VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

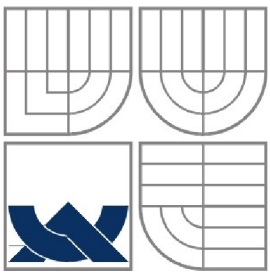
OPEN DOCUMENT FORMAT PRO PŘENOS DAT Z IS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

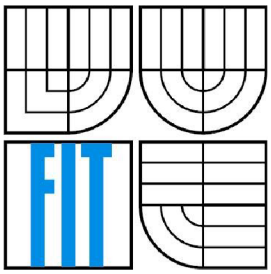
AUTOR PRÁCE
AUTHOR

PETR HALAŠKA

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

OPEN DOCUMENT FORMAT PRO PŘENOS DAT Z IS

OPEN DOCUMENT FORMAT FOR DATA EXPORT FROM IS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

PETR HALAŠKA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. ALOIS KUŽELA

BRNO 2008

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Halaška Petr**
Obor: Informační technologie
Téma: **Open Document Format pro přenos dat z IS**
Kategorie: Web

Pokyny:

1. Prostudujte technologii výstavby informačních systémů (zvláště AJAX) a exportní formáty tabulkových procesorů (zvláště Open Document Format - ODF). Prozkoumejte možnost exportu obrázků a grafů.
2. Navrhněte informační systém malé firmy s využitím technologie AJAX a možným exportem dat do ODF.
3. Informační systém realizujte jako prototypové řešení na vhodném vzorku dat.
4. Zhodnoťte přínosy či případné problémy plynoucí z nasazení technologie AJAX do IS.

Literatura:

- Dle pokynů vedoucího práce.

Při obhajobě semestrální části projektu je požadováno:

- Bez požadavků.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kužela Alois, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2
L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Petr Halaška**
Id studenta: 88709
Bytem: Sportovní 151, 664 61 Opatovice
Narozen: 22. 07. 1983, Brno
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   bakalářská práce

Název VŠKP: Open Document Format pro přenos dat z IS
Vedoucí/školitel VŠKP: Kužela Alois, Ing.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Abstrakt

Účelem práce je prozkoumat způsoby přenosu dat z informačních systémů do formátů, které mohou být zpracovány kancelářskými aplikacemi. Úkolem je navhrnout a vytvořit prototyp takového systému, využívajícího technologie AJAX, zhodnotit přínosy a rizika nasazení této technologie v oblasti informačních systémů. Na vybraném formátu demonstrovat export dat.

Klíčová slova

Open Document Format, Ajax, XML, informační systém

Abstract

Purpose of this work is to examine means of data export from information system into formats that can be processed by office applications. Task is to design and create prototype of such system using AJAX technology, evaluate benefits and risks of deploying the technology in area of information systems. Demonstrate data export into selected format.

Keywords

Open Document Format, Ajax, XML, information system

Citace

Petr Halaška: Open Document Format pro přenost dat z IS, bakalářská práce, Brno, FIT VUT v Brně, 2008

Open Document Format pro přenos dat z IS

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Aloise Kužely. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Petr Halaška
13.5.2008

Poděkování

Děkuji především svému vedoucímu, Ing. Aloisi Kuželovi, za cenné rady a připomínky, nezbytné pro zdárné dokončení této práce. Velký dík patří Ludmile Kuželové za podporu a posilující energii, bez níž by práce nemohla vzniknout.

Obsah

Úvod

Informační systémy poskytují široké možnosti pro uložení dat, manipulaci nebo analýzu a prezentaci shromážděných informací uživateli. Firmám a institucím umožňují zlepšit výkonnost, šetří nejenom čas, ale v nezanedbatelné míře i peníze. V praxi často dochází k situacím kdy potřeba sdílet data sahá mimo oblast pokrytou konkrétním informačním systémem. Údaje uložené v systému je nutné předat uživatelům bez přístupu do systému. Přenos je nutné provést do formátu, který jim umožní dále s daty efektivně pracovat, provádět jejich analýzu a zároveň zabezpečit před neoprávněnými zásahy. Tuto možnost poskytují různé balíky programů souhrně označovaných jako kancelářské aplikace. Práce má za cíl prozkoumat způsoby ukládání do formátů používaných kancelářskými aplikacemi a vybraný formát použít k uložení vzorových dat. V rámci práce budou také zhodnoceny možnosti nasazení technologie AJAX při tvorbě informačních systémů a nastíněny výhody a případné nevýhody využití této technologie.

V první části práce krátce shrneme pojem informačního systému a podíváme se na jednotlivé technologie na kterých jsou stavěny informační systémy. Blíže se zaměříme především na AJAX a způsob jakým spojuje existující přístupy. Bude nutné se seznámit s technologiemi tvorby informačních systémů, které použijeme pro tvorbu systému a export dat.

Druhá část nám představí dva nejrozšířenější formáty používané kancelářskými aplikacemi. Porovnáme jejich základní vlastnosti především v oblasti tabulkových procesorů, zaměříme se především na formát OpenDocument, jeho strukturu a možnosti pro ukládání tabulkových dat, obrázků a grafů – tedy data a objekty, které je třeba exportovat nejčastěji.

V další, třetí části bude proveden návrh informačního systému, navržen vhodný způsob exportu dat ze systému do formátu OpenDocument, aby bylo možné začlenit modul pro export do systému. Seznámíme se s daty a položkami kterými bude databáze plněna a budou popsány operace se záznamy v databázi. Exportovat budeme chtít kromě klasických tabulkových dat také obrázky a grafy. V kapitole také popíšeme implementaci prototypu informačního systému, především modulů a knihoven zajišťujících export dat do zvoleného formátu OpenDocument.

Závěrem budou shrnuty poznatky získané při navrhování a vytváření systému, především z hlediska zapojení a použitelnosti knihoven realizujících export z OpenDocument formátu. Zváženy budou důsledky nasazení AJAXu do informačního systému, výhody a možné nevýhody použití této technologie v informačních systémech.

1 Technologie pro tvorbu IS

Obecný systém je cíleně vytvořený soubor komponent, mezi kterými existují určité vztahy, a které splňují nějaký cíl. Skládá se z atributů (veličiny jež charakterizují určitý prvek systému), událostí (změna atributu nebo změna konfigurace systému - například komponenty) a časových množin (hodnoty vztažené k určitému okamžiku). Systémy obecně dělíme na tvrdé, spojené s konkrétním problémem, nebo měkké systémy kde vystupuje řada obecných faktorů. Existují další druhy dělení, například na uzavřené a otevřené (podle interakce s okolím) nebo na statické a dynamické. V systémech může nastat zpětná vazba, stav v němž výstupní veličina opětovně ovlivňuje vstupní veličinu a tím také samotný systém. Každý systém má tedy tendence být nestabilní, což nemusí nutně znamenat nevýhodu nebo poruchu. U mnoha systémů je tento efekt dokonce velice žádaný.

Informační systém je speciální případ systému pro sběr, udržování, zpracování a poskytování informací a dat. Příkladem informačního systému může být telefonní seznam nebo evidenční kartotéka v knihovně. Systém nemusí být nutně realizovaný pomocí počítačů, může se například jednat o kombinaci elektronické a papírové formy. Jde o konceptuální, otevřený systém s uzavřeným cyklem, kde základním úkolem je zpracování dat a zabezpečení komunikace. Koncový uživatel používá informační systém pro ukládání a hledání dat, jejich organizaci a analýzu.

Velmi důležitým hlediskem je architektura systému. V současné době se téměř výhradně používá třívrstvá architektura. Klient pracuje pouze s uživatelským rozhraním. Datové služby, aplikační služby a logika jsou od sebe odděleny do samostatných logických modelů, které mohou být umístěny buď na stejném nebo na dvou různých serverech. Vrstvy spolu vzájemně komunikují prostřednictvím standardních protokolů, přičemž jednotlivé vrstvy vždy komunikují pouze se sousední vrstvou. Třívrstvý model nám umožňuje získat vyšší úroveň stability.

- **prezentační vrstva**, někdy také nazývána klientská vrstva, má za úkol prezentovat data informačního systému uživatelům a poskytovat jim zároveň přístup k jednotlivým službám systému. Může být realizována jako stránky webového prohlížeče, samostatná aplikace, případně kombinovat oba přístupy.
- **funkční vrstva**, neboli aplikační vrstva, představuje vlastní logiku informačního systému. Základním účelem je poskytovat klientským aplikacím potřebnou funkcionalitu. Vrstva zajišťuje kontrolu správnosti a integrity dat, komunikaci mezi klientskou a databázovou vrstvou, ale také s externími informačními systémy. Umožňuje tedy výměnu dat mezi nimi. Dále také komunikuje s případnými ostatními instancemi (uzly) vlastního informačního systému.

- **databázová vrstva** slouží především jako bezpečné a spolehlivé úložiště dat informačního systému. Kromě toho také zajišťuje základní integritu dat (např. aby každá položka faktury měla přiřazenu fakturu apod.) Dále tato vrstva poskytuje základní zabezpečení přístupu k datům a v neposlední řadě také prostředky pro rychlý přístup k datům a jejich vyhledávání.

Každá z vrstev architektury využívá jiných technologií pro zajištění potřebné funkcionality. Výhody, které přináší použití třívrstvé architektury zahrnují možnosti lepší ochrany a zprostředkování sdíleného přístupu, využití většího škálování - lze použít více serverů a výkon klientských zařízení není tolik omezen. Umožněno je využití více druhů klientských aplikací napojených na aplikační server. Následující část kapitoly popisuje technologie použité ve vytvářeném systému. Historické pozadí vývoje jednotlivých technologií v této kapitole bylo čerpáno z [3], spolu s ověřením údajů pomocí zdrojů odkazovaných v příslušných článcích na [3].

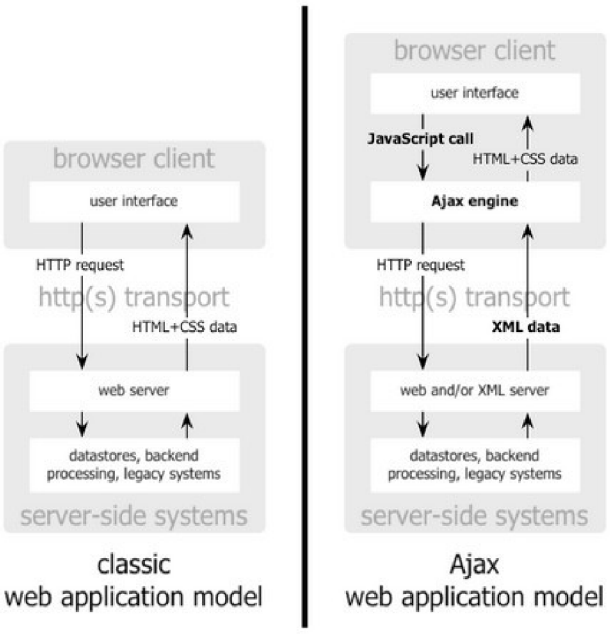
1.1 MySQL

Informační systém musí obsahovat prostředky pro ukládání a správu dat v databázové vrstvě. Uložená data v databázi jsou charakteristická velkým množstvím podobných dat, trvalostí a nutností umožnit k datům sdílený a zabezpečený přístup pro více uživatelů. K těmto účelům se používají nástroje označované pojmem systém řízení báze dat (SŘBD). Systémy umožňují definovat typy dat, vytvářet slovníky dat, zajišťují bezpečnost a integritu dat. Slouží pro zajištění souběžného přístupu a co nejvyšší výkonnosti. V současné době se používají především objektové databáze a objektově relační databázové systémy. Pro potřeby navrhovaného informačního systému zvolíme opensource databázový systém MySQL.

Jedná se o systém vytvořený švédskou firmou MySQL AB. Jeho hlavními autory jsou Michael Widenius a David Axmark. MySQL je multiplatformní databáze. Komunikace s ní probíhá – jak už název napovídá – pomocí jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními. MySQL bylo od počátku optimalizováno především na rychlost, a to i za cenu některých zjednodušení: má jen jednoduché způsoby zálohování, a až donedávna nepodporovalo pohledy, triggery, a uložené procedury. Tyto vlastnosti jsou doplňovány teprve v posledních verzích. Velmi oblíbená a často používaná je kombinace MySQL, PHP a webového serveru Apache jako základů pro webové aplikace. Má vysoký podíl na databázích používaných v současné době. Hlavními důvody používání MySQL je mutliplatformnost (lze provozovat na Linux, MS Windows a dalších operačních systémech), licencování jako volně šiřitelný software a relativně vysoká rychlost provádění SQL dotazů. Pro své výhody je používán ve velké většině neziskových organizací, škol, univerzit a poskytovatelů internetu.

