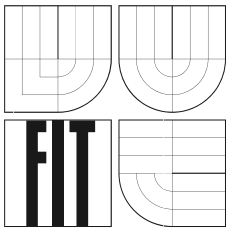
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



MMS Webstudio

Ročníkový projekt

2006 **Vladimír Talaš**

MMS Webstudio

Odevzdáno na Fakultě informačních technologií Vysokého učení technického v Brně
dne 4. května 2006

Prohlášení

Prohlašuji, že jsem tento ročníkový projekt vypracoval samostatně pod vedením Ing. Pavla Slavíčka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Vladimír Talaš
4. května 2006

Abstrakt

Cílem projektu bylo vytvoření webové aplikace pro vytváření, editaci a ukládání multimediálních zpráv pro mobilní telefony MMS. Kromě informací o technologii MMS dokument obsahuje popis open source projektu Apache Cocoon, pro jehož pochopení je nutná znalost jeho principu založeném na datových rourách a komponentech jako jsou generátory, transformátory a serializátory. Je věnován prostor pro popis metod tvorby a zpracování formulářů v Cocoonu. Dále je zde popis tvorby dynamických webových stránek, servletů a také multimediální jazyk SMIL. Dalším bodem projektu byl návrh jednoduché www aplikace umožňující operace s multimediálními zprávami MMS a tuto koncepci realizovat pomocí frameworku Cocoon. S tímto úzce souvisí jazyky HTML a XML, XSL transformace XML dokumentů a v neposlední řadě také jazyk SQL a databázový systém MySQL.

Klíčová slova

Apache Cocoon, XML, XSL, HTML, MySql, Multimedia Messaging Service (MMS), Synchronized Multimedia Integration Language (SMIL)

Obsah

Obsah **5**

1 Úvod **6**

2 Teorie dynamických webových stránek **7**
2.1 Common Gateway Interface (CGI) 7
2.2 Servlety 7

3 Framework Apache Cocoon **9**
3.1 Úvod do Cocoonu 9
3.2 Historie 9
3.3 Architektura 10
3.3.1 Roura (pipeline) 10
3.3.2 Mapa (sitemap) 11
3.3.3 Komponenty 12
3.4 XSP 13
3.5 Cocoon forms 13

4 Technologie Multimedia Messaging Services (MMS) **15**
4.1 Úvod 15
4.2 Princip 15
4.3 Synchronized Multimedia Integration Language (SMIL) 16
4.3.1 Specifikace 16
4.3.2 Podpora 16

5 Aplikace **18**
5.1 Návrh 18
5.1.1 Definice požadavků 18
5.1.2 Specifikace požadavků 18
5.1.3 Případy použití 18
5.1.4 Návrh databáze 18
5.2 Implementace 19
5.2.1 Layout 19
5.2.2 Registrace uživatelů, upload souborů 19
5.2.3 Vytváření a ukládání MMS 20

6 Závěr **23**

Kapitola 1

Úvod

Technologie Multimedia Messaging Services (MMS) [1] přináší nový rozměr do mobilní komunikace prostřednictvím sítí GSM [2]. Tyto zprávy kromě prostého textu umožňují přenos obrazu a zvuku, což významně rozšiřuje možnosti komunikace. S tím jsou ovšem spojeny i vyšší nároky jak na poskytovatele této služby (způsob přenosu, větší objem dat atd.) tak na samotného uživatele. Vytvoření MMS zprávy je komplikovanější, tento projekt se pokusil usnadnit tvorbu MMS zpráv a zároveň demonstrovat možnosti této technologie.

Celý dokument můžeme rozdělit do dvou tematických celků. První část přináší osvětlení použitých technologií. Nejprve se seznámíme s tvorbou dynamických webových prezentací. Jedná se hlavně o popis základních principů fungování dynamických webových stránek, relativně podrobný popis frameworku Cocoon[3, 4], v menší míře je zde uveden popis technologií s Cocoonem spojených, jako je XML [5, 6] či XSL transformace XML dokumentů. Resp. předpokládá se jistá elementární znalost těchto technologií, tudíž jim v dokumentu není věnováno tolik prostoru. Framework Cocoon je založen na datových rourách (pipes). Pro pochopení principu činnosti Cocoonu je nutné pochopení této problematiky.

V projektu jsou použity také technologie multimediálních zpráv pro mobilní telefony (MMS) a s tím související multimediální programovací jazyk SMIL[7]. SMIL slouží k operacím nad MMS zprávami, tj. jejich vytváření, editace atd.. Těmto technologiím je věnována čtvrtá kapitola.

Druhý tematický celek se pak věnuje aplikaci její a implementaci. V této části se dozvíme, jak výsledná aplikace vypadá. Po definici a specifikaci požadavků je zde návrh databáze a jednoduché případy použití. Také je zde několik slov o vizuální podobě MMS Webstudia. Kromě konkrétních postupů při tvorbě aplikace jsou zde také popsány problémy, které při řešení projektu vznikly.

Jako samostatná část pak vystupuje závěr, kde jsou zhodnoceny dosažené výsledky, prodiskutovány nedostatky a zváženy možnosti dalšího postupu při tvorbě projektu.

Kapitola 2

Teorie dynamických webových stránek

První webové servery nebyly v podstatě nic jiného, než obyčejné souborové servery. Pracovaly tak, že pokud si prohlížeč vyžádal nějaký soubor, server mu ho vrátil (pokud existoval) a poté prohlížeč mohl získaný soubor zpracovat, obvykle vykreslil na obrazovku. Pokud dokument obsahoval hypertextový odkaz, mohl uživatel kliknutím na ně zažádat o další soubor. Výhodou tohoto systému je jeho značná jednoduchost, nevýhody spočívají v jeho statičnosti, interakce s uživatelem je omezena na hypertextové odkazy. Uživatel nemá možnost odesílat složitější dotazy a dostávat na ně odpovědi. Naprosto nevhodný je tento systém při rychle se měnících se prezentovaných informací, data musí dostupné v podobě HTML souborů. Je velmi nepraktické zajišťovat neustálou aktualizaci těchto souborů.

Jako reakce na tento stav byly vyvinuty technologie pro dynamické vytváření HTML stránek. Zdrojová data mohou pocházet z dotazované databáze, XML souboru, ale také výpis z FTP serveru atd.. Je zřejmé, že pro generování stránek podle požadavku uživatele je potřeba nějaké aplikace. Webový server se tak rázem mění na aplikační - poskytuje jak soubory, tak aplikace.

