VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

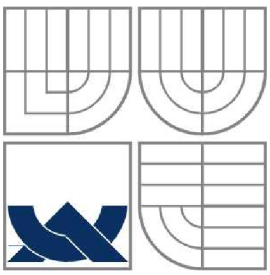
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

TABULKOVÉ PROCESORY JAKO ZOBRAZENÍ DAT PRO OLAP

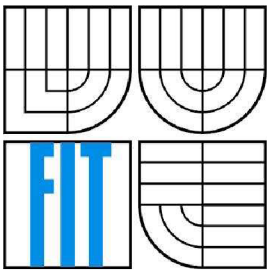
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Alois Kužela
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

TABULKOVÉ PROCESORY JAKO ZOBRAZENÍ DAT PRO OLAP

OLAP INTERFACE IN SPREADSHEET FORM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Alois Kužela

VEDOUCÍ PRÁCE
SUPERVISOR

Prof. Ing. Tomáš Hruška, CSc.

BRNO 2007

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2006/2007

Zadání diplomové práce

Řešitel: **Kužela Alois**
Obor: Výpočetní technika a informatika
Téma: **Tabulkové procesory jako zobrazení dat OLAP**
Kategorie: Databáze

Pokyny:

1. Prostudujte možná zobrazení výstupních údajů systémů OLAP.
2. Seznamte se s vnitřními přenositelnými formáty údajů pro tabulkové procesory (SYLK, XML formát MS Excelu apod.) .
3. Seznamte se se způsobem exportu údajů z portálových informačních systémů do tabulkových procesorů.
4. Po zhodnocení systémů navrhněte systém exportu dat do tvaru tabulkového procesoru pro informační systém pracující v prostředí internetu.
5. Realizujte prototypové řešení navrženého systému.

Literatura:

- dle pokynů vedoucího práce

Při obhajobě semestrální části diplomového projektu je požadováno:

- Bez požadavků.

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

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% 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 paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Hruška Tomáš, prof. Ing., CSc.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 22. května 2007

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

Licenční smlouva je uložena v archivu Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Tato diplomová práce zkoumá možné způsoby přenosu dat z portálových aplikací do tabulkových procesorů. Cílem je navrhnout řešení, které by za použití vhodných formátů souborů umožnilo získat data pro použití v aplikaci MS Excel.

Práce také stručně vysvětluje pojem OLAP, možnosti jeho použití, vnitřní datové struktury a způsob exportu údajů. Podrobněji pak rozebírá různé používané přístupy pro získání dat z webových portálů. Detailně rozebírá textové formáty SYLK a XML, které jednoduchým způsobem umožňují přenos dat mezi různými aplikacemi a tedy jsou vhodným prostředkem pro generování dat také pro tabulkové procesory.

Klíčová slova

CSV, databáze, Excel, export dat, formát souboru, HTML, informační systém, internet, OLAP, OpenOffice, portál, spreadsheet, SYLK, tabulkový procesor, web, XML

Abstract

This thesis envisages the possibilities of data transport from portal applications into spreadsheets. The main goal is to find out a solution to export data in suitable form for MS Excel application.

The thesis also shows principles of the OLAP. It describes OLAP internal data model and shows the way, how data could be exported from a website into spreadsheets. It enlarges SYLK and XML file formats, which are suitable for data representation in MS Office Excel spreadsheet.

Keywords

CSV, database, Excel, data export, file format, HTML, information system, internet, OLAP, OpenOffice, portal, spreadsheet, SYLK, web, XML

Tabulkové procesory jako zobrazení dat pro OLAP

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením
Prof. Ing. Tomáše Hrušky, CSc.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jméno Příjmení
Datum

Poděkování

Tímto způsobem bych chtěl poděkovat vedoucímu projektu Prof. Ing. Tomáši Hruškovi, Csc., především za pomoc v počátcích zpracování této práce a dalšímu směrování ke zdárnému výsledku. Lence Zouharové za pomoc, podporu a motivaci především v dobách *temna*.

Obsah

1. Úvod

V dnešní době je již samozřejmostí využívat internet pro přístup k nejrůznějším druhům informací. Existuje řada různých aplikací a technologií, které umožňují skladovat data a prezentovat je rozličným způsobem. Hardwarově špičkově vybavené servery mohou uchovávat obrovské množství dat v databázích. S využitím příslušného softwaru mohou nad těmito daty provádět složité a časově náročné analýzy a připravit výsledné požadované informace pro snadné zobrazení uživatelům.

Výkony uživatelských počítačů se stěží dají srovnávat s výkony takovýchto serverů, stejně tak valná většina z nich nemá k dispozici potřebné programové vybavení, které by umožňovalo zpracovávat výsledky takových analýz. Uživatelé pracují především prostřednictvím internetových prohlížečů. Portálová aplikace tedy musí zajistit vhodnou reprezentaci výsledků. Aby bylo výsledky možné nejenom prohlížet, ale případně s nimi také rozumným způsobem pracovat, je vhodné nabídnout je v některém alternativním formátu pro poměrně rozšířené uživatelské aplikace, jako jsou tabulkové procesory. Tyto se dají v dnešní době již považovat za jedny z esenciálních programů při práci s počítačem.

Mezi dnes zřejmě nejpoužívanější patří tabulkový procesor Microsoft Excel, především na platformě Windows. Dalším poměrně známou a navíc volně dostupnou možností je využití programu Open Office Calc.

Tato práce čtenáři stručně představí pojem OLAP. Ukáže jeho typické operace, možnosti transformace multidimenzionální reprezentace dat OLAPu do formy vhodné pro tabulkové procesory.
Dále popisuje některé známé možnosti exportu dat z portálů k uživatelům. Zkoumá dva zajímavé textové formáty SYLK a XML, které umožňují v textové podobě přenést informace pro použití v tabulkovém procesoru. Cílem je navrhnout a realizovat způsob, jak lze data z portálu exportovat přímo do aplikace MS Excel nebo i do jiných tabulkových procesorů právě pomocí těchto formátů.

Diplomová práce nenavazuje na žádný ročníkový ani semestrální projekt.

Práce je členěna do několika kapitol, jejichž stručný obsah je uveden níže:

Kapitola 2. OLAP
Vysvětluje pojem OLAP, ukazuje možnosti jeho použití, základní operace a využívaný multidimenzionální datový model. Rovněž poukazuje na srovnání tohoto přístupu a použití „běžných“ transakčních databázových systémů. V závěru zkoumá některé možné výstupy z OLAPu, které lze následně využít jako případné výchozí body dalšího zaměření naší práce.

Kapitola 3. Přenos dat z portálových IS
Probírá možnosti přenosu dat z portálu k uživatelům. Popisuje některé často používané formáty a přístupy, srovnává výhody a nevýhody jejich použití. Stručně nastíní základní informace o formátech SYLK a XML, na které je naše práce detailněji zaměřena a budou tedy podrobněji zpracovány v následujících kapitolách.

Kapitola 4. Formát SYLK
Detailněji seznámí čtenáře s formátem SYLK, který je vhodný pro přenos informací do aplikace

MS Excel. Postupně vysvětluje základy tohoto formátu, rozebírá detailněji některé důležité části a upozorňuje na možné problémy, které se mohou vyskytnou při tvorbě dokumentů.
Jako součást této kapitoly je také uvedena ukázka kompletního dokumentu, který byl vytvořen jako výstup přímo z aplikace Excel. Na této ukázce je vysvětlen význam jednotlivých částí, pro snazší pochopení tohoto formátu.
Kapitola volí spíše obecnější přístup k výkladu formátu a specificky se zaměřuje spíše na některé možné záludnosti, na které je třeba dát si větší pozor. Podrobný popis všech možností formátu je uveden v příloze A.

Kapitola 5. Formát XML
Obdobným způsobem jako kapitola 4 rozebírá formát XML pro MS Excel 2003. Protože je tento formát mnohem propracovanější, je tato kapitola také delší. Uvedená konkrétní ukázka spolu s vysvětlením je také obsáhlejší oproti ukázce souboru SYLK, ale jak je možné z obsahu kapitol vyčíst, nárůst délky přináší své ovoce v podobě přesného zachování zvolené předlohy.

Kapitola 6. Implementace
Tato kapitola shrnuje informace o možnostech realizace exportu informací z portálu do tabulkového procesoru. Vysvětluje základní kroky, které byly voleny před začátkem vývoje aplikace a vedli k rozhodnutí použít právě jazyka PHP.
Výsledkem implementace jsou knihovny určené pro tvorbu formátů z předchozích kapitol. V této kapitole je tedy uveden postupný návrh jednotlivých knihoven. Dále ukázka výstupu knihovny pro zvolené zadání, které je srovnáno s výstupy z tabulkového procesoru. Výhody, nevýhody a výsledné dokumenty tvořené pomocí knihoven jsou srovnány jak vůči tabulkovému procesoru, tak vůči sobě navzájem.
Kapitola se také zmiňuje o tabulkovém procesoru OpenOffice.org Calc a poukazuje na možnosti využití vytvořených knihoven právě v tomto procesoru. Jelikož je tato aplikace poněkud odlišná od primárního MS Excelu, není možné zaručit stoprocentní přenositelnost mezi těmito aplikacemi. Při znalosti uvedených odlišností je však z velké části přenositelnost možná.

2. OLAP

OLAP je zkratkou pro *Online Analytical Processing*, tedy slouží ke zpracování a analýze velkého objemu dat uložených v databázích, případně v datových skladištích *(data warehouse)* nebo může spolupracovat s datovými trhy *(datamart)*. Důležitým požadavkem je zpracování těchto dat v reálném čase, tj. s co nejkratší odezvou. Výstupem analýz mohou být různé souhrny a reporty, které mají vhodnou formu pro prezentaci. Ovšem vygenerování takových výsledků může trvat i několik hodin v závislosti na množství zkoumaných dat. Zpravidla se analýza provádí v noci, aby výsledky byly přichystány k použití v běžné pracovní době.

2.1. OLTP, OLAP a jeho datový model

Velmi rozšířené je využití transakčních databázových systémů OTLP *(Online Transaction Processing)*, neboť oblast použití je téměř neomezená. Tyto systémy jsou využívány pro práci s často modifikovanými záznamy (ukládání, mazání, úpravy záznamů v databázích) v nejrůznějších aplikacích. V dnešní době je možné se s nimi setkat snad ve všech webovských aplikacích, kdy jsou využity *relační databáze* např. pro ukládání informací o zboží či zákaznících internetových obchodů.

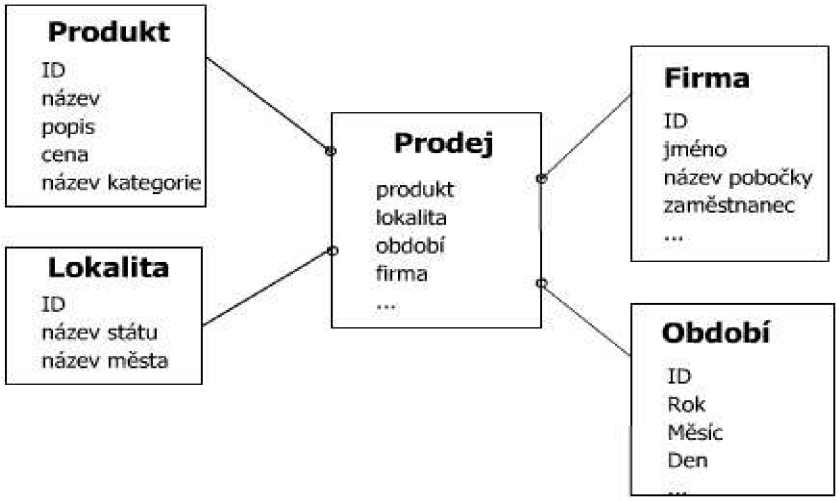
V relačních databázích jsou údaje uloženy v databázových tabulkách. Ty jsou chápány jako dvourozměrné tabulky, skládající se ze sloupců a řádků. Hodnoty jsou pak určeny průsečíky sloupců a řádků. Základy teorie relačních databází položil Dr. E. F. Codd, který byl rovněž i průkopníkem teorie analytických databází.

Databáze tedy slouží jako úložiště dat, které jsou nezávislé na aplikaci pracující s uloženými daty. Data jsou ukládána jako záznamy po řádcích do tabulky, v níž sloupce určují význam a typ uložených dat. Data jsou spolu provázána pomocí *primárních* a *cizích klíčů*.

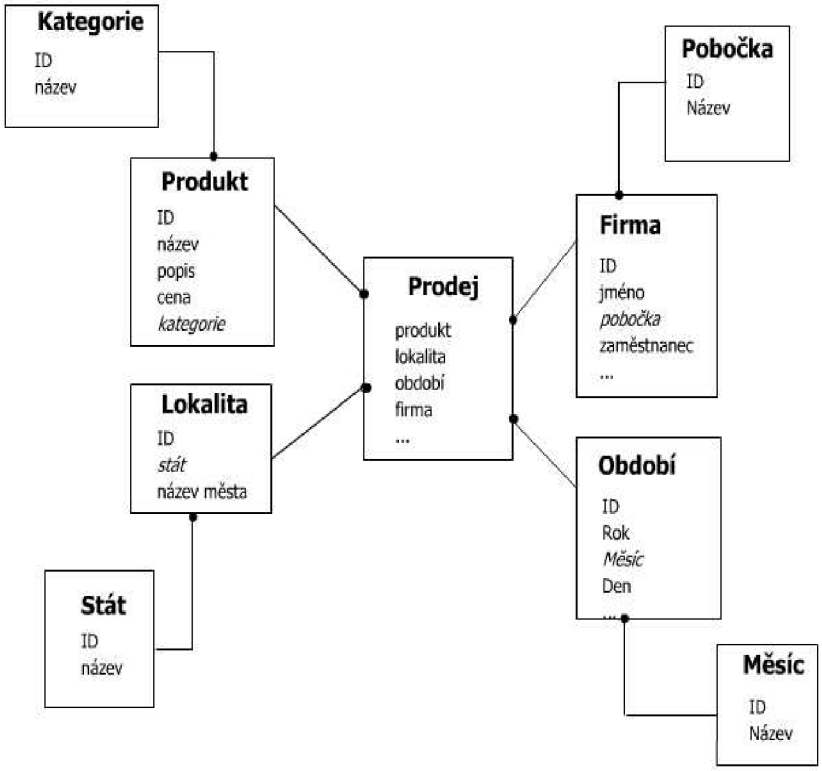
Struktura takových databází by měla vyhovovat pravidlům *normálních forem*. Správně navržená struktura databázových tabulek by měla eliminovat redundanci údajů a vyhovovat alespoň 3NF *(Třetí Normální Forma)*.

Nad takovými databázemi samozřejmě lze vybudovat analytické aplikace, ale už samotná struktura dodržující normální formy je důvodem obtížného sledování závislostí mezi daty. Bude zapotřebí složitějších SQL dotazů pro samotné nalezení a následné propojení souvisejících dat, nad kterými se teprve poté dají vykonávat operace pro požadovanou analýzu informací. To všechno znamená zvýšenou časovou složitost a hlavně zátěž databázového serveru, který je primárně určen k tomu, aby spravoval aktuální příchozí požadavky s co nejkratší odezvou.

Oproti tomu analytické databáze OLAP používají převážně nenormalizované tabulky. Jak bylo zmíněno, kladou důraz na zpracování velkého množství informací v co nejkratším možném čase, proto záznamy často obsahující redundanci dat. Tím je možné získat více detailnějších informací přímo, bez nutnosti využívat další pod dotazy. Tabulky OLAPu lze rozdělit na *tabulky faktů* a *tabulky dimenzí*. Podle druhu propojení těchto tabulek se hovoří o použitém schématu *hvězdicovém* či schématu *sněhové vločky.*



Obr 2.1. Ukázka hvězdicovitého schématu.

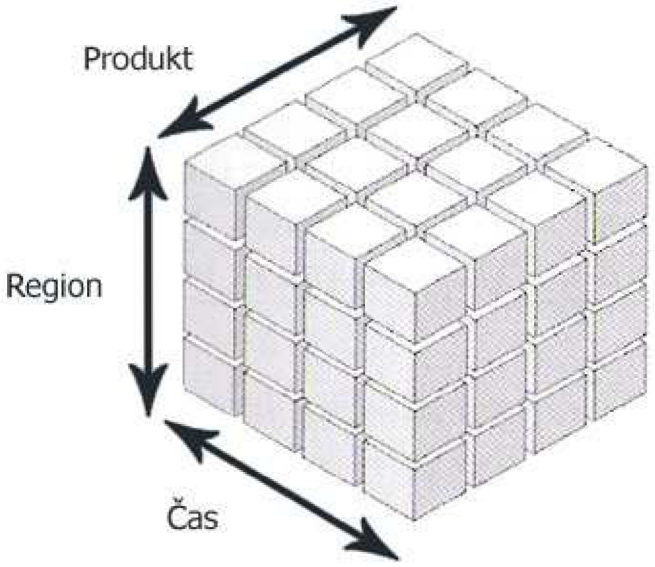


Obr 2.2. Ukázka schématu sněhové vločky.

Většina údajů je uchována v dvourozměrných relačních tabulkách, ale objevuje se zde nový pojem **multidimenzionální datová struktura – krychle**. Krychle je výsledkem různých agregací a analýzy údajů. Pro ilustraci se používá model klasické trojrozměrné krychle, ale dimenze mohou nabývat mnohem větších čísel. Servery podporující OLAP analýzu běžně podporují desítky

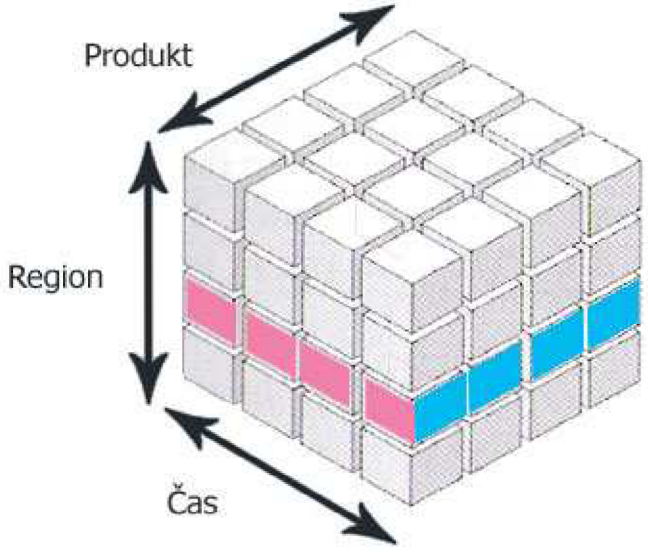
dimenzí.

Příkladem si můžeme uvést krychli s dimenzemi *čas, region, produkt* reprezentující údaje o prodeji různých výrobků nějaké firmy. Konkrétní údaje se nachází v průnicích jednotlivých dimenzí. Dimenze mohou být navíc strukturované. Např. časová dimenze může být dělena na jednotlivé roky, měsíce, týdny a dny. Tím máme možnost nastavit si detail pohledu na informace o které máme momentálně zájem.



Obr. 2.3. Model krychle

Pokud bychom chtěli analyzovat údaje o prodeji v regionech za danou dobu, pouze vybereme požadované dimenze:



Obr. 2.4. Výběr konkrétních dimenzí v krychli.

Oba modely mají samozřejmě své výhody a nevýhody a je nutné správně vybrat použitý model, podle charakteru aplikace.

Relační model:

- **léta zažitý model,**
- **vytvořeno velké množství aplikací a nástrojů pro práci s tímto modelem,**
- **použitelnost v transakčních databázích a datových skladech,**

Nevýhody:

- neobsahuje komplexní analytické nástroje,
- omezené množství údajů zpracovatelných v „rozumném čase“.

Multidimenzionální model:

- **rychlý komplexní přístup k velkému množství informací**
- **přístup k multidimenzionálním i relačním datovým strukturám**
- **možnost provádět komplexní analýzy**
- **silné schopnosti pro modelování a prognózy**

Nevýhody:

- problémy při změně dimenzí, bez přizpůsobení časové dimenzi
- vyšší nároky na kapacitu úložiště

2.2. Operace OLAPu

Mezi hlavní operace OLAPu patří agregace, tj. sumarizace informací ze záznamů či z více tabulek na základě vybraných dimenzí a úrovně detailu. Pro změnu úrovně detailu slouží následující dvě operace. Jejich funkčnost ukážeme na příkladu časové dimenze. Předpokládejme, že zkoumáme momentálně prodané produkty za jednotlivá čtvrtletí.

Roll-up

Vyrolování, zvýšení úrovně agregace a snížení detailu. Na časové ose se projeví například tím, že výpis prodaných produktů bude shrnut za jednotlivé roky. Přesnější dělení na čtvrtletí odpadá.

Drill-down

Opačná operace, snížení úrovně agregace a zvýšení detailu, zavrtání se do dat. Nyní bychom naopak viděli rozpis prodeje např. za jednotlivé měsíce či týdny každého roku. Tedy podrobnější výpis.

Slice & dice

Tyto operace provádí řezy krychle a výběr určitých částí řezu. Nastavují vhodnou (požadovanou) projekci neboli viditelnost výsledků. Vybírají se pouze relevantní části, které jsou promítnuty do výsledku, ostatní zůstanou skryty.

Pivoting

Natočení krychle, nastavení vhodného pohledu na data. Přesun dimenzí, případně přeskládání hierarchie.

2.3. Výstupy OLAPu

Jelikož OLAP pracuje s multidimenzionálním pohledem na data, je potřeba požadovaná data vhodně transformovat pro zobrazení na výstupních zařízeních. Výstupem mohou být různé reporty a souhrny podle aplikovaných operací. Znamená to tedy redukci tří či více dimenzí do dvou rozměrů, které lze snadno zobrazit na dnešních zařízeních. Dvourozměrná interpretace je také snadněji pochopitelná pro člověka. Nejčastější možností tedy jsou *tabulky či grafy.*
Grafy mohou zachycovat i multidimenzionální pohled (3D grafy), přesto se však častěji použije transformovaný 2D pohled na původní multidimenzionální prostor.
Důležitější pro naši práci budou *tabulky.* Tyto lze rozdělit na relační a kontingenční tabulky.

Relační tabulky

Jsou klasické dvourozměrné tabulky, kde řádky tvoří dimenze, sloupce hodnoty. Nad těmito tabulkami lze provádět základní OLAP operace jako

- pivoting – záměna sloupců
- roll-up, drill-down – zakrýváním či zobrazením klíčových sloupců se mění agregace
- slice & dice – zviditelnění či skrytí libovolného sloupce

3	Rok	Měsíc	Oblast	Prodejce	Diskety	Jednotky	Cena	Celkem
4	1999	Prosinec	Olomouc	Stejskal	3M	1367	30	41 010
5	1999	Prosinec	Olomouc	Stejskal	Polaroid	1257	25	31 425
6	1999	Prosinec	Olomouc	Stejskal	Kodak	1346	54	72 684
7	1999	Prosinec	Mohelnice	Stejskal	3M	614	30	18 420
8	1999	Prosinec	Mohelnice	Stejskal	Polaroid	1014	25	25 350
9	1999	Prosinec	Mohelnice	Stejskal	Kodak	1109	54	59 886
10	1999	Prosinec	Litovel	Stejskal	3M	1136	30	34 080
11	1999	Prosinec	Litovel	Stejskal	Polaroid	884	25	22 100
12	1999	Prosinec	Litovel	Stejskal	Kodak	784	54	42 336
13	1999	Prosinec	Olomouc	Hajný	3M	577	30	17 310
14	1999	Prosinec	Olomouc	Hajný	Polaroid	1109	25	27 725
15	1999	Prosinec	Olomouc	Hajný	Kodak	1025	54	55 350
16	1999	Prosinec	Mohelnice	Hajný	3M	854	30	25 620
17	1999	Prosinec	Mohelnice	Hajný	Polaroid	915	25	22 875
18	1999	Prosinec	Mohelnice	Hajný	Kodak	1386	54	74 844
19	1999	Prosinec	Litovel	Hajný	3M	1304	30	39 120
20	1999	Prosinec	Litovel	Hajný	Polaroid	1067	25	26 675
21	1999	Prosinec	Litovel	Hajný	Kodak	588	54	31 752
22	2000	Leden	Olomouc	Stejskal	3M	534	30	16 020
23	2000	Leden	Olomouc	Stejskal	Polaroid	776	25	19 400

Obr. 2.5. Ukázka relační tabulky s daty v aplikaci MS Excel.

Kontingenční tabulka

Dimenze tvoří řádky i sloupce, zachycuje navíc hierarchii. Samotné hodnoty tvoří průsečíky dimenzí. Opět lze provádět základní operace:

- pivoting – záměna dimenzí (řádkových či sloupcových)
- roll-up, drill-down – zakrýváním hierarchie dimenzí se mění úroveň detailu
- slice & dice – zviditelnění či skrytí sloupců

Obě tyto tabulky lze zobrazit aplikací MS Excel, která je momentálně zřejmě nejvíce využívaný prostředek pro práci s výsledky OLAPu.

4					
5	Rok	(Vše)			
6					
7	součet z Jednotky	Oblast			
8	Diskety	Litovel	Mohelnice	Olomouc	Celkový součet
9	3M	10,66%	8,65%	11,35%	30,66%
10	Polaroid	10,32%	11,17%	12,25%	33,74%
11	Kodak	10,10%	12,67%	12,84%	35,60%
12	Celkový součet	31,07%	32,49%	36,44%	100,00%

Obr. 2.6. Ukázka možné kontingenční tabulky v aplikaci MS Excel.

3	Rok	(Vše)				
4	Diskety	3M				
5						
6	součet z Celkem		Měsíc			
7	Prodejce	Oblast	Leden	Únor	Prosinec	Celkový součet
8	Novák	Litovel	10 020 Kč	6 840 Kč	8 460 Kč	25 320 Kč
9		Mohelnice	3 525 Kč	14 175 Kč	5 040 Kč	22 740 Kč
10		Olomouc	12 575 Kč	13 725 Kč	14 160 Kč	40 460 Kč
11	Celkem z Novák		26 120 Kč	34 740 Kč	27 660 Kč	88 520 Kč
12	Zeman	Litovel	10 260 Kč	3 450 Kč	3 725 Kč	17 435 Kč
13		Mohelnice	4 880 Kč	6 500 Kč	6 075 Kč	17 455 Kč
14		Olomouc	9 860 Kč	7 180 Kč	4 950 Kč	21 990 Kč
15	Celkem z Zeman		25 000 Kč	17 130 Kč	14 750 Kč	56 880 Kč
16	Celkový součet		51 120 Kč	51 870 Kč	42 410 Kč	145 400 Kč
17						

Obr. 2.7. Ukázka možné kontingenční tabulky v aplikaci MS Excel.