1.2 AJAX

AJAX, zkratka z anglického Asynchronous JavaScript and XML, není technologie sama o sobě, ale termín popisující nový přístup k webovým technologiím. Kombinací existujících technologií je možné získávat a aktualizovat data na stránce bez nutnostni znovu načítat celou stránku ze serveru. Základním stavebním kamenem AJAXu je objekt XMLHttpRequest, který provádí asynchronní komunikaci skriptů se serverem. Ajax využívá dalších webových technologií především JavaScriptu a XML. Obrázek ukazuje rozdíl klasického přístupu a AJAXu.



Obrázek 1.1: srovnání modelů webových aplikací, převzato z [6]

V klasickém modelu webu je při každé uživatelské akci na server poslán požadavek HTTP. Server poté komunikuje s databázemi, provádí generování kódu a výslednou stránku spolu se styly zašle zpět uživateli. Uživatelův webový klient poté zobrazí výslednou stránku. V průběhu celého procesu komunikace klient - server čeká uživatel na načtení nové verze stránky, mnohdy obsahující nepatrné změny.

Každá uživatelská akce, která v klasickém modelu vyvolá HTTP požadavek na server, má podobu JavaScriptového volání na engine AJAXu. Reakce na uživatelovu akci, které nevyžadují aktualizaci dat přímo ze serveru - jednoduchá validace údajů, úprava dat uložených v paměti nebo zobrazení dodatečných statických informací (například nápovědy) – jsou obslouženy přímo v uživatelově prohlížeči. Pokud rozhraní potřebuje kontaktovat server – například zasílá data pro zpracování, načítá dodatečný kód pro rozhraní, nebo požaduje aktualizované údaje z databáze – jsou požadavky a výsledky zasílány asynchronně, většinou za použití XML, bez přerušování uživatelovy práce s webovou aplikací [6].

AJAX využívá kombinaci technik pro tvorbu webových aplikací, především jazyky HTML nebo XHTML pro prezentaci informací a kaskádové styly (CSS). Pomocí modelu DOM a některého skriptovacího jazyka, většinou JavaScript, umožňuje lépe zobrazozovat a dynamicky měnit obsah stránek. Klíčovou součástí technologie je objekt XMLHttpRequest pro asynchronní výměnu dat

s webovým serverem (typicky je užíván formát XML, ale je možné použít libovolný jiný formát včetně HTML, prostého textu nebo JSON).

1.3 PHP

Jedná se o skriptovací programovací jazyk, určený především pro programování dynamických internetových stránek. Nejčastěji se začleňuje přímo do struktury jazyka HTML nebo XHTML. Ačkoliv primárním použitím je tvorba webových aplikací, lze PHP použít také k tvorbě konzolových a desktopových aplikací. PHP skripty jsou prováděny na straně serveru, k uživateli je přenášen až výsledek jejich činnosti. Syntaxe jazyka kombinuje hned několik programovacích jazyků (C, Java, Pascal). PHP je platformě nezávislý, díky tomu skripty fungují bez úprav na mnoha různých operačních systémech. Obsahuje rozsáhlé knihovny funkcí pro zpracování textu, grafiky, práci se soubory, databázových serverů (PostgreSQL, MySQL) a podporu mnoha internetových protokolů (HTTP, SMTP, SNMP, FTP, POP3). PHP je jazyk dynamicku typový, to znamená že datový typ proměnné se určuje až v okamžiku přiřazení hodnoty. Podporuje přetypování proměnných.

Původní jazyk PHP vytvořil Rasmus Lerdorf jako nástroj k udržování vlastních stránek, který později zkombinoval PHP s dalším svým programem Form Interpreter. Verze PHP2 vypuštěná v roce 1995 veřejně už měla základy funkcionality dnešního PHP, včetně podpory komunikace s databázemi, práci s proměnnými podobně jako jazyk PERL a možnost vkládání HTML kódu. Zeev Suraski a Andi Gutmans v roce 1997 přepsali parser a zformovali tak základ PHP3. Současně byl název změněn na dnešní podobu PHP hypertext procesor. V současné době je jedinou stabilní verzí na které se pracuje PHP 5, podpora pro PHP 4 končí v srpnu 2008. Ve vývoji je také PHP verze 6, která by měla zlepšit podporu kódování Unicode, podporu namespaces a zavést novou správu cache. PHP umožňuje vývojářům psát rozšiřující knihovny v jazyce C, které mohou být zkompilovány na moduly PHP nebo dynamicky načteny za běhu aplikace.

Skripty v PHP jsou většinou uloženy v čitelné textové formě i na produkčních serverech, proto existují různé kódovací nástroje, které šifrují nebo převádí skripty do binárních souborů. Otevřenost a čitelnost PHP skriptů bývá mnohdy hlavním důvodem pro volbu tohoto skriptovacího jazyka, stejně jako celá řada projektů umožňujících šíření a sdílení kódu pod různými licencemi (nejčastěji Open Source). Nejznámější seskupení jsou PEAR (PHP Extension and Application Repository) nebo PECL (PHP Extension Community Library), hostící mnoho balíčků rozšiřujících práci PHP se síťovými protokoly, souborovými systémy a zavádějící nové moduly například pro matematické úlohy a šifrování.

1.4 XML

XML (eXtensible Markup Language, rozšiřitelný značkovací jazyk) je obecný značkovací jazyk, vyvinut a standardizován konsorciem W3C. Umožňuje snadné vytváření konkrétních značkovacích jazyků pro různé účely a široké spektrum různých typů dat. V dnešních informačních technologiích nelze používat proprietární formáty, které jsou svázány s konkrétním softwarem nebo hardwarem. Pro výměnu informací je potřeba používat jednoduchý otevřený formát, který lze číst nezávisle na použité platformě nebo vlastnictví proprietárního software. Takovým otevřeným formátem je právě XML, jehož specifikace je každému zdarma k dispozici na serveru konsorcia W3C. Oproti uzavřeným formátům má XML tu výhodu, že je plně dokumentovaný, zapsaný pomocí čistého textu a proto je na rozdíl od binárních formátů snadno upravitelný. Není problém kdykoliv otevřít XML dokument v libovolném textovém editoru a potřebné úpravy provést ručně. Každý tak může bez problémů do svých aplikací implementovat podporu XML.

Vývojáři aplikací mohou použít mnoha volně přístupných knihoven a balíčků pro práci s XML nebo si vytvořit vlastní. Ačkoliv objem dat může být použitím textového formátu větší, v dnešních systémech je důraz kladen více na srozumitelnost a snadnou práci s daty. Jazyk XML je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů. Umožňuje popsat strukturu dokumentu z hlediska věcného obsahu jednotlivých částí, definovat atributy těchto částí. XML sám o sobě nedefinuje vzhled dokumentu, ten je poté možné popsat pomocí připojeného stylu. Další možností je opět pomocí různých stylů provést transformaci do jiného typu dokumentu, nebo do jiné struktury XML.

1.5 JavaScript

JavaScript je interpretovaný programovací jazyk se základními objektově orientovanými schopnostmi, používaný především v internetových stránkách. Univerzální jádro jazyka bylo vloženo do Mozilly, Netscape Navigatoru, Internet Exploreru a dalších webových prohlížečů. Jeho syntaxe patří do rodiny jazyků C/C++/Java. Slovo Java je však součástí jeho názvu pouze z marketingových důvodů a s programovacím jazykem Java jej vedle názvu spojuje jen podobná syntaxe. JavaScript byl v červenci 1997 standardizován asociací ECMA (European Computer Manufacturers Association) a v srpnu 1998 ISO (International Organization for Standardisation). Standardizovaná verze JavaScriptu je pojmenována jako ECMAScript a z ní byly odvozeny i další implementace, jako je například ActionScript. Pro webové programování by jazyk rozšířen přidáním objektu jenž reprezentuje okno webového prohlížece a jeho obsah. Program v JavaScriptu se obvykle spouští až po stažení WWW stránky z Internetu (tzv. na straně klienta), na rozdíl od jiných interpretovaných

programovacích jazyků (např. PHP a ASP), které se spouštějí na straně serveru ještě před stažením z Internetu. Z toho plynou jistá bezpečnostní omezení, JavaScript například nemůže pracovat se soubory, aby tím neohrozil soukromí uživatele. Použití klientského JavaScriptu – jsou jím obvykle ovládány různé interaktivní prvky uživatelského rozhraní nebo tvořeny animace a efekty obrázků.

JavaScript je možné použít i na straně serveru. První implementací JavaScriptu na straně serveru byl LiveWire firmy Netscape vypuštěný roku 1996, dnes existuje několik možností včetně opensource implementace Rhinola založená na Rhino, gcj a Apache.

1.6 XMLHttpRequest

Základním stavebním kamenem AJAXu je objekt XMLHttpRequest, který provádí asynchronní komunikaci skriptů se serverem. Webovým aplikacím umožňuje rozhraní XMLHttpRequest komunikaci mezi serverem a klientem prostřednictvím protokolu HTTP. Rozhraní mohou skriptovací jazyky, především JavaScript, ale i Python a další, použít k přenosu dat mezi klientem zobrazujícím webovou stránku a webovým serverem. Rozhraní umožňuje provádět jak synchronní dotazy tak asynchronní, na druhém způsobu je založena technologie AJAX umožňující změnu části obsahu stránky zobrazené v prohlížeči klienta bez nutnosti jejího opětovného kompletního načtení ze serveru.

Původní koncept vyvinul Microsoft v rámci projektu Outlook Web Access 2000 jako součást serverové aplikační logiky. V Internet Exploreru je XMLHttpRequest k dispozici jako objekt ActiveX od verze 5.0. Vývojáři prohlížeče Mozilla se rozhodli objekt včlenit do prohlížeče jako nativní součást, později tento způsob zvolili i ostatní tvůrci webových klientů. Objekt XMLHttpRequest implementuje Mozilla od verze 1.0, Opera od verze 8.0, Safari 1.2 a vyšší, nebo Konqueror. Internet Explorer ve verzi 7 obsahuje jak nativní podporu XMLHttpRequest, tak z důvodu zpětné kompatibility i realizaci pomocí ActiveX.

Pro přenos údajů je mezi klientem a serverem je využíváno formátu XML. Objekt XMLHttpRequest provádí komunikaci se serverem odesíláním požadavků metodami GET a POST protokolu HTTP. Po obdržení odpovědi vyvolá na stránce předem nadefinovanou funkci, která posléze výsledek akce zpracuje. Přestože na ujednocení a standardizaci se pracuje, odlišná implementace objektu XMLHttpRequest v různých prohlížečích nutí vývojáře vytvářet aplikačně a platformě závislý kód. Organizace W3C pracuje na standardu, založeném na dosud existujících řešeních, který má pomoci sjednotit způsob jeho zpracování v různých prohlížečích.

2 Formáty pro export dat

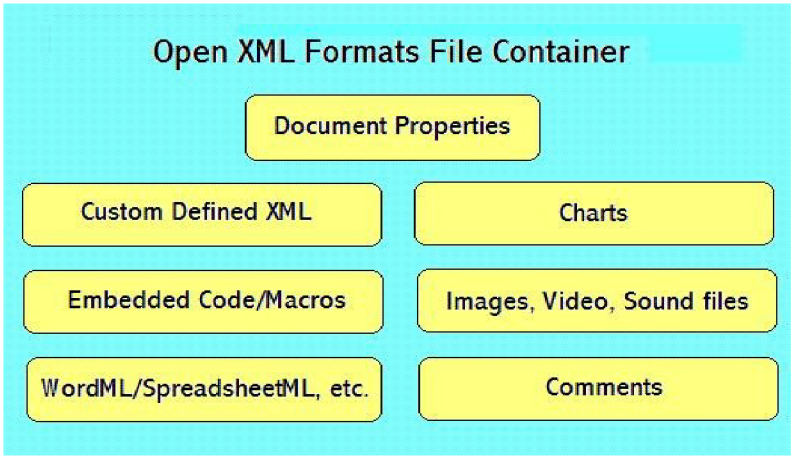
Informační systémy samy o sobě poskytují široké možnosti pro uložení dat, manipulaci nebo analýzu a prezentaci shromážděných informací uživateli. V některých případech jsou však nedostačující z hlediska možností pro šíření dat. V praxi je často nutné archivovat důležité údaje nebo smlouvy po dobu mnoha let i po skončení platnosti. Z legislativních důvodů bývá nutné uchovávat data také v tištěné podobě, obchodní styky vyžadují možnosti pro sdílení informací mimo skupinu uživatelů systému. Export dat z IS vyžaduje existenci široce užívaného a dostatečně komplexního formátu, který data uloží do souborů, aplikací, které s těmito soubory umí pracovat. Používá se také pojem Office Information System označující sadu aplikací určených k podpoře kancelářské práce. V současné době nejrozšířenějšími balíky kancelářských aplikací jsou MS Office a aplikace pracující s formátem Open Document (OpenOffice.org, StarOffice, a další).

