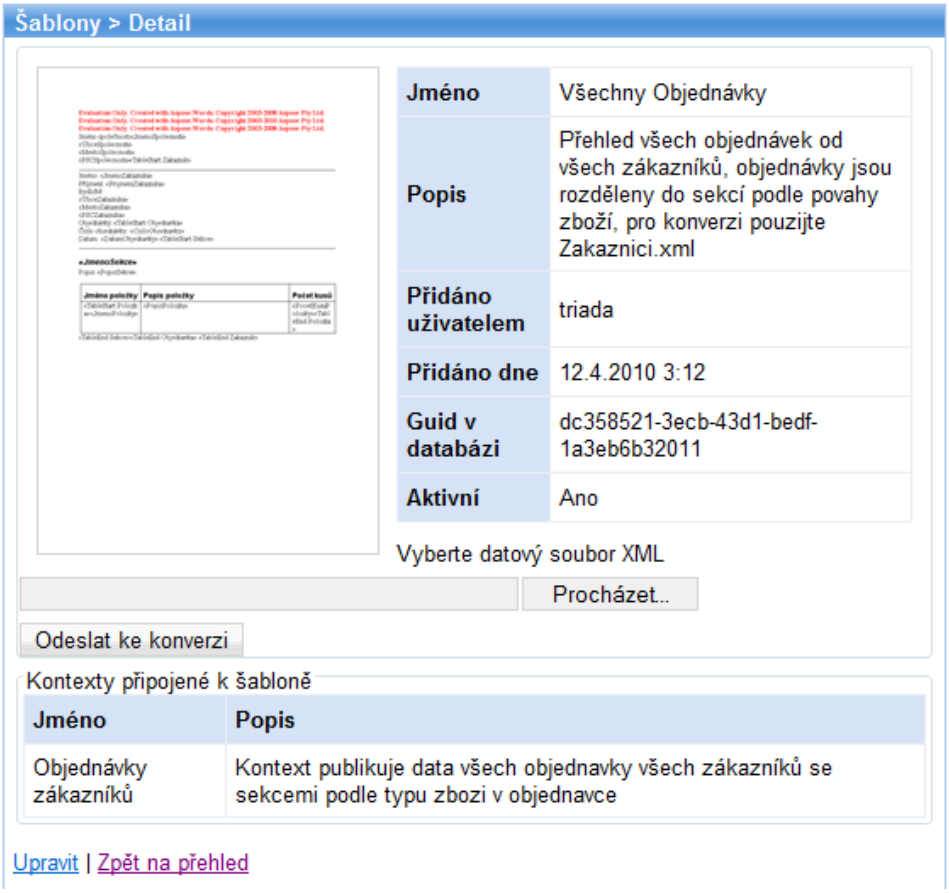
- Šablony
    - Přehled – seznam šablon, umožňuje odeslání šablony ke zpracování
    - Přidat – přidání nové šablony do systému
- Kontexty
    - Přehled – seznam datových typů poskytovaných aplikacemi
- Systém
    - O Systému – zobrazení informací o systému
- Účet
    - Odhlásit – provede odhlášení uživatele ze správy šablon



Obr. 20 Přehled dostupných šablon

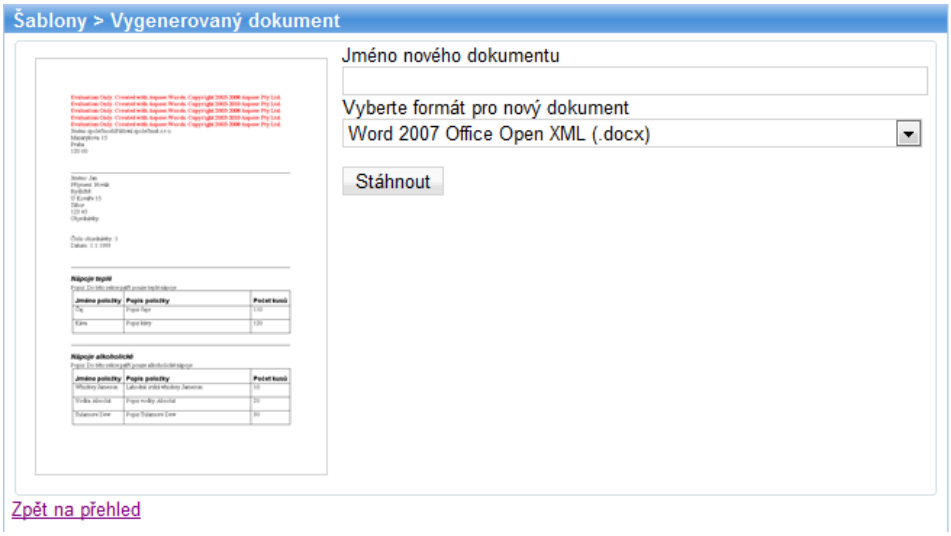
Na stránce Šablony > Přehled je zobrazen seznam všech dostupných šablon, spolu s odkazy na jejich detailní výpis a odkazem pro přidání šablony.