3. Přenos dat z portálových IS

V informačních systémech, stejně jako v dalších webovských aplikacích (např. internetové obchody) je nutné shromažďovat rozličné údaje. Uchovávání dat je obvykle realizováno použitím některého z databázových systémů. Většinou je použito relačních databází jako MySQL či PostgreSql. Větší firmy mohou využívat systém Oracle, rozsáhlá datová skladiště s využitím OLAP. Potřebné informace je nutné zobrazit koncovým uživatelům, kteří pracují s prohlížeči stránek. Proto je potřeba zobrazit výsledky vhodným a přehledným způsobem nezávisle na použitém databázovém systému. Toto je ve většině případů úkolem přímo dané portálové aplikace, která musí umět data získat z jejich úložiště a vhodně prezentovat.

Lze předpokládat, že se bude vycházet z „klasické“ dvourozměrné tabulky, kterou lze snadno zobrazit jako obsah WWW stránky či ji lze snadno vytisknout. Pokud data nemáme uloženy v relačních dvourozměrných tabulkách, lze je do této podoby získat na straně serveru – příprava výsledků analýz OLAPem či vlastní předzpracování dat. Jakým způsobem takové tabulky získáme není pro tuto práci důležité.

Stránky mohou být statické či v dnešní době převážně dynamické. Dynamické pak mohou umožňovat provádění například OLAP operací a podle potřeby měnit agregaci dat. Všechny tyto možnosti závisejí na charakteru a úrovni propracování aplikace.

Celá taková aplikace obvykle běží na serveru a uživatel s ní komunikuje pouze pomocí prohlížeče. Nastává však otázka, jak si uživatel může výsledná data uložit, aby s nimi mohl v *rozumné formě* pracovat aniž by byl on-line připojen k aplikaci?

Uložit výsledky není většinou problém. Obsah WWW stránky lze jednoduše uložit přímo prohlížečem a stejný *obraz* stránky pak pouštět přímo z disku počítače uživatele. Ovšem takový výsledek je vhodný pouze k prezentaci, tzn. že si data můžeme prohlížet v přehledné formě, ale nejsou připravena pro jakoukoliv úpravu či manipulaci. Záleží tedy pouze na tom, co myslíme *rozumnou formou*.

Tato práce je zaměřena na export dat z portálu do tabulkových procesorů, konkrétně do aplikace MS Excel. Ta umožňuje snadnou a efektivní práci s tabulkami dat, tvorbu grafů, práci s kontingenčními tabulkami a mnohé další. Ukážeme si tedy způsob jak nabídnout uživatelům možnost uložení dat právě pro tuto aplikaci. Rozumná forma dat v této práci znamená jejich uložení v tabulkovém procesoru MS Excel.

3.1. Možnosti exportu dat pro MS Excel

3.1.1. CSV

Formát CSV *(Comma Separated Values)* je prostý textový formát. Skládá se ze záznamů oddělených koncem řádku. Záznamy jsou složeny z jednotlivých položek (vlastní hodnoty). Ty jsou od sebe odděleny předem určeným znakem – oddělovačem. Nemusí se jednak o znak čárky jak napovídá název, nýbrž velmi často se používá znak středníku nebo jakýkoliv jiný oddělovač. Vybírá se takový, který se neobjevuje ve vlastních hodnotách, případně se v nich vyskytuje zřídka.

Tento formát umožňuje přenést **pouze hodnoty dat** bez možnosti přenášet strukturu dokumentu či typu dat. Excel umí taková data načíst do jednotlivých buněk mřížky listu a vytvořit tabulku se správně rozmístěnými daty. Nelze tak ale přenést barvy, ohraničení jednotlivých buněk

apod.

	A	B	C	D	E	F	G	H
1	**Rok**	**Měsíc**	**Oblast**	**Prodejce**	**Diskety**	**Jednotky**	**Cena**	**Celkem**
2	1999	Prosinec	Olomouc	Stejskal	3M	1367	30	41 010
3	1999	Prosinec	Olomouc	Stejskal	Polaroid	1257	25	31 425
4	1999	Prosinec	Olomouc	Stejskal	Kodak	1346	54	72 684
5	1999	Prosinec	Mohelnice	Stejskal	3M	614	30	18 420
6	1999	Prosinec	Mohelnice	Stejskal	Polaroid	1014	25	25 350
7	1999	Prosinec	Mohelnice	Stejskal	Kodak	1109	54	59 886
8	1999	Prosinec	Litovel	Stejskal	3M	1136	30	34 080
9	1999	Prosinec	Litovel	Stejskal	Polaroid	884	25	22 100
10	1999	Prosinec	Litovel	Stejskal	Kodak	784	54	42 336

Obr. 3.1. Tabulka v Excelu před převodem do CSV

```
Rok;Měsíc;Oblast;Prodejce;Diskety;Jednotky;Cena;Celkem
1999;Prosinec;Olomouc;Stejskal;3M;1367;30;41 010
1999;Prosinec;Olomouc;Stejskal;Polaroid;1257;25;31 425
1999;Prosinec;Olomouc;Stejskal;Kodak;1346;54;72 684
1999;Prosinec;Mohelnice;Stejskal;3M;614;30;18 420
1999;Prosinec;Mohelnice;Stejskal;Polaroid;1014;25;25 350
1999;Prosinec;Mohelnice;Stejskal;Kodak;1109;54;59 886
1999;Prosinec;Litovel;Stejskal;3M;1136;30;34 080
1999;Prosinec;Litovel;Stejskal;Polaroid;884;25;22 100
1999;Prosinec;Litovel;Stejskal;Kodak;784;54;42 336
```

Obr. 3.2. Výsledný CSV soubor

	A	B	C	D	E	F	G	H
1	Rok	Měsíc	Oblast	Prodejce	Diskety	Jednotky	Cena	Celkem
2	1999	Prosinec	Olomouc	Stejskal	3M	1367	30	41 010
3	1999	Prosinec	Olomouc	Stejskal	Polaroid	1257	25	31 425
4	1999	Prosinec	Olomouc	Stejskal	Kodak	1346	54	72 684
5	1999	Prosinec	Mohelnice	Stejskal	3M	614	30	18 420
6	1999	Prosinec	Mohelnice	Stejskal	Polaroid	1014	25	25 350
7	1999	Prosinec	Mohelnice	Stejskal	Kodak	1109	54	59 886
8	1999	Prosinec	Litovel	Stejskal	3M	1136	30	34 080
9	1999	Prosinec	Litovel	Stejskal	Polaroid	884	25	22 100
10	1999	Prosinec	Litovel	Stejskal	Kodak	784	54	42 336

Obr. 3.3. Soubor CSV načtený zpět do Excelu

Obdobnou možností je text oddělený tabulátory.

3.1.2. HTML

Excel umožňuje importovat HTML stránku. Výhodou je možnost přenést **data i strukturu**. Je tedy

možné připravit uživatelům tabulku barevně, s použitím různých ohraničení buněk, excelovských vzorců a dalších vymožeností.

Formátování takového HTML souboru je však mírně omezeno. Z experimentálních poznatků vyplývá, že formátování jednotlivých buněk lze nastavit pouze přímo. Např. použitím formátovacích značek jako <*font color=“red“*> což se v dnešní době považuje za překonané. Při tvorbě stránek se upřednostňuje oddělení obsahu od formy. K tomuto účelu dobře slouží CSS neboli *tabulky kaskádových stylů*. K HTML značkám lze tedy připsat definici stylu pomocí atributu *style*. Pokud bychom použili definici pravidel (např. definice tříd) CSS na začátku souboru, pak Excel použije **pouze jedno pravidlo.** Nelze aplikovat více pravidel jako u samotné HTML stránky.

Pokud chceme zachytit snadno strukturu tabulky Excelu, pak je vhodné použít tabulku HTML:

<*table*>
<*tr*><*td*>*první buňka prvního řádku*</*td*><*td*>*druhá buňka*</*td*></*tr*>
<*tr*><*td*>*první buňka druhého řádku*</*td*><*td*>*druhá buňka*</*td*></*tr*>
</*table*>

Kde elementy <tr> označují řádky tabulky a elementy <td> buňky tabulky. Tyto se ztotožní se strukturou tabulky v pracovním listu Excelu. Pak snadno můžeme zapisovat obsah buněk na konkrétních souřadnicích, slučovat buňky apod.

Lze použít samozřejmě i jinou strukturu HTML kódu použitím odstavců či bloků <*div*> na které pak aplikujeme CSS předpisy, nicméně správně nastavit pozice buněk ve výsledném sešitu Excelu je pak obtížnější.

3.1.3. SYLK

Tento formát umožňuje přenést též **data i strukturu a navíc informace o typu dat.** Nevýhodou jsou omezené možnosti formátování obsahu. Nedosahuje takových možností jako formát HTML. Je omezen např. v nastavování barvy pozadí buněk, typů ohraničení buněk atd. Formát pracuje přímo s obsahem jednotlivých buněk, takže na rozdíl od HTML není potřeba vytvářet umělé tabulkové struktury pro přesné rozmístění dat. Tento formát bude podrobněji popsán v kapitole 4.

3.1.4. XML (Excel 2003)

Tento formát je asi nejvhodnějším ze všech uvedených. Umožňuje stejně jako formát sylk přenos **obsahu, formy a informací o typu dat**. Navíc **není omezen** v možnostech f**ormátování**. Pracovní sešit Excelu lze uložit ve formátu XML (možnost *uložit jako tabulku XML*) téměř shodně jako při použití binárního výchozího formátu Excelu (*xls*). Nelze takto přenášet grafy a další objekty (to platí i pro předchozí formáty).

3.1.5. Shrnutí

Excel podporuje poměrně velké množství nejrůznějších formátů. Výše uvedené jsou jen malým zlomkem ze všech možností nabídnutých při volbě uložení souboru. Protože je naším cílem generovat snadno a rychle soubory použitelné v této aplikaci, jsou pro nás vhodné textové formáty. Z nich jsou nejlepšími kandidáty výše uvedené formáty. Protože formáty HTML a CSV jsou známé, jednoduché a proto často používané nebudeme se jimi nadále zabývat. Data je možné také exportovat ve formě prostého textu (soubor typu .txt), ale to by nám nepřineslo žádné výhody

využití tabulkového procesoru. Proto se tato práce se dále bude zabývat jen formáty SYLK a XML.

3.2. Způsob exportu dat z portálů

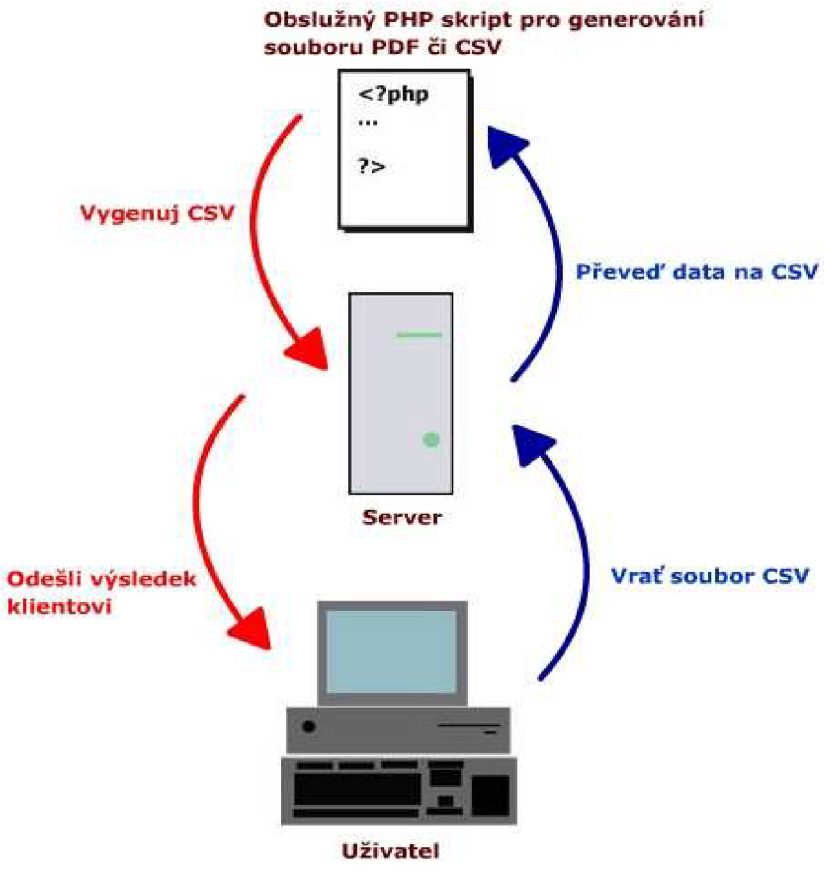
Vždy záleží na tom, jakým způsobem je webová aplikace vytvořena. Lze předpokládat, že je použit některý z vhodných skriptovacích jazyků pro tvorbu dynamických WWW stránek, které pracují s daty v databázi v závislosti na požadavcích uživatele systému. Jak ale uživateli předat data pro uložení?

Nejčastějším řešením je nabídnout uživateli odkazy, směřující na konkrétní skript, který má za úkol zjistit jaká data uživatel požaduje a na základě těchto dat vytvořit vybraný typ souboru. Tento soubor buď nabídnout ke stažení a uložení na disk uživatele, nebo přímo nabídnout možnost otevření v aplikaci (pokud ji uživatel má k dispozici).

Často nabízené formy exportu jsou:

- Exportovat do Excelu, za čímž se převážně skrývá vygenerování formátu CSV.
- Exportovat do PDF, které sice umožní uložit strukturu dat v přesně daném formátování, ale data jsou vhodná pouze pro prohlížení či tisk. S obsahem PDF nelze manipulovat (kromě případného vykopírování obsahu).

Schéma zpracování požadavků může vypadat například následovně:



Obr. 3.4. Schéma generování požadovaného typu souboru

Účelem této práce je vytvořit vhodný prostředek pro umožnění exportu údajů pro aplikaci MS Excel v použitelné formě. Podle obrázku 3.4. bude naším úkolem implementovat právě obslužný skript, který podle požadavků serveru vygeneruje soubor v zadaném formátu. Cílem práce je zajistit vytváření souborů ve formátech SYLK a XML.

Pro implementaci byl zvolen jazyk PHP 5. Skriptovací jazyk PHP je hodně rozšířený a je použit v mnoha internetových aplikacích. Jeho verze 5 již obsahuje dobře propracovaný objektový model, který pro aplikaci rovněž využijeme.

Pro ukázku možné spolupráce s databází byla v jednom z příkladů použita spolupráce volně dostupným databázovým systémem MySQL. Nicméně výsledná aplikace není závislá na databázovém systému a není potřebné databázi vůbec využívat – opět záleží na charakteru aplikace.

Více o jednotlivých formátech bude uvedeno v následujících kapitolách. Podrobnosti o implementaci dále v kapitole šesté. Ukázkové příklady a podrobnější dokumentace o vytvořených skriptech je uvedena na přiloženém médiu.

4. Formát SYLK

Formát SYLK umožňuje přenášet data mezi aplikacemi ve formě prostého textového souboru za použití ASCII kódování. Takový soubor by měl být správně interpretován jakoukoliv aplikací, která tento formát podporuje.

V kapitole 4.1. následuje stručný popis tohoto formátu a některé důležité poznatky či komplikace, které se objevily při studování podrobností tohoto formátu. Detailnější popis, který je sestaven na základě všech nalezených či experimentálně zjištěných informací je uveden v příloze A. Tento přehled však v žádném případě není úplný, protože je formát neustále rozšiřován zejména podle potřeb společnosti Microsoft.

Samotný formát je poměrně jednoduchý a snadno generovatelný, nedovoluje ale plné využití formátovacích možností tabulkového procesoru.

4.1. Popis formátu SYLK

Soubor ve formátu SYLK je prostý textový soubor, který se skládá ze záznamů popisujících obsah a strukturu jednoho listu aplikace Excel. Záznamy jsou od sebe odděleny koncem řádku. Prázdné řádky jsou ignorovány, což umožňuje naformátovat záznamy do spolu souvisejících bloků a tím zvýšit přehlednost zdrojového kódu. Samotný MS Excel žádné prázdné řádky nepoužívá a generuje veškeré záznamy souvisle. Samotné záznamy se dále skládají z několika částí, které budou popsány v následující podkapitole.

4.1.1. Záznamy

Záznamy lze rozdělit na tři části. Strukturu záznamu pak lze popsat následovně:

<RTD><FTD><Fields>

- **<RTD> =** *Record Type Descriptor*, určuje typ záznamu. Skládá se z jednoho či dvou písmen. Z konvence se pro větší přehlednost používají velká písmena.
- **<FTD>** = *Field Type Descriptor*, určuje vlastnost v rámci udaného typu záznamu, která bude nastavována. Možnosti závisí na typu záznamu, a budou uvedeny v přehledu dále. FDT se vždy skládá z posloupnosti: *;Písmenko* např. ;X, ;F apod. Vždy má délku 2 znaky.
- **<Fields>** = jsou vlastní hodnoty vybraného FTD. Následuje bezprostředně za uvedeným FTD, není potřeba uvádět žádný oddělovač. Délka je proměnlivá, závisí na nastavovaných hodnotách. Pokud hodnotou bude řetězec, *musí být uzavřen v uvozovkách!*

Soubor se pak skládá z posloupnosti takovýchto záznamů oddělených koncem řádku. Což bychom obecně mohli zapsat:

```
<RTD><FTD><Fields>
<RTD><FTD><Fields>
<RTD><FTD><Fields>
...
```

Konkrétní záznamy výsledného souboru mohou vypadat například následovně:

```
ID;PWXL;N;E
P;PGeneral
P;P0
P;P0.00
F;P0;FG0C;SDLRTBSM8;X4
C;K"B"
…
```

Nyní si ukážeme jednotlivé části prvního řádku příkladu. Obsahem řádku je záznam: *ID;PWXL;N;E.*
Podle popisu částí záznamu na začátku této kapitoly můžeme určit jednotlivé části záznamu následovně:

- ID – je <RTD>, neboli typ záznamu
- ;P – <FTD>vlastnost záznamu ID
- WXL – <Fields>, aneb hodnota vlastnosti ;P
- ;N – vlastnost záznamu ID
- ;E – vlastnost záznamu ID
- jak vidíme vlastnosti ;E a ;N nemají žádné hodnoty; část Fields je nepovinná, takže je to zcela v pořádku.

Možné hodnoty a jejich význam v jednotlivých částech záznamů jsou popsány v příloze A. Pro pochopení tohoto formátu a porozumění následujícím ukázkám souboru doporučuji zájemcům, aby po přečtení této stručné úvodní kapitolky do problematiky struktury formátu, nahlédli právě do přílohy A, kde se mohou seznámit s jednotlivými možnostmi podrobněji. Jelikož je SYLK poměrně jednoduchý formát, není tato příloha tolik rozsáhlá a náročná. Přesto pokud nechcete procházet všechny detaily doporučuji projít alespoň ty nejdůležitější typy záznamů uvedené dále.

V přehledu možných kombinací hodnot RTD a FTD se objevuje označení **(diff)**. Toto označení bylo převzato z materiálů dokumentujících formát SYLK. Označení znamená, že není nutné uvádět všechny možnosti, protože poslední hodnota bude automaticky určena z předchozích záznamů.

Např. pro souřadnice buněk X, Y platí, že pokud nastavíme novou hodnotu pouze pro souřadnici X (horizontální souřadnice, tedy posun na sloupec číslo X), předpokládá se, že pracujeme ve stejném řádku a tedy hodnota Y by měla být automaticky doplněna na poslední, která byla použita.

Souřadnice v SYLKu mají počátek **vždy v 1:1** a vždy využívají číselné označení buněk, i když má Excel nastaveny souřadnice na typ A1, C4, atd. jak ukazuje obrázek 4.1. Obrázek zachycuje mapování konkrétních buněk listu Excelu formátem SYLK. Další buňky se mapují obdobně. Posunem po horizontální ose směrem zleva doprava roste první, tedy X-ová, souřadnice. Posunem po vertikální ose roste souřadnice druhá, tedy Y-ová, směrem shora dolů.

	A	B	C	D
1	1:1	2:1	3:1	...
2	1:2	2:2	3:2	...
3	1:3	2:3	3:3	...
4	...	...	...	...

Obr. 4.1. Reprezentace souřadnic Excelu ve formátu SYLK

Jak již bylo zmíněno, význam a možnosti kombinací hodnot jednotlivých částí záznamu jsou uvedeny v příloze A. Zde uvedu nejdůležitější typy záznamů, které doporučuji k nastudování, protože jsou základními stavebními kameny tvořeného souboru.

Nejdůležitější typy záznamů:

- ID – povinný první záznam v souboru SYLK
- E – povinný poslední záznam v souboru SYLK
- P – nastavení formátu čísla pro obsah buňky v listu Excelu
- F – nastavení vzhledu buňky
- C – definice vlastního obsahu buňky

Těchto pět typů záznamů nám postačí pro tvorbu převážné většiny souborů (pokud nebudeme potřebovat vyloženě nějaké speciální vlastnosti či nastavení buněk). Bohužel možných hodnot pro nastavení vlastností jednotlivých typů je již více, takže výsledné kombinace jsou rozsáhlejší, a zde nebudou uvedeny. Potěšující může být, že není bezpodmínečně nutné znát všechny z nich k tomu, abychom porozuměli obsahu SYLK souboru, či dokázali jednoduchý soubor vytvořit ručně. Při experimentech se víceméně opakovala pouze jistá podmnožina vlastností a měnily se pouze vlastní hodnoty buněk v částech <*Fields*>.

4.1.2. Struktura SYLK dokumentu

Již víme, že dokument se skládá z řádků na kterých jsou uvedeny záznamy, které udávají vzhled a obsah požadovaného listu. Je potřeba dodržet některá další pravidla pro pořadí jednotlivých záznamů.
Základní struktura dokumentu vypadá následovně:

```
ID...
požadované záznamy tvořící obsah výsledného listu
E
```

Každý SYLK dokument povinně začíná záznamem ID a končí záznamem E. Toto je jednoduchá, ale opravdu nezbytná podmínka, jinak nebude soubor Excelem načten a zahlásí chybu při otvírání.

Záznamy tvořící obsah listu by měly dodržet jistá pravidla a doporučení:

a) Pokud chceme použít číselný formát obsahu buněk, je nutné jej definovat před aplikací na buňku.
b) Pokud chceme použít nastavení vzhledu buňky (písmo, barva, atd.) je nutné jej definovat před aplikací na buňku.

c) Pokud definujeme více číselných formátů obsahu, definujeme je v souvislé posloupnosti; tj. všechny definice budou následovat bezprostředně za sebou.
d) Pokud definujeme více stylů vzhledu, opět je definujeme bezprostředně za sebou.
e) Odkazy na definice stylů a formátů musí být platné = definice musí existovat.
f) Při definici obsahu musí být vždy známy souřadnice buňky; je nutné uvést alespoň jednu souřadnici, což je ale možné pouze v případě, že druhou lze odvodit z předchozích záznamů. V praxi to znamená že první buňku řádku určíme oběma souřadnicemi a dále se v řádku posouváme pouze změnou jedné souřadnice.
g) Je dobré dodržovat jednotný styl zápisu souřadnic; Excel upřednostňuje uvádět souřadnice při nastavování vzhledu a formátu buňky (záznam F) a zadávání souřadnic v pořadí *;Y ;X*. Při zadávání obsahu buňky (záznam C) se využijí poslední definované souřadnice. Tento bod však není povinný, slouží jen jako doporučení.
h) Obsah buněk lze definovat v libovolném pořadí; je ale doporučeno vkládat obsah po řádcích či sloupcích především proto, aby šlo využít změn pouze v jedné souřadnici při zápisu obsahu buněk.

Body a) a b) vycházejí z toho, že aplikace definovaného stylu či formátu se děje prostřednictvím odkazu na tuto definici. Proto je samozřejmě nutné ji znát dříve, než bude použita.

Body c) a d) vychází ze způsobů tvorby odkazů na definice. Definice jsou číslovány od nuly (pozor na výjimky v číslování, kapitola 4.1.5). První definice má tedy přiřazeno číslo (index) 0, druhá 1, třetí 2 atd. To platí jak pro definice číselných formátů tak pro definice stylů. Aby bylo možné je snadno a jednoznačně určit, musí být tyto definice sepsány pohromadě. Získají tak vzestupně unikátní číslo a aplikace načítající formát SYLK je pak schopna ověřit zda požadovaná definice existuje a umožní její použití.

Bod e) vychází z předchozího – pokud číslo použitého formátu či stylu není známo, Excel hlásí chybu a nenačte vytvořený dokument. Zřejmě není definováno žádné standardní chování aplikace, které by řešilo problémy s neexistujícími definicemi. Tento postup je pochopitelný, protože předepisujeme aplikaci, že požadujeme např. nějaký specifický formát čísla v buňce, na který se odkážeme do definice. Pokud definice neexistuje, co je lepší – zobrazit případná data chybně nastavením nevhodného formátu nebo upozornit na chybu a vynutit si opravu? Přikláníme se k druhé možnosti, která je koneckonců implementována, že pokus o načtení takového souboru skončí s chybovým hlášením. Jinak by se chybná hodnota způsobená nevhodně doplněným formátem mohla snadno přehlédnout a uživatelům by se distribuovaly nesprávné dokumenty.

Body f) – h) se zabývají nastavováním souřadnic buněk. Je doporučeno postupovat po řádcích či sloupcích. Lze si vybrat podle okolností – např. při vkládání dat z databáze je jednodušší plnit jednotlivé sloupce, protože hodnoty jsou stejného datového typu a tudíž využívají stejný formát obsahu. Pak lze uvést na začátku první buňky sloupce obě souřadnice X i Y a v dalších buňkách nastavovat souřadnici Y. X-ová souřadnice se automaticky doplní jako poslední, která byla zadána.