2.1 Open XML formát

Přestože je práce zaměřena především na formát OpenDocument, pro účely srovnání zde nastíníme také nový souborový formát pro aplikace Microsoft Office. Open XML formát tvoří části (parts) sdílené všemi programy. Formát je založen na jednoduchém balíčku souborů komprimovaném metodou ZIP. Soubor ve formátu Open XML je tvořen kolekcí libovolného počtu částí, které obdobně jako u OpenDocument formátu tvoří společně výsledný dokument. Údaje v této části byly získány především prostudováním [5] a několika vzorových dokumentů.

2.1.1 Základní koncept Open XML formátu

Většina částí jsou soubory XML, které popisují aplikační data, metadata, individuální uživatelská data a další. Ostatní části zahrnují binární soubory představující obrázky, objekty OLE nebo makra v jazyce VBA. Navíc speciálním druhem části jsou vztahy (doslova relationship parts), které definují vazby mezi ostatními částmi dokumentu. Každá vazba má vlastní



Obr 2.1: kontejner formátu Open XML, převzato z [5]

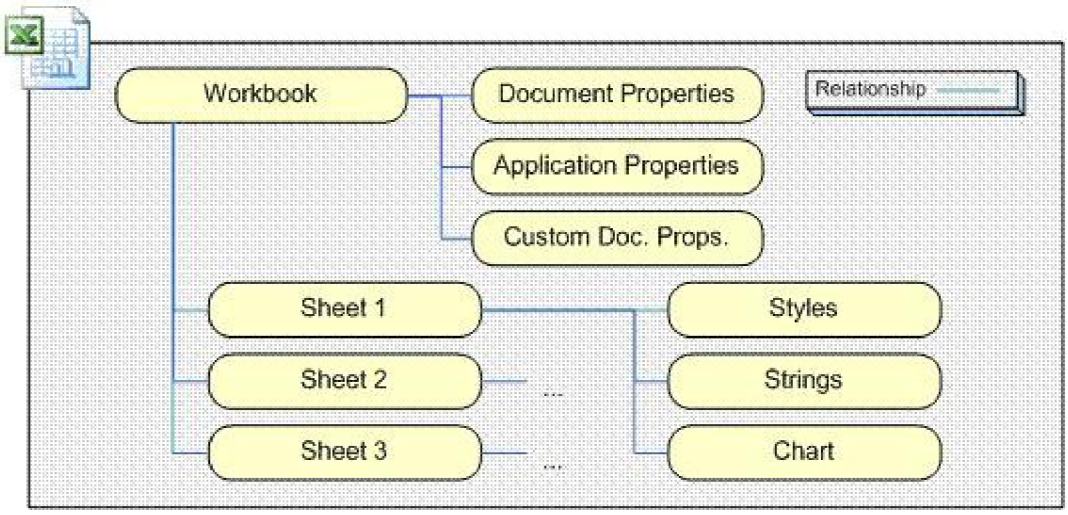
identifikátor a zatímco obsah dokumentu je tvořen částmi, vztahy popisují způsob, kterým jednotlivé části tvoří výsledný dokument Office. Vztahy jsou definované ve formátu XML a každý vztah má jednoznačný identifikátor RelationshipID, který určuje propojení mezi zdrojovou částí a cílem. Například spojení mezi pracovním sešitem a jeho listem je identifikované vztahem, který je uložen přímo v části workbook.xml.rels.

2.1.1.1 Balíček (Package)

Kontejner pro dokumenty ve formátu Open XML je balíček ZIP, který sdružuje jednotlivé části tvořící dokument. V balíčku jsou uložena jednak data společná a sdílená všemi aplikacemi používající formát Open XmL, například vlastnosti dokumentu, styly, grafy, odkazy, kresby a diagramy. Další částí balíku jsou soubory specifické pro danou aplikaci. V Excelu jsou to pracovní sešity, PowerPoint má snímky prezentace a pro Word kromě dokumentu samotného také záhlaví nebo zápatí.

2.1.1.2 Část (Part)

Odpovídá jednomu fyzickému souboru v balíčku. V balíčku existují části společné pro všechny aplikace, jako náhled, metadata a část se vztahy (relationships). Ostatní existují konzistentně ve všech souborech jako specifické části, např. vlastnosti dokumentu (properties) nebo binární soubory. Zajímavé je, že OpenXML odděluje vlastnosti aplikační a souborové. Existuje část docProps/app.xml nesoucí informace o aplikaci, ve které byl soubor vytvořen a část docProps/core.xml nesoucí údaje o tvůrci (creator) a úpravách (created, modified, lastModifiedBy). Mnoho částí je unikátních pro každou aplikaci. Například pokud rozbalíme soubor z Excelu, získáme části workbook.xml a několik sheetN.xml částí očíslovaných podle jednotlivých listů.



Obr 2.2: vztahy uvnitř dokumentu Open XML, převzato z [5]

2.1.1.3 Vztahy (Relationships)

Vztahy specifikují celkovou strukturu dokumentu. Každý vztah určuje propojení mezi zdrojovou částí (source part) a cílem (target resource). Vztahy jsou uloženy opět v jednotlivých částech balíčku ve formátu XML (například /_rels/.rels). Částí určených k uložení jednotlivých vztahů může být více, například kromě vztahů sešit - list uložených ve workbook.xml.rels má každý list uložené vztahy s dalšími částmi v odpovídajícím sheetN.xml.rels souboru. Příklad části nesoucí hlavní vazby v souboru OpenXML (konkrétně soubor pracovního sešitu Excel 2007).

```
<Relationships
xmlns="http://schemas.microsoft.com/package/2005/06/relationships">
<Relationship ID="rId3"
Type="http://schemas.microsoft.com/office/2005/8/relationships/
xlStyles "Target="styles.xml"/>
<Relationship ID="rId1"
Type="http://schemas.microsoft.com/office/2005/8/relationships/
xlWorksheet" Target="worksheets/Sheet1.xml"/>
<Relationship ID="rId5"
Type="http://schemas.microsoft.com/office/2005/8/relationships/
xlMetadata" Target="metadata.xml"/>
<Relationship ID="rId4"
Type="http://schemas.microsoft.com/office/2005/8/relationships/
xlSharedStrings" Target="strings.xml"/>
</Relationships>
```

2.1.2 Makra a programový kód v Open XML

Za delší zmínku stojí nové způsoby uložení maker a dalšího programového kódu. Jednoduché dokumenty uložené v Open XML formátu nemohou nést binární soubory obsahující makra a jsou uloženy bez maker (macro-free). Dokumenty mohou obsahovat a používat makra, ale uživatel nebo vývojář je bude muset uložit jako dokumenty s makry (macro-enabled). Pokud se soubor s binárním kódem makra objeví v souboru označeném jako dokument bez maker, ať schválně nebo náhodou, Office aplikace pozná podle vztahů a obsahu archívu část s makry a nepovolí spuštnění kódu. Toto chování zajišťuje, že škodlivý kód nemůže být nechtěně spuštěn. Na druhou stranu přítomnost maker je zřejmá na první pohled podle přípony souboru (xlsm). Jednoduchým extrahováním obsahu souboru lze zpřístupnit obsah dokumentu bez nutnosti jej otevírat. Vývojáři nyní budou moci zjistit existenci jakéhokoli kódu v dokumentu před jeho otevřením a v případě nutnosti tento kód odstranit bez ztráty ostatních dat uložených v souboru.

2.1.3 Shrnutí

Zajímavou vlastností je optimalizace nahrávání, ukládání a velikosti souboru. OpenXML formát pro tabulkový soubor umožnil tzv. "sdílenou tabulku textových řetězců" v části nazvané strings.xml. Aplikace mohou uložit každou unikátní hodnotu textu nalezenou v sešitu. Různé buňky listu obsahující tuto hodnotu pak místo celého řetězce pouze referencují tabulku řetězců. Nabízí se zřejmá výhoda tohoto přístupu - vývojáři ve vícejazyčných organizacích mohou tuto funkcionalitu využít k podpoře více jazyků - stačí pouze úpravou části strings.xml vytvořit novou jazykovou verzi stejného sešitu. Přístup k těmto částem může být uplatněn při vyhledávání.

Mechanismus vztahů se ve srovnání s formátem OpenDocument jeví jako poněkud složitý, v tomto směru můžeme pro účely exportu dat považovat za lepší a přehlednější strukturu manifestu formátu OpenDocument. Interní i externí zdroje lze v OpenDocument připojit pomocí odkazů. V Open XML jsou externí linky uloženy v samostatné části, zamýšleným přínos mohlo být umožnění omezit nebezpečí přístupu ke škodlivým obsahům odstraněním této části. Podstatně zajímavější se jeví použití odkazů na tabulky textových řetězců v souborech string.xml, vhodné například pro generování vícejazyčných dokumentů. Zda toto řešení přináší úsporu při práci s dokumentem není jisté, například při přidávání textových údajů do sešitu může dojít k prodlevě při kontrole již existujících řetězců v souboru strings.xml.

Zabezpečení dokumentů před spouštěním maker a především odlišení příponou je přínosnější spíše pro klasického uživatele, který tak nespustí nevědomky škodlivý kód v podvrženém souboru. Naopak z pohledu IT pracovníků není v tomto směru mezi formáty Open Document a Open XML větší rozdíl - OLE aplikace, skripty a makra jsou na definovaných místech a není tedy problém je rychle najít a buď odstranit nebo zakázat spuštění v obou případech.

2.2 Open Document formát

Open Document Format je otevřený, volně dostupný souborový formát určený pro kancelářské aplikace (textové editory, tabulkové procesory a další). Formát je založený na značkovacím jazyce XML a původní specifikace byly vytvořeny sdružením OASIS (Organization for Advancement of Structured Information Standards), která vycházela z dřívějšího, na XML založeného formátu pro balík aplikací OpenOffice.org. První oficiální schůze technického výboru projednávající standardizaci formátu se uskutečnila koncem roku 2002. OASIS schválil formát OpenDocument jako standard OASIS v květnu 2005 a zaslal specifikaci OpenDocument formátu spojenému výboru Mezinárodní organizace pro standardizaci a Mezinárodní Elektrotechnické komise. Formát OpenDocument byl přijmut a vydán 3. května 2006 mezinárodní organizací pro standardizaci jako standard ISO/IEC 26300:2006. Specifikace je zdarma dostupná a volně šiřitelná, což umožňuje vývojářům mnoha

kancelářských aplikací, programů pro správu dokumentů a systémů spravujících množství dat začleňovat nativní podporu formátu do svých produktů. Standardizace formátu je zárukou do budoucna ,že aplikace a systémy pro tvorbu dokumentů dodržující tyto specifikace budou schopny v budoucnu pracovat nejen se staršími verzemi dokumentů, ale také zajistit, že dokument vytvořený v jedné z aplikací půjde bez potíží zpracovat v zcela jiném programu podporujícím formát. Open Document Format je nyní podporován a prosazován uskupením Open Document Alliance čítajícím mnoho desítek společností z celého světa, mezi jinými firmy Sun Microsystems Inc., Novell, IMB. Poznatky a informace v této kapitole byly čerpány především z [4].

2.2.1 Struktura dokumentu

Specifikace OpenDocument formátu defunuje dva možné způsoby reprezentace dokumentu:

- jako jeden samostatný dokument XML
- kolekci několika poddokumentů v balíku, z nichž každý obsahuje část celkového dokumentu.

Každý z dílčích dokumentů obsahuje určitou část celkového XML dokumentu. Všechny druhy dokumentů formátu Open Document využívají stejnou definici dokumentu XML a jeho poddokumentů (viz tabulka 2.1).

Každý dokument nebo poddokument musí být tvořen právě jedním kořenovým elementem. Existují čtyři typy poddokumentů rozdělených podle obsahu a každý má jiný kořenový element. Dokument reprezentovaný jediným XML dokumentem má vlastní kořenový element. Níže uvedená tabulka shrnuje uvedené kořenové dokumenty a obsah jednotlivých poddokumentů.

Kořenový element	Obsah poddokumentu	Název poddok. balíku
`<office:document>`	Kompletní dokument v jednom souboru XML.	--
`<office:document-content>`	Obsah dokumentu, automatické styly použité v obsahu dokumentu.	content.xml
`<office:document-styles>`	Styly dokumentu použité v obsahu dokumentu a automatické styly, použité v samotných stylech.	styles.xml
`<office:document-meta>`	Meta informace o dokumentu (autor, čas posledního uložení apod.)	meta.xml
`<office:document-settings>`	Aplikačně specifické nastavení jako velikost okna, nastavení tisku apod.	settings.xml

Tabulka 2.1: kořenové elementy dle specifikace ODF [4].