2.1 Common Gateway Interface (CGI)

Standardním API pro webový server je Common Gateway Interface (CGI)[8]. CGI je velmi oblíbené a rozšířené prostředí, je podporováno většinou webových serverů. Jedná se o velmi jednoduché API, které určuje, jakým způsobem je spuštěná webová aplikace, jakým způsobem jsou načteny a zpracovány parametry od uživatele a jakým způsobem musí aplikace vrátit výsledek. Nevýhodou tohoto řešení je jeho nepříliš velká účinnost. Např. pro každý požadavek se musí spustit nová instance aplikace. Z toho pramení vysoké nároky na hardware serveru.

2.2 Servlety

Jednou z mnoha reakcí na nevýhody CGI byly nově vyvinuté API jako např. NSAPI a WAI pro Netscape či ISAPI firmy Microsoft. Rozsahem připomínaly původní CGI, leč byly mnohem výkonnější. Ovšem díky jejich vzájemné nekompatibilitě brzy upadly v zapomnění. Rozumně použitelné řešení přišlo až od vývojářů Javy a jejich servlety, standardní náhradu CGI.

Servlet je Java program běžící v prostředí aplikačního serveru, je zároveň nezávislý na protokolu. Díky tomu, že je napsán v Javě, má přístup ke všem možnostem tohoto jazyka, zároveň je servlet velmi dobře přenositelný na velké množství platforem a je tedy možné je používat na všech webových serverech. Servlety jsou zároveň dosti výkonné, poněvadž se nahrají do paměti

hned při startu serveru a pro komunikaci s klientem používají vlákna, mohou tedy obsluhovat více požadavků najednou.

V poslední řadě bych se ještě rád zmínil o technologii Java Server Pages (JSP). JSP umožňují kombinovat v jednom WWW dokumentu HTML kód i příkazy Javy, přičemž Java příkazy se zpracovávají a jejich výstupy generují do obsahu WWW dokumentu ještě na serveru. To znamená, že při použití JSP se podobně jako v ASP nebo v PHP přímo do HTML kódu zapisují příkazy. Ty jsou nyní zapisovány v jazyce Java. Speciální servlet se stará o to, aby byla JSP stránka vždy po své modifikaci automaticky přeložena do byte-code. To usnadňuje vývoj, protože se vývojář nemusí starat o kompilaci, i tak je aplikace výkonná, protože jsou skripty vyvolávány v již přeloženém stavu.

Kapitola 3

Framework Apache Cocoon

3.1 Úvod do Cocoonu

Apache Cocoon je „open source“ framework pod záštitou Apache.org pro vytváření webových aplikací založených na XML. Dá se říci, že Cocoon je jedním z hlavních projektů vyvíjených v rámci iniciativy XML Apache. V podstatě je to aplikace, která běží jako nadstavba webového serveru. Funguje tak, že konvertuje stránky z formátu XML do HTML, PDF, VRML a dalších formátů vhodných pro zobrazování na různých platformách a v různých situacích.

Cocoon jako takový je založen XML a proto nutnou podmínkou pro práci v něm je dobrá znalost XML a také aspoň částečná znalost XSL. Vzhledem ktomu, že Cocoon je napsán v Javě, je také vhodná, aspoň okrajová znalost programování v Javě a jejích technologií jako je JSP či servlety.

3.2 Historie

V roce 1998 vytvořili Jon Stevens a Stefano Mazzocchi jednoduchý skript, který řešil automatické aktualizace www stránek java.apache.org. Nastal však problém, že jednotlivé příspěvky neměly jednotný styl prezentace. Řešením bylo oddělit obsah a vzhled příspěvků.

V tomtéž roce byl uveřejněn první návrh XSL (eXtensible Stylesheet Language) a byl vyvinut také v Javě napsaný první XSL procesor LotusXSL od firmy IBM. XSL měl původně sloužit jak k transformaci dokumentů, tak i k definování jejich vzhledu. Stefano Mazzocchi však začal pracovat na oddělení těchto částí - pracoval na tom, čemu se později začalo říkat XSLT (XSL Tranformations)[5, 6].

Při jeho práci se mu nelíbilo, jak musí neustále psát příkazy do příkazové řádky a přepnout do prohlížeče, aby viděl výsledek. Proto napsal program (servlet), který mu po změně ve stylu a po reload v prohlížeči ukázal nový výsledek. Bylo to koncem roku 1998 a v televizi vysílali film Rona Howarda - Zámotek (Cocoon) - podle něho svůj program pojmenoval.

Apache Cocoon 1.0 byl servlet, se zdrojovým kódem o 100 řádcích, který využíval XSLT procesor pro generování výsledného dokumentu z XML souboru a XSL stylu. Mazzocchi si ale nemyslel, že by někdo jiný chtěl jeho servlet používat. Po několika měsících se někdo v mailové konferenci ptal na XSL a Mazzocchi mu odpověděl, že si vytvořil program, který provádí tranformaci XML souborů podle XSL stylu na straně serveru. Mnoho lidí na jeho odpověď reagovalo, a tak začal projekt Apache Cocoon pod záštitou java.apache.org.

Další generace projektu Cocoon je nazývána Cocoon2 (nebo jen C2). Je to vylepšený systém, opět založený na XML a XSLT technologiích pro serverové aplikace. Cocoon 2 využívá pro zpracování požadavků SAX pipeline. V dalším textu popisuji architekturu modelu Cocoon 2.

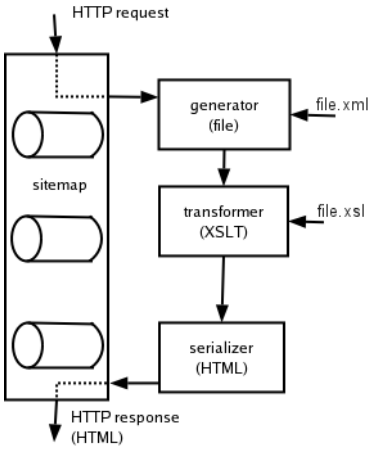
3.3 Architektura

Pro webového klienta nemají stránky generované cocoonem žádné úskalí, jeví se např. jako statické HTML dokumenty (možností může být daleko více, jako např. PDF, VRML atd. záleží na zvoleném typy serializátoru, viz níže). Za srdce a principem celé činnosti Cocoonu se dá považovat roura a sitemap, neboli mapa, která slouží jako jakýsi popis roury. Architektura je založena na 3 typech rour:

- generátory: komponenta pro práci s XML dokumenty z rozdílných zdrojů, zajišťuje vstupní data pro aplikaci.
- transformátory: zpracovávají vstupní data, produkují nové SAX události
- serializátory: komponenty pro převod dat do zvoleného formátu, jde o finální úpravu zpracovávaných dat, vytvářejí výstupy.