Zmínka o pořadí souřadnic X a Y je uvedena na základě výsledků experimentů při generování souborů Excelem. Toto pořadí nemá vliv na konečný výsledek. Excel rovněž doplňuje souřadnice u záznamů F, které nastavují formátovací vlastnosti buněk. Zřejmě je to záměrně proto, aby bylo možné snadno nastavit požadovaný formát buňky, i když do buňky následně neuložíme žádný obsah. Není pak nutná kontrola, zda obsah existuje či nikoliv a podle toho se rozhodovat zda vygenerovat souřadnice do záznamu s formátem či obsahem.

Z předešlých poznámek vyplývá, že valná většina vytvořených dokumentů bude mít strukturu podobnou této:

```
ID...   = povinný první záznam
P;P...  =  definice formátu číselného obsahu buňky
P;P...
P;P...
P;F...  = definice stylů písma atd.
P;F...
P;F...

F...;Xx;Yy    = aplikace stylů a formátu buňky, souřadnice buňky
C...    = obsah buňky
F...
C...

E
```

4.1.3. Kódování souborů

SYLK předpokládá ASCII kódování, proto musí existovat způsob, jak jeho prostřednictvím uložit libovolné znaky - s diakritikou či další speciální znaky. Používají se k tomu únikové (*escape*) sekvence:

ISO sekvence:

Struktura zápisu speciálního znaku pomocí ISO sekvencí vypadá následovně:

- **<ESC>N** následuje označení akcentu a původní písmenko bez akcentu.
  - Pro zápis akcentovaných znaků. Například *í* by se zakódovalo jako *<ESC>N´i*
  - znaky akcentu leží v rozsahu hexadecimálního vyjádření znaku 40-4F
- **<ESC>N** následovaný speciálním znakem v rozsahu 21-3F a 50-7E pro neakcentované speciální ISO znaky.

Označení <ESC> zde reprezentuje znak po stisknutí klávesy Escape.
Je doporučeno využívat právě ISO sekvence, pokud je to možné.

ASCII sekvence:

Je možné využít v případech, že není možné použít ISO sekvenci. Struktura vypadá obdobně jako výše:

- **<Esc> 2M 3N** bude trojznakové kódování znaku, který měl původní kód v hexadecimálním vyjádření *MN*.

Pokud jako hodnotu potřebujeme uvést znak středníku, je potřeba jej zdvojit. Středník se totiž používá jako speciální znak pro oddělování jednotlivých FTD.

Pokud vkládáme jako hodnotu textový řetězec či logické hodnoty TRUE/FALSE je nutné je uzavřít do uvozovek. Pokud je hodnota chybové hlášení (typ Error) předchází jej znak mřížky.

4.1.4. Výsledný formát buněk, datové typy

Pro nastavení vzhledu obsahu v buňkách existuje několik možností podle definice formátu. Experimentování však ukázalo, že například definice stylu písma v záznamu ***F***, kde by se mělo dát nastavit např. tučné písmo, je Excelem zcela **ignorována**.

Největší a v podstatě jediný vliv má na výsledek použitý formát buňky, definovaný v záznamech *P* a dále definované písmo, opět v záznamech *P*.

Základní formátování se vztahuje pouze na číselné hodnoty, jejichž vzhled lze nastavit na celá čísla, reálná, měnu včetně nebo bez symbolů měny, datum, čas apod.

Reprezentace data a času

Pro reprezentaci data používá SYLK celočíselné vyjádření. To se získá jako počet uplynulých dnů od počáteční *epochy*. Epocha je definována pro Windows Excel jako 1. 1. 1900 a tento rok je považován za přestupný (což je chyba, neboť za přestupný rok se považuje rok dělitelný čtyřmi; pokud je dělitelný 100, musí být rovněž dělitelný 400, jinak přestupný není).

Aby norma pro Excel MAC odstranila tuto nesrovnalost, vychází z epochy 1. 1. 1904, který již opravdu přestupný je.

Proto **záleží na nastavení ID záznamu**, aby bylo datum správně interpretováno. Jinak může dojít k neočekávanému posunu data o 4 roky.

Nezáleží na tom, na jaké platformě Excel použijeme, je možné i na MS Windows pracovat s dokumentem SYLK pro MAC Excel, ale je nutné dopočítat správné vyjádření data.

Pokud chceme vyjádřit letopočet před začátkem epochy, je jedinou možností uložit jej jako obyčejný text.

Tento nepatrný detail, nastavení *ID;PXL* namísto *ID;PWXL* a následující nepochopitelné odchylky v zobrazovaných datech při zkoumání tohoto formátu, byl důvodem k opětovnému studování souvislostí a detailů SYLKu. Tato *chybka* nám však pomohla nalézt výše uvedené závislosti mezi datem a prvním záznamem souboru. Donutila nás důkladně zkoumat další internetové zdroje a hledat detaily implementace reprezentace data v tomto formátu. Nakonec se nám podařilo objevit vysvětlení, proč je použito dvou různých epoch od kterých se odvíjí výpočet data.

Čas

Je vyjádřen jako reálné číslo. Vypočítá se jako poměr uplynulé doby dne ku délce dle. Tedy požadovaný čas můžeme převést na počet vteřin od půlnoci a podělit počtem vteřin za celý den (24 * 60 * 60). S reprezentací času se nevyskytl žádný problém v souvislosti s vybraným ID záznamem.

Pokud se však těmto údajům přiřadí špatný formát obsahu buněk, bude se s nimi zacházet jako s obyčejným číslem! Proto je nutno dbát na správný výběr formátu pro datum a čas.

4.1.5. Experimentálně zjištěné problémy či nesrovnalosti

Nyní bychom již měli mít ucelený přehled o tom, jak vypadá korektně vytvořený dokument typu SYLK. Závěrem této kapitoly však chci ještě uvést všechny komplikace, se kterými jsme se setkali při zkoumání a experimentech s přenosem dokumentů do aplikace Excel 2003. Přestože, a podle všech uvedených materiálů oprávněně, jsme se domnívali, že soubor je vytvořen správně, samotný Excel měl v mnohých případech jiný názor.

Pokud se v souboru vyskytne nějaká nesrovnalost, Excel dokument neotevře vůbec. Při spouštění zahlásí chybu a ve většině případů se otevře prázdný list.

Druhým typem chyb je nesrovnalost v očekávaných a v zobrazených hodnotách. Takový dokument je samozřejmě načten bez problémů, ale hodnoty zobrazené Excelem jsou odlišné od těch, které požadujeme.

1. **Záznam ID**
   ID musí být prvním záznamem dokumentu. Jedná se o identifikaci typu souboru, aby mohl být správně interpretován. Pokud není v dokumentu obsažen, Excel hlásí chybu a soubor není otevřen. V tomto záznamu by mělo být přinejmenším nastaven typ aplikace, pro kterou je vytvořený SYLK soubor určen. Každá aplikace totiž může tento formát interpretovat odlišně a některé značky záznamů či vlastností mohou mít v různých aplikacích odlišný význam. Pro nás je důležité nastavit SYLK soubor pro aplikaci MS Excel. První řádek by tedy měl vždy obsahovat přinejmenším tento záznam:
   ID;PWXL
   Aplikace je určena ve vlastností ;P, nastavení WXL označuje MS Excel pro Windows. Pokud bychom vložili pouze hodnotu XL, bude obsah interpretován jako verze pro MAC Excel. Problémy se pak mohou projevit při práci s datem. Vysvětlení této problematiky je uvedeno v kapitole 4.1.4.
2. **Záznam E**
   E označuje konec (End) souboru. Pokud není přítomen, Excel opět hlásí chybu při otevírání souboru a nenačte požadovaný soubor. Nutnost výskytu tohoto záznamu není zcela jasná. Aplikace by měla zvládnout detekovat skutečný konec souboru, a případně záznam E simulovat. Excel jej však nejenže vyžaduje, ale tento záznam musí být ukončen koncem řádku. Pokud není tento záznam ukončen, Excel se chová jako kdyby záznam nebyl vůbec uveden. Podle definice jsou záznamy odděleny koncem řádku, takže jej zřejmě jinak nepovažuje za záznam.
3. **Kódování**
   Problematika kódování byla nastíněna v kapitole 4.1.3. Soubor pracuje nativně v ASCII kódování, takže všechny „speciální“ znaky musí být zakódovány pomocí ISO či ASCII sekvencí, které tyto zapíší pomocí ASCII znaků.
   Problém nastává se samotným generováním ASCII obsahu. Excel poněkud *záhadně* vytváří ISO sekvence a těžko se hledají souvislosti jakým způsobem se nahrazují akcenty. Například písmenko s čárkou je nahrazeno písmenkem B následované požadovaným písmenkem bez čárky. Ale háček je nahrazen např. písmenkem A ***ale následuje některé písmenko až za požadovaným písmenkem bez diakritiky.***
   Například zakódování písmenka „á“ vypadá následovně:
   #NBa = á
   Zde je zápis intuitivní.
   Naopak pokud má Excel zakódovat písmenko „č“ je výsledkem:
   #NAe = č
   Tedy namísto očekávaného #NAc se objevilo písmenko *e*.
   Pozn.: Znak '#' je reprezentuje znak Esc.

   Jelikož toto není vždy pravidlem (posun písmenek může být různý), nebylo prozatím možné odhadnout algoritmus, jakým jsou ISO sekvence v Excelu realizovány. Pokud bychom se drželi poznatků v kapitole 1.4.3. nebudou znaky zobrazeny správně dle našeho očekávání.

   Světlou stránkou však je, že Excel (2003) bez problémů načítá soubory i s použitou

diakritikou, pokud jsou uloženy v kódování podle verze aplikace. U naší zkušební české verze tedy v kódování Windows 1250. Texty jsou pak zobrazeny správně. Není tedy zcela nezbytné využívat původní požadavek ASCII kódování a používat sekvencí pro zakódování všech speciálních znaků. Otázkou zbývá problematika opravdu zvláštních znaků a symbolů. Správné chování této aplikace možná souvisí s českou verzí ASCII tabulky v operačním systému.

4. **Formát čísla, záznamy *P;P***
   Použití formátu čísla v buňce je asi jediný možný způsob, jak Excel přinutit zobrazit číslo opravdu tak jak potřebujeme. V příloze A se můžeme dočíst o nastavení typu obsahu přímo v určení formátu buňky, kde lze nastavit např. že se jedná o reprezentaci měny. Excel toto ovšem zcela ignoruje a pokud není styl nadefinován v záznamech typu *P;P...* a formát buňky na tento záznam neodkazuje, pak je použito výchozí nastavení čísel.
   Excel standardně generuje sadu asi čtyřiceti formátů, aniž by některé z nich vůbec použil. Pro propojení s buňkami se využívá číslo každého záznamu, přičemž se začíná číslovat od nuly.
   Tvorba určení formátu by měla být podle získaných informací popsána v manuálu k Excelu, který jsme bohužel neměli k dispozici, takže toto tvrzení nemůžeme potvrdit. Nicméně se bez něj lze obejít. Řadu nejčastějších formátů lze vysledovat přímo v aplikaci Excel v možnostech formátování buněk, pokud zvolíme formát čísla.
   Důležité je vědět, že formát čísla se využívá rovněž při práci s datem a časem. Jak víme, datum je reprezentováno jako celé číslo, čas jako číslo desetinné. Je třeba jim pouze nastavit některý z implementovaných formátů pro datum či čas. V opačném případě bude v buňce namísto očekávaného data resp. času zobrazeno pouze nic neříkající celé resp. desetinné číslo.
5. **Nastavení stylu, záznamy *P;F* a *P;E***
   Tyto záznamy se tvoří obdobně jako záznamy v bodu 4. Slouží k určení stylu buňky. Tzn. nastavení použitého typu písma, velikosti či barev. Velkým problémem bylo rozlišit význam záznamů *P;F* a *P;E*. Po zdlouhavém zkoumání jsme snad jen díky náhodě dospěli k výsledku, který byl potvrzen při dalším experimentování a tedy jej považujeme za správný.
   Záznamy *P;F* jsou zřejmě pouze *pomocné*. Mají stejnou strukturu jako záznamy *P;E*, ale vždy jsou uvedeny právě čtyři. Slouží většinou k nastavení nějakých obecných stylů. Je to z důvodu, že odkazy na záznamy definic vzhledu nevyužívají právě hodnotu indexu 4! Stejně jako záznamy *P;P* jsou číslovány od nuly, ale odkaz na čtvrtý záznam je považován za chybu. Domníváme se tedy, že záznamy *P;E* jsou Excelem číslovány od hodnoty pět, aby bylo zajištěno toto chování.
   Tato teorie ve všech zkoumaných příkladech fungovala správně. Bohužel se nám nepodařilo najít nějaké rozumné objasnění takového chování. Zřejmě je to interní záležitost implementace Excelu (možná pouze pod OS Windows).
   Vynechání některého dalšího indexu záznamu se dále neprojevilo pro několik desítek nadefinovaných záznamů, které by měly pro tvorbu jakéhokoliv dokumentu postačovat, proto způsob použití stylů považujeme za objasněný.
6. **Datum a čas**
   Prvotním problémem bylo zjištění jak je datum a čas vůbec interpretován. Bylo zřejmé, že na buňky je aplikován číselný formát, v němž je nastaven formát data či času. Při vygenerování několika dokumentů Excelem a jejich zkoumání se ukázalo, že datum je vždy vyjádřeno nějakým celým číslem, čas desetinným číslem menším než 1 a jejich spojením vznikne očekávaný výsledek – buňka obsahující časové razítko.

Určení čísel je popsáno v kapitole 4.1.4. Zde zmíníme pouze fakt, že datum je vyjádřeno jako počet uplynulých dnů od jistého *počátečního data* a čas je vyjádřen jako poměr uplynulé části dne ku délce dne.

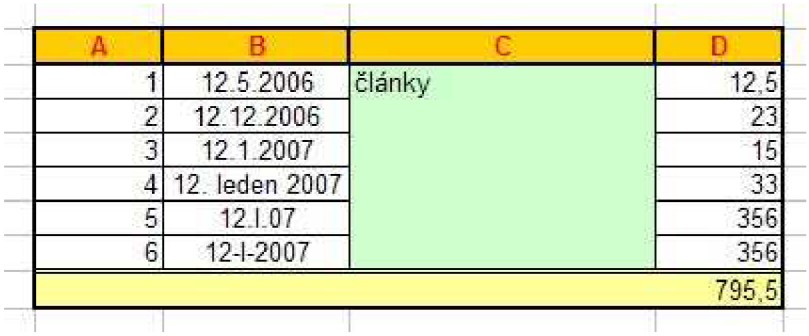
7. **Barvy**
   Sylk nedovoluje přenášet informace o barvě pozadí buněk. Dovoluje nastavit některé barvy písma. Na barvy se opět odkazuje pomocí čísel (indexů) barvy. Indexy nabývají hodnot od 0 do 63. Jejich interpretace je uvedena v následujícím obrázku. Tento byl zjištěn opět experimentálně, uložením různých kombinací barev v dokumentu Excelu jako formát SYLK. Zajímavostí je, že při opačném postupu – vytvořit ručně sylk soubor s odkazy na jednotlivé indexy barev – byly některé barvy Excelem interpretovány odlišně než jsou v uvedené tabulce, tzn. indexy nebyly shodné jako níže.

Obr. 4.2. Tabulka barev a jejich indexů pro použití v SYLK souboru

4.2. Ukázka výstupu z aplikace MS Excel

V této kapitole si uvedeme skutečnou ukázku souboru typu SYLK, která je použitelná v aplikaci MS Excel. Části souboru budou stručně vysvětleny pro objasnění jeho problematiky. Pro více informací opět odkazujeme na přílohu A.
Tato kapitola je uvedena také proto, aby bylo možné pozdější srovnání souboru vygenerovaného tabulkovým procesorem a souboru vygenerovaného vytvořenou aplikací. Jak uvidíme naše aplikace dokáže vygenerovat stejně funkční a přesto kratší SYLK soubory. Je to z toho důvodu, že Excel generuje některé „zbytečné“ informace, jež se v samotném zpracování dokumentu nikdy neuplatní.

Vytvořme si v aplikaci MS Excel 2003 jednoduchou tabulku. Abychom mohli ohodnotit vlastnosti formátu SYLK použijeme co nejvíce formátovacích možností – aplikace barvy pozadí, barvy písma, tučné písmo, různá zarovnání obsahu buněk, sloučení buněk, ohraničení různými typy čar, použití vzorců.
Výchozí tabulka může vypadat například takto:



A	B	C	D
1	12.5.2006	články	12,5
2	12.12.2006		23
3	12.1.2007		15
4	12. leden 2007		33
5	12.I.07		356
6	12-I-2007		356
			795,5

Obr. 4.3. Ukázka tvořené tabulky.

Nyní soubor uložíme jako formát SYLK (*slk, symbolické propojení*). Vygenerovaný soubor lze otevřít v některém z textových editorů, aby byla vidět posloupnost záznamů, který jej tvoří. Pokud otevřeme tento soubor v samotném Excelu, rovnou interpretuje záznamy a zobrazí tabulku! Jednotlivé záznamy jsou uvedeny a okomentovány v následující tabulce: (prázdné řádky a zvýraznění jednotlivých částí je pouze pro zvýšení přehlednosti. Skutečný vygenerovaný soubor je spojitý prostý text, kde jsou záznamy uvedeny na řádcích bezprostředně za sebou.)

Vygenerovaný kód	Komentář
ID;**PWXL;N**;E	První povinný záznam. Nastavuje aplikaci MS Excel, buňky budou chráněny
P;PGeneral	Definice formátů buněk, tento záznam bude mít identifikátor 0 a bude na něj odkazováno pomocí ;P0. Jak si můžeme všimnout, mnohé záznamy nebudou vůbec využity
P;P0	;P1
P;P0.00	;P2
P;P#,##0	;P3
P;P#,##0.00	;P4
P;P#,##0\ _K_#NAe;;\-#,##0\ _K_#NAe	;P5
P;P#,##0\ _K_#NAe;;[Red]\-#,##0\ _K_#NAe	;P6
P;P#,##0.00\ _K_#NAe;;\-#,##0.00\ _K_#NAe	;P7
P;P#,##0.00\ _K_#NAe;;[Red]\-#,##0.00\ _K_#NAe	;P8
P;P#,##0\ "$";;\-#,##0\ "$"	;P9
P;P#,##0\ "$";;[Red]\-#,##0\ "$"	;P10
P;P#,##0.00\ "$";;\-#,##0.00\ "$"	;P11
P;P#,##0.00\ "$";;[Red]\-#,##0.00\ "$"	;P12
P;P0%	;P13
P;P0.00%	;P14
P;P0.00E+00	;P15
P;P##0.0E+0	;P16
P;P#" "?/?	;P17
P;P#" "??/??	;P18

P;Pd/m/yyyy	;P19
P;Pd/mmm/yy	;P20
P;Pd/mmm	;P21
P;Pmmm/yy	;P22
P;Ph:mm\ AM/PM	;P23
P;Ph:mm:ss\ AM/PM	;P24
P;Ph:mm	;P25
P;Ph:mm:ss	;P26
P;Pd/m/yyyy\ h:mm	;P27
P;Pmm:ss	;P28
P;Pmm:ss.0	;P29
P;P@	;P30
P;P[h]:mm:ss	;P31
P;P_-* #,##0\ "$"_-;;\-* #,##0\ "$"_-;;_-* "-"\ "$"_-;;_-@_-	;P32
P;P_-* #,##0\ _K_#NAe_-;;\-* #,##0\ _K_#NAe_-;;_-* "-"\ _K_#NAe_-;;_-@_-	;P33
P;P_-* #,##0.00\ "$"_-;;\-* #,##0.00\ "$"_-;;_-* "-"??\ "$"_-;;_-@_-	;P34
P;P_-* #,##0.00\ _K_#NAe_-;;\-* #,##0.00\ _K_#NAe_-;;_-* "-"??\ _K_#NAe_-;;_-@_-	;P35
P;P[$-405]d\.\ mmmm\ yyyy	;P36
P;P[$-F800]dddd\,\ mmmm\ dd\,\ yyyy	;P37
P;P[$-405]d/mmm/yy;;@	;P38
P;P[$-405]d\-mmm\-yyyy;;@	;P39
P;FArial;M200	Následuje definice použitých písem. Zřejmě proto, že je 4. definice přeskočena, Excel vytvoří 4 definice ;F, které mají jinak stejnou strukturu jako definice ;E. Je nutné zde nastavit veškeré informace o písmu včetně kurzívy, tučného písma, barev... Definice má index 0 a je na ni odkazováno pomocí M0 ze záznamu F, vlastnosti ;F
P;FArial;M200	M1
P;FArial;M200	M2
P;FArial;M200	M3
P;EArial;M200	M5 !!! čtvrtá je opravdu vynechána
P;EArial;M200;SU;L13	M6
P;EArial;M200;SU;L37	M7
P;EArial;M200;SB;L11	M8 – tučné písmo barvou 11
F;**P0**;**DG0G8**;**M255**	Výchozí formátování, nastaví typ obsahu buněk jako obecný (P0 se odkazuje na definici PGeneral), výchozí nastavení buněk ve vlastnosti ;D jako obecné čísla bez desetinných

	míst zarovnány obecně, šířka sloupců je přednastavena na 8 znaků. Nastavuje výšku řádků na 255. (;M255)
B;**Y11;X6;**D2 2 10 5	Významné jsou informace o počtu využitých sloupců a řádků
O**;L**;D;V0;K47;G100 0.001	Nastavení vlastností ;L zajistí zobrazení názvů sloupců v Excelu klasicky A, B, C... Ostatní význam není znám
F;**W4 4 13**	Definice vlastností sloupců, upravují se šířky potřebných sloupců, V tomto záznamu se šířka sloupce 4 nastaví na 13 znaků
F;W5 5 21 F;**M270;R3** F;M270;R4 F;M270;R10 F;M285;R11 F;Y3;X3 F;X4 F;X5 F;X6	Úprava výšky řádků
F;**P0;FG0C;S**D**LRTBSM8**;Y4;X3	Následuje definice obsahu buněk: Formát buňky na souřadnicích 3, 4 bude Obsah: obecný, číslo obecné bez desetinných míst, zarovnání na střed. Styl: rámečky okolo, výplň a tučné písmo (není zajištěno definicí ;SD ale M8 – odkaz na definici písma 8 výše.)
C**;K"A"**	Obsah této buňky (3, 4) bude text A (musí být v uvozovkách)
F;P0;FG0C;SDLRTBSM8;X4 C;K"B" F;P0;FG0C;SDLRTBSM8;X5 C;K"C" F;P0;FG0C;SDLRTBSM8;X6 C;K"D" F;SLRTBM0;Y5;X3 C;K1 F;P19;FG0C;SLRTBM0;X4 C;K38849 F;SLRTSM0;X5 C;K"#NAel#NBanky"	Obsah *články* je zakódován do ASCII znaků

	pomocí ISO sekvence (# reprezentuje Esc) #NAe = č #NBa = á
F;SLRTBM0;X6 C;K12.5 F;SLRTBM0;Y6;X3 C;K2 F;P19;FG0C;SLRTBM0;X4 C;K39063 F;SLRSM0;X5 F;SLRTBM0;X6 C;K23 F;SLRTBM0;Y7;X3 C;K3	
F;P19;FG0C;SLRTBM0;X4 C;K39094 F;SLRSM0;X5 F;SLRTBM0;X6 C;K15 F;SLRTBM0;Y8;X3 C;K4	Odkaz na typ formátu 19 - datum Hodnota je opravdu vyjádřena celým číslem!
F;P37;FG0C;SLRTBM0;X4 C;K39094 F;SLRSM0;X5 F;SLRTBM0;X6 C;K33 F;SLRTBM0;Y9;X3 C;K5 F;P38;FG0C;SLRTBM0;X4 C;K39094 F;SLRSM0;X5 F;SLRTBM0;X6 C;K356 F;SLRTM0;Y10;X3 C;K6 F;P39;FG0C;SLRTM0;X4 C;K39094 F;SLRBSM0;X5 F;SLRTM0;X6 C;K356 F;SLTBSM0;Y11;X3	;P37 stejný datum vyjádřen v jiném tvaru
C;K795.5;ESUM(R[-6]C[+3]:R[-1]C[+3])	Zde bude hodnota buňky číslo 795,5. Je zde však uveden také vzorec, který sečte hodnoty uvedené ve sloupci D. Při automatické rekalkulaci bude hodnota přepočítána vždy při otevření souboru Excelem, tedy hodnota by nemusela být uvedena. V rámci kompatibility

	však doporučuji hodnotu uvádět
F;STBSM0;X4 F;STBSM0;X5 F;SRTBSM0;X6	
E	Povinný poslední záznam. End. Je také ukončit znakem pro konec řádku (Enter).

Výsledek po načtení souboru SYLK znovu do Excelu:

	A	B	C	D
1		12.5.2006	články	12,5
2		12.12.2006		23
3		12.1.2007		15
4		12. leden 2007		33
5		12.I.07		356
6		12-I-2007		356
795,5				

Obr. 4.4. Výsledek v SYLKu.

Pokud srovnáme výsledek se zadáním je vidět, že formát SYLK poněkud zaostává v možnostech zachování vzhledu dokumentů. Správně umí použít barvu a styl písma (tučné), zarovnání v buňkách, číselné formáty a vzorce. Naopak zcela ignoruje barvy pozadí, namísto ní vždy použije v obrázku viditelný výplňový vzor. Neumí rovněž zachovat sloučení buněk ani nastavit různé druhy ohraničení buněk. Jako ohraničení umožňuje pouze jednoduchou čáru šířky 1 bod.

4.3. Shrnutí

SYLK je textový formát určen k přenosu informací mezi různými aplikacemi. Velmi vhodný je zejména pro uchování dat pro tabulkové procesory. To, že je tento formát textový a poměrně jednoduchý s sebou nese řadu výhod a nevýhod.

Mezi výhody můžeme zařadit:

- jednoduchost
- nenáročnost na plné pochopení formátu
- snadné generování bez nutnosti využití tabulkového procesoru
- soubory jsou relativně krátké, což vede k úspoře místa
- spolehlivý přenos dat (hodnot a jejich typů)
- snadný přenos vzhledu jednoduchého dokumentu
- podpora vzorců

Také má ale řadu nevýhod. Mnohé z nich jsou viditelné z obrázků v předchozí kapitole. Týkají se zejména zachování vzhledu dokumentu.