XML nemá nativní podporu pro ukládání binárních dat, například obrázků nebo OLE objektů. Specifikace [2] ale umožňuje jejich uložení do balíku spolu s XML obsahem k němuž se vztahují. Tento způsob používají téměř všechny aplikace pracující s formátem OpenDocument, které se zaměřují na složitější dokumenty nebo jsou součástí některé kancelářské sady. Balík musí obsahovat

seznam - manifest ostatních souborů a údaje o nich. Manifest samotný je XML soubor s kořenovým elementem `<manifest:manifest>`, umístěný v cestě „META-INF/manifest.xml“. Hlavní informace v něm obsažené jsou:

- seznam všech souborů, které jsou v balíku a cesta k nim
- typy média MIME type souborů uložených v balíku
- pokud je soubor šifrován, obsahuje informace potřebné k dešifrování souboru

Celý balík tvořící výsledný dokument je komprimován metodou ZIP a obsahuje následující soubory tvořící základ většiny kancelářských dokumentů:

Adresář / soubor	**Obsah**
mimetype	Obsahuje pouze jedinou řádku, která definuje typ MIME dokumentu (podle tabulky 2.3).
content.xml	Jednotlivé poddokumenty tvořící samotný Open Document podle tabulky 2.1.
styles.xml	
meta.xml	
settings.xml	
META-INF/manifest.xml	Seznam souborů v balíku a další informace o nich – cesty k souborům, jejich typy médií (mime typ).
Pictures/	Adresář obsahující obrázky v nativním, binárním formátu
ObjectN/	Adresář, který obsahuje další vložené objekty (např. grafy), může být více číslovaných adresářů podle počtu objektů.
Thumbnails/	Adresář obsahující náhled první stránky dokumentu.
Configurations2/	Adresář obsahující individuální nastavení aplikace .

Tabulka 2.2: části a struktura balíku Open Document

Výše uvedená struktura představuje klasickou strukturu balíku OpenDocument, nemusí ale vždy přesně odpovídat u konrétních dokumentů z praxe. Informace o nastaveních aplikace nebo náhled první strany jsou většinou generovány kancelářskými aplikacemi a během implementace exportních funkcí se ukázalo, že nepřítomnost těchto informací nezabraňuje korektnímu otevření dokumentu. Například adresáře obsahující obrázky nebo objekty jsou vytvářeny až pokud je do dokumentu skutečně vkládán obrázek nebo graf. Pokud tedy obsahuje dokument vložené objekty nebo obrázky v binárním formátu, musí být uvedeny v manifestu cesty k těmto vloženým souborům a jejich mime typy. Příklad takového souboru META-INF/manifest.xml generovaného aplikací OpenOffice.org (verze 2.0.4)

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest:manifest
xmlns:manifest="urn:oasis:names:tc:opendocument:xmlns:manifest:1.0">
<manifest:file-entry
manifest:media-type="application/vnd.oasis.opendocument.spreadsheet"
manifest:full-path="/"/>
<manifest:file-entry
manifest:media-type="application/vnd.sun.xml.ui.configuration"
manifest:full-path="Configurations2/"/>
<manifest:file-entry manifest:media-type="image/jpeg"
manifest:full-path="Pictures/10000000000000AB000000D36A9BBBFE.jpg"/>
<manifest:file-entry manifest:media-type="image/jpeg"
manifest:full-path="Pictures/1000000000000167000000F0B5205E0F.jpg"/>
<manifest:file-entry manifest:media-type="text/xml"
manifest:full-path="content.xml"/>
<manifest:file-entry manifest:media-type="text/xml"
manifest:full-path="styles.xml"/>
<manifest:file-entry manifest:media-type="text/xml"
manifest:full-path="meta.xml"/>
<manifest:file-entry manifest:media-type=""
manifest:full-path="Thumbnails/thumbnail.png"/>
<manifest:file-entry manifest:media-type=""
manifest:full-path="Thumbnails/"/>
<manifest:file-entry manifest:media-type="text/xml"
manifest:full-path="settings.xml"/>
</manifest:manifest>
```

V následující tabulce jsou uvedeny řetězce, které udávají jednotlivé MIME typy souborů s dokumenty v OpenDocument formátu. Každému MIME typu odpovídá jiný typ dokumentu tak jak jsou uvedeny ve specifikaci [4]. Pro každý typ souboru je vlastní přípona vycházející ze zkratky přirozeného pojmenování dokumentu (například „ott“ z „open text template, „ods“ pro „open document sheet“). Názvy jednotlivých typů začínají na „`application/vnd.oasis.opendocument`“, dále se liší podle druhu dokumentu (textový dokument, obrázek) a případně následuje údaj o tom, že jde o šablonu pro dokument daného typu.

Mime typ / typ média	Přípona	popis
`application/vnd.oasis.opendocument.text`	`odt`	Textový dokument
`application/vnd.oasis.opendocument.text-template`	`ott`	Šablona textového dokumentu
`application/vnd.oasis.opendocument. graphics`	`odg`	Grafický dokument

Mime typ / typ média	Přípona	popis
application/vnd.oasis.opendocument.graphics-template	otg	Šablona grafického dokumentu
application/vnd.oasis.opendocument.presentation	odp	Dokument prezentace
application/vnd.oasis.opendocument.presentation-template	otp	Šablona dokumentu prezentace
application/vnd.oasis.opendocument.spreadsheet	ods	Dokument tabulkového procesoru
application/vnd.oasis.opendocument.spreadsheet-template	ots	Šablona tabulkového dokumentu
application/vnd.oasis.opendocument.chart	odc	Graf
application/vnd.oasis.opendocument.chart-template	otc	Šablona grafu
application/vnd.oasis.opendocument.image		Obrázek
application/vnd.oasis.opendocument.image-template		Obrázek použitý jako šablona
application/vnd.oasis.opendocument. formula	odf	Vzorce
application/vnd.oasis.opendocument. formula-template	otf	Vzorce použité jako šablona
application/vnd.oasis.opendocument.text-master	odm	Globální textový dokument
application/vnd.oasis.opendocument.text -web	oth	Textový okument použitý jako šablona pro HTML

Tabulka 2.3: typy média - MIME typy - jednotlivých dokumentů (podle [4])

2.2.2 Metadata

Metadata jsou v širším slova smyslu „data o datech“, v informačních technologiích se jimi rozumí statistické údaje o souborech a dokumentech. Existuje též množina předdefinovaných metadata elementů, které by měly být zpracovávány a aktualizovány aplikací. Mohou být vynechány, nebo se objevovat vícekrát. Ošetření aktualizace při vícenásobném opakovaní je ponecháno na zpracovávající aplikaci. Mezi předdefinovaná metadata patří například autor, datum poslední změny, klíčová slova, sledování změn v dokumentech, nebo třeba statistické údaje o dokumentu jako počet obrázků a slov. Lze také vytvářet vlastní metadata, která by měla být zpracovávajícími aplikacemi uchovávána. Vzhledem k tomu, že pro tento cizí obsah není definována sémantika, aplikace jej nijak nemusí zpracovávat, měla by ho pouze beze změn uchovat. Přehled nejpoužívanějších metadat předdefinovaných kancelářskými aplikacemi je uveden v tabulce níže.

Typ dokumentu	Statistiky dokumentu
Text	meta:page-count
	meta:word-count
	meta:character-count
	meta:paragraph-count
	meta:frame-count
	meta:draw-count
Tabulky	meta:page-count
	meta:table-count
	meta:image-count
	meta:cell-count

2.2.3 Styly

OpenDocument podporuje několik druhů stylů, které jsou podle definice rozděleny do tří hlavních kategorií. Z pohledu uživatele jsou všechny tyto styly nedílnou součástí dokumentu. Společně obsahují informace o rozložení a formátování dokumentu, které jsou nezávislé na zobrazovacím zařízení. Výchozím předpokladem je, že autor dokumentu chce zachovat toto formátování a informace ať je dokument zobrazen na kterémkoliv výstupním zařízení.

2.2.3.1 Bežné styly (Common styles)

Většina aplikací v kancelářských balíčcích umožňuje používat a definovat styly, například nadpisu nebo vlastní písma v odstavci. Ve specifikaci OpenDocument se na ně odkazuje termínem styly, protože tento typ má nejblíže k stylům jak je chápe uživatel. V případě kdy je nutné rozlišení od jiných stylů používá se pro ně termínu „bežný styl“.

2.2.3.2 Automatické styly (Automatic styles)

Za automatické se označují styly, které z uživatelova pohledu slouží k formátování odstavců nebo tabulek v dokumentech. Formátování, která jsou okamžitě přiřazena určitému objektu (odstavec), jsou reprezentována automatickým stylem. Tímto způsobem je oddělen obsah dokumentu od vzhledu.

2.2.3.3 Hlavní styly Master styles

Za hlavní styl se označuje obecný styl, který nese kromě formátovacích údajů ještě dodatečný obsah. Tento obsah navíc je při použití stylu zobrazen spolu s běžným obsahem dokumentu. Příkladem jsou hlavní stránky, které například v grafických aplikacích obsahují dodatečné tvary a kresby vykreslené na pozadí každé stránky. V textových dokumentech je hlavními styly tvořeno záhlaví a zápatí stránky.

Obsah „navíc“ v hlavním stylu ovlivňuje konečný vzhled dokumentu, nemění ale žádným způsobem samotný obsah dokumentu. Hlavním stylem lze například v grafických dokumentech definovat jednotné pozadí pro všechny strany (lze použít například k umístnění firemního loga).

2.2.4 Tabulky v Open Document

Specifikace Open Document Format v části věnované tabulkám uvádí strukturu tabulek podobnou struktuře podle specifikací XSL a HTML. Stejně jako tyto tabulky mohou být i tabulky ve formátu OpenDocument zanořeny jedna do druhé. Základní specifikace tabulek pro OpenDocument je stejná pro tabulkové procesory stejně jako textové editory. Speciální vlastnosti tabulek podporované pouze tabulkovými procesory jsou poté popsány samostatně.

Zápis tabulek v dokumentu OpenDocument je založen na mřížce řádků a sloupců. Řádky tabulky jsou v hierarchii nadřazeny sloupcům. Tabulka se dělí na řádky a každý řádek se následně dělí na buňky. Každý sloupec obsahuje popis sloupce, ale jednotlivé sloupce již neobsahují popisy buněk. Jednotlivé řádky tabulky mohou být prázdné a různé řádky mohou obsahovat různý počet buněk tabulky. V souboru je uložena pouze skutečně použitá část tabulky, záměrem je snížit velikost balíčku. Každá aplikace toto řeší jiným způsobem, většinou jsou při načítání souboru tabulkovým procesorem prázdné a neúplné řádky doplněny buňkami. Ostatní aplikace používají tabulky pevných rozměrů a případné neúplné řádky vykreslí jako by měly potřebný počet prázdných buněk.

Specifikace formátu OpenDocument je komplexnější a definuje také rozšířený model tabulky. Tabulky mohou být za určitých podmínek vnořeny jedna do druhé, v případě vícestránkových tabulek umožňuje opakovat hlavičky řádků i sloupců tabulky na následujících stranách. Rozšířený model umožňuje definovat rozsah buněk, nad kterými lze provádět třídění, filtrování obsahu a databázové operace. Nad takovými tabulkami lze provádět operace pomocí jazyka SQL. V následující části je popsán především základní model tabulek.

Kořenovým elementem tabulek je `<table:table>`, ke kterému bývá připojeno několik atributů. Obsah elementu tabulky v základě tvoří řádkové a sloupcové elementy. Atributy tvořící součástí kořenového elementu tabulky jsou název tabulky, styly použité v tabulce, atribut „protected“ a další. Některé z těchto atributů jsou podrobněji popsány dále.

2.2.4.1 Název tabulky a styl tabulky

Název tabulky je reprezentován atributem `<table:name>` a má hodnotu řetězce, například „List1“. Styl tabulky je reprezentován atributem `<table:style-name>` jehož hodnota odkazuje na styl tabulky představovaný elementem `<style:style>`. Styl tabulky obsahuje formátovací údaje tabulky například barvu pozadí a šířku. Styl tabulky může být buď automatický nebo běžný.