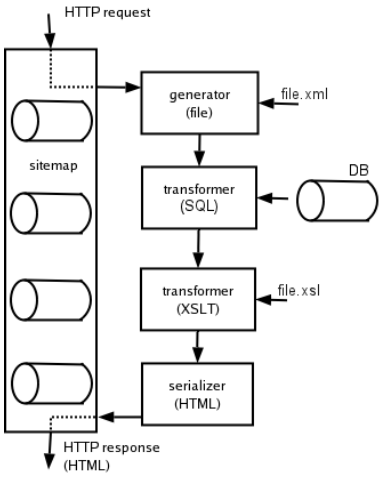
3.3.1 Roura (pipeline)

Roura je vlastně jedna z komponent, ježto se využívá v 99 procentech aplikací vyvíjených v Cocoonu (popis dalších standardních komponent se nachází níže). Je ekvivalentem roury používané na Unixových systémech, tzn. že např. při zápisu A—B—C je výstup programu A automaticky přesměrován na vstup programu B atd. Obdrží-li Cocoon nějaký požadavek na data, vybere podle typu požadovaných dat správnou rouru, která data zpracuje a vrátí výsledek ve zvoleném formátu. Na obrázku 3.1 vidíme nejjednodušší příklad na použití roury v Cocoonu. Z obrázku je patrné, že vstup je zapsán v souboru file.xml, načtení tohoto souboru má na starosti komponenta generátor, posléze je soubor transformován za použití šablony file.xsl a výsledek je poslán na výstup aplikace pomocí HTML serializátoru. Zjednodušeně probíhá každý požadavek takto: generátor načte vstup, ten je za pomocí transformátoru přeměněn či upraven a o výstup se postará komponenta serializer.



Obrázek 3.1: roura v Cocoonu

Na prvém příkladu bylo prezentováno velmi triviální použití roury. Při vývoji aplikací by takový přístup jistě nebyl dostačující. Naštěstí možnosti Cocoonu nejsou zdaleka tak omezené, můžeme používat více transformátorů za sebou, programovat své vlastní komponenty a mnoho dalšího. Na obrázku 3.2 vidíme využití dynamického obsahu souboru. Opět máme soubor XML a XSL styl. Obsahem XML souboru je požadavek na databázi.



Obrázek 3.2: roura v Cocoonu s SQL dotazem

Požadavek v XML souboru je zpracován standardní komponentou Cocoonu pro zpracování SQL dotazu, výsledek ve formě XML dat je rourou zaslán k dalšímu zpracování (v tomto případě se bude jednat o formátování pomocí XSL šablony), výsledek je zpracován serializátorem, standardně nastaveným na HTML.

3.3.2 Mapa (sitemap)

Popis všech rour k dané aplikaci se nachází ve speciálním XML souboru zvaném sitemap.xmap. Je to dokument obsahující deklaraci použitých komponent (pokud byly použity jiné než standardní) a popis rour. Ve praxi to vypadá tak, že pokud se vyskytne požadavek na data (např. HTML dokument, obrázek JPG atd.) je na základě tohoto požadavku vybrána roura, která jej popsaným způsobem zpracuje. Pokud není k požadavku definovaná žádná roura, vypíše Cocoon chybovou hlášku. Vše je nejlépe patrné z obrázku 3.3.

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

<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">

<map:pipelines>
  <map:pipeline>
    <map:match pattern="index.html">
      <map:generate src="index.xml"/>
      <map:transform src="xml2html.xsl"/>
      <map:serialize type="html"/>
    </map:match>
  </map:pipeline>
</map:pipelines>

