СОФИЙСКИ УНИВЕРСИЕТ СВ "КЛИМЕНТ ОХРИДСКИ"

ФАКУЛТЕТ "МАТЕМАТИКА И ИНФОРМАТИКА"

ДИПЛОМНА РАБОТА

На тема: "Разработка на библиотека за генериране на RSS (Really Simple Syndication) и RDF(Resource Description Framework) потоци за WAP портали"

Дипломант:
Владимир Щраков
Ф.№ М21411
Спец.Разпределени системи и мобилни технологии

Научен ръководител:
Доц.д-р Силвия Илиева: .....

СОФИЯ, 2007

СЪДЪРЖАНИЕ

1 УВОД

Бързото развитие на Интернет, мобилните технологии и на цифровите технологии способства за навлизането им в нашето ежедневие, променяйки стойностите, стандартите и представите ни за комуникация и бизнес.

В съвременния живот все повече се акцентира върху интеграцията на Интернет с мобилните устройства, което довежда до революция в досегашните технологии - всеки човек, разполагащ с мобилно устройство, независимо в коя точка на планетата се намира, може да има достъп до лична или корпоративна информация в Интернет, да обработва и изпраща данни до други мобилни устройства и компютри, да пазарува, осъществява банкови операции, забавлява и всичко това в едно малко компактно мобилно устройство.

Следвайки тази тенденция се засилва нуждата от приложения за мобилни телефони, позволяващи обмен на информация към отдалечени компютри и други мобилни устройства, като същевременно начинът, по който става тази комуникация, е скрит за потребителя.

Тази тенденция от своя страна доведе по естествен начин до възникването на така наречените WAP портали. WAP порталите са средата чрез която доставчиците на съдържание (фигури, снимки, рингтонове, игри, новини, видео и т.н.) за мобилни устройства успяват да предоставят по лесен и удобен начин това съдържание, както и множество други услуги.

Един от основните проблеми, който срещат разработчиците на подобни WAP портали е свързан със съдържанието, което те предоставят на клиентите си. Този проблем се състой в това как съдържанието да бъде най-удобно предоставено на клиента. За да бъде постигнато това трябва да бъдат налице следните няколко фактора:

- Стилно и разбираемо представяне на съдържанието
- Предоставяне на клиента лесен и интуитивен достъп до съдържанието
- Обединяването на съдържанието от няколко web източника на едно място
- Категоризиране на съдържанието
- Индексиране на съдържанието с цел, лесно и бързо търсене в него

Постигането на тези цели може да се осъществи по различни начини. Една от всичките възможности е използването на технологии като RSS (Really Simple

Syndication) и RDF (Resource Description Framework). За първите три предпоставки, изброени по-горе, най-често се използват RSS потоците, които осигуряват постоянно обновяване на съдържанието, което е предназначено за клиентите. Другите два фактора обикновенно се постигат с помощта на технологията RDF. Тези две технологии ще бъдат разгледани по-подробно в отделни точки на втората част на дипломната работа, а тяхното място в спектъра от възможните (и съществуващи) други решения се дискутира в третата част.

1.1 Описание на задачата. Практическа полза от реализацията.

Дипломната работа има за цел да се проектира и разработи библиотека, позволяваща лесен и надежден начин за генериране на RSS и RDF съдържание, което да бъде използвано за изграждането на така наречените RSS/RDF feeds, които от своя страна могат да бъдат интегрирани в WAP портал.

Основните изисквания, които трябва да се реализират за пълноценната работа на системата са следните:

- Да се проектира и разработи библиотека която да предоставя удобен интерфейс, чрез който да може да бъде генерирано разнообразно RSS/RDF съдържание.
- Да се разработи примерен вариант на генериране на RSS feed отразяващ последните новини предоставени от виртуална (фиктивна) новинарска агенция.
- Да се интегрира разработения примерен вариант описан по-горе в WAP портал.

Осъществяването на конкретната разработка ще бъде базирано на стандартни библиотеки и спецификации. Това ще осигури гъвкавостта на библиотеката и възможността за използването й не само в WAP портали но и в други WEB или стандартни приложения.

1.2 Структура на дипломната работа

Първа глава е уводна, съдържа кратко въведение в темата и описание на структурата на дипломната работа.