Na detailu šablony je zobrazen její náhled a jsou uvedeny informace o šabloně uchovávané v systému. Důležitou položkou jsou kontexty připojené k šabloně, což je seznam datových typů, se kterými lze šablonu použít. Jelikož systém ještě není plně nasazen a nejsou definovány datové typy informačního systému *Munis* je pro potřeby bakalářské práce tato informace uvedena pouze pro ilustraci budoucího chování systému a data v ní zobrazené jsou fiktivní.



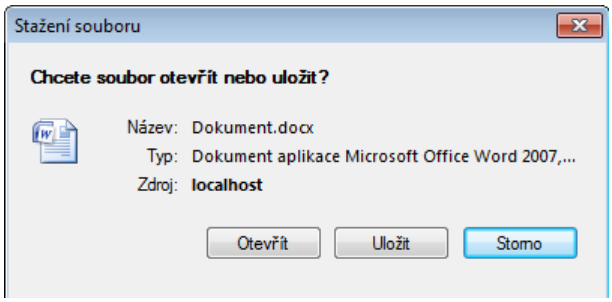
Obr. 21 Detail šablony

Po zobrazení detailu šablony lze tuto po přiložení správného datového XML stisknutím tlačítka *„Odeslat ke konverzi“* odeslat pro zpracování. Datové XML jsou umístěny ve složce Aplikace\Příklady\Data, v popisu šablony je vždy uveden název datového souboru XML, který má být použit pro vygenerování textového dokumentu.

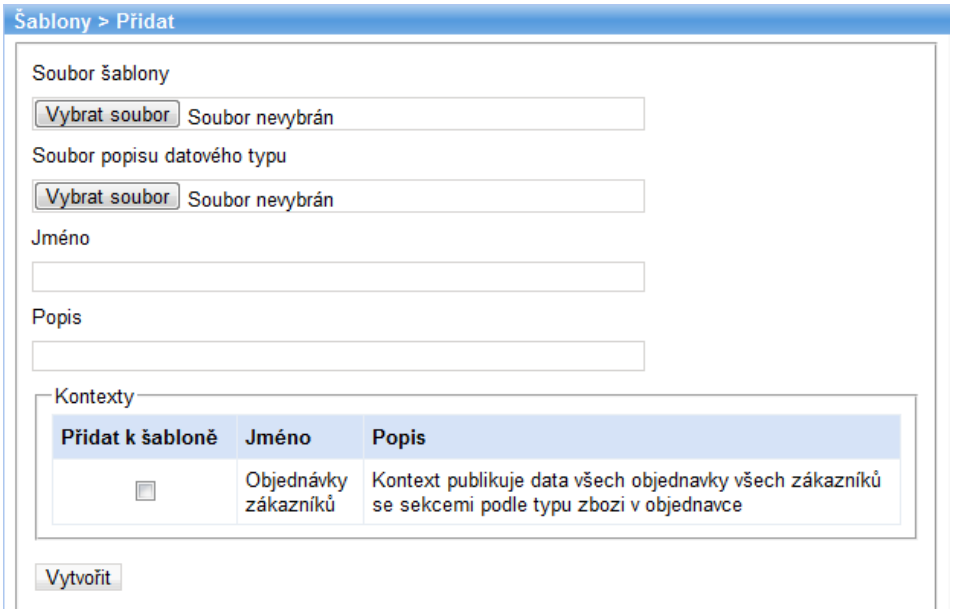


Obr. 22 Výběr formátu výsledného textového dokumentu

Vygenerování textového dokumentu je provedeno na serveru a výsledek je zaslán zpět do klienta. Náhled vygenerovaného dokumentu je zobrazen na stránce Šablony > Vygenerovaný dokument. Po výběru formátu, ve kterém bude dokument uložen, a zadání jména, pod kterým bude dokument uložen na místní disk, lze stiskem tlačítka *„Stáhnout“* uložit vygenerovaný dokument na místní disk. Textový dokument lze uložit ve formátu *Word 2007 Office Open XML (.docx)*, *Portable Document Format (.pdf)*, *OpenOffice OpenDocument (.odt)* a *Word 2003 (.doc)*.