</map:sitemap>
```

Obrázek 3.3: soubor sitemap.xmap

Definice roury je zapsána jako *map:pipeline*, uvnitř této definice najdeme co se stane, je-li požadavek na soubor index.html. V tomto případě se nejprve načte soubor index.xml, obsahující data ve formě XML souboru, ten je po načtení zaslán transformátoru, jež k transformaci použije XSL šablonu xml2html.xsl, výsledek transformace je zaslán serializátoru, který data převede do zvoleného formátu, v tomto případě se jedná o HTML soubor. Mapa nejenže deklaruje a konfiguruje

roury a komponenty, ale vytváří také nový, virtuální jmenný prostor pro klienta. Tento jmenný prostor nijak nesouvisí se skutečnými soubory na disku serveru, ve zmíněném příkladu z obrázku 3.3 žádný soubour index.hmtl neexistuje. Tímto se zvyšuje bezpečnost aplikace, poněvadž klient standardní cestou nemůže zjistit strukturu skutečných souborů na serveru.

3.3.3 Komponenty

Jak už bylo řečeno, existují 3 základní typy komponent Cocoonu - generátor, transormer, serializer. Při definici roury je na prvním místě obvykle generátor, následuje jeden či více transformerů, výstup je většinou zpracováván za pomocí serializeru. Cocoon nabízí celou škálu standardních komponent všech zmíněných typů, pokud ani jeden z nich nevyhovuje požadavkům programátora, lze si jednoduše v Javě naprogramovat komponenty vlastní.

Generátory: Generátory mají za úkol na začátku roury vygenerovat XML data ve formě SAX událostí. Cocoon obsahuje mnoho vestavěných generátorů, zde je výpis těch nejdůležitějších.

- File Generator: čte XML soubor z disku a předává jej následující komponentě v rouře. Je to implicitní generátor.
- XPath Generator: čte fragment XML souboru specifikovaný pomocí XPath a předává jej další komponentě v rouře.
- Directory Generator: vypisuje obsah zadaného adresáře. Výpis je převeden do XML a předán další komponentě v rouře.
- Image Directory Generator: pro soubory s bitmapovými obrázky rozšiřuje výše uvedený generátor o atributy výšku a šířku obrázku.
- Request Generator: předává v XML hodnoty hlavičky a parametrů HTTP požadavku.
- Server Pages Generator:důležitý generátor pro XSP (XML Server Pages - obdoba Java Server Pages), kterými se budeme zabývat později.

Transformátory: Transformátor je komponenta, která za vstup dostává ve formě SAX událostí XML data. Tyto data podle jistého klíče přetransformuje a opět ve formě SAX události pošle dál. Z toho plyne, že transformátor nemůže být ani na začátku ani na konci roury. Zde je několik nejzajímavějších transformátorů:

- XSLT Transformer: implicitní a také nejužívanější transformátor. Na základě XSL dokumentu přečteného z disku nebo z libovolného URL aplikuje XSL transformaci na vstupní data. Výbornou vlastností je, že tomuto transformátoru můžeme předat přes mapu parametry - například hodnoty parametrů HTTP požadavku.
- SQL Transformer: umí se dotázat SQL databáze. Dotaz specifikují XML data na vstupu transformátoru, výsledek dotazu je převeden do XML a odeslán na výstup transformátoru.
- LDAP Transformer: umí se dotázat LDAP databáze. Princip je podobný jako u SQL transformátoru.
- CInclude/XInclude Transformer:umí do XML dat ze vstupu vložit text nebo další XML elementy z vnějšího zdroje. Transformátor XInclude používá specifikaci XInclude.
- Session Transformer: umí do transformovaných XML dat začlenit hodnoty z kontextu uživatelské relace (session) nebo naopak z transformovaných dat zapsat hodnoty do kontextu relace. Hodí se například pro nastavení implicitních hodnot webových formulářů.

- Write/Read DOM Session Transformer: může zapsat do kontextu uživatelské relace (session) část DOM (Document Object Model: stromová reprezentace XML dat) transformovaných dat (Write DOM Transformer), resp. jej umí z kontextu relace přečíst a začlenit do transformovaných dat (Read DOM Transformer). Tato dvojice transformátorů je podobná předchozímu transformátoru.
- I18n Transformer: umožňuje internacionalizaci aplikace. Označené texty jsou nahrazovány z katalogu podle aktuálně nastaveného jazyka.
- Paginator: umožňuje efektivně rozdělit na více „stránek“ data, která by jinak při zobrazení byla příliš rozsáhlá. Je konfigurovatelný, takže můžete například nechat zobrazovat rozsáhlou tabulku na počítačích typu desktop po sto řádcích, na PDA po dvaceti a na WAP stránkách pro mobilní telefony po pěti řádcích.