Във втората глава са описани основните понятия и технологиите, които се използват за реализиране на основните задачи. Обосновани са архитектурните и технологични решения, взети по време на разработката на дипломната работа. Разгледани са подробно RSS и RDF форматите, както и WAP технологията, която е в основата за изграждане на приложенията за мобилните устройства.

Третата глава е посветена на сравняване на решението, което се предлага в дипломната работа със съществуващите до този момент подобни решения. Описани са предимствата на библиотеките за генериране пред „традиционните" технологии, доколкото може да се говори за традиционност в една толкова бързо развиваща се област на комуникациите. Накратко се представени и други технологии, които се използват за реализацията на приложението, каквито са PHP 5 и MySQL база данни.

Четвърта глава, озаглавена "Проектиране и реализация", описва архитектурата на отделните модули на библиотеката и каква функционалност се очаква от тях. В последствие се описва подробно механизмът на реализация на отделните модули и връзките между тях

В пета глава "Тестване и интеграция" са разгледани другите два етапа при изграждане на едно приложение—тестване и интеграция. В теза глава се описва необходимия софтуер за експлоатация на системата и физическата мрежа за внедряване на системата.

В заключението на дипломната работа са разгледани някои възможности за бъдещо развитие и обогатяване на функционалността на разработваната библиотека.

2 ОСНОВНИ ПОНЯТИЯ И ИЗПОЛЗВАНИ ТЕХНОЛОГИИ

2.1 Безжични комуникации

Безжичните комуникации са една огромна сфера, включваща в себе си всичко от радио и телевизионните вълни до пейджърите, мобилните телефони и сателитните комуникации. Напоследък се наблюдава бързо и динамично развитие в сферата на мобилните телефони, като същевременно, стандартите и протоколите, които се използват за комуникация се подобряват все повече и повече. В комбинация и в двете насоки, това развитие довежда до днешните мобилните технологии, изправяйки производителите на мобилни телефони и мобилните оператори пред нови предизвикателства.

Другата динамично развиваща се сфера от безжичния свят е тази на доставчиците на съдържание (content providers). Ежедневната нуждата от предоставяне на актуално (новини) и забавно (картинки, мелодии, игри, видео клипове) съдържание е важен фактор за съществуването на всеки WEB или WAP портал. Възниква обаче проблемът как да бъде представено това съдържание по възможно най-добрия и компактен начин, как да бъде реализирано лесното и ефективно търсене на това съдържание в дадения портал. Това води до въвеждането на различни формати на описване на данни за данните (metadata). Тема на тази дипломна работа са именно използването на два от тези формата - RSS (Really Simple Syndication) и RDF(Resource Description Framework).

В повечето от настоящите решения на проблема за предоставяне на съдържанието по най-добър и компактен начин са застъпени използването на технологии, които позволяват индексиране на съдържанието. Индексирането се използва с цел категоризиране, подреждане и бързо търсене в него, както и удобен и ефективен начин за предоставяне на бързо променящото се съдържание на клиентите.

Повечето web сайтове постигат това най-често с използването на RSS и RDF потоци. Съществуват множество библиотеки и приложения които генерират, обработват и предоставят RSS или RDF потоци в web пространството. Използването на такива потоци в света на мобилните устройства и най-вече на мобилните портали представлява интерес, на който е посветена предложената дипломна работа.

В настоящия момент почти всеки нов модел мобилен телефон има вградени приложения (така наречените RSS-четци), които могат периодично да проверяват вместо вас сайтовете, разрешени за RSS, и да извлича актуализираното съдържание.

Внедряването на такива RSS-четци или още агрегатори на информация в мобилните портали е интересен проблем, който е предмет на настоящата дипломна работа. В следващите точки ще разгледаме подробно характеристиките и възможностите на RSS и RDF технологиите.

2.2 Представяне на RSS (Really Simple Syndication) и RDF(Resource Description Framework)

В този параграф ще бъдат представени два от форматите използвани в момента за описване на данните които съществуват в интернет пространството, а именно RSS (Really Simple Syndication) и RDF(Resource Description Framework). Първо ще започнем с RSS (Really Simple Syndication).