```
<style:style style:name="Table 1" style:family="table">
<style:table-properties style:width="12cm"
fo:background-color="light-grey"/></style:style>
<table:table table:name="Table 1" table:style-name="Table 1">
</table:table>
```

2.2.4.2 Atribut protected

Atribut `<table:protected>` udává zda je tabulka uzamčena například pro zápis. Pokud je tabulka uzamčena může další atribut `<table:protection-key>` určovat heslo použité k uzamčení a zabraňující uživateli zrušit nastavení atributu `<table:protected>` bez znalosti hesla. Z bezpečnostních důvodů není v XML souboru uloženo přímo heslo, ale pouze hashovaná hodnota hesla.

2.2.4.3 Atributy print a print-ranges

Atribut `<table:print>` specifikuje zda má být tabulka vytištěna a nabývá booleovské hodnoty. Pro hodnotu „true“ je tabulka vytištěna, pokud je hodnota „false“ není tabulka tištěna. Pokud není atribut v dokumentu uveden, jeho výchozí hodnota je „true“. Atributu print je nadřízen atribut `<table:display>`, to znamená pokud tabulka není viditelná, nemůže být vytištěna nezávisle na hodnotě atributu `<table:print>`. Skutečný rozsah tištěné tabuky je definován atributem `<table:print-ranges>`, v případě, že atribut chybí, je vytištěna celá oblast tabulky.

2.2.5 Obsah tabulky

Element `<table:table-row>` představuje právě jeden řádek tabulky. Obsahem řádku tabulky jsou elementy představující buňky v tomto řádku. Každý element může mít jeden nebo více atributů z nichž nejpodstatnější jsou popsány níže. Pomocí elementu `<table:row>` lze popsat i více řádků tabulky použitím atributu `<table:number-rows-repeated>`. Další element tvořící obsah tabulky je element `<table:table-column>`. Stejně jako v případě elementu `<table:row>` lze pomocí jednoho sloupce popsat i více sloupců, za použití atributu `<table:number-columns-repeated>`. Příklad ukazuje tabulku popsanou pomocí řádků a použití atributu pro opakované sloupce. Každý element `<table:table-cell>` představuje právě jednu buňku tabulky. Na níže uvedeném příkladu je tabulka o dvou řádcích a třech sloupcích, která využívá právě atributu pro opakování sloupců. Další atribut, který je často součástí tabulek, konkrétně buněk, je `<office:value-type>`. Tento atribut specifikuje jakého datového typu (číslo, text, datum, apod.) jsou buňky v tabulce a pokud není přítomen, pro buňky bývá využit výchozí formát.

```
<table:table table:name="Sheet1" table:print="false">
<table:table-column table:number-columns-repeated="3"/>
<table:table-row>
<table:table-cell office:value-type="string">
<text:p>a</text:p></table:table-cell>
<table:table-cell office:value-type="string">
<text:p>b</text:p></table:table-cell>
<table:table-cell office:value-type="string">
<text:p>C</text:p></table:table-cell>
</table:table-row><table:table-row>
<table:table-cell office:value-type="string">
<text:p>E</text:p></table:table-cell>
<table:table-cell office:value-type="string">
<text:p>F</text:p></table:table-cell>
<table:table-cell office:value-type="string">
<text:p>D</text:p></table:table-cell>
</table:table-row>
</table:table>
```

2.2.6 Rámce

Rámcem je obdélníkový kontejner obsahující rozšířený obsah, jako jsou obrázky, textboxy nebo objekty. Rámec může obsahovat více zobrazení jednoho objektu. Například může obsahovat objekt a obrázek, aplikace si poté může vybrat obsah, který podporuje nejlépe a ten zobrazit. Aplikace nesmí zobrazit více než právě jeden z elementů obsažených v rámci. Pořadí elementů obsahu uložených v rámci určuje autor dokumentu a aplikace by měla zobrazit první element v pořadí, který umí zpracovat. Rámec musí obsahovat alespoň jeden element, použití vícenásobných elementů je volitelné. Pokud aplikace neumí zobrazit některý element v rámci, může ho ponechat, ale není to povinné. Rámce lze také použít například k umístění textu mimo klasický tok textu v dokumentu.

Každý rámec je v dokumentu tvořen elementem `<draw:frame>`. Rámce mohou mít připojeny různé atributy, které jsou společné pro ostatní tvary a kresby (drawing shapes). Nejdůležitější z nich jsou velikost, umístění a název rámce. Další atribut vlastní pro tabulkové dokumenty je koncová pozice. Rámce jsou klíčovou částí dokumentu, protože obrázky nebo grafy se do struktury dokumentu vkládají právě jako podelementy rámců. Rámce se také využívají jako kontainer pro různé pluginy, tzn. binární objekty, které kancelářská aplikace neumí sama zpracovat. Pluginy jsou charakterizovány svým typem média (MIME-type).

2.2.6.1 Název rámce

Název rámce je určen hodnotou atributu <draw:name>, aplikace jej mohou použít například k vykreslení alternativního textu pokud není zobrazen obsah rámce. Název rámce lze také použít v odkazech v rámci dokumentu.

2.2.6.2 Umístění rámce a velikost

Atributy <svg:x> a <svg:y> udávají umístění levého horního rohu rámce v dokumentu, jeho velikost je určena atributy <svg:width> a <svg:height>, udávající šířku a výšku rámce vzhledem k levému hornímu rohu. Šířka a výška mohou být absolutní, nebo mít relativní hodnoty vzhledem k nadřazenému objektu (například 50% šířky stránky).

2.2.6.3 Koncová pozice

V dokumentu tabulkového procesoru lze ukotvit rámec do konkrétní buňky dokumentu. V tom případě lze pomocí atributu <table:end-cell-address> určit buňku, do které bude rámec ukotven a atributy <table:end-x> a <table:end-y> upřesnit konečnou pozici rámce. V tomto případě jsou ignorovány atributy udávající klasickou velikost rámce. Souřadnice koncové pozice jsou odvozeny od levého horního rohu buňky.

2.2.7 Obrázek v Open Document

Obrázek je v dokumentu reprezentován elementem <draw:image> a může být:

- obsažen v dokumentu jako odkaz na zdroj vně obsahu (do balíčku)
- přímo vložen do těla dokumentu

Atributy, které mohou být spojeny s elementem obrázku <draw:image> jsou jednak samotná obrazová data a také název filtru který aplikace použije. Obrazová data mohou mít libovolný formát, doporučuje se použití formátu SVG pro vektorovou grafiku a formátu PNG pro bitmapy. Většina aplikací pro OpenOffice podporuje také obrázky ve formátech JPG, EPS, TIFF.

Pokud je obrázek vložený do dokumentu jako odkaz pak obrazová data bývají uložena v externím souboru (většinou obsaženém v balíčku dokumentu). Element <draw:image> obsahuje několik atributů <xlink:href> popisujících sémantiku odkazu.

Obrazová data mohou být také vložena přímo v těle dokumentu. V tom případě jsou data v elementu <office:binary-data>, který je obsažen v elementu <draw:image> a uložena binárním kódováním BASE-64. Za tohoto uspořádání není vyžadována přítomnost atributu <xlink:href>. Možnost uložení obrázku jako odkazu na binární soubor a jeho umístění ve složce Pictures balíčku využívají například kancelářské aplikace. Příklad obrázku vloženého pomocí odkazu je uveden níže. Ve výsledném dokumentu bývá element obrázku obsahem rámce.