- Neumožňuje nastavovat barvy pozadí: chceme-li výplň, použije výplňový vzor.
- Neumožňuje nastavit druh ohraničení buněk, pouze zda linka kolem buňky bude či nikoliv.

- Neumožňuje sloučit buňky.
- Neumožňuje natočení textu či vertikální text.
- Neumožňuje export speciálních objektů (grafy, …).
- Neumožňuje přenášet více listů v pracovním sešitu.
- Neumožňuje listy pojmenovat. Použije se jméno podle uloženého souboru.
- Obtížná práce s datem a časem.
- Zřejmě není příliš známým formátem.

Pokud tedy známe možnosti tohoto formátu, určitě pro něj nalezneme využití v mnoha aplikacích, v nichž potřebujeme přenést informace do tabulkových procesorů v jiném formátu nežli v CSV. Pokud bychom využili předepsaného ASCII kódování, měl by takto vytvořený dokument být přenositelný mezi různými platformami zcela bez problémů.

5. Formát XML

XML umožňuje přenést kompletní informace o vzhledu požadovaného dokumentu stejně jako data a informace o datových typech. Používá vyhrazených jmenných prostorů, bez jejichž použití nelze XML interpretovat aplikací MS Excel.

Oproti SYLKu má výhodu v tom, že se jedná o strukturovaný dokument, který je díky vhodnému pojmenování použitých XML elementů snadno pochopitelný a přehledný i bez dokumentace k tomuto formátu. Ovšem bez dokumentace je nemožné experimentálně zjistit všechny možné kombinace a výskyty elementů pro zápis požadovaného výsledku.

Dále odstraňuje problematiku kódování, neboť v XML dokumentu lze určit kódování obsahu a Excel jej pak interpretuje správně. (Při experimentech bylo použito kódování UTF-8.)

Nevýhodou nadále zůstává nemožnost přenášet objekty.

5.1. Popis formátu XML obecně

XML (eXtensible Markup Language) je dnes již hodně používaným a rozšířeným formátem. Existuje řada publikací i internetových zdrojů věnovaných tomuto formátu, protože si našel velké uplatnění zejména v oblasti předávání informací na internetu. Přestože je tento formát snad už poměrně známý, sluší se jej alespoň stručně uvést, stejně jako u předchozího zkoumaného formátu SYLK.

XML byl navržen pro zavedení logické struktury do vytvářených dokumentů. Má umožnit popsat obsah dokumentů jednotnou formou, která bude snadno zpracovatelná a usnadní hledání a shromažďování informací na internetu. Jednou z dalších diskutovaných možností je nahrazení například databázových systémů dokumenty XML. Ve své podstatě je to snadné, protože v XML můžeme ukládat pevně dané typy záznamů, takže místo tabulky v relační databázi jsme schopni vytvořit XML soubor se stejným obsahem. Otázkou je pak např. rychlost přístupu k takto uchovaným informacím, nároky na diskovou kapacitu ve srovnání s databázovými systémy.

Logické členění dokumentů je v dnešní době důležité také pro podporu budování sémantického webu, snazšího zpřístupnění obsahu webových stránek uživatelům se zrakovým postižením atd. Nutno však podotknout, že XML neumožňuje definovat sémantický význam dokumentů! Umí pouze nadefinovat byť i pevnou strukturu dokumentu o přesně daných datových typech obsahu s pojmenováním značek takovým, aby co nejlépe vystihovaly opravdový obsah, ale to je stále pouze oblast syntaxe. Získáme tak pouze logické členění dokumentu a můžeme podle předepsaných pravidel ověřit, zda zkoumaný dokument vyhoví či nikoliv. Sémantickou stránku nijak ověřit nemůžeme, protože nikde není uvedeno, co přesně by jednotlivé značky měly obsahovat. Pojmenování značek je čistě na autorech dokumentu a nikdo nemůže zaručit, že ve značce pojmenované jako LETADLO si autor bude uchovávat informace o takto pojmenovaném hotelu či restauraci. V jiném dokumentu tato značka třeba obsahuje informace o typu dopravního prostředku.

Z kontextu dokumentu si samozřejmě ve většině případů správný význam jednotlivých značek dokážeme odvodit (pokud jsou použity značky tomu napomáhající), důležitější však je, algoritmické zpracování počítačem, pro které toto není zajištěno.

Značky

Jazyk XML je jako většina značkovacích jazyků odvozen od návrhu jazyka SGML. Jakožto značkovací jazyk je tvořen jednotlivými značkami, popisující obsah a samotným obsahem – daty.

Pro lepší představu pro ty, kdo tento jazyk ještě neznají, je velmi známým podobným jazykem jazyk HTML či XHTML, který je primárně určen pro vytváření webových prezentací.

Značky se zapisují do úhlových závorek <*značka*> a jsou vždy párové. Tedy k počáteční značce existuje vždy koncová značka <*/značka*> případně je počáteční značka rovnou uzavřena <*značka/*>. Každý pár značek tvoří **element**. Můžeme tedy říci, že XML soubor je tvořen z větší části elementy, které obklopují jednotlivá data.

Pokud známe HTML, pak je snadné zvládnout základní strukturu XML s tím, že je nutno přísně dodržovat některá pravidla, zejména tyto:

- Každý XML dokument je definován uvnitř *kořenového elementu*, tzn. že všechny další elementy jsou uzavřeny právě tímto elementem.
- Názvy značek jsou *case-sensitive*. Počáteční značka musí přesně odpovídat koncové značce.
- Elementy se nesmí křížit –

nelze zapsat např. <znacka1><znacka2>data </znacka1></znacka2>
nýbrž správně <znacka1><znacka2>data </znacka2></znacka1>

- Uvnitř elementů se nesmí objevit některé *zakázané* znaky – ty musí být nahrazeny například entitami. Například nelze zapsat jako data úhlové závorky, protože by pak nebylo rozpoznatelné, zda-li se nejedná o součást některého elementu.

Atributy

Každý element může mít ve své počáteční značce definovánu řadu atributů. Ty se zapisují jako *název*=*"hodnota"* a jsou od sebe odděleny mezerami. Atributy můžeme rovněž využít pro uchování různých informací, případně můžou sloužit k upřesnění významu hodnot obsahu elementů.

Dokumenty lze vytvořit různými způsoby. Buď obsah vložit do elementů nebo atributů nebo použít jejich kombinaci. Nyní si ukážeme tyto tři způsoby:

```
<lide>
<osoba>
<jmeno>Josef</jmeno>
<vek>32</vek>
<vyska><cm>178</cm></vyska>
</osoba>
</lide>
```

Veškeré hodnoty jsou zapsány v elementech. Je vidět trochu nešikovného použití <*cm*> uvnitř elementu *vyska*.

```
<lide>
<osoba jmeno="Josef" vek="32" vyska="1.78" jednotky="m" />
</osoba>
</lide>
```

Zde jsou všechny hodnoty přesunuty do atributů.

```
<lide>
<osoba>
<jmeno>Josef</jmeno>
<vek>32</vek>
<vyska jednotky="cm">178</vyska>
</osoba>
</lide>
```

V poslední ukázce převažuje zápis hodnot v elementech, uvnitř elementu *vyska* je použit atribut *jednotky*, který nám zpřesňuje význam hodnoty v elementu.

Zřejmě nezáleží na tom, který způsob použijeme. Hodnoty pak získáme vhodným zpracováním zkoumanému dokumentu. V XML pro Excel je využit způsob třetí. Většina hodnot je zapsána v elementech s tím, že upřesňující nastavení či informace jsou zaznačeny v atributech elementů. Tento způsob pak eliminuje zbytečný rozsah vytvořených XML souborů nutným zápisem doplňujících elementů a také stále zachovává přehlednost výsledného souboru.

Jmenné prostory (namespaces)
Protože v XML si můžeme názvy značek libovolně volit, mohlo by při propojení některých dokumentů dojít ke kolizi mezi jejich názvy. Proto byly navrhnuty jmenné prostory, které omezují kontextový prostor, k němuž se dané názvy vztahují. Pokud tedy elementy zařadíme do nějakého jmenného prostoru, nebudou v kontextu mimo tento prostor známy a tudíž názvy nemůžou kolidovat s jinými stejně pojmenovanými elementy.

Jmenné prostory se vytvoří např. v atributu *xmlns* některého elementu, v němž nadefinujeme tzv. *prefix* jmenného prostoru, který bude provázán s umístěním určeném pomocí *URI (Uniform Resource Identifier)*

V následující ukázce si uvedeme definici dvou jmenných prostorů a jejich použití na některých elementech a atributech.

```
<lide xmlns:os='urn:example-org:osoby'
      xmlns:jed='urn:example-org:jednotky'>
<osoba>
<os:jmeno>Josef</os:jmeno>
<vek>32</vek>
<vyska jed:jednotky="cm">178</vyska>
</osoba>
</lide>
```

Deklarace XML
Dokument může začínat tzv. deklarací XML, ve které se nastavuje typ obsahu dokumentu a je možné nastavit také znakovou sadu, kterou dokument využívá. Protože je tato deklarace aplikací MS Excel využívána uvedu zde její ukázku. Deklaraci tvoří jeden jednoduchý řádek:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

Atributy encoding a standalone jsou nepovinné. Encoding určuje znakovou sadu dokumentu, v

ukázce je nastavena na UTF-8 (Excel tento atribut negeneruje). Standalone určuje, zda je dokument samostatný, tj. bez externích zdrojů.

Kompletní XML dokument by pak mohl vypadat následovně:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<lide xmlns:os='urn:example-org:osoby'
      xmlns:jed='urn:example-org:jednotky'>
<osoba>
<os:jmeno>Josef</os:jmeno>
<vek>32</vek>
<vyska jed:jednotky="cm">178</vyska>
</osoba>

<osoba>
<os:jmeno>Jan</os:jmeno>
<vek>50</vek>
<vyska jed:jednotky="m">1.8</vyska>
</osoba>
</lide>
```

DTD (document type definition)
DTD umožňuje předepsat jistá pravidla dokumentům. Protože tato problematika pro naši práci není důležitá, uvedu pouze využití DTD a nebudu zde uvádět vnitřní syntaxi.

DTD umožní nadefinovat pravidla pro posloupnost elementů, které budou tvořit výsledný korektní dokument. Zároveň určuje, které elementy mohou následovat či být uzavřeny v jednotlivých elementech. Může předepsat typ obsahu elementů či jejich opakování. Opakování elementů znamená kolikrát se mohou v daném kontextu vyskytovat – vůbec, jednou, vícekrát.

Excel samotný tato pravidla zřejmě používá, protože hlásí chybu při nevhodném umístění elementů. Nicméně umístění samotného DTD, podle kterého by bylo možné zjistit všechny závislosti, nebylo nalezeno.

5.2. Základní struktura XML pro MS Excel 2003

Pro vnitřní strukturu tohoto typu dokumentu platí poznatky uvedené v kapitole 5.1. o XML. Excel využívá svou podmnožinu pojmenovaných elementů a jmenných prostorů, které je nutné správně použít, jinak nastavené vlastnosti nebudou rozpoznány a dokument se nepodaří naformátovat podle našich představ.

Protože popis jednotlivých elementů, jejich významu, možných atributech a očekávaných hodnotách je velmi rozsáhlý, je pro zájemce uveden v příloze B. V této příloze jsou velmi podrobně popsány všechny informace, které byly nezbytné k vytvoření výsledné aplikace, která umí požadované dokumenty generovat. Tato příloha je poměrně rozsáhlá, přesto však neobsahuje vyčerpávající popis všech elementů a jejich možností. Formát XML pro uchování dokumentů v aplikaci Excel je natolik propracován, že jeho možnosti téměř odpovídají binární verzi Excelovských dokumentů XLS a tedy nebylo možné v omezeném časovém horizontu prostudovat podrobně všechny jeho detaily. Cílem tvořené aplikace bylo přenášet nejčastější a nejpoužívanější typy údajů – jednoduché tabulky, základní formátování textů, nastavení datových typů buněk apod. a proto ani nebylo nutné zkoumat v příloze nepopsané části.

V této kapitole tedy uvedeme pouze základy struktury korektního XML souboru

interpretovatelného Excelem s popisem jednotlivých částí, aby bylo možné se snadněji orientovat v uvedených ukázkách v dalších kapitolách.

Základní zjednodušená struktura vypadá následovně:

```
<?xml version="1.0"?>
<?mso-application progid="Excel.Sheet"?>
<Workbook xmlns="urn:schemas-microsoft-com:office:spreadsheet"
 xmlns:o="urn:schemas-microsoft-com:office:office"
 xmlns:x="urn:schemas-microsoft-com:office:excel"
 xmlns:ss="urn:schemas-microsoft-com:office:spreadsheet"
 xmlns:html="http://www.w3.org/TR/REC-html40">
 <DocumentProperties xmlns="urn:schemas-microsoft-
com:office:office">
 ...
 </DocumentProperties>
 <ExcelWorkbook xmlns="urn:schemas-microsoft-com:office:excel">
 ...
 </ExcelWorkbook>

 <Styles>
  ...
</Styles>
 <Worksheet ss:Name="List1">
  <Table>
   ...
  </Table>
  <WorksheetOptions xmlns="urn:schemas-microsoft-com:office:excel">
   ...
  </WorksheetOptions>
 </Worksheet>

 <Worksheet ss:Name="List2">
  <Table>
   ...
  </Table>
  <WorksheetOptions xmlns="urn:schemas-microsoft-com:office:excel">
   ...
  </WorksheetOptions>
 </Worksheet>
</Workbook>
```

XML soubor začíná deklarací. Kořenový element je nazván ***Workbook***, který obaluje veškerý obsah dokumentu. Název, stejně jako názvy ostatních elementů, je zvolen velmi výstižně. Reprezentuje opravdu celý pracovní sešit s jeho listy a základními vlastnostmi dokumentu. Uvnitř sešitu je možné vytvářet více listů, což otvírá nové možnosti, které u formátu SYLK nebyly k dispozici. V tomto elementu jsou rovněž nadefinovány jmenné prostory, které se vztahují k některým atributům u některých elementů. Tyto je nutno zřejmě zjistit experimentálně zkoumáním různých výstupů přímo z aplikace MS Excel.

Dalším elementem je ***DocumentProperties***, který nastavuje základní vlastnosti dokumentu. Tento element není nezbytný, ale je vždy vygenerován, proto je dobré alespoň některé vlastnosti uvádět.

Následuje ***ExcelWorkbook***, který má za úkol nastavit vlastnosti čistě pracovního sešitu. Tyto dále budou společné pro všechny případné pracovní listy. Opět není nutné jej uvádět.

Velmi důležitý je element ***Styles***, který v sobě uzavírá definice jednotlivých použitých formátovacích stylů, aplikovaných na buňky listů dokumentu. Každý styl je popsán v elementu

Style, který má povinný atribut ID, sloužící pro provázání stylu s buňkou. Počet vygenerovaných elementů a jejich rozsah závisí přímo na počtu různých použitých formátovacích prvků uvnitř dokumentu. Excel samotný nevyužívá možného dědění stylů na základě společných vlastností, ale pro každé formátování vytvoří vždy kompletní definici, proto počet roste podle požadavků. Např. pokud použijeme červené písmo, tučné červené písmo a červenou kurzívu, pak Excel vygeneruje tři samostatné styly, kdy v prvním nastaví červenou barvu písma, v dalším červenou barvu písma a duktus písma tučný, a v posledním červenou barvu písma a styl kurzívy namísto možnosti vytvořit například styl pro červenou barvu písma a dvou odvozených stylů, v nichž by bylo uvedenou pouze, že se jedná o písmo tučné či kurzívu. Odvozování působí snadným dojmem a mohlo by ušetřit v některých případech i poměrně velké množství zapisovaných informací. Každý rub má však svůj líc, takže na druhou stranu by vyvstaly další problémy se sledováním vztahů mezi styly a hledání závislostí, což by prodloužilo dobu zpracování dokumentu před kompletním načtením do Excelu.

Definice stylu může vypadat například následovně:

```
 <Style ss:ID="s1">
 <Alignment ss:Vertical="Top"/>
   <Borders>
    <Border ss:Position="Bottom" ss:LineStyle="Double"
ss:Weight="3"/>
    <Border ss:Position="Left" ss:LineStyle="Continuous"
ss:Weight="1"/>
    <Border ss:Position="Right" ss:LineStyle="Continuous"
ss:Weight="1"/>
    <Border ss:Position="Top" ss:LineStyle="Continuous"
ss:Weight="2"/>
   </Borders>
   <Interior ss:Color="#CCFFCC" ss:Pattern="Solid"/>
  </Style>
```

Zde jsme nastavili ohraničení vlevo a vpravo čarou o tloušťce 1 pixel, nahoře 2 pixely a dolů čarou dvojitou. Dále barvu pozadí na oranžovo-žlutou, vyjádřenu hexadecimálním RGB kódem #ccffcc. Styl má nastaven identifikátor s1. Všechny buňky, které mají využít tyto vlastnosti jej budou volat právě přes tento název.

Po seznamu všech aplikovaných stylů jsou definovány obsahy jednotlivých listů podle potřeby. Definice listu je uzavřena v elementu ***Worksheet***, a vyžaduje povinné pojmenování listu v atributu *ss:Name="nazev"*. Jméno listu je standardně v Excelu viditelné v záložkách listů ve spodní části okna aplikace. Celý list je chápán jako jedna rozsáhlá tabulka, jejíž buňky tvoří průsečíky sloupců (které jsou ve většině případů označeny písmeny) a řádků (které jsou číslovány). Do buněk této tabulky pak zapisujeme potřebná data, včetně případných *vlastních tabulek*, které jsou pouze ohraničená podmnožina vybraných buněk z tabulky listu.

Proto Excel očekává obsah listu uzavřen v elementu ***Table***. V každém listu je proto povolen pouze jeden tento element (jak bylo uvedeno výše, celý list je vlastně jedna velká tabulka). Tabulka je tvořena buňkami, které jsou v XML reprezentovány po jednotlivých řádcích. Využívá se elementů ***Row*** a ***Cell***, kdy pomocí indexů vymezíme řádek, ve kterém definujeme obsah a tento uložíme do konkrétní buňky *Cell*. Toto je elegantní a přehledný způsob jak vytvářet obsah i rozsáhlých listů, je nutné si pouze uvědomit, že musíme pracovat v kontextu některého řádku. Menší nevýhodou je ztráta možnosti zapsat obsah buněk v libovolném pořadí, jak je tomu u formátu SYLK. Pro ukázku lze zapsat tabulku například:

```
  <Table>
   <Row ss:Index="3" ss:AutoFitHeight="0" ss:Height="13.5"/>
   <Row ss:AutoFitHeight="0" ss:Height="13.5">
    <Cell ss:Index="3" ss:StyleID="s21"><Data
ss:Type="String">A</Data></Cell>
    <Cell ss:StyleID="s21"><Data ss:Type="String">B</Data></Cell>
    <Cell ss:StyleID="s21"><Data ss:Type="String">C</Data></Cell>
    <Cell ss:StyleID="s21"><Data ss:Type="String">D</Data></Cell>
   </Row>
   <Row>
    <Cell ss:Index="3" ss:StyleID="s22"><Data
ss:Type="Number">1</Data></Cell>
    <Cell ss:StyleID="s23"><Data ss:Type="DateTime">2006-05-
12T00:00:00.000</Data></Cell>
    <Cell ss:MergeDown="5" ss:StyleID="m46613552"><Data
ss:Type="String">články</Data></Cell>
    <Cell ss:StyleID="s31"><Data ss:Type="Number">12.5</Data></Cell>
   </Row>
</Table>
```

Po ukončení zápisu obsahu listu v elementu *Table* můžeme nastavit ještě případné nutné vlastnosti konkrétního listu v elementu ***WorksheetOptions***.

Detailnější popis elementů je uveden v příloze B. Pro pochopení řešené problematiky je tento stručný úvod dostačující. V následující kapitole si ukážeme opravdový výstup, který vznikne při uložení dokumentu ve formátu XML. Jednotlivé části budou rovněž stručně vysvětleny.

5.3. Ukázka výstupu z aplikace MS Excel 2003

V této kapitole si uvedeme ukázku skutečného Excelovského XML souboru. Částí souboru budou stručně vysvětleny pro objasnění jeho problematiky. Pro více informací opět odkazuji na přílohu B. Ukázka je uvedena také proto, aby bylo možné pozdější srovnání souboru vygenerovaného tabulkovým procesorem a souboru vygenerovaného vytvořenou aplikací a umožní nám také stručné srovnání se soubory SYLK.

Pro ukázku použijeme stejný příklad jako u formátu SYLK. Chceme uložit následující tabulku pomocí formátu XML (Soubor | Uložit jako tabulka XML)

A	B	C	D
1	12.5.2006	články	12,5
2	12.12.2006		23
3	12.1.2007		15
4	12. leden 2007		33
5	12.I.07		356
6	12-I-2007		356
			795,5

Obr. 5.1. Ukázka tvořené tabulky.

Začátek souboru tvoří hlavička XML a definice použitých jmenných prostorů na které se budou odkazovat některé elementy. Jak bylo zmíněno výše, tyto jmenné prostory je třeba použít, jinak obsah nebude interpretován správně.

```
<?xml version="1.0"?>
<?mso-application progid="Excel.Sheet"?>
<Workbook xmlns="urn:schemas-microsoft-com:office:spreadsheet"
 xmlns:o="urn:schemas-microsoft-com:office:office"
 xmlns:x="urn:schemas-microsoft-com:office:excel"
 xmlns:ss="urn:schemas-microsoft-com:office:spreadsheet"
 xmlns:html="http://www.w3.org/TR/REC-html40">
```

Dále se ukládají některé informace o dokumentu a pracovním sešitu. Tyto informace jsou víceméně volitelné, ale některé z nich je vhodné uvádět. Např. v elementu LastAuthor je uveden uživatel počítače, který dokument naposledy uložil, dále jsou zde informace o datu vytvoření dokumentu, nastavení rozměrů oken. Excel tyto údaje generuje vždy, proto je doporučuji používat taktéž.

```
 <DocumentProperties xmlns="urn:schemas-microsoft-
com:office:office">
  <LastAuthor>Zap</LastAuthor>
  <Created>2006-12-18T15:13:39Z</Created>
  <LastSaved>2006-12-18T15:13:39Z</LastSaved>
  <Version>11.6568</Version>
 </DocumentProperties>
 <ExcelWorkbook xmlns="urn:schemas-microsoft-com:office:excel">
  <WindowHeight>6540</WindowHeight>
  <WindowWidth>13260</WindowWidth>
  <WindowTopX>480</WindowTopX>
  <WindowTopY>45</WindowTopY>
  <ProtectStructure>False</ProtectStructure>
  <ProtectWindows>False</ProtectWindows>
 </ExcelWorkbook>
```

Definice stylů je velmi důležitá pro zachování shodného vzhledu. Každý styl má svůj identifikátor ID, pomocí něhož je propojen s konkrétní buňkou. Styly dle definice lze od sebe odvozovat (dědit), ale Excel samotný tuto možnost nevyužívá a pro každý styl definuje všechny vlastnosti znovu. Pro všechny buňky, které mají nastaveno nějaké formátování zde nalezneme definovaný styl.

```
<Styles>
  <Style ss:ID="Default" ss:Name="Normal">
   <Alignment ss:Vertical="Bottom"/>
   <Borders/>
   <Font x:CharSet="238"/>
   <Interior/>
   <NumberFormat/>
   <Protection/>
  </Style>
  <Style ss:ID="m46613552">
```

```
    <Alignment ss:Vertical="Top"/>
    <Borders>
     <Border ss:Position="Bottom" ss:LineStyle="Double"
ss:Weight="3"/>
     <Border ss:Position="Left" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Right" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Top" ss:LineStyle="Continuous"
ss:Weight="2"/>
    </Borders>
    <Interior ss:Color="#CCFFCC" ss:Pattern="Solid"/>
   </Style>
   <Style ss:ID="m46613562">
    <Alignment ss:Vertical="Bottom"/>
    <Borders>
     <Border ss:Position="Bottom" ss:LineStyle="Continuous"
ss:Weight="2"/>
     <Border ss:Position="Left" ss:LineStyle="Continuous"
ss:Weight="2"/>
     <Border ss:Position="Right" ss:LineStyle="Continuous"
ss:Weight="2"/>
     <Border ss:Position="Top" ss:LineStyle="Double" ss:Weight="3"/>
    </Borders>
    <Interior ss:Color="#FFFF99" ss:Pattern="Solid"/>
   </Style>
   <Style ss:ID="s21">
    <Alignment ss:Horizontal="Center" ss:Vertical="Bottom"/>
    <Borders>
     <Border ss:Position="Bottom" ss:LineStyle="Continuous"
ss:Weight="2"/>
     <Border ss:Position="Left" ss:LineStyle="Continuous"
ss:Weight="2"/>
     <Border ss:Position="Right" ss:LineStyle="Continuous"
ss:Weight="2"/>
     <Border ss:Position="Top" ss:LineStyle="Continuous"
ss:Weight="2"/>
    </Borders>
    <Font x:CharSet="238" x:Family="Swiss" ss:Color="#FF0000"
ss:Bold="1"/>
    <Interior ss:Color="#FFCC00" ss:Pattern="Solid"/>
   </Style>
```

Popíšeme si např. následující styl:
Definuje ohraničení buňky plnou čarou tloušťky 1 pixel nahoře a dole, 2 pixely vpravo a vlevo.

```
   <Style ss:ID="s22">
    <Borders>
     <Border ss:Position="Bottom" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Left" ss:LineStyle="Continuous"
ss:Weight="2"/>
     <Border ss:Position="Right" ss:LineStyle="Continuous"
ss:Weight="1"/>
```

```
     <Border ss:Position="Top" ss:LineStyle="Continuous"
ss:Weight="2"/>
    </Borders>
   </Style>
```

Tento styl kromě ohraničení nastavuje zarovnání textu vertikálně dolů, horizontálně na střed.
V elementu NumberFormat definuje typ obsažených dat jako ShortDate, tedy bude očekávat datum.