2.2.1 RSS (Really Simple Syndication)

Ще започнем нашето описание с първия от форматите, които ще бъдат ползвани от библиотеката за генериране на метаданни, а именно RSS.

2.2.1.1 Дефиниция на RSS

RSS е семейство от web feed формати използвани за публикуването на често обновяващо се цифрово съдържание. Примери за такова съдържание са блоговете, онлайн новините и други.

Зад инициалите RSS най-често се крие едно от следните значения:

- Really Simple Syndication (RSS 2.0)
- Rich Site Summary (RSS 0.91, RSS 1.0)
- RDF Site Summary (RSS 0.9, RSS 1.0)

RSS форматите всъщност представляват структуриран по определен начин XML, който служи за обединяване на метаданни, с цел повишаване скоростта и ефективността на обмена на данни, RSS доставя данните до клиентите си под формата на XML файл, който се нарича “*RSS feed*”, “*webfeed*”, “*RSS stream*” или “*RSS channel*”.

RSS предоставя удобен начин, за тези, които искат да публикуват съдържание, да могат да разпространяват информацията в стандартизиран формат. Стандартният

формат на XML файл позволява информацията да бъде публикувана веднъж и да се разглежда от много различни програми. Често срещан пример на RSS съдържание са източниците на информация, като заглвия на новини, които периодично се актуализират.

Ползата от RSS се състои в обобщаването на цялото съдържание от няколко уеб източника на едно място. Вече не трябва да се посещават различни сайтове, за да се получи най-новата информация по темите, от които се интересуват клиентите. С RSS се доставя обобщение на съдържание и след това се взима решение кои статии да се четат, чрез щракване върху връзка.

RSS съдържанието обикновено е текстово базирано, публикувано от различни източници, но главно медийни канали или лични уеблогове – познати също като блогове. Блогът може да бъде сравнен с онлайн списание. Популярността на RSS нараства и се появяват нови типове съдържание, включително мултимедийно базирано съдържание. Споделянето на такъв тип съдържание е известно като blogcasting или podcasting. Например някои медийни канали предлагат аудио копие на техните отделни новинарски емисии.

Механизмът на доставяне на RSS съдържание е познат като RSS поток (канал). Има множество RSS канали, които се състоят от заглавия или кратки обзори на съдържанието и връзка, водеща до първоначалния източник. Каналите могат да съдържат също пълното съдържание и да включват прикачени документи от почти всеки тип. Други имена за RSS каналите са уеб канали, XML канали, RSS предавания и обобщено съдържание.

Публикуващите RSS използват канала си, като начин да персонализират своето съдържание за потребителите и да предложат връзки обратно към техните сайтове. RSS каналите са лесен начин да следите уеб или wap сайтове, новини или блогове, които четете често. Вместо да преглеждате wap пространството от много сайтове, последните новини или актуализации от тези сайтове пристигат при вас автоматично. Сравнете RSS каналите с това да имате свой собствен личен помощник, който изрязва заглавия или пише кратки обзори на уеб съдържанието, за да ви спести време. Можете бързо да видите кои елементи ви интересуват и след това да решите кои от пълните статии да прочетете.

Тъй като RSS каналите са по желание и базирани на абонаментни отношения, новото съдържание, което се доставя до вас, е точно по темите, които ви интересуват. Вие избирате за кои RSS канали да се абонирате и от кои RSS канали да се откажете.

Едно от предимствата на абонирането за RSS канал в сравнение с абонирането за имейл бюлетин или пощенски списък е това, че не предоставяте името си, имейл адреса или друга лична информация на никого. Публикуващият съдържанието няма как да се свърже с вас освен чрез RSS канали, тъй като няма вашия имейл адрес. Той също не може да предаде вашата информация на никой друг. В момента повечето RSS каналите са безплатни. Въпреки това някои съдържания, които се отнасят или са свързани с RSS канала може да изискват такса за достъп.

Обикновено един такъв *webfeed* е представен в уеб или wap сайта с една от следните три иконки ( , 

или 

).

Полезна информация и примери за RSS може да бъдат намерени на Free Validator [2], както и на RSS 2.0 at Harward Law [3].