```
<draw:image xlink:href="Pictures/10000000000000AB000000D36A9BBBFE.jpg"
xlink:type="simple" xlink:show="embed" xlink:actuate="onLoad"><text:p/>
</draw:image>
```

2.2.8 Objekty v Open Document

Grafy jsou ve formátu Open Document chápány jako objekty. Graf je jednak samotným dokumentem, ale může být v rámci jiných dokumentů (například sešitů tabulek) uložen jako objekt. Specifikace formátu rozlišuje mezi dvěma typy objektů, a to následujícím způsobem:

- objekty, které je možné vyjádřit zápisem v XML, mezi jinými to jsou textové dokumenty, dokumenty tabulkových procesorů, prezentace, kresby a grafy
- objekty, které nemohou být vyjádřeny zápisem v XML, (OLE objekty, skripty).

Každý z typů objektů je v dokumentu reprezentován jedním z dvou elementů, `<draw:object>` pro objekty reprezentované XML zápisem a `<draw:object-ole>` pro objekty s reprezentací pouze binární. Aplikace podporující ukládání objektů v dokumetu by měly podporovat odkaz na objekty uložené přímo v balíčku dokumentu. Také mohou podporovat odkazování na objekty uložené samostatně mimo balíček, například na soubory médií. Každý element může mít připojeny některé z následujících atributů:

data objektu (object data)

oznámení o změně tabulky (table change notification)

ID třídy (class ID)

2.2.8.1 Data objektu

Mohou být uložena v dokumentu některým z následujích způsobů podobně jako obrázky:

- jako atribut `<xlink:href>`, odkazující na vnější reprezentaci dokumentu. Rozlišují se objekty které jsou zapsané pomocí XML. V tom případě atribut odkazuje na balíček, ve kterém je obsah objektu zapsán stejně, jako by se jednalo o samostatný dokument. Objekt zapsaný v XML je popsaný například `<office:document>` nebo `<math:math>` elementem. Pro objekty, které jsou zapsány jinak než v XML odkazuje atribut `<xlink:href>` na soubor v balíku, který obsahuje binární data objektu.
- uložena přímo do dokumentu, v elementu `<draw:object>` nebo `<draw:object-ole>`, kdy použitý element závisí na zápisu objektu. V těchto případech nemusí být atribut `<xlink:href>` v elementu uveden. Objekty uložené přímo v dokumentu, které jsou zapsány binárně, musí obsahovat v sobě element `<office:binary-data>`, který obsahuje data objektu v binárním kódování BASE-64.

2.2.8.2 Oznámení o změně tabulky

Protože některé objekty, především grafy, mohou potřebovat oznámit změnu tabulky v dokumentu, OpenDocument specifikace obsahuje atribut `<draw:notify-on-change-of-table>`. Atribut lze připojit k elementu obsahujícímu příslušný objekt, jeho hodnotou je název tabulky, při jejíž změně má být zaslána notifikace příslušnému objektu. V případě, že chceme oznámit pouze změnu určitého obsahu tabulky, můžeme použít atribut `<draw:notify-on-update-of-ranges>`. Pokud dojde ke změně v oblasti tabulky je o tom zaslána objektu zpráva a dojde k požadované akci, například překreslení grafu.

2.2.9 Grafy

Dokumenty grafů jsou vždy součástí ostatních dokumentů XML. Celý graf tvoří element `<chart:chart>`, který představuje všechen obsah grafu včetně nadpisu, legendy a grafického obsahu tvořícího vykreslovací oblast grafu. Data, která tvoří základ obsahu grafu jsou reprezentována tabulkou. Element grafu obsahuje informace o druhu grafu, elementy a atributy popisující vykreslovanou oblast grafu – oblast tisku (plot area), a datovou část grafu. Datová část grafu je realizována tabulkou. V elementu `<chart:plot-area>` je blíže popsána oblast tisku grafu.

Atribut `<chart:class>` udává typ grafu, jeho hodnotou je vždy řetězec „chart:type“, kde „type“ nahrazuje název konkrétního typu grafu. Mezi typy definované ve specifikaci patří například sloupcový (bar), koláčový (circle), spojnicový (line) nebo spojnicový s výplní (area).

2.2.9.1 Velikost grafu

Zobrazení grafu se řídí hodnotami atributů `<svg:width>` a `<svg:height>`. Pokud jsou tyto atributy vynechány, velikost grafu se řídí velikostí okna v němž je graf zobrazen (například rámec).

2.2.9.2 Mapování řádků a sloupců

Atributy `<chart:column-mapping>` a `<chart:row-mapping>` jsou volitelné a mohou poskytovat seznam tzv. indexů. Tyto indexy lze použít k změně řazení údajů pocházejících z rodičovského dokumentu. V jednom grafu nesmí být použity oba atributy současně.

2.2.9.3 Legenda grafu

Popisky grafu jsou obsaženy v elementu `<chart:legend>`, jehož přítomnost udává zda je zobrazena legenda nebo ne. Umístění popisků je určeno atributem `<chart:legend-position>`.

2.2.9.4 Kreslící plátno

Je definováno elementem <chart:plot-area> a obsahuje kromě samotné plochy pro vykreslení také údaje o hlavní a vedlejší ose grafu, popiscích grafu nebo rozměrech vykreslovaného plátna.

2.2.9.5 Tabulka v grafu

Datová část grafu je uložena v tabulce, která je zapsaná podle specifikace platné pro tabulky. Popisky hodnot grafu tvoří jeden řádek tabulky, druhý a další řádek jsou pak samotné údaje zobrazované do grafu. Tyto hodnoty mohou být přejmuty z tabulky v sešitu, kde je graf umístěn, nebo zapsány přímo v tabulce. Obecně vzato platí pro tabulky nesoucí data grafu stejné specifikace jako pro klasické tabulky sešitu. Pouze jsou uloženy v dokumentu obsahující graf a odlišují se hodnotou atributu <table:name>. Pojmenovány mohou být například jako „local-table“, místní tabulky.

3 Návrh systému a exportu

Záměrem práce je vytvořit aplikaci nebo modul, která uživatelům informačního systému umožní údaje uložené v systému přenést do jejich počítačů ve formátu, se kterým pracují tabulkové procesory. Aplikace musí umět spolupracovat se systémem a být přístupná pro všechny uživatele. Rozšíření a aktualizace aplikace by měla jít snadno nasadit do již existujícího prostředí. Z těchto důvodů je lepší realizovat program jako součást portálu než klientskou aplikaci, kterou by uživatelé individuálně instalovali do svých počítačů.

Pro tvorbu portálových řešení existuje mnoho jazyků, například ASP, PHP, nebo JAVA. Pro naše účely zvolíme jazyk PHP, konkrétně verzi 5. Jazyk má přehlednou syntaxi, je široce používán a podporován. Moduly vytvářené v jazyce PHP lze také snadno napojovat na systémy založené na jiných jazycích. V neposlední řadě hrál roli také vylepšený objektový model oproti starším verzím a stále se rozšiřující počet webhostingů podporujících novou verzi. Objektový model jazyka nám dovolí zpřístupnit funkcionalitu výsledné aplikace několika jednoduchými metodami. Pro pochopení dalšího návrhu je důležité si připomenout, že chování programu se v objektovém modelu nazývá metodami a vlastnosti jsou označovány jako atributy.

3.1.1 Princip exportu

Výsledkem exportu dat má být kompletní dokument ve formátu OpenDocument pro tabulkový procesor. Z popisu formátu víme, že na soubor v tomto formátu lze nahlížet jako na balík jednotlivých dílčích souborů, ať už binárních nebo ve formátu XML, komprimovaný pomocí metody ZIP. Po prostudování PHP rozšíření pro ZIP a dalších jsme se rozhodli použít volně šiřitelného modulu File_Archive, který je k dispozici v rámci distribučního systému PEAR. Balíček je šiřitelný pod podmínkami GNU General Public Licence. Vytvářená aplikace může s výhodou využívat jeho funkcí pro vytváření souborů a manipulaci se soubory v rámci archívu. Navíc použití již existujícího řešení nám umožní soustředit se na podstatné části problému, tedy vytvoření správné struktury balíku a uložení dat do struktury formátu OpenDocument.

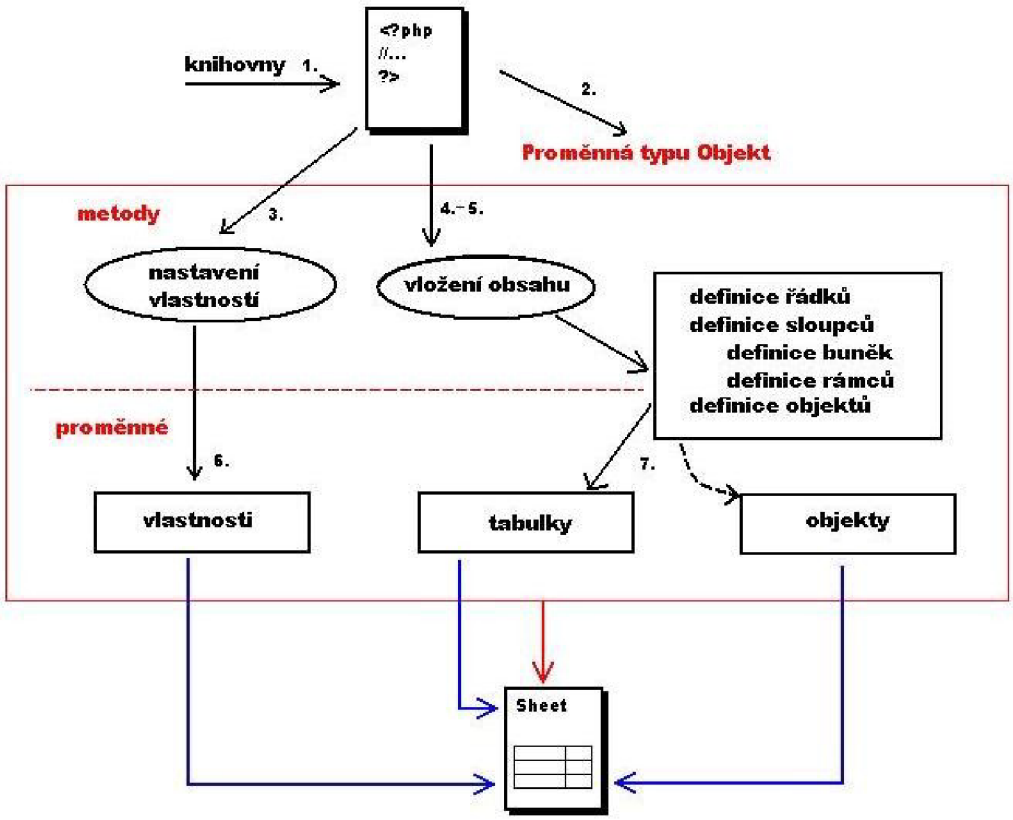
Chceme realizovat skripty pro export dat tak, aby byly co nejvíce použitelné a snadno začlenitelné do systému. Knihovny zajišťující export dat by měly obsahovat proměnné, do kterých můžeme pomocí veřejných metod ukládat data získaná skriptem z databáze. Bude nutné vytvořit třídu, z níž budou odvozeny objekty, definovat vlastnosti třídy a potřebné metody. Jedna ze tříd (respektive její objekt) bude obsahovat metody označované jako „setters“ pro ukládání dat získaných skriptem do vlastností tohoto objektu. Druhá třída poté pomocí „getters“ tyto vlastnosti získá a použije při vytváření XML struktur výsledného dokumentu. Zde je na místě vysvětlit co znamená

pojem getters a setters. Jedná se o speciální metody, které se volají při zápisu nebo čtení některé vlastnosti. Významem tohoto rozdělení je oddělit metody starající se o shromažďování dat ze systému od metod, které vytvářejí samotný dokument.

3.1.2 Třídy pro export

Exportní modul se sestává ze dvou hlavních knihoven, první je spreadsheet.php. V této knihovně je definována třída SpreadSheet, obsahující funkcionalitu pro manipulaci s daty předávanými z informačního systému. Knihovna nevytváří sama o sobě soubor, pro jejich použití je nutné připojení knihovny do skriptu. Skript volá metody z třídy Spreadsheet a předáním dat jako prarametrů metod plní struktury této třídy potřebnými daty. Pro provedení exportu je nutné nejdříve vytvořit v skriptu nový objekt třídy Spreadsheet a zavolat jeho metody AddRow, AddChart nebo AddPicture. Z pohledu třídy Spreadsheet je dokument vlastnostmi, které je možné naplnit těmito metodami. Vlastnosti třídy mají hodnoty reprezentující obsah dokumentu a statistiky exportovaných dat. Přístup k nim je možný pomocí veřejných metod (například getContent, getPictures a getCharts). Třída Spreadsheet také v konstuktoru třídy nastavuje základní metadata a typ mime dokumentu. Uložení souboru je v exportním skriptu provedeno voláním metody save ze třídy Spreadsheet.