Serializery: Serializery slouží ke konečné úpravě XML dat (dodávaných na jejich vstup opět ve formě SAX událostí). Tyto data převádí na výstupní proud, kterému rozumí webový klient, popřípadě může být zpracován jinak, nezávisle na Cocoonu. Opět tu máme několik nejzajkímavějších serializátorů

- HTML/XHTML Serializer: serializuje vstupní data XML na HTML nebo XHTML. Pozor, vstupní XML data už musí obsahovat značky (X)HTML, žádná transformace se zde už nekoná. Jde skutečně jen o serializaci událostí SAX do proudu. Tento serializátor je implicitní a také nejčastěji používaný.
- XML Serializer: nechává data v XML, nicméně není zbytečný, jak by se mohlo na první pohled zdát. Opět serializuje události SAX do proudu.
- WAP/WML Serializer: používá se pro vytváření WML stránek pro WAP prohlížeče mobilních telefonů.
- Text Serializer: vytváří obyčejný text.
- PDF Serializer: vytváří PDF. Zajímavý serializátor, PDF lze použít například na skutečně „printer-friendly“ variantu webových stránek.
- SVG Serializer: Umožňuje výstup do různých grafických rastrových formátů. Podporovány jsou následující formáty: JPEG, PNG, TIFF.

3.4 XSP

XSP (eXtensible Server Pages)[3] je technologií jež byla prvně použita v Cocoonu a slouží pro práci s dynamickým obsahem XML stránek. XSP dokumenty mají možnost smíchat XML data s kódem v Javě, Javascriptu či Pythonu. Problémem je, že takto použité propojení míchá zároveň logiku dat s její datovým obsahem což není v souladu s přístupem SoC (Separation of Concerns). V Cocoonu je tento problém vyřešen za pomocí knihoven značek neboli logicsheets. Jedná se vlastně o XSL dokumenty, které za běhu transformují tyto značky na programový kód. Při prvním spuštění se projeví se malá výkonová ztráta neboť XSP se překládají na třídu v Javě. Přeložené XSP se ovšem dají uložit ve formě souborů class, které se pak dají použít jako uživatelský generátor.ogicsheets a jejich jmenné prostory musejí být nadeklarovány v souboru „cocoon.xconf“.

3.5 Cocoon forms

Framework Cocoon je zaměřen na oddělení formy od obsahu, díky tomu lze aplikace vytvořené v Cocoonu použít na mnoha platformách s minimálními úpravami. Vzhledem k požadavkům na mo-

derní aplikaci kdy je potřeba prezentovaná data nejen prohlížet, ale i měnit, a to např. prostřednictvím formulářů, obsahuje framework Cocoon pokročilé prostředky pro tvorbu a zpracování formulářů(validace vstupních dat apod.).

Kromě jednoduchých prostředků pro práci s formuláři jako je *FormValidatorAction*, *precept*, *JXForms*[3] obsahuje Cocoon i blok *Cocoon Forms*. Původně vznikl jako samostatný doplněk pod jménem *Woody*, časem byl však přijat jako standardní součást Cocoonu. Nyní se oficiálně nazývá jako Cocoon Forms.

Tvorba formuláře za pomoci Cocoon Forms je závislá na jeho *definici*, která je zcela nezávislá na prezentaci a je to jakási šablona. Je to vlastně obecný popis formuláře. Další částí je *instance* což je vlastně konkrétní podoba formuláře v XML formátu vytvořená na základě šablony. Kromě vzhledu(např. rozmístění polí) může obsahovat i aktuální hodnoty polí formuláře. Finální podoba, tu kterou uživatel uvidí na svém prohlížeči, vznikne XSL transformací instance formuláře. Kromě základních datových typů jako je *string, integer, long, date* obsahuje Cocoon forms vlastní, složitější datové typy např. *e-mail*. V neposlední řadě je nutno zmínit se ještě *flowscriptu* s nímž jsou Cocoon Forms těsně spjaty. Je to funkce např. v Javascriptu, která vyvolá akce související s kontrolou zadaných dat ve formuláři.

Kapitola 4

Technologie Multimedia Messaging Services (MMS)

4.1 Úvod

Jde o otevřený standard pro bezdrátové sítě, který sestavilo wap fórum pro organizaci 3GPP (Partnerský projekt mobilních sítí třetí generace, dohoda o spolupráci vytvořená v prosinci roku 1998) [2]. Ačkoli byl standard MMS původně určen až pro sítě třetí generace, operátoři jej implementovali už do svých sítí druhé generace (2G)[2]. K přenosu MMS zpráv byl použit bezdrátový přenosový protokol WAP za využití technologie přenosu dat GPRS.

Zprávy MMS jsou v podstatě rozšířením zpráv SMS o možnost přenosu obecných dat. Jedná se o technologii umožňující vytvářet, posílat a přijímat textové zprávy které kromě textu obsahují grafiku, zvuk a zároveň také může obsahovat video. Přenos MMS zpráv je proto značně komplikovanější. Data, jež se kromě textu přenášejí, mohly být podle původní specifikace 3GPP R4[2] pouze tohoto typu:

- obrázky ve formátech JPEG, GIF, WBMP
- zvuky ve formátu AMR
- data typu PIM, tedy položky kalendáře vCalendar 1.0 a vizitky vCard 2.1.

Později k těmto základním formátům dat přibyly i další, včetně formátů pro video. K přenosu takových dat je ale třeba prostředek, který současné odeslání takto odlišných typů dat zajistí. K tomu slouží jazyk SMIL, na němž jsou zprávy MMS založeny. Slouží k přenosu obecných multimediálních dat mj. také přes spoje s nižší přenosovou kapacitou, což se je případ i systému GSM. Jazyk SMIL disponuje prostředky, kterými dokáže současný přenos data různého charakteru zajistit.

SMIL zapouzdřuje text zprávy a přiložená data do jedné jednotky, která je pak odeslána stylem, který se označuje jako „slideshow“, což znamená, že celý obsah zprávy je podle potřeby rozdělen na jeden nebo více rámců, jež jsou jeden po druhém odeslány k cíli. V každém z přenášených rámců je pak samostatně vyhrazená část pro data a pro text. Tento princip je třeba k tomu, aby mohla být taková data přenášena i přes nízkokapacitní sítě.

4.2 Princip

Služba MMS patří podobně jako SMS do skupiny služeb s anglickým označením „store-and-forward“, což lze přeložit zhruba jako „ulož a předej“. Tím je už naznačen princip fungování služby. Zprávy

MMS jsou v mobilním telefonu prostřednictvím klienta MMS zpracovány podle protokolu SMIL, pak jsou podle předpisů protokolu WAP poslány přes GPRS na adresu MMS serveru.

Server, pro který se často používá označení MMSC (Multimedia Message Service Center), zprávu MMS uloží ve svých registrech. Podle cílové adresy, kterou může být telefonní číslo nebo e-mail, se pak server rozhodne, zda zprávu pošle ihned k příjemci, nebo zda adresáta nejprve upozorní prostřednictvím SMS. Pokud příjemce potvrdí přijetí zprávy a má správně nastaveny datové profily, obdrží zprávu MMS ihned poté. Když ale zprávu příjemci nelze z jakéhokoli důvodu doručit, zůstane MMS po určitou dobu v závislosti na nastavení operátora uložena v registrech MMS serveru.

Specifikace standardu MMS je nezávislá na vlastní přenosové technologii, což znamená, že kromě zmíněného GPRS je stejně tak dobře použitelná např. v sítích WCDMA.[2] Technologie GPRS a sítě GSM jsou ale v současnosti pro MMS zprávy nejčastěji používanou platformou. Nutno dodat, že zprávy MMS mají svá poměrně přísná omezení velikosti. Nejvíc se týká velikosti datové přílohy – obvykle je v rozmezí 30 kB až 100 kB. Obsáhlejší přílohy by pak mohla umožnit třeba technologie EDGE[9]. To ale záleží na rozhodnutí operátora. T-Mobile má bez ohledu na EDGE maximální velikost příloh nastavenu na 100 kB, stejně tak i Oskar, a Eurotel podporuje velikost zpráv až do 300 kB.

4.3 Synchronized Multimedia Integration Language (SMIL)

4.3.1 Specifikace

Jazyk SMIL byl vyvinut konsorciem W3C (World Wide Web Consortium) a umožňuje tvorbu multimediálních prezentací (obdoba prezentací Power Pointu) určených především pro internet. Je to jazyk značkovací, vycházející z XML. Do prezentací je možno vkládat text, obrázky, zvuk nebo i video.

V prvé řadě je nutno si uvědomit, že SMIL neslouží k tvorbě multimediálních dat jako jsou obrázky video či hudba. Umožňuje integrovat tyto multimediální objekty do HTML nebo XML dokumentů. Pracuje s již hotovými objekty, jeho úkolem je vkládat je do dokumentů, vymezuje jim prostor, určuje jejich polohy v závislosti na čase atd. Výsledný dokument je jakousi dynamickou prezentací srovnatelnou s prezentací vytvořenou v PowerPointu. Vlastnosti SMILu se dají shrnout do několika základních bodů:

- přesný popis vizuální podoby prezentace přesné umístění a velikost vkládaných objektů změny prezentace v čase
- hyperodkazy a parametrizace prezentace (umožňují uživateli reagovat na objekty - zapínat, vypínat - simulují ovládací prvky přehrávače)
- testování podmínek - SMIL disponuje větvením (switch), které umožňuje rozlišit spoustu vlastností prohlížeče - rychlost připojení, systémový jazyk, velikost obrazovky a další

4.3.2 Podpora