2.2.1.2 Кратка история на RSS

Преди да се появи RSS форматът, са съществували няколко други формата за обединяване(syndication) на метаданните, но нито един не се е наложил като стандарт. Идеята за преструктуриране на метаданните свързани в уеб сайтовете възниква през 1995 когато се появява Meta Content Framework разработена в Apple Computer's Advanced Technology Group. Друга такава разработка за създаване на формат за обединяване на метаданните е осъществена от Netscape, Userland Software и Microsoft.

Историята на RSS в дати изглежда по следния начин :

- Март 1999 – Ramanathan V. Guha от Netscape създава първата версия на RSS (наречена RDF Site Summary и известна също така като RSS 0.9), която се използва в портала My Netscape.
- Юли 1999 – Dann Libby създава прототип наречен RSS 0.91 (като тук RSS означава Rich Site Summary), който опростява формата RSS 0.9 и добавя така наречения *scripting news format*, който е ориентиран преди всичко към обмен на актуалното (новините).
- 2000 – Създадена е групата RSS_DEV. Тя създава RSS 1.0, който е базиран на RDF спецификацията, но въвежда използването на повече модули в описанието на метаданните.
- 2000 – Dave Winer публикува RSS 0.92, който представлява надграждане на RSS 0.91.

- 2001 – Dave Winer създава две поредни версии RSS 0.93 и RSS 0.94.
- 2002 – Winer издава крайния вариант на RSS 0.92, който е известен като RSS 2.0 (Really Simple Syndication). Той позволява да бъдат добавяни нови елементи в XML-а, чрез използването на XML namespaces.
- 2002 – New York Times предоставя на читателите си възможността да се абонират за новинарски RSS feed.
- 2003 – Winer и Userland Software, предоставят собствеността върху формата на Harvard's Berkman Center for the Interner & Society.

Последващите исторически данни за RSS са предимно свързани с разработването на различни програми за „четене" и представяне на RSS feeds. Примери за такива са Safari 2.0 на Apple Computer, Microsoft IE, Outlook и други.

Подробности около всяка една от гореспоменатите важни дати от историята на RSS могат да бъдат намерени на сайта Wikipedia [8].

2.2.1.3 Пример за RSS

Примерни XML файлове съдържащи RSS могат да бъдат намерени в интернет пространството, но тъй като разработваната библиотека ще генерира RSS 2.0 представяме един примерен XML, в който се съдържа RSS.

Листинг 1

```
<?xml version="1.0"?>
<rss version="2.0">
   <channel>
      <title>Liftoff News</title>
      <link>http://liftoff.msfc.nasa.gov/</link>
      <description>Liftoff to Space Exploration.</description>
      <language>en-us</language>
      <pubDate>Tue, 10 Jun 2003 04:00:00 GMT</pubDate>
      <lastBuildDate>Tue, 10 Jun 2003 09:41:01 GMT</lastBuildDate>
      <docs>http://blogs.law.harvard.edu/tech/rss</docs>
      <generator>Weblog Editor 2.0</generator>
      <managingEditor>editor@example.com</managingEditor>
      <webMaster>webmaster@example.com</webMaster>
      <item>
         <title>Star City</title>
         <link>http://liftoff.msfc.nasa.gov/news/2003/news-starcity.asp</link>
         <description>How do Americans get ready to work with Russians aboard the
 International Space Station? They take a crash course in culture, language and
 protocol at Russia's &lt;a
href="http://howe.iki.rssi.ru/GCTC/gctc_e.htm"&gt;Star
City&lt;/a&gt;.</description>
         <pubDate>Tue, 03 Jun 2003 09:39:21 GMT</pubDate>
         <guid>http://liftoff.msfc.nasa.gov/2003/06/03.html#item573</guid>
      </item>
      <item>
```

```
26          <description>Sky watchers in Europe, Asia, and parts of Alaska and
27Canada will experience a &lt;a
28href="http://science.nasa.gov/headlines/y2003/30may_solareclipse.htm"&gt;partial
29 eclipse of the Sun&lt;/a&gt; on Saturday, May 31st.</description>
30          <pubDate>Fri, 30 May 2003 11:06:42 GMT</pubDate>
31          <guid>http://liftoff.msfc.nasa.gov/2003/05/30.html#item572</guid>
32       </item>
33       <item>
34          <title>The Engine That Does More</title>
35          <link>http://liftoff.msfc.nasa.gov/news/2003/news-VASIMR.asp</link>
36          <description>Before man travels to Mars, NASA hopes to design new
37engines that will let us fly through the Solar System more quickly.  The proposed
38 VASIMR engine would do that.</description>
39          <pubDate>Tue, 27 May 2003 08:37:32 GMT</pubDate>
40          <guid>http://liftoff.msfc.nasa.gov/2003/05/27.html#item571</guid>
41       </item>
42       <item>
43          <title>Astronauts' Dirty Laundry</title>
44          <link>http://liftoff.msfc.nasa.gov/news/2003/news-laundry.asp</link>
45          <description>Compared to earlier spacecraft, the International Space
46Station has many luxuries, but laundry facilities are not one of them.  Instead,
47 astronauts have other options.</description>
48          <pubDate>Tue, 20 May 2003 08:56:02 GMT</pubDate>
49          <guid>http://liftoff.msfc.nasa.gov/2003/05/20.html#item570</guid>
50       </item>
51    </channel>
52 </rss>
```