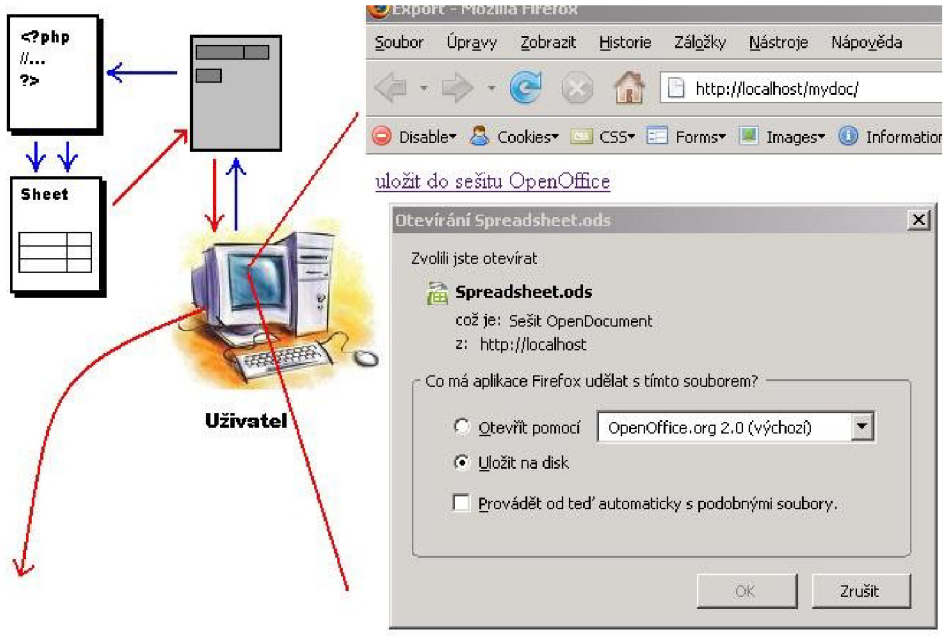
:alizováno



Obr 3.1: Schéma exportu dokumentu

Ve vytvářeném dokumentu formátu OpenDocument jsou použity binární formáty obrázků, které jsou do výsledného dokumentu vloženy v rámcích jako odkazy na soubory uložené ve složce Pictures v balíku. Vkládání souboru je prováděno funkcí ve třídě SpreadsheetWriter, která originální soubor otevře a přečte. Data prečtená jako binární řetězec znaků uloží do řetězce a poté řetězec zapíše jako obsah nově vytvořeného souboru pomocí metod z balíku funkcí File Archive. Grafy jsou v obsahu dokumentu uloženy jako odkazy na objekty v balíku, reprezentované vlastními soubory XML. Schéma na obrázku (3.1) demonstruje proces vytváření exportovaného dokumentu.

3.1.3 Výsledný dokument



Obr 3.2: průběh exportu z pohledu uživatele

Výsledný dokument nabídnutý uživateli ke stažení je již komprimovaný soubor pro tabulkový procesor. Dokument lze otevřít v aplikaci OpenOffice.org Calc (verze 2.0.4) a také na platformě Linux v aplikaci KSpread (součásti KOffice). Výsledkem jsou knihovny implementované v jazyce PHP verze 5, které díky svému objektovému modelu umožňují připojení ke skriptu a voláním jejich metod vytvořit výsledný dokument.

Export dat z systému je pro běžného uživatele realizován co nejjednodušeji. Uživatel si bude přát získat uložené údaje na serveru a uložit ve formě dokumentu pro tabulkový procesor, v našem případě například OpenOffice.org Calc. Nabídneme tedy uživateli odkaz po jehož stisknutí se spustí na serveru skript, který shromáždí požadované údaje, vygeneruje soubor ve formátu OpenDocument a nabídne jej uživateli ke stažení. Pokud nastane při vytváření souboru chyba, bude to uživateli oznámeno pomocí výpisu v internetovém prohlížeči.

3.2 Návrh informačního systému

Uživatelem systému bude malá firma zabývající se prováděním jednoduchých zakázek především pro běžné lidi. Například vyřizování reklamací nebo drobné opravy. Očekávanými součástmi systému je především evidence zaměstanců, zákazníků, dále přehled o vyřizovaných zakázkách a jejich splacení. Systém by měl také umožnit pracovníkům ukládat si do nějaké formy poznámky, na těchto částech lze demonstrovat nasazení technologie AJAX do informačního systému.

Systém budeme implementovat jako prototypové řešení. Prototyp se bude řídit návrhem popsaným dále, nicméně některé navržené komponenty mohou mít pouze omezenou funkčnost. Primárním účelem námi navrženého sysému bude demostrace využití technologie AJAX a především ověření možnosti exportu dat z informačního systému do formátu Open Document, používaného tabulkovými procesory.

3.2.1 Data v systému

Systém bude navržen především jako demonstrační prostředek, na kterém bude implementován export dat z databáze systému. V návrhu systému se proto soustředíme především na datovou strukturu a provázání jednotlivých entit v systému. Nejdůležitější data v systému jsou uložena v těchto prvcích (tabulkách) databáze: zaměstnanec, zákazník, zakázka, faktura. Každý z prvků obsahuje důležité specifické atributy. Jednotlivé prvky se navzájem velice ovlivňují a jsou provázány.

3.2.1.1 Zaměstnanec

Zaměstnance bude moci do systému přidat jen zaměstnanec s právy administrátor. Identifikace bude probíhat přihlašovacím jménem, které má mít každý uživatel unikátní, a autentizace heslem. Každý zaměstnanec disponuje určitými právy v systému, takže například řadový pracovník může přidávat zakázky nebo zákazníky, ale ne zasahovat do evidence zaměstnanců. Nezbytné jsou údaje o jménu a bydlišti každého zaměstnance. Volitelnými údaji poté mohou být kontaktní informace. V systému musí být vždy vytvořen pracovník s administrátorským oprávněním. Přehled údajů o zaměstnanci:

- Id, login a heslo
- Práva
- Jméno a příjmení
- Město, ulice, směrovací číslo
- Telefon, email
- Mzda

3.2.1.2 Zákazník

Zákazník je v systému chápán jako jedinec nebo zástupce firmy, která má nějakou adresu na níž společnost sídlí. Je tedy potřeba uchovat jméno, adresu a kontaktní údaje (telefon a e-mail) konkrétní osoby, která objednávku zadala. Zákazníkům budou vystavovány faktury, proto je nutné zavést nějaký osobní identifikátor. Zákazník jej nemusí samozřejmě znát, identifikátor slouží pro vnitřní potřebu.

- Id
- Jméno a příjmení
- Město, ulice, směrovací číslo
- Telefon, email

3.2.1.3 Zakázka

Zakázkou se rozumí požadavek zadaný firmě zákazníkem, například oprava přístroje nebo vypracování technické zprávy. Za každý požadavek zadaný zákazníkem přebírá zodpovědnost pracovník, který jej přijal, proto v tabulce musí být kromě identifikátoru zakázky jednoznačné identifikátory zadavatele a zodpovědného pracovníka. Popis zakázky udává např. druh nebo upřesňuje některý z parametrů. Podstatné je také datum zadání zakázky, jejího dokončení a poté datum předání zákazníkovi. Cena zakázky může být dohodnuta předem nebo stanovena později.

- Identifikátory zadavatele zakázky a zodpovědného pracovníka
- Popis zakázky, upřesňující její parametry nebo účel
- Datum zadání, datum dokončení
- Datum vyřízení (předání zákazníkovi)
- Cena zakázky (předem dohodnutá nebo určená později podle ceníku)

3.2.1.4 Faktura

Faktura představuje účet za provedenou práci a každá zakázka je splatná právě jednou fakturou. Protože zákazník může mít zadáno několik požadavků současně, případně může dojít k nahromadění faktur, musí každá faktura odpovídat právě jedné zakázce a také být přiřazena právě zákazníkovi, který objednávku zadal. Základní údaje pro fakturu tvoří:

- Především číslo faktury, identifikátory zakázky a zadavatele
- Datum splatnosti faktury
- Částka, na kterou je faktura vystavena a údaj zda už byla zaplacena

3.2.1.5 Úkoly

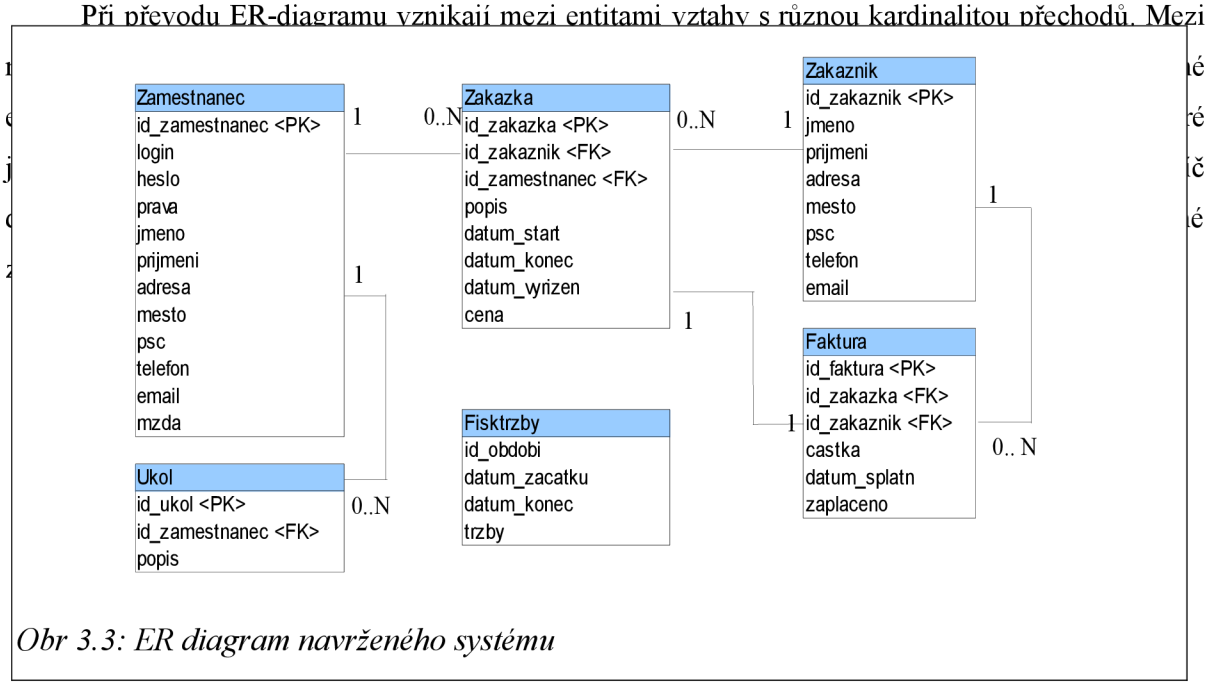
Pomocí úkolů chceme zaměstnancům – uživatelům systému umožnit jednoduchým způsobem vytvářet a do systému ukládat vlastní poznámky a připomínky. Z pohledu uživatele by měl být

způsob ukládání a práce s poznámkami intuitivní. Uživateli bude postačovat zadat popisek pro nový úkol a vytvořením se automaticky prováže s údaji uživatele.

3.2.2 Zjednodušený diagram tříd (ER-diagram)

Entity-relationship diagram nám umožňuje představit jednotlivé entity v systému a vztahy mezi nimi, které musíme poté převést do tabulek databáze. Normalizace je pojem, který značí proces převodu modelu na tabulky databáze a pravidla, která je nutno dodržet. Pomocí normalizace můžeme zjednodušit databázovou strukturu do stavu vhodného pro užití relační databáze, která má být v našem případě výsledkem tohoto procesu.

Při převodu ER-diagramu vznikají mezi entitami vztahy s různou kardinalitou přechodů. Mezi



Obr 3.3: ER diagram navrženého systému

3.2.2.1 Příklad převodu na tabulky databáze

Entitu faktury chceme převést na tabulku v databázi. Proto potřebujeme vztahy, které má entita Faktura s ostatními entitami v databázi převést podle uvedených pravidel. V tabulce Faktura realizujeme vztah 1:N s entitou Zákazník vytvořením cizího klíče, který se bude odkazovat na primární klíč v tabulce Zákazník. V případě realizace vztahu 1:1 stačí vytvořit v tabulce Faktura cizí klíč odkazující se na primární klíč v tabulce Zakázka. Bylo by možné zvolit i opačnou realizaci vztahu kde cizí klíč by byl v tabulce Zakázka a ukazoval na primární klíč Faktury.

3.3 Použití AJAXu

AJAX můžeme v systému využít nejlépe k poskytnutí funkcí, které uživateli urychlují nebo usnadňují práci s webovou stránkou. Mohou to být různé našeptávače a pomocníci pro vyhledávání, také

ověření údajů můžeme díky AJAXu provádět během činnosti uživatele, bez nutnosti načíst v prohlížeči jinou stránku, takže se vše blíží běžné desktopové aplikaci. Díky tomu je uživatel příjemně překvapen plynulostí procesů na ní. Většina akcí se dá provést bez nutnosti načítat celý obsah stránky znovu, provádějí se pouze aktualizace skutečně změněných informací. Musíme se snažit, aby uživatel neměl tendenci používat tlačítko Zpět (například pokud bychom vyvolali dojem nahrání nové stránky).

3.3.1 Implementace objektu XMLHttpRequest

Prohlížeče Firefox a Opera v posledních verzích podporují objekt XMLHttpRequest jako nativní objekt prohlížeče a implementují jej přímo, takže k vytvoření objektu stačí pouze zavolat příslušný konstruktor. Internet Explorer podporuje od verze 7 také přímou implementaci XMLHttpRequest, ale z důvodu zpětné kompatibility zachovává i implementaci pomocí ActiveX. Verze Internet Explorer 6 (se kterou je ještě dnes nutné počítat) používá k vytvoření objekt ActiveX. Při vytváření objektu XMLHttpRequest musíme proto vytvořit funkci, která by měla zakrýt různé způsoby vytváření objektů. Podrobným úvodem do problematiky použití AJAXu byla při tvorbě této práce publikace [2].