Obr. 23 Dialog uložení vygenerovaného dokumentu

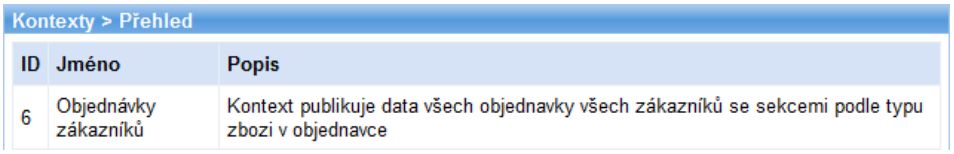


Obr. 24 Přidání nové šablony

Šablonu lze přidat pomocí stránky Šablony > Přidat. Pro přidání šablony je potřebné zadat soubor šablony, popis jeho nativního datového typu, jméno a popis šablony. Pole kontexty obsahuje seznam všech datových typů v systému, zaškrtnutím pole *„přidat k šabloně“* bude tento typ dat přiřazen k šabloně. K přiřazenému kontextu (datovému typu) je potřeba přiřadit soubor XSLT definující převod do nativního datového typu

šablony. Jelikož nejsou tyto asociace v prezentaci bakalářské práce podporovány je toto přiřazení pouze ilustrativní.

Po úspěšném přidání šablony je zobrazena stránka Šablony > Přehled.


Kontexty > Přehled

ID	Jméno	Popis
6	Objednávky zákazníků	Kontext publikuje data všech objednavky všech zákazníků se sekcemi podle typu zbozi v objednavce

Obr. 25 Přehled

Stránka Kontexty > Přehled zobrazuje všechny zaregistrované datové typy zasílané aplikacemi.

7 Závěr

V bakalářské práci byl prezentován popis systému pro tvorbu dokumentů ze šablon, jeho návrh a implementace. Návrh a implementace systému práce byla ovlivněna požadavky zákazníka – firmy Triada. Domluva se zákazníkem byla také jednou z časově nejnáročnějších fází vývoje, kdy bylo potřeba vysvětlit všechny aspekty různých návrhů a rozhodnout mezi nimi. Systém se i díky tomu podařilo vytvořit k plné spokojenosti zákazníka. Oproti stávajícím řešením nabízí množství nových možností, jakými jsou konstrukce master-detail do libovolné úrovně, jednoduše rozšiřitelné vyhodnocování výrazů vkládaných do textových dokumentů, možnost centralizovaného provozu a nezávislost na dalším softwaru. Systém je navržen tak, aby byl mnohem lépe upravitelný pro konkrétní potřeby zákazníka než jakýkoliv nalezený nástroj, zabývající se podobným tématem.

Systém bude v budoucnu dále rozšiřován. Prvním z těchto rozšíření bude formulování přesného rozsahu funkcí použitelných ve výrazech. Dále se uvažuje například přidání podpory pro kolekce všech položek některého seznamu v datech, nebo formátování tisku hodnoty výrazu. Po zapojení do systému a nahrazení hromadné tvorby dokumentů budou šablony s ostatními aplikacemi sdíleny přes centrální datové úložiště. Podle vytížení služby generování dokumentů může být zavedena distribuce zátěže mezi několik serverů v součásti správa šablon. S tím je spojeno přidání rozhraní pro sledování průběhu požadované operace.

Pro zjednodušení tvorby šablon by mohl vzniknout jednoduchý editor textových dokumentů nebo vytvořen doplněk pro aplikaci *Microsoft Office Word*, který by pomáhal určit dostupné datové položky a nabízel podporu tvorby výrazů a sekcí v dokumentu.

Literatura

[1] http://msdn.microsoft.com/en-us/library/dt80be78.aspx

[2] http://support.microsoft.com/kb/257757/

[3] ftp://sineadown:aSx2000@ftp.sinea.cz/pub/SiNavod4.zip

[4] http://cs.wikipedia.org/wiki/PostScript

[5] http://www.w3schools.com/xslfo/default.asp

[6] http://cs.wikipedia.org/wiki/DocBook

[7] http://www.aspose.com/categories/.net-components/aspose.words-for-.net/default.aspx

[8] http://ima.udg.edu/~dagush/papers/RECURSIV.pdf

8 Přílohy

Na přiloženém CD se nachází text práce ve formátu PDF (složka Bakalářská práce), aplikace (složka Aplikace), její zdrojové kódy (složka Source) a podrobná programátorská dokumentace (složka Dokumentace).