Този пример се намира на сайта на Harvard's Berkman Center for the Internet & Society.

RSS тук е представен с описание на <channel> - таг (ред 3), който съдържа различни елементи, обединени в него.

Първите няколко елемента описват източника на този канал (от 4-и до 13-и ред). Те предоставят данни като, заглавието на канала, линк на който може да бъде посетен сайта на самия източник, кратко описание, езика използван в съдържанието, датата на публикуване и други системни данни за доставчика на канала.

Следват изброяване на текущото съдържание, като последователност от <item> елементи (ред 14). Всеки един такъв елемент представлява отделна новина в конкретния случай. Всеки елемент съдържа заглавие на новината (ред 15), линк който води към самата новина (ред 16), кратко описание на новината (teaser – ред 17) и датата на публикуване (ред 22).

2.2.2 RDF (Resource Description Framework)

В този параграф ще опишем втория формат, който ще бъде използван за разработване на библиотеката за генериране. Това е RDF (Resource Description Framework).

2.2.2.1 Дефиниция на RDF

RDF (Resource Description Framework) е фамилия от спецификации, предоставени от World Wide Web Consortium (W3C), която първоначално е била замислена да служи като модел за метаданни, но в последствие прераства в метод за моделиране на информацията, чрез множество и разнообразни синтактични формати.

Моделът на метаданните, представен от RDF е базиран на идеята за описване на ресурсите с помощта на изрази от типа *предмет-предикат-обект*.

Предметът описва ресурса, като Uniform Resource Identifier (URI). Допустимо е да се използват в този формат и ресурси, които не са строго идентифицирани т.е. без имена. Някои ресурси остават не именувани и се наричат *анонимни* ресурси.

Предикатът представлява връзката между предмета и обекта.

Обектът също представлява ресурс или литерал. В семантичните уеб (web) приложения ресурсите обикновено се описват с URI, който „сочат" към истински данни, които са достъпни в интернет.

Но RDF форматът не е ограничен само с това да описва различни ресурси достъпни в интернет пространството. Не е задължително URI да сочи към реален и достъпен ресурс. Затова са дефинирани правила които са приложими за URI идентификаторите, които се описват в така наречените *речници*, какъвто е например Dublin Core Metadata.

Спецификацията на RDF може да бъде намерена на (http://www.w3.org/RDF/).

Полезна информация за RDF и примери за неговото използване в реални решения и приложения може да бъде открита и в онлайн енциклопедията Wikipedia [9].

2.2.2.2 Кратка история на RDF

Спецификацията на RDF формата има няколко предшественици. Технически най-близък е метода Meta Content Framework започнат от Ramanathan V. Guha по времето когато работи за Apple Computer и продължена с помощта на Tim Bray, в периода през който работи в Netscape Communications Corporation.

Консорциумът W3C публикува спецификация на модела на RDF и XML синтаксис през 1999. Последваща версия е публикуване през 2004. Според някои

автори ([1] , стр. 698) той предоставя способи за създаване на метаданни. Позволява създаването на именувани свойства за вашите ресурси, така че да може да описвате данните както искате.

RDF конструкцията има три характеристики:

- Subject – ресурса, който описваме
- Predicate – името на свойството, което създаваме
- Object – стойността на това свойство

Следния пример демонстрира смисъла на тези три основни конструкции.