```
  <Style ss:ID="s23">
    <Alignment ss:Horizontal="Center" ss:Vertical="Bottom"/>
    <Borders>
     <Border ss:Position="Bottom" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Left" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Right" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Top" ss:LineStyle="Continuous"
ss:Weight="2"/>
    </Borders>
    <NumberFormat ss:Format="Short Date"/>
   </Style>
   <Style ss:ID="s31">
    <Borders>
     <Border ss:Position="Bottom" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Left" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Right" ss:LineStyle="Continuous"
ss:Weight="2"/>
     <Border ss:Position="Top" ss:LineStyle="Continuous"
ss:Weight="2"/>
    </Borders>
   </Style>
   <Style ss:ID="s32">
    <Borders>
     <Border ss:Position="Bottom" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Left" ss:LineStyle="Continuous"
ss:Weight="2"/>
     <Border ss:Position="Right" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Top" ss:LineStyle="Continuous"
ss:Weight="1"/>
    </Borders>
   </Style>
   <Style ss:ID="s33">
    <Alignment ss:Horizontal="Center" ss:Vertical="Bottom"/>
    <Borders>
     <Border ss:Position="Bottom" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Left" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Right" ss:LineStyle="Continuous"
```

```
ss:Weight="1"/>
     <Border ss:Position="Top" ss:LineStyle="Continuous"
ss:Weight="1"/>
    </Borders>
    <NumberFormat ss:Format="Short Date"/>
   </Style>
   <Style ss:ID="s34">
    <Borders>
     <Border ss:Position="Bottom" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Left" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Right" ss:LineStyle="Continuous"
ss:Weight="2"/>
     <Border ss:Position="Top" ss:LineStyle="Continuous"
ss:Weight="1"/>
    </Borders>
   </Style>
   <Style ss:ID="s35">
    <Alignment ss:Horizontal="Center" ss:Vertical="Bottom"/>
    <Borders>
     <Border ss:Position="Bottom" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Left" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Right" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Top" ss:LineStyle="Continuous"
ss:Weight="1"/>
    </Borders>
    <NumberFormat ss:Format="[$-F800]dddd\,\ mmmm\ dd\,\ yyyy"/>
   </Style>
   <Style ss:ID="s36">
    <Alignment ss:Horizontal="Center" ss:Vertical="Bottom"/>
    <Borders>
     <Border ss:Position="Bottom" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Left" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Right" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Top" ss:LineStyle="Continuous"
ss:Weight="1"/>
    </Borders>
    <NumberFormat ss:Format="[$-405]d/mmm/yy;@"/>
   </Style>
   <Style ss:ID="s37">
    <Borders>
     <Border ss:Position="Left" ss:LineStyle="Continuous"
ss:Weight="2"/>
     <Border ss:Position="Right" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Top" ss:LineStyle="Continuous"
ss:Weight="1"/>
    </Borders>
   </Style>
   <Style ss:ID="s38">
    <Alignment ss:Horizontal="Center" ss:Vertical="Bottom"/>
```

```
    <Borders>
     <Border ss:Position="Left" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Right" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Top" ss:LineStyle="Continuous"
ss:Weight="1"/>
    </Borders>
    <NumberFormat ss:Format="[$-405]d\-mmm\-yyyy;@"/>
   </Style>
   <Style ss:ID="s39">
    <Borders>
     <Border ss:Position="Left" ss:LineStyle="Continuous"
ss:Weight="1"/>
     <Border ss:Position="Right" ss:LineStyle="Continuous"
ss:Weight="2"/>
     <Border ss:Position="Top" ss:LineStyle="Continuous"
ss:Weight="1"/>
    </Borders>
   </Style>
  </Styles>
```

Následuje definice konkrétního pracovního listu. Těchto definic může následovat vícero. Každý list musí být pojmenován. Jméno se pak projeví na záložkách v aplikaci Excel. Výchozí hodnoty bývají tři listy List 1, List 2, List 3.

Veškerá data v listu jsou součástí jedné *tabulky*. V níž se definují použité sloupce a řádky a jednotlivé hodnoty buněk se určují v elementech *Cell* (buňka tabulky).

```
<Worksheet ss:Name="test1">
  <Table ss:ExpandedColumnCount="6" ss:ExpandedRowCount="11"
x:FullColumns="1"
   x:FullRows="1">
   <Column ss:Index="4" ss:Width="69.75"/>
```

Nejdříve se nastaví potřebné vlastnosti sloupcům a řádkům (jako šířky sloupců a výšky řádků)

```
  <Column ss:AutoFitWidth="0" ss:Width="114.75"/>
   <Row ss:Index="3" ss:AutoFitHeight="0" ss:Height="13.5"/>
```

A po té se definují kompletní řádky s obsahem. Tento řádek bude řádek 4. Není zde uveden Index, proto se předpokládá, že bude následovat za posledním uvedeným, který má index 3 (předchozí rámeček).
V řádku budou uvedeny 4 buňky, na které bude aplikován styl ID=21, ten zajistí vykreslení oranžového pozadí, tučného textu červeně, zarovnaného na střed buňky. Data budou řetězce (string) a vlastní hodnoty budou písmenka A, B, C, D

```
<Row ss:AutoFitHeight="0" ss:Height="13.5">
    <Cell ss:Index="3" ss:StyleID="s21"><Data
ss:Type="String">A</Data></Cell>
    <Cell ss:StyleID="s21"><Data ss:Type="String">B</Data></Cell>
    <Cell ss:StyleID="s21"><Data ss:Type="String">C</Data></Cell>
    <Cell ss:StyleID="s21"><Data ss:Type="String">D</Data></Cell>
   </Row>
```

Následuje definice ostatních řádků.

```
  <Row>
    <Cell ss:Index="3" ss:StyleID="s22"><Data
ss:Type="Number">1</Data></Cell>
    <Cell ss:StyleID="s23"><Data ss:Type="DateTime">2006-05-
12T00:00:00.000</Data></Cell>
    <Cell ss:MergeDown="5" ss:StyleID="m46613552"><Data
ss:Type="String">články</Data></Cell>
    <Cell ss:StyleID="s31"><Data ss:Type="Number">12.5</Data></Cell>
   </Row>
   <Row>
    <Cell ss:Index="3" ss:StyleID="s32"><Data
ss:Type="Number">2</Data></Cell>
    <Cell ss:StyleID="s33"><Data ss:Type="DateTime">2006-12-
12T00:00:00.000</Data></Cell>
    <Cell ss:Index="6" ss:StyleID="s34"><Data
ss:Type="Number">23</Data></Cell>
   </Row>
   <Row>
    <Cell ss:Index="3" ss:StyleID="s32"><Data
ss:Type="Number">3</Data></Cell>
    <Cell ss:StyleID="s33"><Data ss:Type="DateTime">2007-01-
12T00:00:00.000</Data></Cell>
    <Cell ss:Index="6" ss:StyleID="s34"><Data
ss:Type="Number">15</Data></Cell>
   </Row>
   <Row>
    <Cell ss:Index="3" ss:StyleID="s32"><Data
ss:Type="Number">4</Data></Cell>
    <Cell ss:StyleID="s35"><Data ss:Type="DateTime">2007-01-
12T00:00:00.000</Data></Cell>
    <Cell ss:Index="6" ss:StyleID="s34"><Data
ss:Type="Number">33</Data></Cell>
   </Row>
   <Row>
    <Cell ss:Index="3" ss:StyleID="s32"><Data
ss:Type="Number">5</Data></Cell>
    <Cell ss:StyleID="s36"><Data ss:Type="DateTime">2007-01-
12T00:00:00.000</Data></Cell>
    <Cell ss:Index="6" ss:StyleID="s34"><Data
ss:Type="Number">356</Data></Cell>
   </Row>
   <Row ss:AutoFitHeight="0" ss:Height="13.5">
    <Cell ss:Index="3" ss:StyleID="s37"><Data
ss:Type="Number">6</Data></Cell>
    <Cell ss:StyleID="s38"><Data ss:Type="DateTime">2007-01-
```

```
12T00:00:00.000</Data></Cell>
    <Cell ss:Index="6" ss:StyleID="s39"><Data
ss:Type="Number">356</Data></Cell>
   </Row>
```

V tomto řádku definujeme sloučenou buňku, která bude obsahovat vzorec SUMA pro součet hodnot v posledním sloupci.

```
 <Row ss:AutoFitHeight="0" ss:Height="14.25">
    <Cell ss:Index="3" ss:MergeAcross="3" ss:StyleID="m46613562"
     ss:Formula="=SUM(R[-6]C[3]:R[-1]C[3])"><Data
ss:Type="Number">795.5</Data></Cell>
   </Row>
```

Tabulku je nutno ukončit a lze nadefinovat některé vlastnosti samotného listu. Například skrýt mřížku, či celý list. Zde jsou nastaveny vlastnosti pro tisk (velikost záhlaví a zápatí), aktivní buňka při načtení souboru.

```
</Table>
  <WorksheetOptions xmlns="urn:schemas-microsoft-com:office:excel">
   <PageSetup>
    <Header x:Margin="0.4921259845"/>
    <Footer x:Margin="0.4921259845"/>
    <PageMargins x:Bottom="0.984251969" x:Left="0.78740157499999996"
     x:Right="0.78740157499999996" x:Top="0.984251969"/>
   </PageSetup>
   <Print>
    <ValidPrinterInfo/>
    <PaperSizeIndex>9</PaperSizeIndex>
    <HorizontalResolution>-3</HorizontalResolution>
    <VerticalResolution>0</VerticalResolution>
   </Print>
   <Selected/>
   <Panes>
    <Pane>
     <Number>3</Number>
     <ActiveRow>10</ActiveRow>
     <ActiveCol>9</ActiveCol>
    </Pane>
   </Panes>
   <ProtectObjects>False</ProtectObjects>
   <ProtectScenarios>False</ProtectScenarios>
  </WorksheetOptions>
```

Ukončí se definice pracovního listu, a za ní může následovat nový list. Zde nebude využit, proto následuje ukončení celého pracovního sešitu.