SMIL jako takový podporuje většinu multimediálních souborů jako jsou např. obrázky JPG, GIF, PNG, videu soubory AVI, MPG, MOV, audio ve formátu WAV, MID či MP3 ale také text jakožto prostý text nebo ve formě HMTL. Horší je podpora ze strany současných webových prohlížečů do nichž nelze SMIL integrovat klasickou cestou, tedy vložení kódu napsaného v jazyce do HTML fungovat nebude. Existují sice možnosti zobrazení pomocí Javy, či jiných, specializovaných programů, leč co se týče nativní podpory SMILu, tak jak byl definován zatím ještě neexistuje.

Jednoduché, avšak nepříliš elegantní řešení vyvstává při vložení odkazu do dokumentu a ten se při kliknutí na něj zobrazí v externím prohlížeči. Bohužel otázkou zůstává, co se stane při kliknutí na takový odkaz. Možnosti jsou, že se soubor otevře v externím programu nebo se otevře nová stránka s vloženým objektem, ale může dojít i k situaci kdy na straně klienta není nainstalován vhodný prohlížeč a dojde pouze k zobrazení zdrojového kódu. Možným řešením je projekt HTML+TIME Microsoftu ve spolupráci s firmou Macromedia. Tato problematika už ale přesahuje možnosti tohoto projektu a proto se spokojíme s řešením výše uvedeným (použití odkazu a externího programu)

Co se týče oněch zmiňovaných externích programů, máme na výběr z mnoha možností, pro ukázku uvedu několik z nich:

- Real One Player
- Apple Quick Time Player
- Java Applety na straně serveru - zde je nutná podpora Javy na straně klienta

Kapitola 5

Aplikace

5.1 Návrh

5.1.1 Definice požadavků

Vytvořit webovou aplikaci, schopnou provádět operace s multimediálními zprávami pro mobilní telefony MMS. Operacemi se myslí vytvoření editace a uložení MMS zprávy. Uživatel bude mít možnost vkládat vlastní motivy, ať zvuk tak i obraz. Součástí MMS zprávy je také text.

5.1.2 Specifikace požadavků

Ve frameworku Apache Cocoon vytvořit aplikaci schopnou provádět operace vytvoření, editace a uložení MMS zprávy. Obrázky, které tvoří základ MMS zprávy mohou být ve formátech JPG, PNG, GIF. Budou uloženy v databázi jako binární data. Uživatel bude mít možnost si obrázky prohlížet v galerii, nové obrázky bude moci do systému nahrávat přes webové rozhraní. Při nahrávání může zvolit zda budou obrázky veřejné(obrázek je umístěn v galerii, viditelný pro všechny uživatele systému) či privátní. S tímto souvisí možnost sytému registrovat nové uživatele, vytvářet jim uživatelské účty. Tyto účty budou chráněny unikátní uživatelským jménem a heslem. Jako databázový systém bude použit systém MySQL. Obrázky se v databázi vyskytují ve formě binárních dat, musí být proto použit datový typ BLOB. K MMS zprávám je také možno přidávat zvuk, bude se jednat o krátké skladby ve formátu MID. Pro ukládání již hotových MMS zpráv poslouží jazyk pro zpracování multimediálních dat - SMIL. MMS zpráva resp. její reprezentace v kódu SMIL bude rovněž uložena v databázi.

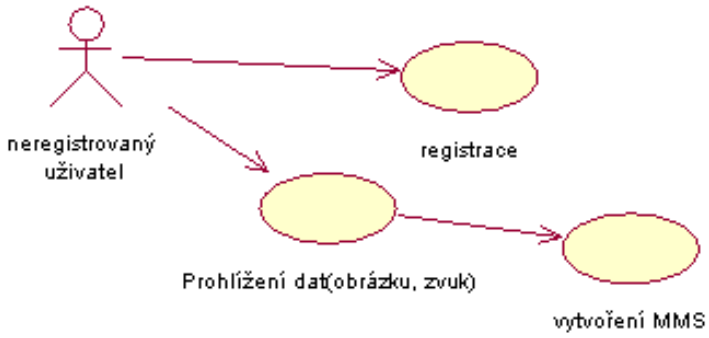
5.1.3 Případy použití

Na obrázcích 5.1 a 5.2vidíme jednoduché případu použití pro jednotlivé role. Jak je ze schémat patrné, vytvářet MMS zprávy může i neregistrovaný uživatel. Jde však pouze o jakýsi náhled, uložení MMS zprávy do databáze je uživateli umožněno až po registraci.

5.1.4 Návrh databáze

Jako databázový systém je použit systém MySQL. Na obrázku 5.3 je vidět struktura jednoduché databáze.

Tabulky *obrazky* a *zvuk* obsahují zdrojová data jako binární data. Obsahují také informace vlastníkovi, pokud vlastník není zadán, je automaticky nastavena hodnota *public* a obrázek či zvu-



Obrázek 5.1: role neregistrovaný uživatel

ková data jsou automaticky považována za veřejná. Zároveň je uchovávána informace o datumu a čase vložení. Pro uložení dat v binární podobě je třeba použít datový typ BLOB.

Tabulka *Osoba* obsahuje základní informace o uživatelích, kteří již byli registrováni do systému. Nakonec v tabulce *MMS* se nachází vytvořené MMS zprávy. Jednotlivé MMS zprávy náleží jistému vlastníkovi, neregistrovaný uživatel nemá možnost zprávy ukládat. Zpráva je v databázi uložena jako kód v multimediálním jazyce SMIL.

5.2 Implementace

5.2.1 Layout

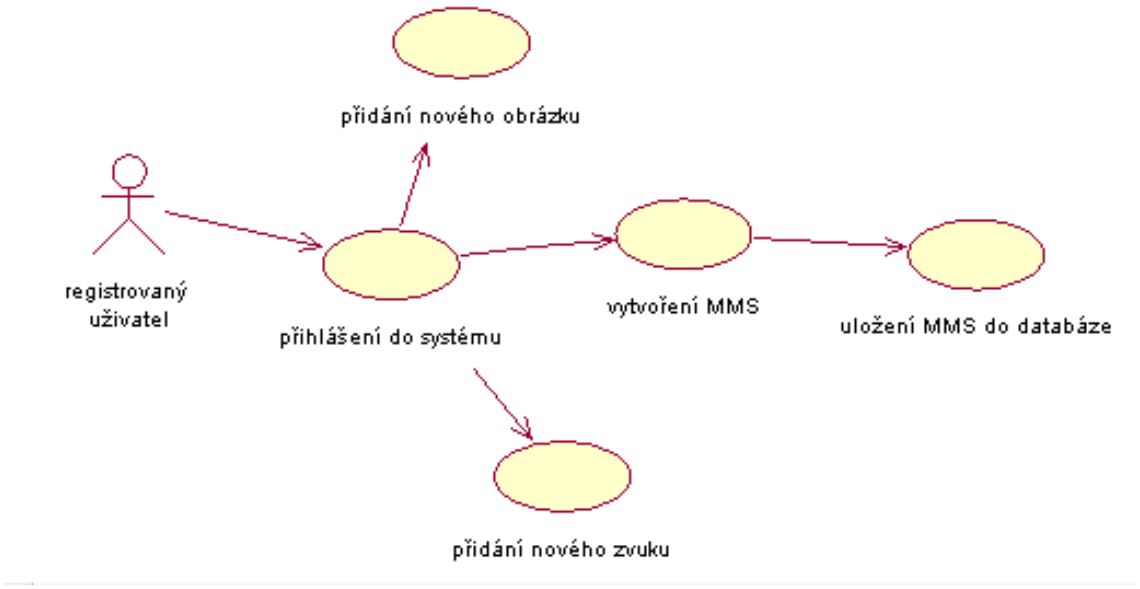
Na obrázku 5.4 vidíme základní schéma grafické podoby aplikace. Hlavní okno je rozděleno na 3 části (v obrázku označeny 1,2,3, oblasti jsou vyznčeny silnou čarou). První část slouží pouze k zobrazení loga, má jen informativní charakter, druhá část zobrazuje menu a odkazy do sekcí aplikace, ve třetí části jsou zobrazována data. Tečkované linky vymezují oblasti blokových elementů.

5.2.2 Registrace uživatelů, upload souborů