Subject: Използването на RSS и RDF в мобилните портали

Predicate: автор

Object: Владимир Щраков

В истинските случаи на използване на RDF, субектът винаги ще бъде URI, което води към истинското съдържание. Ползата от този RDF модел е преди всичко във възможността за индексиране на съдържанието. Например, машините за търсене индексират всички думи в интернет пространството (HTML документи), които могат да намерят. Това води до неефективно търсене и предоставяне на разнородни резултати, които често нямат нищо общо с това което търсим.

Как RDF помага да избегнем този ефект? Отговорът на този въпрос се крие в така наречените RDF речници (vocabularies). Те притежават свойството да описват конкретните неща. Един от най-поулярните RDF речници е Dublin Core.

2.2.2.3 Пример за RDF

Листинг 2

```
<?xml version="1.0"?>
<rdf:description xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  about="http://www.vladeto.com/vl/contact#">
  <title>Vladimir Shtrakov Home Page</title>
  <creator>vladeto@vladeto.com</creator>
  <description>All the information you need to know about me, can be found
here!</description>
</rdf:description>
```

Нека разгледаме основните елементи изграждащи това RDF описание. Елементът <rdf: description> описва дадения ресурс. Атрибута about на <rdf: description> представя субекта, който описваме, а всеки елемент на <rdf: description> описва свойство на ресурса, като стойностите на тези елементи са стойности на свойсвтвата.

Технологията RDF е много повече от предоставеното кратко описание. Нейните характеристики са твърде разнообразни, поради което няма да можем да ги обясним изцяло в настоящата дипломна работа. Ще споменем няколко от основните проблема, които могат да бъдат решени с използването на RDF. Технологията може да бъде използвана при изграждането на карти на сайтове (site maps), класифициране на съдържанието в web, както и за описанието на статии, новини и друга информация, съхраняващи се в електорнни масиви.

Подробна информация какво представлява RDF може да бъде намерена на сайта съдържащ спецификацията на RDF [4]. Там може да бъде намерена и информация за стандарта Dublin Core.

Разработчиците на приложения използващи RDF технологията могат да намерят полезна информация, примери и насоки за своите разработки на известния и популярен сайт www.w3schools.com, и още по-конкретно в тази негова част, посветена на RDF [5]. Информацията и насоките, предоставени на този сайт бяха особено полезни в процеса на разработката на библиотеката, описана в дипломната работа.

2.3 Wireless Application Protocol (WAP)

Съкратено от *Wireless Application Protocol* (Протокол за безжични приложения) - приложна среда и пакет от протоколи, позволяващи незабавен достъп на потребителят до информация и услуги през безжични устройства, като мобилни телефони, пейджъри, смарт-телефони и комуникатори (WAP устройства). Тази технология е разработена от Phone.com (първоначално Unwired Planet) заедно с Motorola, Nokia и Ericsson и е базирана на съществуващите Интернет (Internet) технологии като XML (eXtended Mark-up Language) и IP (Internet Protocol). Тя използва така наречените микробраузери (microbrowsers) – браузери (browsers), които обработват малки по обем, информационни блокове които са приспособени за сравнително малката памет на WAP-устройствата и ниската скорост на трансфер на данни, характерни за масово разпространените безжични мрежи в момента.

Протоколът WAP се поддържа от всички операционни системи (ОС). Има и такива ОС, специално разработени за безжични устройства: PalmOS, EPOC, Windows CE, FLEXOS, OS/9 и JavaOS. Целта на този стандарт е да предоставя Интернет информация и услуги до потребителите на WAP устройства.

2.3.1 Преимущества на WAP-технологията

WAP предоставя на крайните потребители бърз и лесен достъп до необходимите Интернет данни и услуги, като унифициран трансфер на информацията, банкиране и забавления през техните мобилни устройства. Интранет (Intranet) информация, като корпоративна база от данни например, също може да бъде достъпна чрез WAP технологията. Потребителят има възможност да получава и изисква информация чрез сигурна, бърза и ниско струваща среда, факт, който прави WAP услугата по-атрактивна за потребителите, изискващи повече стойност и функционалност от своите мобилни устройства.