```
  </Worksheet>
</Workbook>
```

A to je celý vygenerovaný dokument. Myslím, že na rozdíl od formátu SYLK lze alespoň přibližně odvodit význam jednotlivých částí pouze se znalostí anglického jazyka.
Po načtení XML do aplikace není patrný žádný rozdíl mezi původním souborem xls a tímto xml:

A	B	C	D
1	12.5.2006	články	12,5
2	12.12.2006		23
3	12.1.2007		15
4	12. leden 2007		33
5	12.I.07		356
6	12-I-2007		356
			795,5

Obr. 5.2. Výsledek v XML.

5.4. Shrnutí

Formát XML odstraňuje téměř všechny nevýhody formátu SYLK. Stále přetrvává nemožnost přenášet objekty a grafy, ale ostatní nastavení obsahu a vzhledu požadovaného dokumentu je zde plně pod kontrolou.

Tento formát je striktně strukturovaný (podstata XML), proto je zapotřebí znát strukturu a možnosti vnořených elementů přesně. Samotné názvy elementů a atributů jsou však voleny tak, aby odrážely nastavované části tvořeného dokumentu a tudíž jsou snadno pochopitelné. Problémem může být nastavení jmenných prostorů některých elementů. Pokud není nastaven správný jmenný prostor, Excel element nerozpozná a ignoruje jeho definici.

Proto je při ruční tvorbě kompletního dokumentu nutné znát detailněji popis tohoto formátu. Částečný popis je uveden v příloze B.

Mezi výhody můžeme zařadit:

- obecně známé principy formátu
- možnost komplexního nastavení vzhledu obsahu
- snadné generování bez nutnosti využití tabulkového procesoru
- úspora místa vzhledem k binární verzi téhož dokumentu
- spolehlivý přenos dat (hodnot a jejich typů)
- podpora vzorců
- možnost přenosu kompletního pracovního sešitů více listy

Některé nevýhody:

- neumožňuje export speciálních objektů (grafy, …)
- velmi rozsáhlá definice formátu, neumožňuje plné pochopení všech možností

5.5. Srovnání výsledků SYLK a XML

Pokud srovnáme vytvořené dokumenty ukázkového příkladu z kapitol 4.2. a 5.3. jsou zřejmé nedostatečné schopnosti SYLKu zachovat vzhled dokumentů. Snížení formátovacích možností má výhodu v redukci paměťového prostoru pro vytvoření celistvého dokumentu. V ukázkách vychází pro dokument typu SYLK 127 řádků zdrojového kódu, a soubor má velikost asi 2,15 kB. Oproti tomu dokument XML již potřebuje 232 řádků zdrojového kódu o celkové velikosti souboru 10 kB.

Tedy zachování shodného vzhledu pomocí XML nás stojí téměř pětkrát více místa oproti použití formátu SYLK. Je třeba se zamyslet, zda-li je pro nás nutné zachovat grafický vzhled výsledku, nebo se spíše soustředit na redukci velikosti souboru. V dnešní době tyto rozdíly nejsou tolik dramatické. Pokud se velikost souboru bude pohybovat v řádech kB případně několika MB, není příliš nutné používat méně kvalitní formát pro úsporu. Výhodné je to zejména při předpokladu použití pomalého síťového připojení, či generování velmi objemných souborů o velikostech větších než desítky MB.

Poměr úspory místa SYLK souboru proti XML je zachován i v jiných testovaných souborech. SYLK soubory zabírají vždy zhruba pětinovou velikost souborů XML. Ovšem s více použitými formátovacími možnostmi se různě mění úspora XML oproti původnímu binárnímu souboru Excelu XLS. V našem příkladu původní soubor zabíral 15,5 kB, takže úspora XML formátu je necelých 40 % velikosti původního souboru. Ve složitějších případech může být vygenerovaný XML soubor dokonce větší než původní. V testovaných příkladech samozřejmě předpokládáme použití pouze dat, která lze uložit v obou textových formátech – tedy bez externích obrázků, grafů či dalších *vymožeností*.

6. Implementace

Úkolem bylo navrhnout a realizovat řešení exportu dat do tvaru, kterému rozumí tabulkové procesory. Konkrétně se zaměřením na aplikaci MS Excel a formáty SYLK a XML.

Výsledkem jsou knihovny implementované v jazyku PHP 5 za použití jeho objektového modelu. Tyto knihovny stačí připojit k PHP skriptu a pomocí jejich metod lze vytvořit soubor v požadovaném formátu.

Funkčnost byla vyzkoušena pod OS MS Windows XP a aplikacích MS Excel 2003, Excel 2000 a OpenOffice.org 2.1 Calc. Pod těmito verzemi fungují správně. Ve starší verzi Excel 2000 není ještě implementována podpora XML, formát SYLK zde však funguje správně. Podrobnější srovnání výsledků v jednotlivých aplikacích bude popsáno v závěrečné podkapitole.

Tato kapitola si však neklade za cíl podrobně rozebírat zdrojový kód vytvořené aplikace, ale ukázat postupy návrhu a zpracování úkolu, stručné ukázky použití vytvořených skriptů a předvést získané výsledky.

Kompletní zdrojové kódy aplikace včetně dokumentace a ukázky použití vytvořené aplikace jsou dostupné na přiloženém CD-ROMu, případně jsou k dispozici na URL adrese http://zap.profitux.cz/.

Tato kapitola se rovněž nezabývá problematikou programování v jazyku PHP a pro pochopení některých ukázkových pasáží předpokládá alespoň základní znalosti tohoto jazyka.

6.1. Výběr vývojových prostředků

V této kapitole si projdeme některé důležité počáteční úvahy od kterých se odvíjel vývoj aplikace. Shrneme-li požadavky, pak je naším úkolem vytvořit aplikaci, která bude schopná informace z portálového informačního systému přenést na uživatelský počítač ve formátu interpretovatelném tabulkovým procesorem.

Prvním důležitým požadavkem tedy je, aby aplikace uměla spolupracovat s portálovým systémem. Protože budou data nabídnuta různým uživatelům, je vhodné, aby samotná aplikace pracovala přímo jako součást portálu, aby nebylo nutné vyvíjet klientské programy, které by si uživatelé podle potřeby zprovoznily na svých počítačích a pomocí nich získávali potřebná data. Pro programové řešení portálů existuje řada nejrůznějších jazyků, namátkou ASP, PHP, JAVA, Perl, Python, proto je nutné si vybrat pro naše účely vhodný z nich.

Volba padla na jazyk PHP, který je jednoduchý a ve velké míře podporovaný. Lze jej relativně snadno propojit se systémy, na kterých je využito jiné řešení než právě uvedený jazyk PHP. Rovněž značnou roli hrál subjektivní názor na vhodnost a kvalitu tohoto jazyka, a po zvážení všech jeho schopností, byl shledán plně vyhovujícím.

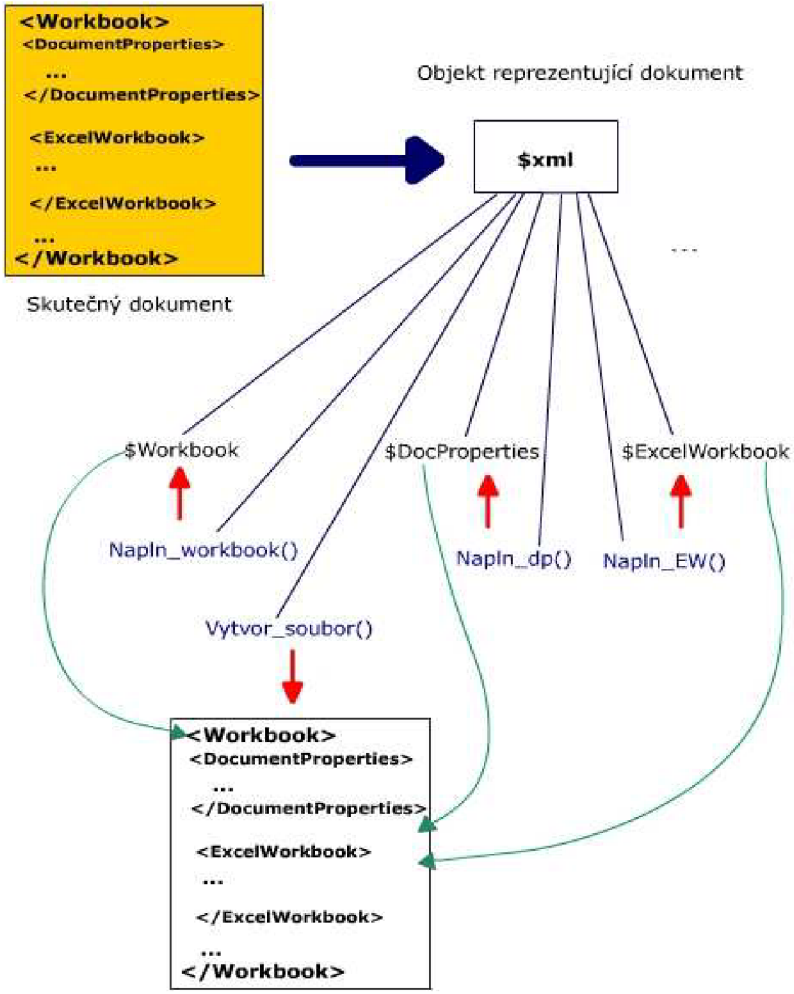
Proč verze 5? Odpovědí je vylepšený objektový model, který už se blíží objektovým modelům vyšších programovacích jazyků jako C++ či Java. Možná tato verze ještě není plně podporována na každém *FreeHostingu*, ale toto by pro nás nemělo být limitním omezením. Protože se to do budoucna určitě změní, a pak dobrý program je nutné postavit na dobrém základu. Objektový model je ve starších verzích jazyka PHP velmi jednoduchý a pro rozsáhlejší aplikace nevhodný.

Proč se ubírat cestou OOP (Objektově Orientované Programování)? V dnešní době se všechny moderní a nově vznikající jazyky zaměřují na důkladnou podporu OOP. Programovací metody prošly dlouhým vývojem a v poslední době začal převažovat právě objektový přístup. Stručně řečeno to znamená, že se na věci reálného světa díváme jako na celek a takto se je snažíme zachytit pomocí programovacího jazyka. Jako příklad můžeme uvést například domácí zvíře psa.

Abychom jej reprezentovali v programu, vytvoříme jeho model. Vznikne objekt, ve kterém budou zahrnuty nějakou formou jeho vlastnosti jako barva, výška, jména apod. a také v něm bude zahrnuto jeho chování tedy, že umí štěkat, běhat, hrabat atd. Vlastnosti jsou v programu obvykle nazývány jako atributy a chování jako metody.

OOP má řadu výhod, mezi nejvýznamnější z nich patří *zapouzdření* a *dědičnost*. Má také své nevýhody, např. je složitější na pochopení na rozdíl od *klasického přístupu*. Pro má také řadu příznivců a odpůrců. Celkově je tato problematika mnohem náročnější a není účelem této práce se jí zabývat. Tato část je jen velmi jemnou zmínkou o OOP a měla by sloužit jako vysvětlení volby objektového přístupu a volby verze programovacího jazyka.

Výsledkem úspěšné činnosti spuštěné aplikace bude vytvořený dokument. Dokument můžeme modelovat jako samostatný objekt, který se skládá z jednotlivých částí. Jak víme z popisu formátu, u SYLKu jsou to různé typy záznamů a u XML jsou to různé elementy, které tvoří celkový obsah. Abychom mohli tento obsah snadněji generovat, je vhodné implementovat metody, které nám podle požadavků zajistí správné vytvoření jednotlivých části dokumentu. Konečným spojením těchto částí ve správném pořadí pak získáme kompletní soubor v požadovaném formátu, který můžeme uživateli nabídnout ke stažení.



Obr. 6.1. Schéma reprezentace vstupního dokumentu v objektovém modelu

Na obrázku 6.1. je uvedeno schéma, které znázorňuje tento proces na příkladu tvorby XML souboru. Vstupní soubor uvnitř programu chápeme jako objekt, neboli proměnnou *$xml*, který má pro uchování svých částí definovánu řadu dalších proměnných (začínají znakem $). Také má implementovány metody (modře), které naplní jednotlivé proměnné správným obsahem. Metoda *Vytvor_soubor()* pak zajistí spojení obsahu rozděleného do proměnných tak, aby vznikl korektní soubor.

Je také vhodné předem uvažovat o způsobu zpracování možných chyb, které nastanou za běhu skriptů. Pro ošetření možných chybových stavů bude použit systém výjimek, které umožňují zachytit neočekávané dění uvnitř programu a vyvolat akci, která na takové chování bude reagovat.

Systém výjimek je rovněž k dispozici až od verze PHP 5.

Dalším požadavkem byla zpracovatelnost výsledku v tabulkovém procesoru. To znamenalo výběr vhodných přenosových formátů souborů, tedy takových, aby je bylo možné vygenerovat bez použití tabulkového procesoru a aby přesto byly v tabulkovém procesoru použitelné. Jako takové vyhovují formáty textové, protože generování prostého textu je jednoduché. Více o zvolených formátech SYLK a XML je popsáno v předchozích kapitolách a přílohách A a B.

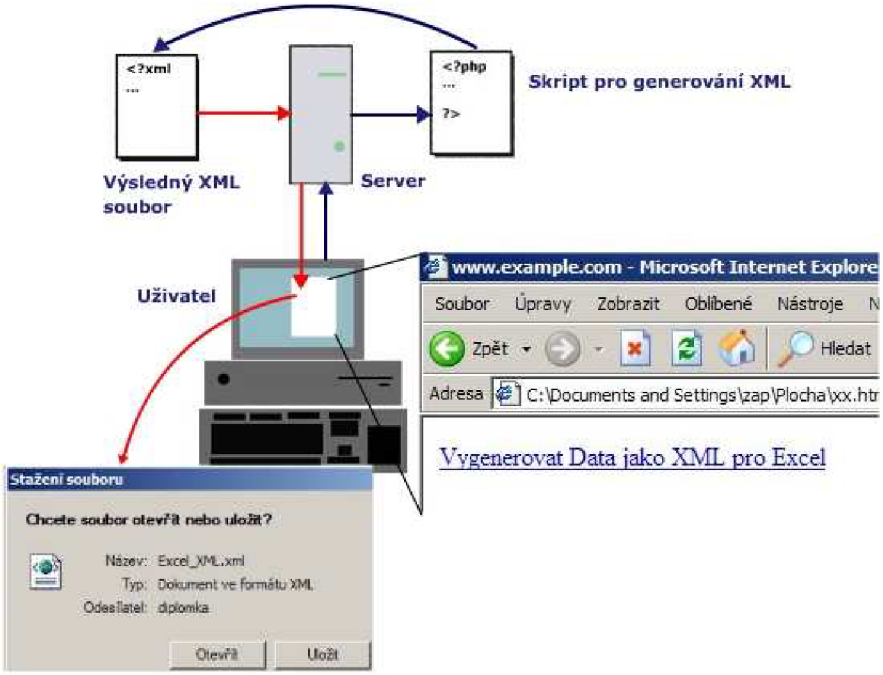
Protože byl jako primární tabulkový procesor vybrán MS Excel, bylo také nutné, přizpůsobit tomu cílovou platformu, kterou je MS Windows. Samotný PHP skript je na platformě nezávislý, pokud nebudeme používat jisté specifické vlastnosti konkrétních operačních systémů. Volbou univerzálního kódování UTF-8 je přenositelnost podpořena i do cizojazyčných projektů.

Nyní máme k dispozici stručný ucelený přehled o základních požadavcích a výběru prostředků. Můžeme se tedy pustit do konkrétnějšího návrhu aplikace.

6.2. Základní návrh funkcionality

Před samotným programováním je u každého projektu nejdůležitější důkladný návrh. Zahrnuje jak detailní rozbor zadání, tak předběžnou představu o podobě programového kódu. Je nutné ověřit na smyšlených scénářích průběhy různých situací, které mohou ve výsledku nastat, abychom odhalili případné nedostatky v našem návrhu. Část návrhu je časově náročná, nicméně pokud je návrh správný, ušetří nám spoustu problémů a času se samotným programováním a opravováním objevených chyb, které se objevily v důsledku opomenutí některé možnosti.

Scénář pro naši aplikaci je velmi jednoduchý: uživatel si přeje získat data uložená na nějakém portálu ve vhodné formě pro tabulkový procesor MS Excel. Nemusíme řešit žádné zvláštní situace. Pokud vše proběhne v pořádku, nabídne se uživateli soubor ke stažení. Pokud nastane chyba při vytváření požadovaného souboru, vypíše se uživateli zpráva prostřednictvím internetového prohlížeče.



Obr. 6.2. Scénář práce vytvářené aplikace

Uživatel tedy bude pracovat s portálovým systémem prostřednictvím svého internetového prohlížeče. Pro export dat do tabulkového procesoru, mu tento systém nabídne odkaz, jehož aktivováním se odešle požadavek na server. Tento požadavek vyvolá obslužný skript, který bude mít za úkol vytvořit kompletní soubor požadovaného formátu (na obrázku generujeme formát XML). Po dokončení práce skriptu, je nový soubor předán serveru, aby jej mohl vrátit zpět klientovi. Soubor je dále nabídnut uživateli opět prostřednictvím internetového prohlížeče ke stažení, nebo rovnou k otevření. Pokud má uživatel k dispozici MS Excel, bude soubor po otevření rovnou načten do této aplikace.

Potřebné prostředky k realizaci takového řešení máme k dispozici. Jako vhodné formáty jsme zvolili formát SYLK a XML. Podrobněji jsou popsány v předchozích kapitolách této práce a v přílohách. Vhodný programovací jazyk, který je možno využít ve webovských aplikacích máme rovněž vybrán. V předchozí kapitole je uveden návrh objektového přístupu, který při implementaci využijeme.

Poslední otázkou zůstává, jak vyřešit implementaci těchto dvou formátů? Jelikož se jedná o zcela odlišné formáty, které spolu nemají žádné vnitřní spojitosti, bude nejlepší metodou vytvořit dvě odlišné knihovny. Každá z nich bude umět generovat právě jeden typ souboru. Toto rozdělení jednak zvýší přehlednost zdrojových kódů oddělením jednotlivých celků, dále pak může přinést časovou úsporu při generování výsledků na straně serveru. Jelikož se v jednom okamžiku dá vygenerovat pouze jeden soubor, pak se právě podle požadavků na typ výsledného souboru využije pouze kód příslušné knihovny namísto toho, aby server musel zbytečně zpracovat veškerý kód obou knihoven.

Můžeme se tedy pustit do programování. Začneme jednodušším z formátů, tedy formátem SYLK.

6.3. SYLK

Naším úkolem bude nyní vytvořit samostatný PHP skript, který zajistí obsluhu požadavků pro generování SYLK dokumentů. Protože se jedná o samostatný celek provádějící určitou činnost, budeme jej v další části nazývat knihovnou.

Protože výsledné dokumenty budou zřejmě zcela odlišné, je důležité vytvořit vhodné rozhraní pro práci s výslednou knihovnou. Základní idea je vytvořit sadu funkcí, které může uživatel této knihovny využít ke generování SYLK souborů. Obsah dokumentu bude nutné předávat pomocí nastavovaných proměnných. Aby byl uživatel odstíněn od nutnosti znát detaily tohoto formátu, je potřeba navrhnout rozumný, obecný a snadno pochopitelný způsob nastavování obsahu dokumentů. Samozřejmě se výsledné rozhraní a způsob práce nebude zamlouvat všem, ale po nastudování ukázkových příkladů bude snad pochopitelné.

Pokud budeme vycházet z objektově orientovaného návrhu uvedeného v předchozích kapitolách, bude nutné vytvořit *třídu*, která je základem pro odvozování objektů. Třída v sobě zahrnuje veškeré informace o vlastnostech a metodách objektů. Tedy naše výsledná knihovna je ve své podstatě jedna rozsáhlejší třída v jazyku PHP 5.

6.3.1. Popis knihovny c_sylk.php

Vytvořená knihovna je pojmenována c_sylk.php, aby název vypovídal o jejím účelu. Prefix *c_* pro rozlišení, že se jedná o třídu a *sylk* pak proto, že je cíleně zaměřena na generování sylk dokumentů. Obdobně bude pojmenována knihovna pro tvorbu XML souborů.

V této kapitole nebudou popsány podrobné detaily o způsobu zápisu ani významu zdrojového kódu. Tyto informace jsou součástí dokumentace k vytvořenému projektu. Dokumentace je k dispozici na přiloženém CD-ROMu nebo na webovských stránkách projektu: http://zap.profitux.cz/.

Tato knihovna je téměř samostatný celek, který dokáže vytvořit korektní SYLK soubor. Je nutné pouze dodržovat předem určený způsob předávání informací o požadovaném obsahu pro zpracování touto knihovnou. Vesměs se jedná o nastavování parametrů metodám této třídy. Pomocí nich pak postupně tvoříme obsah dokumentu v jeho objektové reprezentaci uvnitř skriptu. Knihovna ke své funkčnosti vyžaduje ještě třídu c_exceptions.php, která je rovněž součástí projektu. Tato třída byla navržena pro implementaci zpracování *výjimek*. V konečném důsledku se jedná pouze o výpis chybového hlášení, které upozorní autora generujících skriptů na závažné chyby, jejichž důsledkem nelze vytvořit požadovaný soubor. Součástí některých těchto zpráv je rovněž podrobný popis v jaké části pravděpodobně došlo k chybě a jaký je možný způsob její nápravy. Pokud toto z nápovědy není zřejmé, je nutno prozkoumat příslušnou část knihovny a pokusit se z komentářů k významu jednotlivých části problém odhalit.

Pro plné pochopení funkčnosti vytvořené knihovny je nutné seznámení se s formátem SYLK. Jeho neznalost však nebrání použití knihovny pro generování takovýchto souborů. Stačí podle návodů a dokumentace nastudovat principy rozhraní dané knihovny a lze ji snadno a plně využívat.

6.3.2. Práce s knihovnou c_sylk.php

Máme-li k dispozici obě potřebné knihovny, můžeme se pustit do tvorby skriptu, který nám vygeneruje soubor pro MS Excel. Knihovny sami o sobě nedokáží vytvořit žádný soubor, je nutné je připojit do skriptů, které zajistí vložení požadovaného obsahu souboru.

Podrobnější detaily jsou rovněž součástí dokumentace a návodů. Ukážeme si pouze

nejdůležitější kroky, které je třeba dodržet, abychom dosáhli správného výsledku. Podotkněme, že průběh běhu skriptu musí být bez problémů, protože Excel reaguje velmi citlivě na výskyt jakéhokoliv neočekávaného záznamu ve zdrojovém souboru dokumentu SYLK.

Tvorbu generujícího skriptu lze shrnout do osmi kroků, které budou uvedeny společně s jejich obrazem v PHP kódu**:**

1. Do skriptu připojíme obě potřebné khihovny
   ```
   include_once('c_exceptions.php'); include_once('c_sylk.php');
   ```
2. Vytvoříme si instanci třídy, tedy objekt reprezentující tvořený dokument se kterým budeme pracovat:
   ```
   $slk = new c_sylk();
   ```
3. Pokud je potřeba, nastavíme výchozí šířky sloupců.
   ```
   $slk->formatCol(...);
   ```
4. Pokud je potřeba nastavíme výšky řádků.
   ```
   $slk->formatRow(...);
   ```
5. Je vhodné nastavit výchozí formát buněk.
   ```
   $slk->defaultFormat(...);
   ```
6. Postupně vkládáme jednotlivé buňky listu Excelu.
   ```
   // metoda cell(...)
   ```
7. Spojíme jednotlivé části dokumentu ve správném pořadí.
   ```
   $slk->setSylk(...);
   ```
8. Nabídneme soubor ke stažení.
   ```
   header("Content-Type: application/vnd.ms-excel");
   header("Content-Disposition: attachment;filename=$filename");
   echo iconv('UTF-8', 'WINDOWS-1250//IGNORE', $slk->getSylk());
   ```

Schéma 6.1. Postup tvorby SYLK dokumentu s využitím knihovny c_sylk.php

Jak vidíme, základní myšlenka je velmi jednoduchá. Vytvoření celého funkčního skriptu bude mírně komplikovanější, protože metody v ukázce vyžadují řadu parametrů, které zajišťují vložení požadovaného obsahu. Jejich vytvářením a použitím se kód rozroste.

Nejdůležitější je pro nás tvorba právě těchto parametrů. Ukážeme si tedy možnosti, jakým způsobem lze předávat metody potřebné informace, aby je mohla převést na části sylk dokumentu.

Ve většině případů je uložení relevantních hodnot pro tvořenou část realizováno použitím asociativního pole. Asociativní pole je standardní vnitřní datový typ jazyka. Umožňuje uložit kolekci různých druhů informací, přístupných pod jedinečnými názvy klíčů (indexů pole). Nyní stačí znát, jaké typy klíčů a jejich hodnot vyžadují jednotlivé nastavované části. V knihovně jsou tyto hodnoty voleny záměrně tak, aby co nejblíže vystihovali účel jejich použití. Pro tvorbu názvů je využito anglického jazyka, protože jím lze stručně a jednoznačně vystihnout požadovaný pojem. Pro většinu programátorů jsou anglické termíny zažité a zcela přirozené, proto je právě tento jazyk volen jako nejschůdnější cesta.

Při nastavování formátů sloupců či řádků se objevují klíče jako *WIDTH*, *NUM_FORMAT, DIGITS, ALIGN* apod., které mají význam *nastavení šířky sloupce, základní formát čísla, počet desetinných míst čísla, zarovnání obsahu v buňkách.* Klíče v tomto případě jsou snad zcela srozumitelné a vystihují přesně svůj účel. Pro snadné odlišení významových klíčů je nutné uvádět velkými písmeny. Pro zvýšení přehlednosti názvů odvozených od více slov, jsou jednotlivá slova

oddělena znakem podtržítka.

Možné hodnoty, které lze přiřadit těmto klíčům je nutné vysledovat z dokumentace knihoven. Např. šířka očekává číslo, které udává šířku sloupce ve znacích, ale určení číselného formátu či zarovnání již není jednoznačné. Kdyby knihovna měla kontrolovat všechny možné hodnoty, vyjadřující totéž, značně by vzrostla časová složitost provádění skriptu. Zarovnání na střed buňky je možné vyjádřit například hodnotou *center, C, stred, middle, cent,* či čísly a dalšími kombinacemi. Záleží na zvyku každého z nás, co by sám upřednostnil.

Pro zavedení jednoznačnosti a tím rovnou mírné optimalizace rychlosti při zpracování jsou možné hodnoty pevně dány a je nutno se s nimi seznámit. Tam kde je to možné, hodnotu nastavované vlastnosti tvoří jedno velké písmenko, které je voleno rovněž tak, aby charakterizovalo nastavovanou hodnotu a bylo tak snadno zapamatovatelné. V ostatních případech se očekávají celé řetězce či čísla, které již dále musí být testovány, zda vyjadřují smysluplnou a povolenou hodnotu. Žádné jiné hodnoty nejsou platné a v případě jejich výskytu buď dojde k ignorování vkládaného údaje, nebo výpisu chybového hlášení, které upozorní uživatele, že je nutno hodnotu změnit.

Pro uvedený příklad hodnoty pro zarovnání, tedy klíč *WEIGHT* vyhoví dle předchozího popisu písmenko *C*. Jelikož se vychází opět z angličtiny a také z terminologie vnitřního formátu SYLK, *C* jako *Center* vyhovuje. Pozor ovšem na kontext použití. Při nastavování formátu čísla *C* označuje, že se jedná o měnu, anglicky *currency.*

Nastavení obecných formátů není nikterak složitá záležitost. Podle schématu 6.1. je toto nastavení dokonce nepovinné. Excel má totiž vždy nastaven implicitní formát řádků a sloupců, který bude použit, pokud není řečeno jinak.

Zbývá osvětlit problematiku vkládání vlastních buněk listu. Buňkami rozumějme prostor v aplikaci MS Excel, který je vymezen průsečíky řádků a sloupců listu. Ve většině případů jsou buňky ohraničeny světle šedou mřížkou a jsou obdélníkového tvaru, orientovány na šířku. Každá buňka má jedinečné kartézské souřadnice, kterými můžeme přímo přistoupit k jejich obsahu.

Jak bylo popsáno v kapitole věnované formátu SYLK, umožňuje tento formát přímý přístup k jednotlivým souřadnicím nezávisle na okolním kontextu. Umožňuje nastavit vzhled a obsah první buňky listu, hned na to například buňky uprostřed, a pak např. pokračovat vedle první buňky.

Z hlediska efektivity je tento přístup špatný. Excel si musí shromáždit informace o všech nastavených buňkách, tudíž si je musí zařadit do správného pořadí výskytu. Takovéto zbytečné přeskakování po buňkách prodlouží načítání dokumentu do tohoto tabulkového procesoru. Proto je výhodnější definovat jednotlivé buňky po řádcích či po sloupcích a toto dodržovat jednotně v celém dokumentu. Navíc mají obvykle řádky či sloupce společné vlastnosti týkající se vzhledu nebo formátu obsahu, které lze využít a ušetřit tak velikost vytvářeného zdrojového skriptu.

Excel samotný generuje soubory SYLK po řádcích, proto je nejpřirozenějším řešením využít také tento postup. Zápis obsahu v jiném pořadí žádným způsobem neporušuje pravidla tohoto formátu, proto je vhodné využít si způsob nejvíce vyhovující použité aplikaci.

Samotná tvorba buňky probíhá v SYLKU ve dvou krocích. V jednom kroku se nastaví formát buňky a ve druhém vlastní obsah. Každý krok znamená jeden typ záznamu v souboru. Nezávisí na pořadí těchto kroků, je možné nejdříve vložit hodnotu do buňky a následně nastavit její formát a vzhled. Buňka je určena nastavenými souřadnicemi.

Excel využívá způsobu prvního. Nejdříve nastavuje formát buňky. Zřejmě je to z důvodu, že naformátovat lze i buňku, která nebude obsahovat žádnou hodnotu. Např. orámujeme buňky, a vytvoříme si tak tabulku, do které budeme později ručně zadávat hodnoty. Zároveň ve formátovacím záznamu nastavuje souřadnice buňky. Každá buňka má vždy nastaven formátovací záznam bez ohledu na to, zda je nezbytný či nikoliv. Pokud nemá žádné specifické nastavení, obsahuje odkaz na výchozí formát obsahu buňky.

Dalším poznatkem je, že Excel neuvádí souřadnice v záznamu s definicí obsahu buňky. Protože má každá buňka přiřazena jistý formát a tento formát má nastaveny souřadnice buňky, ke které se vztahuje, jsou při definici obsahu buňky automaticky nastaveny souřadnice určené v předchozím záznamu.

Pokud to samotný Excel provádí tímto způsobem, je zřejmě nejlepší tento styl dodržet. Opět to není nezbytné, nicméně naše knihovna generuje záznamy pro vytváření obsahu listu právě v tomto duchu.

Knihovna však tyto kroky spojuje do jediného volání příslušné metody, která zajistí vytvoření správných záznamů. Je ale nutné vytvořit více proměnných, které zajistí naplnění obou záznamů. Pro vytvoření buňky je potřeba zadat souřadnice, na kterých se má buňka v listu objevit. Dále vlastnosti vzhledu, které se předávají ve formě asociativního pole s vhodnými klíči a možnými hodnotami. Musíme předat také vlastní obsah buňky, který vložíme opět do jiného pevně stanového asociativního pole a nakonec ještě typ formátu čísla obsahu, který se má na danou buňku vztahovat. Formáty byly rovněž popsány v kapitolách o SYLKu, udávají např. jak má vypadat datum, čas, měna, počet desetinných míst.

Vysvětlíme si nyní, proč je pro předání obsahu použito také asociativní pole namísto jednoduché proměnné. Je to z důvodu rozlišení typu obsahu, který předáváme. Pole použité pro obsah má pevně dané klíče, které je možno použití. Např. klíč *VAL* slouží k určení hodnoty obsahu. Oproti tomu na indexu *EXPR* předáváme vzorec, který má být použit pro výpočet obsahu z ostatních buněk v listu. Je možné nastavit některé další vlastnosti buňky, případně její sdílení. Na detailní popis opět odkazuji na dokumentaci k projektu. Použití asociativního pole je přehlednější a snazší než nastavování řady dalších parametrů pro metodu, která bude mít tvorbu obsahu na starosti. Navíc se ve většině případů řada z těchto vlastností nepoužije. Záleží na charakteru výsledného dokumentu. Tato funkčnost je víceméně přidána pro zachování všech možností formátu SYLK.

Pro předávání vzorců, se využívá standardního zápisu vzorců v Excelu. Lze využít jak relativní určení buněk, jejichž hodnoty se mají ve vzorci použít (buňka v předchozím řádku od aktuální apod.) tak absolutní přímým zápisem souřadnic. Rovněž lze využívat rozsahy a oblasti, tedy vzorec pro součet hodnot ve sloupci může vypadat *SUM(R[-12]C:R[-1]C).* Pro zachování kompatibility doporučuji používat spíše absolutní určení buněk a také výchozí hodnoty buňky pro případ, že vzorec nebude interpretován správně. Více bude uvedeno v kapitole věnující se srovnání výsledků v OpenOffice.org Calc.

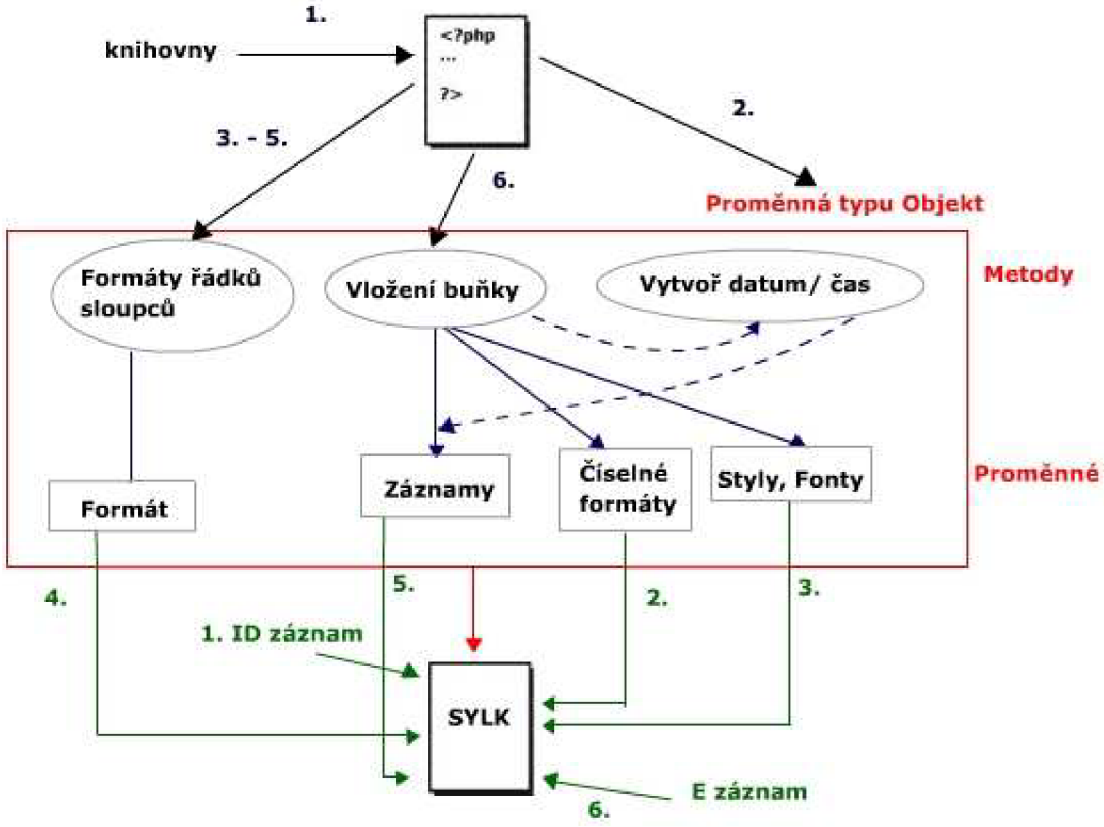
Ostatní hodnoty se předávají přímo jako čísla či textové řetězce. Připomeňme si pouze trochu zvláštní způsob práce s datem a časem. Pro korektní zobrazení časového údaje je potřeba jednak nastavit správný číselný formát buňky a pak také vložit správnou číselnou reprezentaci tohoto údaje. Pro datum se využívá celé číslo, pro čas číslo desetinné. Pro usnadnění práce s těmito typy údajů vytvořená třída disponuje funkcemi, které snadno vytvoří správnou hodnotu z některého z běžných zápisů data či času.

Pokud máme celý soubor vytvořen, zbývá nám jej předat prohlížeči. To lze zařídit řadou způsobů, pro ukázku jsme zvolili přímý výstup, který umožní otevření v tabulkovém procesoru, nebo jeho uložení na disk. Tento způsob vyžaduje, nějakým způsobem informovat prohlížeč o tom, že předávaný obsah má chápat jako soubor určený právě pro aplikaci Excel. To lze snadno určit vložení příslušné hlavičky, která bude odeslána serverem při http dotazu.

Znovu upozorňuji, že knihovna je navržena v kódování UTF-8 a tedy i používané skripty by měly být napsány v tomto kódování. SYLK však pracuje v ASCII kódování, a protože tohoto nejsme schopni dosáhnout, je nutné převést výsledek do nativního kódování tabulkového procesoru. Proto výstup překódujeme do znakové sady Windows-1250. Pak je obsah interpretován

správně.

Závěrem této kapitoly předvedeme stručné schéma, jak probíhá vlastní generování požadovaného SYLK souboru s použitím naší knihovny.

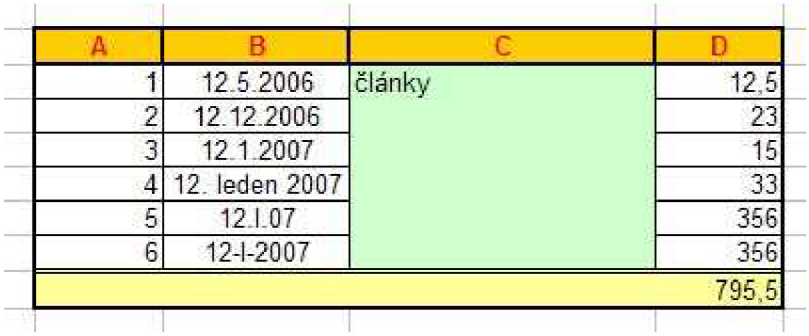


Obr. 6.3. Schéma tvorby SYLK souboru použitím knihovny c_sylk.php

6.3.3. Ukázka použití knihovny

Detailní postup tvorby skriptu pro získání dokumentu typu SYLK je uveden v ukázkách na přiloženém médiu či na webu. Zde si ukážeme pouze obecný postup, jak by se při vytvoření vzorového dokumentu postupovalo v několika krocích a předvedeme konečný výsledek. V následující kapitole porovnáme výsledek práce naší knihovny s výstupem, který pro stejné zadání vygeneroval Excel. Výpis zdrojového kódu vygenerovaného SYLK souboru Excelem, stejně jako snímek jeho reprezentace po opětovném načtení je uveden v kapitole 4.2.

Jako příklad použijeme zadání uvedené v předchozích kapitolách, abychom mohli srovnávat úspěšnost a efektivitu práce naší aplikace:



A	B	C	D
1	12.5.2006	články	12,5
2	12.12.2006		23
3	12.1.2007		15
4	12. leden 2007		33
5	12.I.07		356
6	12-I-2007		356
			795,5

Obr. 6.4. Zadání tvořené tabulky.

Nyní se pokusíme dosáhnout stejného či lepšího výsledku za pomoci naší knihovny. Budeme postupovat přesně podle schématu 6.1.:

1. Nejdříve vytvoříme PHP skript, do kterého připojíme naše třídy pro práci s formátem SYLK.
2. Dále vytvoříme instanci třídy c_sylk – proměnnou typu objekt.
3. Z obrázku je patrné že je vhodné upravit šířky sloupců B a C; nastavíme tedy jejich šířku na potřebný počet znaků, což nám ušetří nastavování těchto informací pro jednotlivé buňky. Řádky není třeba nijak zvlášť upravovat, výchozí formát také ne.
4. Vkládáme jednotlivé buňky. Vhodný postup se jeví tento: nejdříve vložíme buňky prvního řádku, který slouží jako záhlaví tabulky. Všechny buňky mají stejné formátování, které lze takto využít bez nutnosti přenastavování parametrů volaných metod. Je nutné nastavit vzhled podle možností samotného SYLKU, tedy můžeme vyžadovat pozadí těchto buněk, ale musíme počítat s tím, že SYLK použije pouze černobílý výplňový vzor.
   Další buňky je ze stejného důvodu, využití společných vlastností, vhodné vkládat po sloupcích. Vložíme tedy šest buněk prvního sloupce, který obsahuje pouze čísla. Ve sloupci B jsou různé formáty data, tedy bude nutné přenastavovat číselný formát podle požadovaného výsledku. Pozor také na reprezentaci data v tomto formátu, je nutné využít funkci, která datum převede na vhodný tvar.
   Sloupec C v originálním zadání obsahuje sloučené buňky s nastavenou barvou pozadí a krátkým textem. SYLK nemá prostředky pro dosažení takového efektu, proto pro zachování optického dojmu sloučení buněk, můžeme všechny buňky tohoto sloupce vyplnit pozadím a ohraničit je tak, aby vytvářely dojem sloučené oblasti.
   Poslední sloupec opět obsahuje čísla. V první buňce je potřebné nastavit počet desetinných míst zobrazovaného čísla na 1, jinak bude číslo zobrazeno jako celé.
   Poslední řádek by měly opět tvořit sloučené buňky. Jak bylo uvedeno u sloupce C, toto není možné, proto naformátujeme buňky tak, aby vypadaly obdobně. Obsah poslední buňky je vzorec, konkrétně součet hodnot posledního sloupce. Naše knihovna podporuje vkládání Excelovských formulí, takže jej lze bez problémů vytvořit.
5. Zbývá nám generovaný obsah odeslat na stranu klienta. Tato problematika byla rovněž zmíněna v předchozí kapitole. Stačí zavolat k tomuto účelu vytvořenou metodu, překódovat výsledek do správné znakové sady a pomocí nastavené hlavičky odeslat prohlížeči, který nabídne jeho otevření či stažení.

Pro porovnání s výstupem v kapitole 4.2. uvedeme obsah vygenerovaného souboru typu SYLK:

```
ID;PWXL;N;E
P;PGeneral
P;Pd/m/yyyy
P;P[$-F800]dddd\,\ mmmm\ dd\,\ yyyy
P;P[$-405]d/mmm/yy;;@
P;P[$-405]d\-mmm\-yyyy;;@
P;EArial;M200
P;EArial;M200;SB;L3
F;P0;DG0G8;SM0;M255
B;Y8;X4
O;L
F;SM0;W 2 2 13;C2
F;SM0;W 3 3 21;C3
F;P0;FI0C;SDLRBTSM1;Y1;X1
C;K"A"
F;P0;FI0C;SDLRBTSM1;X2
C;K"B"
F;P0;FI0C;SDLRBTSM1;X3
C;K"C"
F;P0;FI0C;SDLRBTSM1;X4
C;K"D"
F;P0;SLRBTM0;Y2;X1
C;K1
F;P0;SLRBTM0;Y3;X1
C;K2
F;P0;SLRBTM0;Y4;X1
C;K3
F;P0;SLRBTM0;Y5;X1
C;K4
F;P0;SLRBTM0;Y6;X1
C;K5
F;P0;SLRBTM0;Y7;X1
C;K6
F;P1;FG0C;SLRBTM0;Y2;X2
C;K38849
F;P1;FG0C;SLRBTM0;Y3;X2
C;K39063
F;P1;FG0C;SLRBTM0;Y4;X2
C;K39094
F;P2;FG0C;SLRBTM0;Y5;X2
C;K39094
F;P3;FG0C;SLRBTM0;Y6;X2
C;K39094
F;P4;FG0C;SLRBTM0;Y7;X2
C;K39094
F;P0;FI0L;SLRTSM0;Y2;X3
C;K"články"
F;P0;FI0L;SLRSM0;Y3;X3
C;K""
F;P0;FI0L;SLRSM0;Y4;X3
C;K""
F;P0;FI0L;SLRSM0;Y5;X3
C;K""
F;P0;FI0L;SLRSM0;Y6;X3
C;K""
F;P0;FI0L;SLRBSM0;Y7;X3
C;K""
F;P0;FF1R;SLRBTM0;Y2;X4
```

```
C;K12.5
F;P0;FF0R;SLRBTM0;Y3;X4
C;K23
F;P0;FF0R;SLRBTM0;Y4;X4
C;K15
F;P0;FF0R;SLRBTM0;Y5;X4
C;K33
F;P0;FF0R;SLRBTM0;Y6;X4
C;K356
F;P0;FF0R;SLRBTM0;Y7;X4
C;K356
F;P0;FF1R;SLBTSM0;Y8;X1
F;P0;FF1R;SBTSM0;X2
F;P0;FF1R;SBTSM0;X3
F;P0;FF1R;SRBTSM0;X4
C;K795.5;ESUM(R[-6]C:R[-1]C)
E
```

A jak vypadá výsledek v aplikaci MS Excel?

A	B	C	D
1	12.5.2006	články	12,5
2	12.12.2006		23
3	12.1.2007		15
4	12. leden 2007		33
5	12.I.07		356
6	12-I-2007		356
			795,5

Obr. 6.5. Výsledek za použití knihovny c_sylk.php.

Jak vidíme z obrázku 6.5. dokázali jsme vytvořit na pohled dokonce přesnější výstup než samotný Excel. Je to pochopitelné, protože pokud vytváříme obsah ručně, dokážeme přesně umístit buňky podle potřeby tam, aby zachovávali vzhled původního souboru. Automatické generování si musí poradit s vhodnou aproximací chybějících vlastností formátu SYLK oproti binárnímu souboru XLS, ve kterém bylo původní zadání vytvořeno. Slučování buněk řeší například ignorací sloučení, což se projeví že se obsah posledního řádku objeví úplně vlevo – v původním souboru celý řádek tvořila jediná buňka, po konverzi to bude buňka ve sloupci A. Pokud víme, že se poslední řádek vztahuje k sloupci D, dokážeme její pozici nastavit přesněji.

Potřebný PHP skript pro generování takového výsledku lze přehlednou formou včetně vysvětlujících komentářů zapsat na 186 řádků, celková velikost souboru nedosahuje ani 5 kB.

Výsledný SYLK soubor je vygenerován na 77 řádků a zabírá 1,14 kB. Srovnáme-li tyto údaje s výstupem z Excelu (127 řádků , velikost asi 2,15 kB), vidíme že se nám v tomto případě podařilo dosáhnout poměrně velké úspory místa. Tato úspora závisí na množství použitích formátovacích vlastností v buňkách. Redukce potřebného prostoru lze dosáhnout vypuštěním zbytečných záznamů. Excel vždy generuje svou standardní sadu číselných formátů, i když v souboru nebudou použity. Stejně tak jako generuje několik nadbytečných stylů pro písmo, zřejmě

kvůli problémům se čtvrtým záznamem tohoto typu, který je popsán v kapitole věnované SYLKu. Srovnejme počet *P* záznamů na začátku našeho souboru a souboru uvedeného v kapitole 4.2.

Při otevírání souboru může dojít k problému při použití prohlížeče Internet Explorer. Pokud se nepodaří soubor přímo otevřít, je nutné jej uložit na disk a po té otevřít přímo z disku.

6.3.4. Závěrečné shrnutí

Pro tvorbu dokumentu ve formátu SYLK byla vytvořena knihovna **c_sylk.php**. Jedná se o *třídu* podle objektového modelu jazyka PHP 5. Ke své funkčnosti vyžaduje ještě třídu implementující zpracování některých výjimek **c_exceptions.php**. Po připojení těchto dvou souborů a můžeme snadno napsat php skript, který zajistí generování libovolného dokumentu ve tomto formátu.

Soubor vygenerovaný pomocí knihovny *c_sylk.php* je mnohdy o poznání kratší než soubor vygenerovaný aplikací MS Excel. Knihovna ukládá pouze opravdu použité informace a negeneruje nadbytečné definice formátů obsahu či písmem. Experimenty prozatím neprokázaly, že by toto vynechání „zbytečných údajů“ mělo nějaký negativní dopad na výsledek. Excel data načte a správně zobrazí což bylo přesně naším cílem. Navíc při nasazení do webovského prostředí je každá úspora přenášeného množství dat vítána.

6.4. XML

Druhým úkolem bude vytvořit samostatný PHP skript, který umožní generování XML souborů ve správném tvaru pro tabulkový procesor MS Excel 2003. Základní principy jsou stejné jako u implementace formátu SYLK. Vytvoříme knihovnu, jejíž rozhraní lze využít pro vkládání potřebného obsahu jednotlivých vytvářených dokumentů. Způsob tvorby spočívá v definici vlastností jednotlivých částí dokumentu a vkládání potřebných buněk uvnitř každého listu.

Tyto vlastnosti a hodnoty obsahu jsou předávány pomocí parametrů metod, které následně zajistí vygenerování všech potřebných elementů XLM souboru. Pro jednoznačnost a vyšší rychlost zpracovaní je tvar nastavovaných vlastností stanoven pevně a je potřeba jej nastudovat z dokumentace či návodů k projektu.

Rovněž budeme vycházet z objektově orientovaného návrhu, proto vytvoříme *třídu*, která je základem pro odvozování objektů. Třída v sobě zahrnuje veškeré informace o vlastnostech a metodách objektů. Naše výsledná knihovna je vlastně rozsáhlá třída v jazyku PHP 5, která v sobě zapouzdřuje veškeré potřebné nástroje pro snadné generování našich dokumentů.

6.4.1. Popis knihovny c_xml.php

Vytvořená knihovna je pojmenována c_xml.php, aby název vypovídal o jejím účelu. Prefix *c_* pro rozlišení, že se jedná o třídu a *xml* pak proto, že je cíleně zaměřena na generování XML dokumentů.

V této kapitole nebudou popsány podrobné detaily o způsobu zápisu ani významu zdrojového kódu. Tyto informace jsou součástí dokumentace k vytvořenému projektu. Dokumentace je k dispozici na přiloženém CD-ROMu nebo na webovských stránkách projektu: http://zap.profitux.cz/.

Tato knihovna je téměř samostatný celek, který dokáže vytvořit korektní XML soubor, aby jej správně interpretovat tabulkový procesor MS Excel 2003. Je nutné pouze dodržovat předem určený způsob předávání informací o požadovaném obsahu pro zpracování touto knihovnou.

Vesměs se jedná o nastavování parametrů metodám této třídy. Pomocí nich pak postupně tvoříme obsah dokumentu v jeho objektové reprezentaci uvnitř skriptu. Knihovna ke své funkčnosti vyžaduje ještě třídu *c_exceptions.php*, která je rovněž součástí projektu. Tato třída byla navržena pro implementaci zpracování *výjimek*. V konečném důsledku se jedná pouze o výpis chybového hlášení, které upozorní autora generujících skriptů na závažné chyby, jejichž důsledkem nelze vytvořit požadovaný soubor. Součástí většiny těchto zpráv je podrobný výpis v jaké části a z jakého důvodu došlo k výjimce. U většiny je také uveden možný způsob nápravy. Pokud nebudou všechny způsobené chyby odstraněny, nebude vytvořený dokument správně načten tabulkovým procesorem.

Pro plné pochopení funkčnosti vytvořené knihovny je nutné detailnější seznámení se s formátem XML pro Excel 2003 popsaného např. v příloze B. Po seznámení se s tímto formátem lze knihovnu snadno upravovat případně rozšiřovat pro implementaci více možností podporovaných Excelem. Tento formát je již důkladně propracovaný a umožní tak uchovat v textové podobě téměř všechny důležité Excelovské funkcionality. Naše knihovna umožňuje pouze některé často využívané možnosti. Zejména komplexní nastavení stylů přenášeného obsahu včetně všemožných rámečků, barev písma i pozadí, dále vkládání vzorců, definici pojmenovaných oblastí a také tvorbu více listů v sešitu. Možnosti tohoto formátu jsou omezené a neumožňují přenos externích objektů a obrázků.

Knihovna je vytvořena především za účelem tvorby dokumentů bez nutné znalosti jeho vnitřní reprezentace. Po nastudování návodů a pochopení rozhraní této knihovny je tvorba XML dokumentů snadnou záležitostí.

6.3.4. Práce s knihovnou c_xml.php

Máme-li k dispozici obě potřebné knihovny (c_xml.php a c_exceptions.php), můžeme se pustit do tvorby skriptu, který nám vygeneruje soubor pro MS Excel. Knihovny sami o sobě nedokáží vytvořit žádný soubor, je nutné je jejich schopnosti využít v samostatném, cíleně vytvořeném skriptu, který tvoříme pro konkrétní typ dokumentu. Každý dokument bude obsahovat jiné informace a bude mít jiný vzhled, proto je nutné nastavit potřebné vlastnosti tak, aby výsledek vyhovoval vstupnímu zadání.

Podrobnější detaily o tvorbě takového skriptu jsou rovněž součástí dokumentace a návodů. Ukázka kompletního skriptu je uvedena na přiloženém médiu či na webu. V této části si uvedeme pouze nejdůležitější kroky, které je třeba dodržet, abychom dosáhli správného výsledku. Podotkněme, že průběh skriptu musí být bez problémů, protože Excel reaguje velmi nenačte nekorektní XML soubor. Při výskytu chyby generuje chybové hlášení při otvírání souboru, které uživateli sdělí cestu k souboru, obsahující podrobnější zachycení chyby, na jehož základě lze snadno detekovat a opravit chybu v generujícím skriptu.

Tvorbu generujícího skriptu lze shrnout do devíti kroků, které budou uvedeny společně s jejich obrazem v PHP kódu**:**

1. Do skriptu připojíme obě potřebné khihovny
   ```
   include_once('c_exceptions.php'); include_once('c_xml.php');
   ```
2. Vytvoříme si instanci třídy, tedy objekt reprezentující tvořený dokument se kterým budeme pracovat:
   ```
   $xml = new c_xml();
   ```
3. Pokud je potřeba, nastavíme vlastnosti dokumentu
   ```
   $xml->setDocumentProperties(...)
   ```
4. Pokud je potřeba nastavíme vlastnosti pracovního sešitu
   ```
   $xml->setExcelWorkbook(...);
   ```

5. Vytvoříme jednotlivé listy sešitu, tato část je nejrozsáhlejší, zahrnuje veškeré generování buněk jednotlivých listů. Vytvoření listu se skládá z těchto kroků:
   I. Obsah každého listu je interně reprezentován Excelem jako tabulka. Vytvoříme tedy tabulku
   ```
   $xml->setTable(...);
   ```
   II. Podle potřeby naformátujeme sloupce
   ```
   $xml->addColToTable (...);
   ```
   III. Vkládáme obsah buněk. Je nutno vkládat jej po řádcích. Vytvoříme tedy element řádku
   ```
   $xml->addRowToTable(...);
   ```
   a potřebné buňky uvnitř řádku
   ```
   $xml->addCellToTable (...);
   ```
   a definici řádku ukončíme
   ```
   $xml->addRowEndToTable();
   ```
   IV. Nastavíme potřebné vlastnosti listu
   ```
   $xml->setWorksheetOptions(...);
   ```
   V. Celý list přidáme do obsahu sešitu. Každý list má svůj název.
   ```
   xml->setWorksheet("Nazev");
   ```

6. Zajistíme vygenerování stylů buněk použitých v dokumentu
   ```
   $xml->setStyles();
   ```
7. Spojíme jednotlivé části pracovního sešitu ve správném pořadí.
   ```
   $xml->setWorkbook();
   ```
8. Sestavíme kompletní XML dokument spojením všech částí ve správném pořadí
   ```
   $xml->setXml();
   ```
9. Nabídneme soubor ke stažení.
   ```
   header("Content-Type: application/vnd.ms-excel");
   header("Content-Disposition: attachment;filename=$filename");
   echo $xml->getXml();
   ```

Schéma 6.2. Postup tvorby XML dokumentu s využitím knihovny c_xml.php

Myšlenkou tvorby dokumentu je sledovat strukturu XML souboru, který vygeneruje Excel. Ten logicky sestavuje jednotlivé části ve vhodném pořadí. Vysvětlíme si význam některých kroků a upřesníme způsob předávání opravdových dat do vytvářeného dokumentu. Ze schématu 6.2. je patrné, že veškerá činnost je zajištěna prostřednictvím metod naší knihovny. Data se tedy musí předávat dovnitř metod skrze jejich parametry. Parametry lze chápat jako proměnné, které v sobě uchovají námi požadovaný údaj, který se projeví v buňce výsledného dokumentu.

Nejdůležitější je pro nás tvorba právě těchto parametrů. Ukážeme si tedy možnosti, jakým způsobem lze předávat metody potřebné informace, aby je mohla převést na části xml dokumentu.

Ve většině případů je uložení relevantních hodnot pro tvořenou část realizováno použitím asociativního pole. Asociativní pole je standardní vnitřní datový typ jazyka. Umožňuje uložit kolekci různých druhů informací, přístupných pod jedinečnými názvy klíčů (indexů pole). Nyní stačí znát, jaké typy klíčů a jejich hodnot vyžadují jednotlivé nastavované části. V knihovně jsou tyto hodnoty voleny záměrně tak, aby co nejblíže vystihovali účel jejich použití. Pro tvorbu názvů je využito anglického jazyka, protože jím lze stručně a jednoznačně vystihnout požadovaný pojem. Pro většinu programátorů jsou anglické termíny zažité a zcela přirozené, proto je právě tento jazyk

volen jako nejschůdnější cesta.

Nastavování vlastností dokumentu, pracovního sešitu, listu apod. je zajištěno příslušnými metodami, které jako parametry očekávají právě asociativní pole. Možné klíče každého pole je nutno vyčíst z dokumentace, obvykle ale vhodně reprezentují nastavovanou vlastnosti. Pro ukázku si uvedeme některé z nich jako *Author* pro nastavení jména autora dokumentu, *Title* pro nastavení titulku dokumentu, *HideHorizontalScrollBar* pro skrytí vodorovného posuvníku nebo *DoNotDisplayGridlines* pro skrytí mřížky. Na rozdíl od klíčů u formátů SYLK sledují klíče výslednou reprezentaci v XML dokumentu, který je case-sensitive, tedy rozlišuje velikosti písmen. Je nutno dodržovat přesný zápis uvedený v dokumentaci. Základní pravidlo je však jednoduché – žádné mezery a velké písmenko každého obsaženého slova. Na příkladu klíče *Author* vidíme, že název se skládá pouze z jednoho slova, které tedy začíná velkým písmenkem. U pojmenování, které vzniká z více slov jako např. *Hide horizontal scroll bar*, vypustíme mezery a místo nich slova viditelně oddělíme velkými písmenky.

Proč je použito i takto dlouhých klíčů? Jak jsme zmínili v předchozích částech, XML formát byl navržen opravdu velmi rozumným a logickým způsobem. Klíče jasně vypovídají o svém významu. Jak se říká, je zbytečné znovu objevovat kolo, a proto tam, kde je to vhodné je zachováno pojmenování převzaté přímo z XML elementů namísto vymýšlení kratších variant. Se znalostí anglického jazyka jsou tyto hodnoty snadno zapamatovatelné.

Nicméně pro tvorbu parametrů v některých metodách jsou navrženy vlastní názvy klíčů, které jsou kratší a snaží se stejně tak vystihnout svůj účel.

Možné hodnoty, které lze přiřadit těmto klíčům je také nutno vyčíst z dokumentace knihoven. Některé očekávají zadání logické hodnoty, jiné textového řetězce či čísla. Ve složitějších případech je nutno předat pole nastavovaných vlastností, které zápis mírně zpřehlední.

Po nastavení základních vlastností vytvoříme samotné listy sešitu. Tato část je zřejmě nejrozsáhlejší a nenáročnější z celé tvorby dokumentu. Každý list je Excelem chápán jako tabulka. Její buňky jsou vykresleny šedou mřížkou v dokumentu, pokud není skryta či nemá nastavenu jinou barvu. Každý list obsahuje jedinou tabulku listu, jejíž buňky můžeme nastavit podle potřeby. Pokud si v listu vytvoříme naši tabulku, kterou chápeme jako jisté ohraničené buňky obsahující související data, nemá *nic* společného s tabulkou listu. Takovýchto tabulek můžeme v listu vytvořit libovolný počet.

Tabulka listu je tvořena řádky a sloupci se záhlavím typicky pro řádky číselným, pro sloupce tvořeným velkými písmeny. Pokud je to nutné, nastavíme výchozí vlastnosti některým sloupcům. Jedná se většinou o zvětšení šířky sloupce, případně přidáme výchozí formátování. Při vkládání obsahu je nutno postupovat po jednotlivých řádcích. Toto je menší omezení oproti formátu SYLK, kde bylo možné přistupovat k libovolným buňkám listu. V XML můžeme vždy definovat pouze buňky v kontextu aktuálního řádku se kterým pracujeme. Aby knihovna rozpoznala, kdy má ukončit definici buněk v řádku je nutné tento řádek vymezit voláním příslušných metod – na začátku pro vytvoření elementu řádku, a na konci pro jeho ukončení. Vychází to z podstaty samotného formátu, který vyžaduje párové elementy. Řádek je tedy element, který obklopuje elementy jednotlivých buněk a jejich obsahů.

Definice obsahu buněk je zde také mírně komplikovanější než u předchozího formátu. Vložení všech informací sice obstará metoda s jediným parametrem, ale tento parametr je asociativní pole nesoucí několik důležitých částí. Jednou je definice různých atributů a nastavení požadovaného vzhledu buňky. To bude vysvětleno záhy. Druhou významnou částí je definice obsahu buňky. V každé buňce se mohou vyskytnou tři ty typy obsahu – vlastní hodnota, komentář buňky a určení pojmenované oblasti k níž buňka náleží. Přestože jsou možnosti nastavení poněkud košaté, není nutné všechny využívat. Záleží pouze na požadovaném výsledku.

Nastavování vzhledu buněk a formátu obsahu je řešeno uvnitř XML podobně jako u

formátu SYLK. Excel vyžaduje definici jistých elementů pro styly buněk, na které se z jednotlivých buněk odkazuje. Protože předem nevíme, jaké definice budou zapotřebí, je jejich tvorba vyřešena současně s definicí obsahu buňky. Požadujeme-li nějaké specifické nastavené vzhledu, vytvoříme pomocí metody pro tvorbu stylu tento element a jeho identifikátor předáme jako atributy tvořené buňky. Pro vytvoření stylu byly navrženy dvě metody.

Hlavní z nich je použita vždy a má za úkol převést definice uživatele na vhodný tvar pro zpracování naší knihovnou. Použití vstupního formátu této knihovny je složitější a vyžaduje důkladné nastudování možností a způsobu jejich zápisu přímo z dokumentace či komentářů zdrojových kódů. Na druhou stranu umožňuje nastavit veškeré vlastnosti, které naše knihovna podporuje. Využívá systému označení názvů vlastností a hodnot vhodně zvolenými písmeny či zkratkami.
Pro ukázku uvedeme například:

```
$sid = $xml->addStyle("A:VLB;HZC;".
                      "~B:PTB;LSC;WT2;".
                      "~B:PTL;LSC;WT2;".
                      "~B:PTR;LSC;WT2;".
                      "~B:PTT;LSC;WT2;".
                      "~F:CH238;FMW;CL#993300;BD1;".
                      "~I:CL#FFCC00;PTS;"
                      );
```

Význam tohoto zápisu stylu je následující: Metoda *addStyle* vytvoří na základě vstupního řetězce požadovaný element pro styl a vrátí jeho identifikátor do proměnné *sid*. Parametr pro přehlednost rozdělen na více řádků definuje jednotlivé formátovací vlastnosti. Detailněji rozepíšeme pouze první řádek, kde počáteční *A:* označuje zarovnání obsahu, z anglického slova *alignment*. Hodnoty oddělené středníky jsou definice jednotlivých nastavení pro zarovnání. Znak tildy uvozuje novou nastavovanou vlastnost. *VLB;* určuje vertikální zarovnání dolů (*vertical, bottom*), *HZC;* zase horizontální zarovnání na střed (*horizontal, center*).

Druhý řádek nastavuje ohraničení buňky nahoře, nastavuje souvislou čáru šířky 2 px. Ostatní řádky jsou obdobné.

Tento způsob zápisu je možná z počátku nejasný, ale při jeho použití bude výsledný běh skriptu rychlejší. Zejména z toho důvodu, že druhý způsob zápisu je vytvořen záměrně pro uživatelsky snazší zápis požadavků, ale jeho činnost spočívá v převodu uživatelského vstupu právě do výše zmíněného formátu, který je interně použit pro samotnou definici elementů.

Druhá metoda využívá jako parametr asociativní pole, kde jsou jednotlivé klíče opět voleny tak, aby vystihovaly význam. Tedy obvykle jsou tvořeny zkratkami anglických slov popisující vlastnosti. Pro zkrácení nutného zápisu je rovněž v této metodě implementováno hromadné nastavení některých společných vlastností. Pro srovnání uveďme definici ohraničení předchozím způsobem, kde je potřeba pro každou stranu buňky uvést jednu definici, a způsob zápisu uvedený níže. Uvedená ukázka má zcela stejný význam jako předchozí, proto ji ponecháme bez dalšího komentáře.