Jak při registraci uživatelů tak i při uploadu souborů(obrázky, zvuk) bylo zapotřebí využít formulářů. Framework Cocoon umožňuje velmi dobře a jednoduše zpracovávat formuláře. Při tvorbě aplikace jsem použil různé postupy v kontrole zadaných údajů ve formulářích při registraci uživatelů a uploadu souborů.

Při registraci bylo využito XSP dokumentů, standardní součásti Cocoonu známou jako *form-validator* a jazyku JAVA. Základem byla nově nadefinovaná roura v souboru *sitemap.xmap* pojmenovaná *registrace* při požadavku vedoucí k registraci se vytvoří formulář. Po kliknutí na tlačítko *odeslat* se vyvolá akce kontrolující zadaná data. Možné hodnoty *(parametry)* jsou uloženy v souboru označovaném jako *descriptor* (*descriptor.xml*). V souboru *regerr.xsp* jsou funkce v jazyce JAVA, které na základě povolených dat uložených v descriptoru kontrolují vstupní data ve formuláři. Pokud jsou data v pořádku, je obsah formuláře vložen do databáze a tím zaregistrován nový uživatel. V opačném případě se formulář zobrazí znovu, tentokráte se zobrazenými varováními o nesprávném vyplnění.

Při uploadu souborů jsem použil jiný přístup pro kontrolu vstupních údajů, jedná se hlavně o kontrolu typu vkládaného souboru (podporovány jsou formáty JPG, PNG, GIF), dále se musí

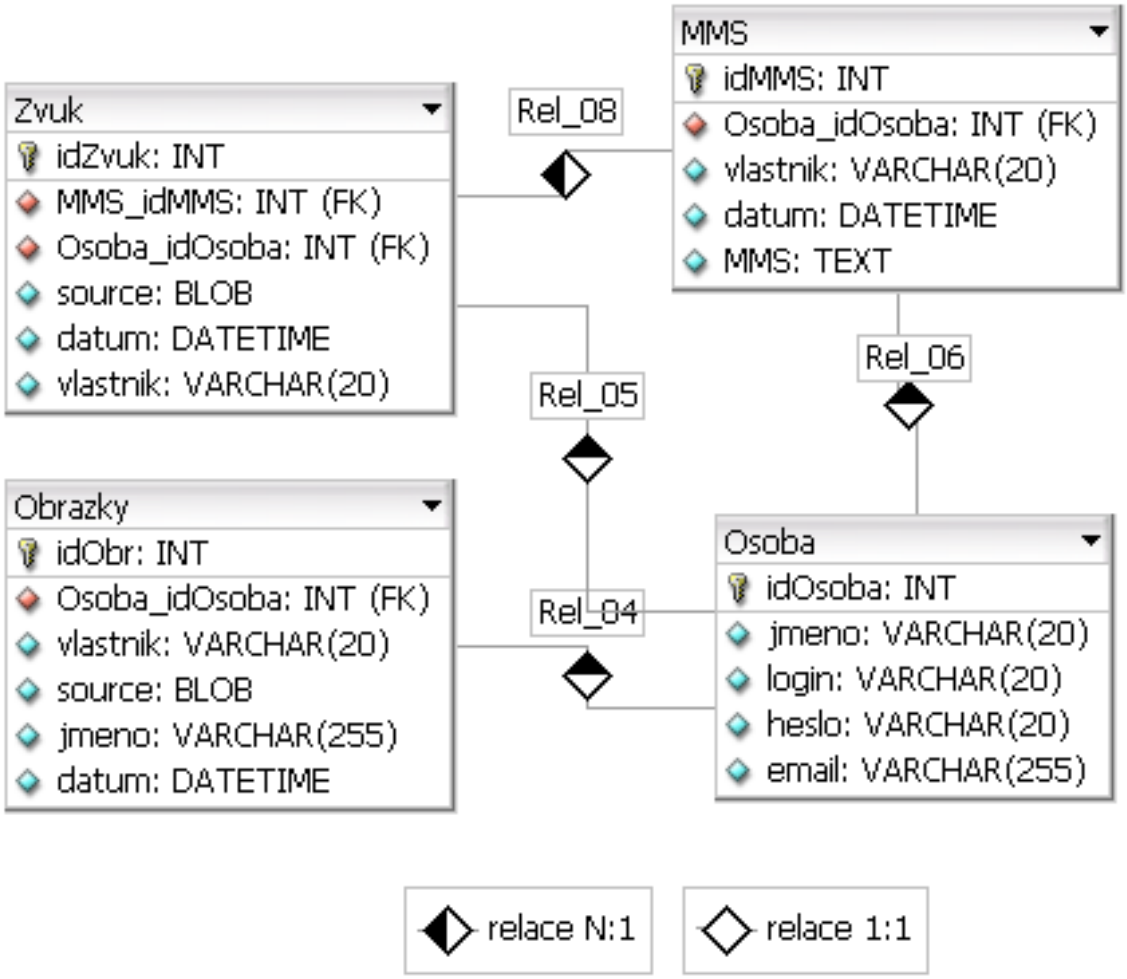


Obrázek 5.2: role registrovaný uživatel

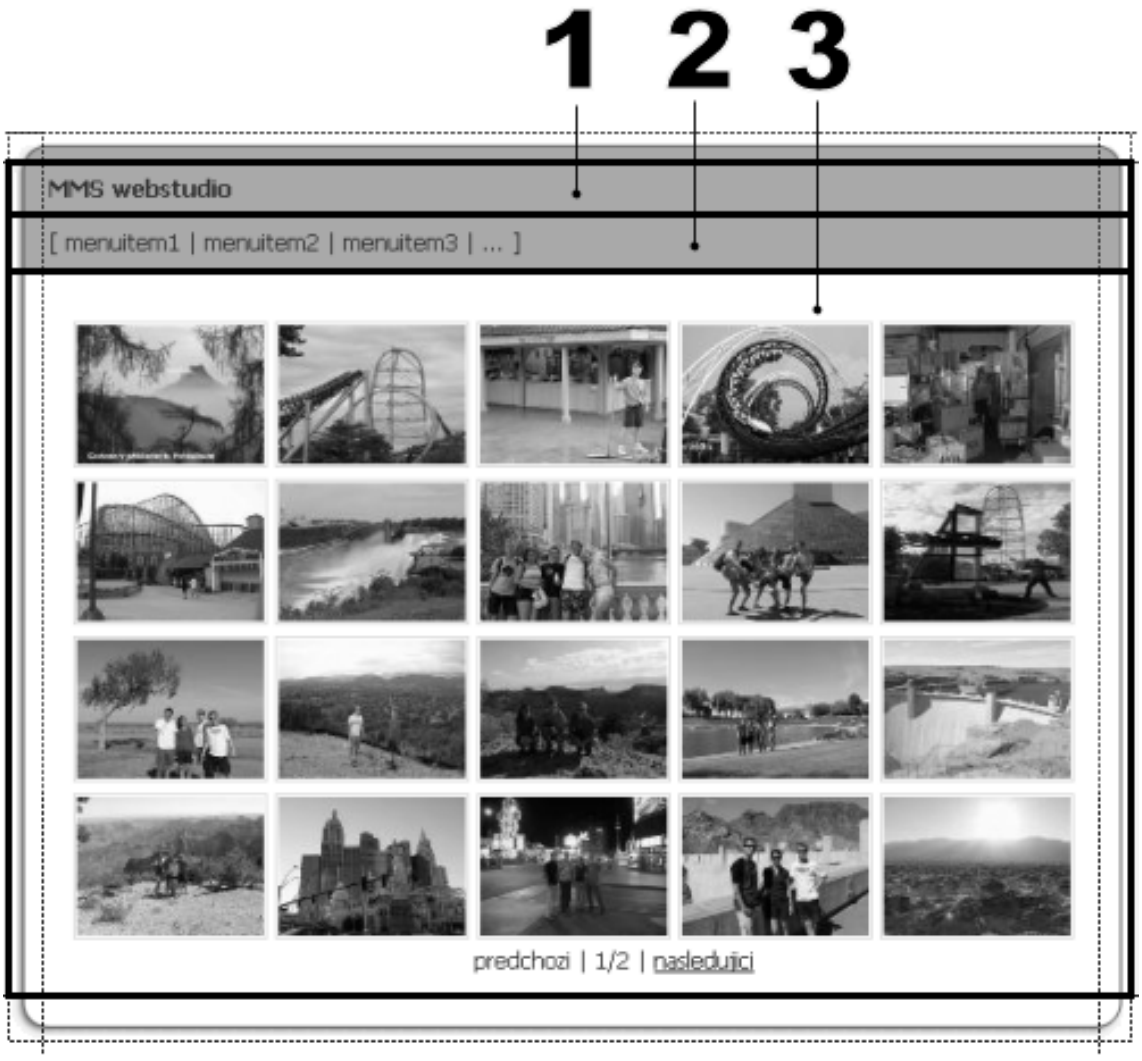
ověřit, zda je uživatel registrován a řádně přihlášen (vkládat nová data mohou jen registrovaní uživatelé). V standardní součásti Cocoonu *Cocoon forms* využívající *flow script*. V *sitemap.xmap* je definována nová roura, při požadavku na upload souboru se vyvolá *flow* funkce napsaná v JavaScriptu. Ta využívá třídy jazyku JAVA. Využil jsem Javovské třídy Cocoonu pro kontrolu formulářů. Pro uložení dat do databáze jsem vytvořil třídu v Javě umožňující převod vstupního datového toku na typ *string*(řetězec), následně uložení této struktury v binárním formátu do tabulku databáze.

Při porovnání těchto dvou přístupů kontroly formulářů je vhodnější druhý způsob pomocí Cocoon forms, tento způsob díky šablonám a modelům formuláře umožňuje efektivní oddělení obsahu od formy, dále díky standardním funkcím není třeba vytvářet vlastní funkce pro kontrolu dat.

5.2.3 Vytváření a ukládání MMS



Obrázek 5.3: schéma databáze



Obrázek 5.4: grafická podoba aplikace

Kapitola 6

Závěr

Podařilo se nastudovat informace o technologii multimediálních zpráv MMS, programovacím jazyce SMIL a open source projektu Apache Cocoon. Na základě těchto dat byla ve frameworku Cocoon vytvořena jednoduchá aplikace pro práci s MMS zprávami využívající možnosti multimediálního jazyka SMIL. Vzhledem k časové náročnosti projektu se nepodařilo implementovat některé věci zmíněné v návrhu. Jedná se o umožnění přidání zvuku ve formátu MID k MMS zprávám. I přes tyto nedostatky se celkově podařilo prezentovat princip technologie tvorby MMS zpráv a frameworku Apache Cocoon.

Co se dalšího vývoje projekty týče je nasnadě odstranění současných nedostatků v podobě přidávání zvuku k MMS zprávám, dále pak možnost vytváření video sekvencí či vytvoření jednoduchého grafického editoru pro úpravu obrázků.

Literatura

[1] Nokia, *Multimedia Messaging Services (MMS)*, Dokument dostupný na URL `http://www.nokia.com/nokia/0,8764,400,00.html` (duben 2006).

[2] Wikipedia, *Global System for Mobile Communications* , Dokument dostupný na URL `http://cs.wikipedia.org/wiki/GSM` (duben 2006).

[3] The apache cocoon project, *oficiální www stránky apache cocoon*, Dokument dostupný na URL `http://cocoon.apache.org` (duben 2006).

[4] www.root.com, *Seriál Cocoon v příkladech* , Dokument dostupný na URL `http://www.root.cz/serialy/cocoon-v-prikladech/` (duben 2006).

[5] Harold, E. R., *XML v kostce: pohotová referenční příručka*, Computer Press, Praha, 2002, ISBN 80-7226-712-4.

[6] Marchal, B., *XML v příkladech*, Computer Press, Praha, 2000, ISBN 80-7226-332-3.

[7] W3c world wide web consortium, *the synchronized multimedia integration language*, Dokument dostupný na URL `http://w3c.org/AudioVideo/` (duben 2006).

[8] Hall, M., *JAVA TM servlety a stránky JSP*, Neocortex, Praha, 2001, ISBN 80-86330-06-0.

[9] T-Mobile, *T-Mobile EDGE* , Dokument dostupný na URL `http://t-mobile.cz/Web/Residential/TarifySluzby/DatoveSluzby/T-MobileEDGE.aspx` (duben 2006).