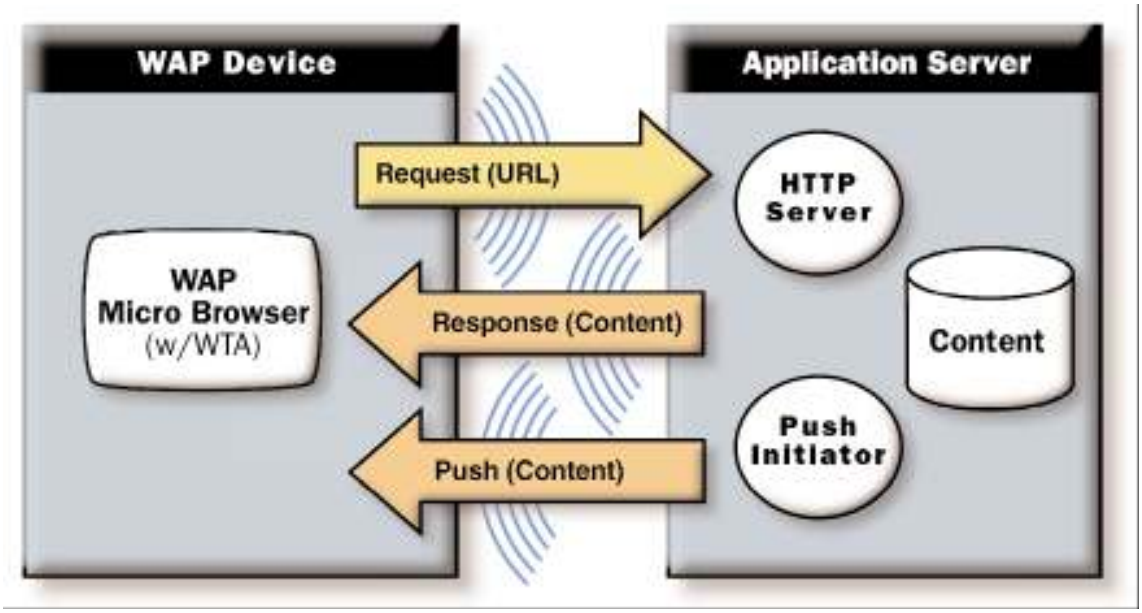
WAP предоставя редица предимства и за мобилните оператори. Технологията способства намаляване на разходите и нарастване на броя на абонатите, благодарение на увеличаващите се по вид и по обем информационни услуги. Това води до популяризиране на услугите предоставяни от мобилните оператори и съответно на доставчиците на тези услуги. Ползите от това са няколко:

- Увеличаване на броя киенти (предизвикване на клиентския интерес с нови и разнообразни новини и услуги)
- Увеличаване на печалбите от трафик и посещение на мобилните портали
- Популяризиране на цялата марка (бранд) на мобилния оператор

2.3.2 WAP портали

В настоящ точка ще дефинираме най-важните атрибути на понятието WAP портал, ще опишем от какви компоненти е изграден, един WAP портал, какви технологии са използвани в изграждането на един такъв портал, както и какви услуги може да предостави на клиентите си даден WAP портал.

Знаейки какво представляват съвременните WEB портали не е трудно да се досетим какво се крие зад понятието WAP портал. Той предоставя онлайн съдържание, което става достъпно за клиентите, чрез браузъра на мобилните им телефони. На *фиг. 1* е изобразен начинът по който се осъществява достъпът до даден WAP портал.



Фиг. 1 Достъп до WAP портал

На *Фиг. 1* можем да идентифицираме основните компоненти върху, които можем да изградим един WAP портал.

Те са:

1.WAP - micro browser

2. HTTP – server

3. Rush – lnltalator

Както се вижда HTTP – сървъра и Rush-инициализатора са елементи на един специален сървер – Application Server.

Технологиите които се използват при разработката на един такъв портал са разнообразни, но основните са:

- WML/xHTML – езикът който „разбират" нашите мобилни телефони
- HTTP – протоколът, който се използва за преноса на данни от мобилното устройство до сървера, на който е инсталиран порталът и обратно.