```
$sid = $xml->cellStyle(
                      array("align" => "center",
                            "valign" => "bottom",
```

```
                                "borders" => "solid;2;",
                                "font" => "b;#993300;",
                                "fontFamily" => "Arial;swiss;238",
                                "bg" => "#FFCC00"
                               ));
```

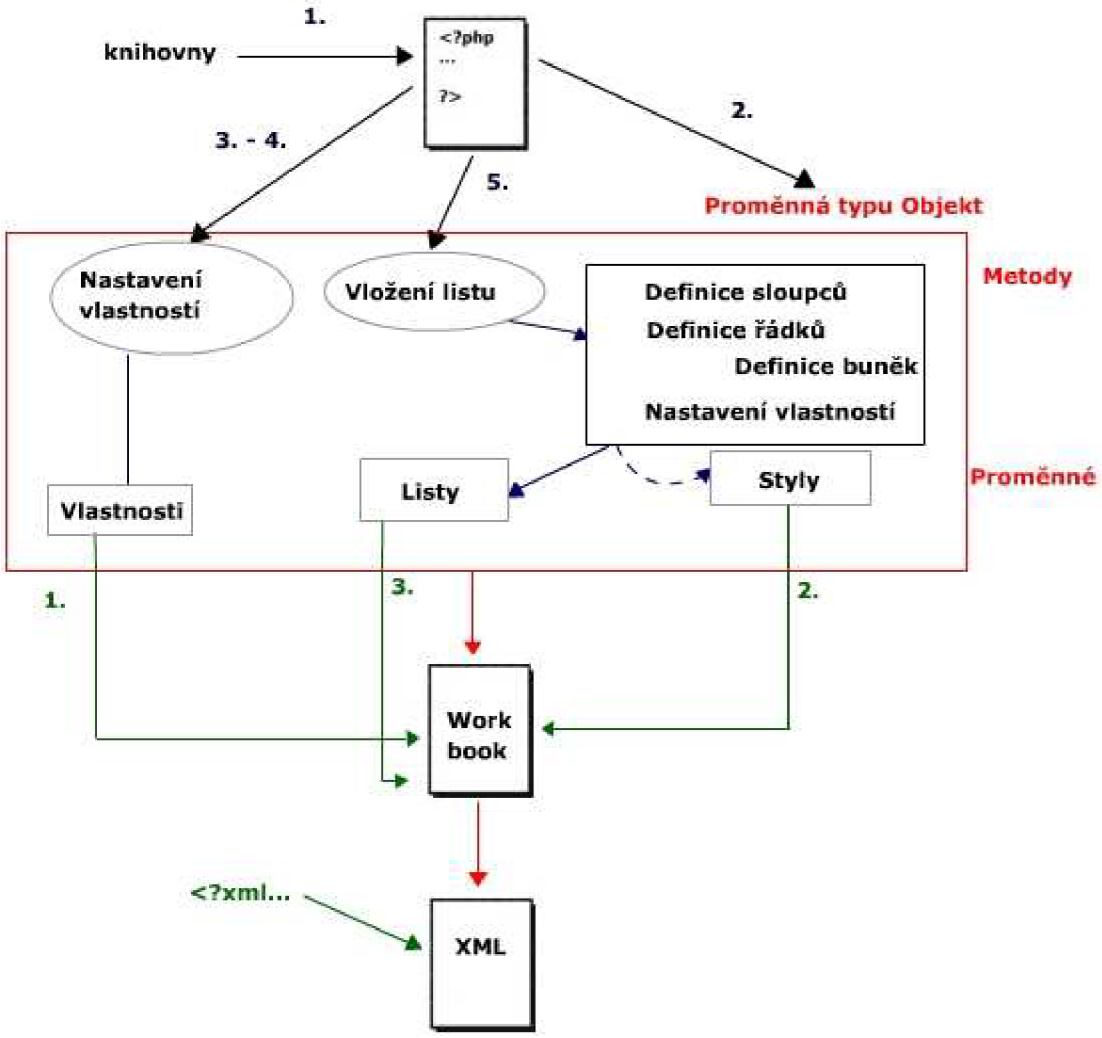
Záleží pouze na našem vkusu, kterou z uvedených možností budeme používat. V ukázkových příkladech jsou uvedeny obě dvě možnosti. V dnešní době je získaná výhody prvního způsobu docela nevýrazná, proto pokud by nám činilo potíže vypořádat se s navrženým způsobem zápisu, můžeme s klidným svědomím používat způsob druhý.

Po dokončení definice obsahu listu a nastavení případných vlastností musíme knihovnu upozornit, že má veškerý doposud vytvořený obsah zařadit do konkrétního listu. V Excelu musí mít každý list své jméno. To je viditelné jako pojmenován záložek v dolní části okna aplikace, kde výchozí názvy pro českou verzi tohoto procesoru jsou List 1, List 2 atd. Pokud nevyžadujeme specifické pojmenování knihovna po volání metody pro definici listu automaticky doplní název List a správné pořadové číslo.

Pokud máme vytvořeny všechny potřebné listy, musíme knihovně předat informaci o tom, že má vygenerovat správnou posloupnost elementů pro použité styly tak, aby se na ně v dokumentu mohlo odkazovat. Styly se sice vytváří průběžně, ale jsou uchovávány v jiné části, aby bylo možné snadno ošetřit existenci stejných stylů a zamezilo se zbytečnému vytváření duplicitních stylů. Tuto činnost nám zajistí volání jediné metody. Jedinou podmínkou je pouze nevynechat její zápis, jinak se výsledku neobjeví žádné definice elementů *Style* a tím pádem budou odkazy buněk neplatné. Takový dokument nebude Excelem načten a způsobí generování chybového hlášení při otvírání souboru.

Na závěr necháme sestavit výsledný dokument pracovního sešitu, který bude kompletně uzavřen v XML elementech *Worbook*. Následné volání pro vytvoření xml souboru opatří vytvořený obsah správnými hlavičkami a je možné výstup odeslat uživateli. Správnou interpretaci odesílaného souboru klientským prohlížečem je možné ovlivnit hlavičkami, které mu server odesílá. Pokud přednastavíme, že typ obsahu je určen pro aplikaci MS Excel, bude uživateli nabídnuta možnost otevření tohoto souboru přímo v této aplikaci, pokud ji má k dispozici. Druhou možností je uložit si výsledný soubor přímo k sobě na disk a následně otevřít podle potřeby. V některých prohlížečích jako např. Internet Explorer se může objevit problém s přímým otevřením souboru a jedinou možností je právě mezikrok uložení souboru na disk.

Závěrem této kapitoly uvedeme stručné schéma, jak probíhá vlastní generování požadovaného XML souboru s použitím naší knihovny. Toto schéma nebude zachycovat všechny možnosti, protože by bylo hodně rozsáhlé.



Obr. 6.6. Schéma tvorby XML souboru použitím knihovny c_sylk.php

6.4.3. Ukázka použití knihovny

Konkrétní ukázku skriptu, který zajistí vygenerování XML dokumentu je možné shlédnout na přiloženém médiu. Tato kapitola pouze obecně naznačí možné kroky, které je nutné projít, abychom dosáhli požadovaného cíle.

Jako ukázkový příklad použijeme opět stejné zadání které je uvedeno ve všech předchozích kapitolách. Zejména z důvodu srovnání úspěšnosti a efektivity práce naší aplikace:

A	B	C	D
1	12.5.2006	články	12,5
2	12.12.2006		23
3	12.1.2007		15
4	12. leden 2007		33
5	12.I.07		356
6	12-I-2007		356
			795,5

Obr. 6.7. Zadání tvořené tabulky.

Nyní se pokusíme dosáhnout stejného výsledku za pomoci naší knihovny. Budeme postupovat podle schématu 6.2.:

1. Nejdříve vytvoříme PHP skript, do kterého připojíme naše třídy pro práci s formátem XML.
2. Dále vytvoříme instanci třídy c_xml – proměnnou typu objekt.
3. Je vhodné uvádět některé základní vlastnosti dokumentu jako autor, klíčová obsah nebo stručný popis. Tento krok však není nepovinnný.
4. Specifické nastavení pracovního sešitu není třeba.
5. Nejdůležitější bude tedy naplnění obsahu. Budeme tvořit pouze jeden list pracovního sešitu Excelu. Inicializujeme nejdříve prázdnou tabulku, do níž postupně vložíme požadované buňky. Z obrázku je patrné že je vhodné upravit šířky sloupců B a C; nastavíme tedy jejich šířky. Dále budeme vkládat jednotlivé řádky.

   Protože tento formát vyžaduje práci po řádcích, budeme tedy postupovat shora dolů. První řádek bude mít stejné nastavení formátu buněk. Nastavíme pomocí vybrané metody veškeré požadované formátování buňky – tučné červené písmo, oranžové pozadí, tučné ohraničení, zarovnání textu na střed – a získáme identifikátor tohoto stylu. Dále pouze vložíme čtyři buňky s odkazem na právě vytvořený styl. Ukončíme první řádek a můžeme pokračovat.

   Ostatní řádky se tvoří obdobně. Bohužel však nebudeme moci využít postupného vkládání buněk stejného formátování, a bude nutné postupně vytvářet vhodný styl pro konkrétní buňku. Pro zkrácení zdrojového textu lze použít různé techniky, např. uložení si všech použitých stylů a následné předávání správného identifikátoru. Toto je otázka konkrétní implementace.

   Mírně odlišný způsob zpracování je u třetího sloupce. Ten obsahuje sloučené buňky. Proto se obsah této buňky bude definovat pouze jednou a to v definici prvního řádku hodnot (tedy skutečného druhého řádku tabulky). V ostatních řádcích je nutné tuto buňku vynechat a pomocí čísla vkládané buňky přímo vložit hodnotu do 4. sloupce.

   Obdobný postup platí pro poslední řádek. Sloučíme buňky a jako jejich obsah vložíme výpočet hodnoty pomocí vzorce. Poslední řádek tedy bude obsahovat definici jediné buňky. Vkládání vzorců funguje stejně jako u formátu SYLK a odráží použití vzorců v samotném Excelu. Lze využít jak relativního tak absolutního určení buněk, podílejících se na tvorbě výsledné hodnoty.

   Máme již nastaven kompletní požadovaný obsah, žádné zvláštní vlastnosti pracovního listu nejsou potřeba proto můžeme přejít k dalšímu bodu.

6. Je nutné zajistit vytvoření stylů. Jednotlivé styly se vytvářejí podle potřeby při

vkládání obsahu do buněk, ale jsou uloženy takovým způsobem, aby knihovna snadno mohla rozpoznat jednotlivé vytvořené styly a použít jejich identifikátory. Předejte se tak zbytečnému duplicitnímu vytváření shodných stylů buněk. Tento formát uložení však není použitelný pro obsah tvořeného XML dokumentu. Proto je nutno po dokončení definic všech stylů zavolat metodu, která zajistí konverzi do správného tvaru.

7. Jedním z posledních kroků je vytvoření kompletního pracovního sešitu. Metoda zajišťující vykonání tohoto požadavku má za úkol projít jednotlivě uložené části dokumentu a jednotlivé vytvořené listy a spojit je ve správném pořadí tak, aby souhlasily s definicí XML formátu Excelu.
8. Prostým zavoláním metody pro vytvoření kompletního XML dokumentu se přidají potřebné hlavičky souboru a celý obsah sešitu vytvořeného v předchozím bodě se převede do struktury xml souboru.
9. Tento soubor můžeme nabídnout ke stažení. Poskytneme vhodné http hlavičky klientskému prohlížeči, které budou předávaný soubor charakterizovat jako soubor pro aplikaci MS Excel. Pokud proběhne vše v pořádku a uživatel bude mít na svém počítači tento tabulkový procesor k dispozici, bude nabídnuta volba otevření souboru přímo v tomto programu. Druhou a mnohdy spolehlivější nabídkou je uložení souboru na disk. Po té lze bezproblémově otevřít ve vhodném programu za předpokladu, že jeho vytváření proběhlo bez problémů.

Pro porovnání s výstupem v kapitole 5.3. je nutné vyzkoušet použití knihovny buď přímo na http://zap.profitux.cz/ nebo pomocí přiložených zdrojových souborů na CD-ROMu. Kompletní obsah vygenerovaného souboru nebudeme uvádět, protože je víceméně podobný souboru generovanému Excelem a pak je zbytečně rozsáhlý. Oproti vygenerovanému formátu SYLK je několikanásobně delší.

Výsledek po načtení do MS Excel 2003:

A	B	C	D
1	12.5.2006	články	12,5
2	12.12.2006		23
3	12.1.2007		15
4	12. leden 2007		33
5	12.I.07		356
6	12-I-2007		356
			795,5

Obr. 6.8. Výsledek za použití knihovny c_xml.php.

Podle očekávání se nárůst délky zdrojového XML kódu projevil v grafické reprezentaci v tabulkovém procesoru. Výsledek je na pohled k nerozeznání od zadání. Jak je vidět, tento formát dokáže zachovat téměř veškeré informace o vzhledu. Mezi nedostatky nadále patří nemožnost přenášení grafů, obrázků a jiných objektů.

Potřebný PHP skript pro generování našeho výsledku lze přehlednou formou včetně vysvětlujících komentářů zapsat na 570 řádků, celková velikost souboru zabírá okolo 14 kB.

Výsledný XML soubor je vygenerován na 212 řádků a zabírá 7,72 kB. Srovnáme-li tyto údaje s výstupem z Excelu (232 řádků , velikost asi 10 kB), vidíme že se nám v tomto případě

podařilo dosáhnout taktéž úspory místa. Úspora vzniká zejména vynecháním některých nejasných nastavení, která generuje samotný Excel a podle testování nemají žádný zásadní vliv na výsledek.

Při otvírání souboru může dojít k problému při použití prohlížeče Internet Explorer. Pokud se nepodaří soubor přímo otevřít, je nutné jej uložit na disk a po té otevřít přímo z disku.

6.4.4. Závěrečné shrnutí

Pro tvorbu dokumentu ve formátu XML byla vytvořena knihovna **c_xml.php**. Jedná se o *třídu* podle objektového modelu jazyka PHP 5. Ke své funkčnosti vyžaduje ještě třídu implementující zpracování některých výjimek **c_exceptions.php**. Po připojení těchto dvou souborů a můžeme snadno napsat php skript, který zajistí generování libovolného dokumentu ve tomto formátu.

Soubor vygenerovaný pomocí knihovny *c_xml.php* je kratší oproti souboru vygenerovanému aplikací MS Excel. Je to díky vynechání nejasných a zřejmě zbytečných nastavení, které nikterak neovlivní výsledný dokument.

6.5. Sylk nebo XML?

Máme k dispozici dvě knihovny, které nám umožní přenést data z datového úložiště na serveru do tabulkového procesoru prostřednictvím souboru vybraného typu. Otázkou který z nich je vhodnější nelze asi jednoznačně zodpovědět.

Z předchozích kapitol je vidět, že tvorba formátu SYLK je jednodušší, skripty jsou kratší ale výsledek je také o poznání horší co se týče nastavení vzhledu. Nefunguje ani slučování buněk, které se v dokumentech může objevit docela často.

Vytvoření souboru XML vyžaduje o něco větší úsilí. Zejména proto, že umožňuje zachovat veškerý vzhled dokumentu, má k dispozici více možných nastavení, což způsobí nárůst zdrojového kódu. Výsledek je však v mnohých případech nerozeznatelný od originálu uloženém v binárním formátu.

Obě knihovny využívají obdobný způsob definice obsahu. Nejdříve je nutné nastavit vzhled a formát buněk a následně připojit spolu s daty uloženými v buňce do kontextu pracovního listu. Pro prozkoumání všech možností je nutné projít si dokumentace ke knihovnám či alespoň některé ukázkové příklady.

Výběr knihovny závisí zřejmě na požadavcích uživatelů a náročnosti vytvářeného obsahu. Pokud nepotřebujeme žádné zvláštní formátování obsahu a vystačíme s jediným listem, lze bez problémů použít soubor typu SYLK. V opačném případě je lepší sáhnout po knihovno c_xml.php.

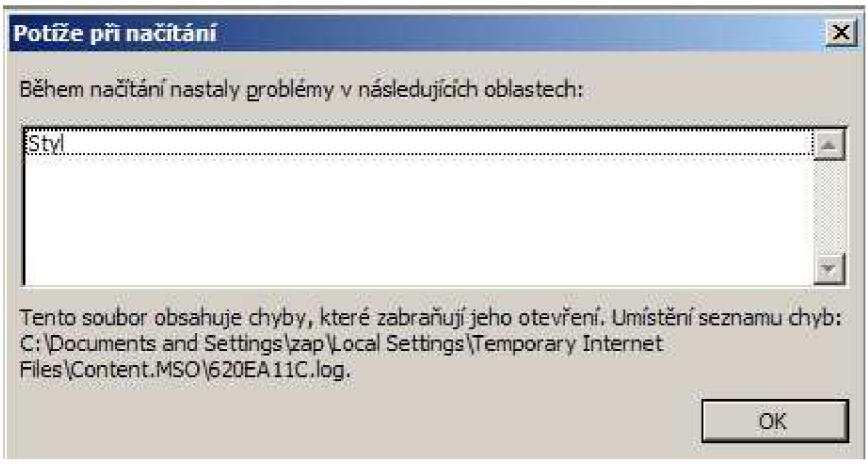
Spíše se ale přikláníme k používání formátu XML, který má velký potenciál pro budoucí použití. Data uchovaná v tomto formátu by měla být poměrně snadno převoditelná do jiného tvaru a budou tak snadno využita i na jiných místech. Podpora tohoto formátu však není implementována ve starších verzích Excelu. Funkčnost byla otestována na verzi 2003, kde funguje bezchybně. Na verzi 2000 tento formát není ještě implementován, tedy jedinou možností ve starších verzích je opět použít SYLK.

S formátem XML pro Excel se také snadněji pracuje již z důvodů kontroly chyb. Pokud nastane chyba uvnitř souboru typu SYLK, Excel zahlásí chybu a číslo záznamu. Přičemž však není zřejmé o jakou chybu se jedná a pro opravu je nutné znát podrobnosti tohoto formátu. Pokud se nejedná o příliš závažné chyby, Excel načte dostupnou část pracovního listu.

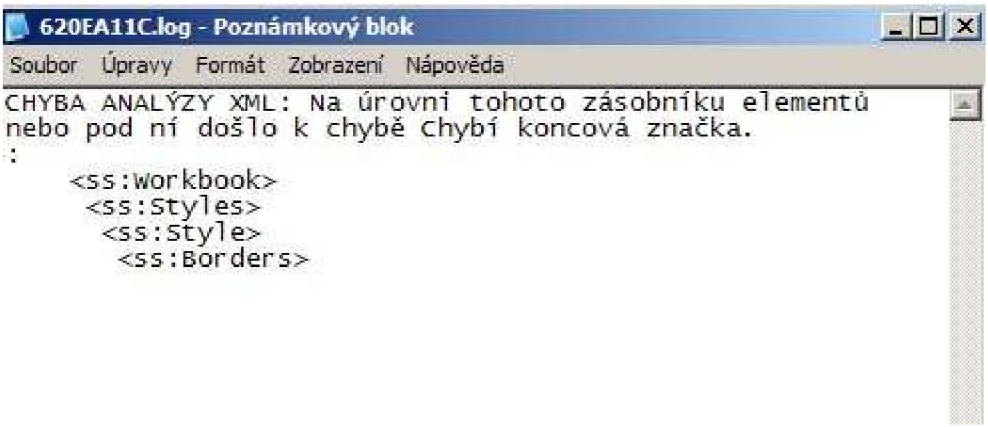
Obr. 6.9. Chybové hlášení při otvírání nekorektního souboru SYLK

Obr. 6.10. Souhrnné chybové hlášení při otvírání nekorektního souboru SYLK

Při otvírání chybného XML dokumentu je však vygenerován chybový soubor s detailnějším popisem a kontextu elementů, v němž k chybě došlo. Soubor je uložen v některém systémovém souboru, a cesta je vypsána při chybovém hlášení Excelu. Takovýto dokument není nenačten. Většinou se otevře pouze prázdný pracovní list.



Obr. 6.11. Chybové hlášení při otvírání nekorektního XML souboru



Obr. 6.12. Vygenerovaný soubor s podrobnostmi o vzniklé chybě

6.6. OpenOffice.org Calc

Knihovny byly primárně vytvořeny pro práci v tabulkovém procesoru MS Excel 2003. Vygenerované soubory však fungují správně i v tabulkovém procesoru OpenOffice.org 2.1. Calc.

V tomto procesoru však nefunguje vše úplně stejně, ale při *běžném* formátování obsahu lze předpokládat přenositelnost mezi oběma tabulkovými procesory.

6.6.1. SYLK

Nefunguje formátování vzhledu. Nezobrazují se nastavené barvy, rámečky, pozadí ani vzorce. Výsledek se zobrazí správně rozložen v buňkách listu bez jakéhokoliv formátování. Většina datových typů hodnot je interpretována správně.

Datum a čas je naformátován také správně. Ale některé složitější Excelovské formáty nejsou podporovány. Při běžném použití jednoduchých formátů nenastává žádný problém (Den. Měsíc. Rok).

Místo vzorců (např. pro součet hodnot) není zobrazeno nic. Vzorce nejsou zřejmě přenositelné. Proto **je vhodné při používání vzorců vyplňovat i výchozí hodnoty buněk** (pokud je dovedeme předpočítat).

Dalším problémem je nastavení pojmenovaných oblastí. V SYLKu nejsou žádné oblasti zachovány.

Podotýkám že tento formát není obecně podporován tímto tabulkovým procesorem. Při uložení obsahu ve formátu SYLK přímo v tomto tabulkovém procesoru dojde rovněž ke ztrátě formátování.

6.6.2. XML

Rozdíl oproti Excelu je v aplikaci vzorců s relativním zadáním hranic ve sloučených buňkách. Pokud do sloučené buňky ve sloupci 5 zadáme vzorec typu `SUM(R[-6]C:R[-1]C)` pak je chybně aplikován na sloupec 6. Tedy nastane posun. Možným řešením je buď nepoužívat slučování buněk nebo používat přesné pozice sloupců `SUM(R[-6]C5:R[-1]C5)`. Tím je tento problém vyřešen.

OpenOffice.org také nerozumí některým formátům Excelu. Například při použití složitých formátů pro datum (*[$-F800]dddd\,\ mmmm\ dd\,\ yyyy*), nemusí být hodnota zobrazena.

Pojmenované oblasti jsou částečně interpretovány, pod podmínkou, že oblast je souvislá. Neumožňuje načíst pojmenovanou oblast, která je tvořena více různými úseky, což však není nikterak typické a tedy zřejmě nezpůsobí příliš problémů.

Ostatní vlastnosti jsou zachovány.

7. Závěr

Cílem této práce bylo hlubší seznámení se s možnostmi přenosu dat z portálových aplikací do tabulkových procesorů. Vycházíme z předpokladu, že data jsou uložena v relačních databázích, nebo je na straně serveru můžeme získat v rozumném jiném tvaru. Při použití složitějších datových struktur a operací OLAPu existují v takovýchto systémech možnosti pro transformaci multidimenzionálních krychlí do dvourozměrných tabulek, které již lze vhodnou formou reprezentovat na zobrazovacích zařízeních či tisknout. Toto je velmi důležitý předpoklad, protože nám umožní zaměřit se pouze na tvorbu požadovaných souborů bez nutnosti zabývat se převodem dat do nějakého uniformního vstupního formátu pro naši aplikaci.

Jako přenosové formáty byly použity formáty SYLK a XML, které jsou podporovány aplikací MS Excel. Jejich hlavní předností je, že se jedná o textové formáty a jsou snadno generovatelné na straně serveru za pomocí běžně dostupných nástrojů. Úkolem tedy bylo nastudovat detaily těchto formátů, jejich zobrazovací schopnosti a vytvořit nástroj pro tvorbu takových dokumentů.

SYLK je prostý formát, který nepodporuje zdaleka tolik vlastností jako formát XML. Jeho zobrazovací schopnosti postačí pro jednoduché dokumenty, které nepotřebují kombinovat barevnost buněk, jejich slučování či různé druhy orámování. Tomu odpovídá i kratší kód pro generování tohoto souboru. Stejně tak vytvořený SYLK souboru bude kratší.

XML umožňuje přenášet v podstatě veškeré vzhledové vlastnosti dokumentu pro MS Excel. Umožňuje vytvořit kompletní pracovní sešity pro tuto aplikaci, nastavovat všechny podporované datové typy a formáty buněk. Potřebný zdrojový kód takovéto aplikace je pak poněkud delší než při generování předešlého formátu, výsledek je však mnohem přesnější a vhodnější zejména pro prezentaci.

Nevýhodou textových formátů je nemožnost přenést grafy, obrázky a různé další objekty. Bylo by to možné pouze pokud bychom uměli generovat binární soubor pro MS Excel. Pro webovské aplikace je však mnohem důležitější použití textových formátů, které jsou pak také svou podstatou typicky menší velikosti ve srovnání s binárním tvarem souboru.

Výsledkem jsou knihovny v jazyku PHP použitelné téměř na všech webovských serverech, které usnadňují tvorbu souborů SYLK a XML vhodných pro Excel. Odstiňují uživatele od nutnosti znát detaily těchto formátů. Uvedené ukázky a dokumentace na přiloženém médiu by měly umožnit snadnou a rychlou tvorbu požadovaných dokumentů.

Tyto knihovny neimplementují všechny možnosti, které tyto formáty poskytují. Zejména formát XML je natolik obsáhlý, že implementace všech detailů vyžaduje plné pochopení funkcionality aplikace MS Excel a také mnohem více času. Aplikace jsou však navrženy v objektovém modelu tak, aby jednotlivé další funkcionality byly relativně snadno doplnitelné. Vhodným leč nepochybně obtížným kandidátem pro rozšíření může být implementace kontingenčních tabulek, které jsou také velmi významné zejména ve spojitosti s použitím OLAPu.

Práce také zmiňuje srovnání kompatibility s dalším velmi rozšířeným tabulkovým procesorem OpenOffice.org Calc. Jak je ukázáno, není výsledek naší práce omezen pouze na svůj primární cíl MS Excel, ale výstupy lze použít i v tomto tabulkovém procesoru. Při dodržení jistých zásad při tvorbě souborů jsou zejména dokumenty XML rovnocenné pro obě aplikace.

Věříme, že naše práce bude přínosem a že se nám podařilo vytvořit v praxi užitečný nástroj, který pomůže při vytváření portálových aplikací.

Literatura

[1] Lacko L. *Databáze: datové sklady, OLAP a dolování dat*. Brno, Computer Press 2003
[2] Staníček P. *CSS Kaskádové styly.* Brno, Comptuer Press 2003
[3] Staníček P. *CSS Hotová řešení.* Brno, Comptuer Press 2006
[4] Berka P. *Dobývání znalostí z databází*. Praha, Academia 2003
[5] Brázda J. *PHP 4 Praktické příklady.* Praha, Grada Publishing 2003
[6] Gilmore J. W. *Velká kniha PHP 5 a MySQL*. Brno, Zoner Press 2005
[7] Skonnard A., Gudgin M. *XML – pohotová referenční příručka*. Praha, Grada Publishing 2006
[8] Harold E.R., Means W. S. *XML v kostce*. Praha, Computer Press 2002

Internetové zdroje

Téma XML pro Excel

- Jmenné prostory v MS Office 2003, URL: http://schemas.stylusstudio.com/msoffice2003/index.html
- MS Office 2003 a formát XML, URL: http://www.simonstl.com/articles/officeXML/index.html
- Přehled některých formátů: URL: http://www.wotsit.org/search.asp?page=2&s=database
- Schemata v MS Office 2003, URL: http://rep.oio.dk/Microsoft.com/officeschemas/SchemasIntros.htm

Téma formát SYLK

- Přehled o formátu SYLK, URL: http://en.wikipedia.org/wiki/SYmbolic_LinK_(SYLK)
- Ukázky použití, URL: http://www.pindari.com/sylk.html
- Základní informace o formátu SYLK, URL: http://www.fileformat.info/format/sylk/
- Podrobnější rozbor formátu SYLK, URL: http://www.jkrieger.de/programming/delphi/sylk.txt
- FAQ, řešení některých problémů, odkazy na další vhodné zdroje, URL: http://www.allanswers.org/software/spreadsheets/faq.htm
- Dokument, detaily formátu SYLK, URL: http://www.hh.iij4u.or.jp/~hayasida/oleo/sylksum.doce

Přílohy

Součástí této práce, byla realizace navrženého řešení. K této práci je přiložen CD-ROM na kterém naleznete zdrojové kódy vypracovaných knihoven, dokumentace knihoven ve formátu HTML a dokumentace k formátům SYLK a XML pro využití v aplikaci MS Excel .

Funkční verze je momentálně dostupná na URL: http://zap.profitux.cz/.
Zde také naleznete veškerou dokumentaci i zdrojové kódy ke stažení.

V textu jsou zmiňovány přílohy A a B. **Příloha A** popisuje detaily formátu SYLK (14 stran). **Příloha B** popisuje detaily formátu XML pro Excel 2003 (47 stran). Obě tyto přílohy jsou k nalezení na přiloženém médiu nebo na URL adresách
http://zap.profitux.cz/bin/sylk_doc.pdf
http://zap.profitux.cz/bin/xml_doc.pdf