```
function createXmlHttpRequestObject() {
  var xmlHttp;
  try  { // pokusí se vytvořit objekt XMLHttpRequest pro nové prohlížeče
    xmlHttp = new XMLHttpRequest();  }
  catch(e)
  { try
      { // pokusí se vytvořit XMLHttpRequest objekt pro IE6 a starší
        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
      }
      catch (e) {}
    }
  // ověří zda je objekt vytvořen a vrátí ho nebo oznámí chybu
  if (!xmlHttp) alert("Nepodařilo se vytvořit objekt XMLHttpRequest");
  else return xmlHttp;}
```

3.3.2 Datová mřížka

Datová mřížka, v originále datagrid, někdy také grid view, je část uživatelského rozhraní umožňující dynamický pohled na data pomocí tabulkové reprezentace. Mřížku mohou například využít prohlížeče souborů pro manipulaci se seznamy souborů. Klasické implementace datové mřížky se vyznačuje několika možnostmi navíc oproti obyčejnému seznamu dat.

Typická mřížka umoží provádět jednu nebo více následujících akcí:

- kliknutím na záhlaví sloupce změnit řazení mřížky
- uchopením a tažením záhlaví sloupce změnit šířku a pořadí sloupců
- přímé editování zobrazených údajů v mřížce
- oddělovače řad a sloupců, změny barevného zvýraznění

Mezi hlavní nevýhody webových aplikací oproti desktopovým systémům patří právě práce s mřížkami. Přepínání mezi stránkami s mřížkami nebo jejich aktualizace vyžadovala vždy úplné znovunačtení stránky. S využitím přístupů, které nabízí AJAX, máme možnost realizovat mřížky ve webovém systému efektivněji a z uživatelského hlediska také příjemněji než dříve. Obsah stránky v prohlížeči klienta bude aktualizován bez nutnosti překreslovat ostatní obsah. Díky AJAXU se také úprava některého záznamu v tabulce stane snadnějším. Můžeme v Javascriptu definovat funkce pro editaci mřížky a pomocí AJAXU poté realizovat aktualizaci dat na pozadí. Protože schéma funkce datové mřížky je komplikované, není zde podrobněji rozebíráno. Zájemci mohou nahlédnout do komentovaných zdrojových textů aplikace na přiloženém CD.

Námi realizovaná datová mřížka umožňuje kliknutím do tabulky provést editaci vybraných údajů a poté změněné údaje ihned uložit. Přenos změněných dat na server se opět děje pomocí odeslání AJAXem v pozadí uživatelovy práce.

3.3.3 Kontrola formulářových údajů

Ověřování vstupních dat je nezbytnou součástí zabezpečení aplikací. Ve všech typech informačních systémů lze zadáním nesprávných údajů, ať již nedopatřením nebo úmyslně, způsobit selhání a ztrátu dat. Kontroly formulářových údajů bývají nejčastěji prováděny na serverech po odeslání formulářů. Některé úlohy můžeme provést díky skriptovacím jazykům přímo ve webových prohlížečích, například ověření správného formátu emailu nebo datumu. Tradiční techniky mají několik nevýhod, jako je nutnost čekat na znovunačtení stránky po odeslání údajů. Také bezstavový protokol HTTP neumožňuje znovu nabídnout uživateli po odeslání nesprávně vyplněného formuláře již předvyplněné údaje. Některé potíže lze obejít různými technikami, například skriptem v PHP data vyplnit a uživateli poslat při novém načtení stránky s formulářem. Využitím AJAXu lze tradiční techniky vylepšit o průběžnou kontrolu dat při zadávání uživatelem. Samozřejmě musí zůstat zachována také kontrola dat na straně serveru, aby se předešlo zneužití (například vypnutím Javascriptu).

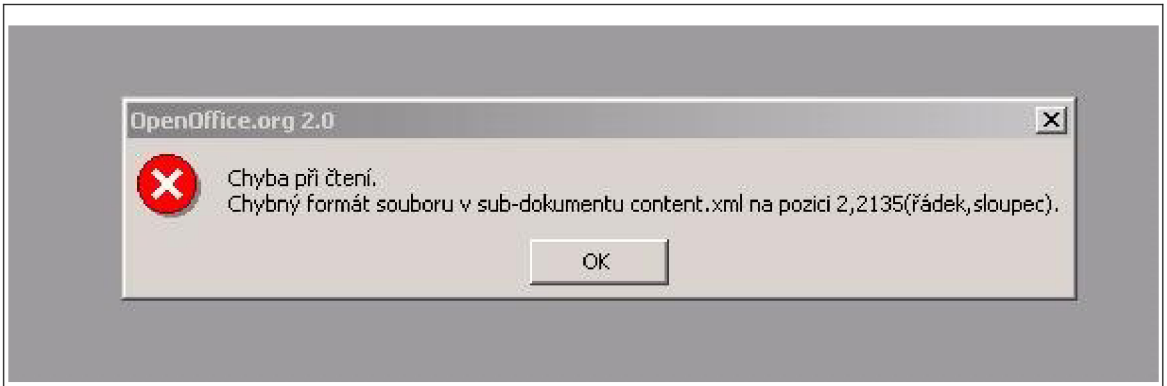
Kombinovaný způsob kontroly opět využívá asynchronního volání serveru. Uživatel při vyplňování formuláře přepíná mezi jednotlivými políčky. Pomocí JavaScriptu lze na přepnutí do jiného pole reagovat, a využitím objektu XMLHttpRequest odeslat právě vyplněná data na server. Výsledky ověření dat jsou poté serverem zaslány zpět prohlížeči a vhodným způsobem sdělena

uživateli, například formou upozornění pod špatně vyplněným polem. Uživatel může data buď opravit nebo pokračovat ve vyplňování formuláře bez nutnosti obnovovat stránku.

Vyplněním formuláře a stisknutím odesílacího tlačítka jsou údaje poslány na server klasickým způsobem. Protože nic nenutí uživatele opravit chybná data, musí na straně serveru znovu proběhnout úplné ověření dat. Správná data se uloží pro další zpracování, zatímco neplatné údaje způsobí opětovné načtení formuláře se zvýrazněnými špatně vyplněnými údaji. Využitím technik AJAXu a jejich spojením se standardními způsoby ověřování tak dostáváme uživatelsky příjemnější a zároveň bezpečný způsob jak zajistit integritu systému a údajů v něm.

3.4 Výsledky přenosu dat

Výsledné dokumenty pro tabulkové procesory, které jsme exportovali během vytváření této práce, mají vždy obecně menší velikost oproti srovnatelným dokumentům vytvářeným přímo kancelářskými aplikacemi. Způsobeno je to tím, že kancelářské aplikace ukládají do dokumentu také výchozí styly a často zahrnují údaje pro obnovu poškozených částí. Oproti tomu naše knihovny jsou zaměřeny na generování obsahové struktury a vnitřní reprezentace dokumentů je bez ukládaných stylů.

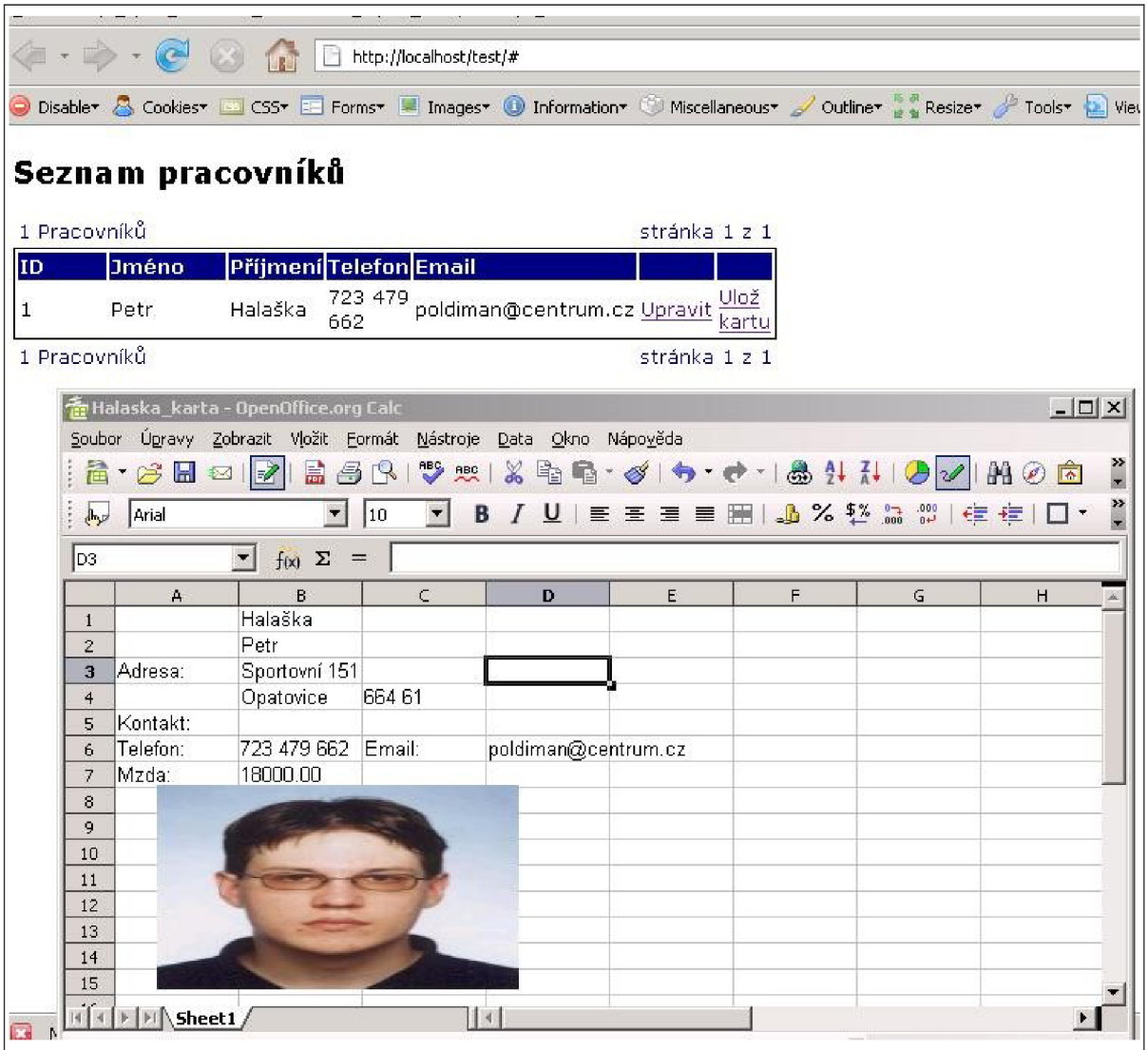


Obr 3.4: Chybové hlášení o poškozeném souboru (aplikace OpenOffice.org)

Při tvorbě funkcí pro export dat jsme se často setkávali s problémy v neúplně nebo nesprávně vytvářené struktuře XML. Vygenerovaný dokument se někdy nezobrazil se zamýšleným obsahem a nebo jej nebylo možné otevřít. Naštěstí aplikace OpenOffice.org, v níž jsme dokumenty po většinu doby testovali, přesněji určuje místo v dokumentu které nemohlo být zpracováno (viz obrázek 3.4).

Dalším problémem, který zahrnoval nejen exportování samotných údajů, ale také ostatní práci se systémem, bylo kódování znakových sad. Formát OpenDocument je založen na XML a kromě jiného používá shodně znakovou sadu UTF-8. Ačkoliv v databázi MySQL, v sytému i exportních skriptech bylo nastaveno kódování také na UTF-8, docházelo k opakovaným problémům se zobazováním a přenosem znaků české abecedy. Tento problém, popsaný v mnoha uživatelských diskuzích, byl nakonec vyřešen explicitním nastavováním kódové sady UTF-8 pro každé připojení.

Následující obrázek ukazuje kartu pracovníka exportovanou ze systému. Na mřížce se seznamem pracovníků je u každého záznamu možnost uložit kartu. Odkaz vede na soubor se skriptem PHP, který díky předanému číslu pracovníka získá z databáze údaje potřebné údaje a pomocí vytvořených metod tyto údaje uloží do generovaného dokumentu. Zatímco hlavičky jsou určeny přímo konkrétním skriptem, údaje o pracovníkovi jsou získány SQL dotazem na tabulku zaměstnanců v databázi informačního systému.



Obr 3.5: Příklad exportovaných dat

4 Závěr

V této práci jsme měli za cíl především prozkoumat strukturu formátů používaných kancelářských aplikacemi pro tabulkové dokumety a demonstrovat možnosti pro přenos dat z informačního systému do zvoleného formátu. Jako vybraný formát pro přenos dat ze systému byl použit OpenDocument. Důvodem volby formátu OpenDocument byla skutečnost, že se jedná o standardizovaný formát ISO/EIC 26300 [4] a je široce používán světovými firmami a vládními insitucemi (například v Brazílii také ozbrojenými silami) [7]. Předností formátu je skutečnost, že umožňuje vytvářet dokumenty složené z XML souborů a binárních souborů obsahujících také obrázky a grafy.

Vytvořené knihovny v jazyce PHP dávají uživatelům systémů možnost přenosu údajů z databáze do tabulkových procesorů. Díky metodám objektového návrhu a implementace těchto knihoven je lze připojit do existujících systémů bez nutnosti podrobné znalosti formátu OpenDocument. Pro zapojení exportních knihoven do systému stačí použití několika jednoduchých metod zastřešujících celkový proces vytvoření dokumentu. Knihovny nezahrnují všechny možnosti, které formát OpenDocument poskytuje. Umožňují vytvářet základní struktury tabulek a připojit k dokumentu vložením například obrázky produktů nebo shrnout údaje o produkci do grafu.

Návrh knihoven umožňuje rozšíření do budoucna o další funkce, které poskytuje formát OpenDocument, například export tabulek obsahující statistické funkce nebo vlastní předdefinované styly. Pro účely firemních systémů by byl vhodným vylepšením modul poskytující možnosti pro výběr a připojení předem uložených maker k vytvářeným dokumentům.

V rámci práce bylo demonstrováno několik možností využití technologie AJAX. Hlavní výhoda využití AJAXu je zpříjemnění a urychlení práce, protože uživatel nemusí čekat na aktualizace stránek. Při vkládání dat umožňuje AJAX zefektivnit ověřování dat a uživateli lépe oznámit nesprávně vložené údaje. Toto chování, spolu s možností editace údajů na stránce pomocí mřížky, je daleko blíže tomu, co zná uživatel z klasických desktopových aplikací. AJAX však stále zůstává pouze nadstavbou nad stávajícími webovými technologiemi. Jeho nevýhodami je nutnost využití JavaScriptu nebo jiného skriptovacího jazyka. Ačkoliv ve firemním informačním systému tato situace nebývá častá, po vypnutí skriptování přestává AJAX fungovat. Při změnách na stránce realizovaných pomocí AJAXu se nemění odkazy v adresním řádku prohlížeče. Stav v konkrétním okamžiku je definován stavem skriptových objektů na stránce a proto není možné takto modifikovanou stránku poslat e-mailem ani uložit do záložek, ovlivněny jsou i funce zpět a vpřed v prohlížečích.

Věříme že v naší práci se podařilo vytvořit praktický a do budoucna použitelný nástroj pro přenos dat z informačního systému do široce využívaného a přenositelného formátu. Velice zajímavým osobním přínosem bylo zkoumání a realizace struktury formátu OpenDocument.

Literatura

[1] Gutmans, A., Bakken, S. S., Rethans, D., *Mistrovství v PHP 5*., Brno, Computer Press 2007

[2] Darie, C., Brinzarea, B., Bucica, M., Chereches-Tosa, F., *AJAX a PHP – tvoříme interaktivní webové aplikace profesionálně*., Brno, Zoner Press 2006

[3] *Wikipedia, the Free Encyclopedia* [online], duben 2008, dostupné <http://en.wikipedia.org/>

[4] ISO/IEC 26300. *Open Document Format for Office Applications (OpenDocument) v1.0 (Second Edition)*, 2006, dostupné z <http://www.oasis-open.org/committees/download.php/19275/OpenDocument-v1.0ed2-cs1.odt>

[5] Rice, F., *Introducing the Office (2007) Open XML File Formats* [online] 2006 dostupné z <http://msdn.microsoft.com/en-us/library/aa338205.aspx>

[6] Garret, J.J. *Ajax: A New Approach to Web Applications* [online]. 2005 [cit. 22.dubna 2008] dostupné z <http://www.adaptivepath.com/ideas/essays/archives/000385.php> poslední úpravy 18. 2. 2005

[7] *OpenDocument Format Alliance* [online]. 2007 [cit. 2. května 2008], dostupné z <http://www.odfalliance.org/> poslední aktualizace 7. 4. 2008

Seznam příloh

Příloha 1. CD se zdrojovými texty