Най-интересното в един WAP портал са наборът и типът услуги, които той предоставя на своите клиенти. Основните услуги които един WAP портал предоставя са новини, картинки, мелодии, рингтонове, телевизия, VoD (video on demand). Също така WAP порталите предоставят и интерактивни услуги като чат, блог, онлайн пазаруване на стоки и др.

2.4 PHP 5 и DOM

Тъй като библиотеката е разработена с програмния език PHP, в този параграф ще обърнем внимание на версията, библиотеките и средата с която е разработена.

Изискванията към разработката наложиха използването на PHP 5 и в частност DOM библиотеката, включена в тази дистрибуция. Функциите, които се изпълняват от DOM библиотеката са използвани за ефективното и коректно генериране на XML съдържанието, което реално RSS и RDF форматите представляват.

Обектно- ориентираният модел на PHP 5, спомага за по-доброто структуриране на отделните модули в библиотеката за генериране на RSS и RDF файлове. Това структуриране позволява последващо разширяване и подобряване на библиотеката.

В описанието на проектирането и реализацията на библиотеката по-нататък ще разгледаме основните функции предоставени от PHP и DOM [7] които са използвани. Информацията и характеристиките на тези функции са предоставени в оналйн наръчника за PHP [6].

3 СРАВНЕНИЕ СЪС СЪЩЕСТВУВАЩИ РЕШЕНИЯ

Преди проектирането и реализацията на библиотеката за генериране на RSS и RDF бяха разгледани и проучени няколко съществуващи вече решения. Тъй като началната идея на библиотеката по спецификация беше основана на използването на PHP разгледаните съществуваща решения също бяха разработени на PHP. В настоящата глава ще опишем накратко две от тях и ще обясним защо се наложи да ги отхвърлим и да развием собствено софтуерно решение.

3.1 RSS Feed Generator v1.0

Първата библиотека, която беше проучена преди започването на проектирането и реализацията беше RSS Feed Generator v1.0 предоставена от 2rss.com. Тази библиотека е разработена в два варианта. Единия е на PHP, а другият на ASP. Някои от възможностите и параметрите, които предоставя библиотеката са следните:

- rss_server – име на сървър на който има инсталирана MySQL или MS SQL база от данни.
- rss_db, rss_user и rss_pass – име на базата от данни и потребителските данни за достъп.
- rss_connection – валиден connection string.
- channel_copyright, channel_editor, channel_webmaster – параметри описващи RSS канала.
- image_title, image_url, image_link, image_width, image_height – параметри описващи картинка/ снимка дефинирана в RSS потока.

Библиотеката предоставя набор от функции с име add_channel, които генерират RSS канал, съдържащ различни данни в зависимост от подадените параметри.

Базирайки се на горното описание възникнаха няколко предпоставки за отхвърлянето на библиотеката, като вероятно решение, което да бъде използвано и доразвито така, че да отговаря на техническата спецификация. Тези предпоставки са:

- фиксиран набор от тагове включени в генерирането – изискванията към библиотеката е да поддържа генерирането на произволен набор от XML тагове (пример: <vf:teaser>Read the latest news …</vf:teaser>)

- невъзможност за разширяване на текущото решение с добавяне на нова функционалност, което да не изисква отделяне на усилия (effort) (1 човекоден - man/day)
- липса на многократна използваемост (reuse)

3.2 RAP - RDF API for PHP V0.9.4

RAP - RDF API for PHP V0.9.4 е PHP библиотека за изграждане на семантични уеб (semantic web) приложения. Разработката на тази библиотека е започнала като open source проект в Свободен университет Берлин през 2002 година и след това продължава да се развива с помощта на вътрешни и външни сътрудници. Последната версия на библиотеката съдържа:

- API за манипулиране на RDF графите като набор от изрази
- API за манипулиране на RDF графите като набор от ресурси
- RDF/XML парсъри
- RDF/XML сериалайзъри
- Съхраняване на модела в паметта или в база данни
- Поддръжка на езика RDQL
- Графичен интерфейс за управление на RDF моделите, съхранени в база данни
- Поддръжка на по-често срещаните RDF речници (vocabulary).
- Други

Освен безспорните качества тази разработка има два съществени недостатъка, които са главната причина да търсим собствено решение. Те са:

- не предоставя интерфейс за генериране на файлове
- не се използват и не се отчитат предимствата на различните мобилни апарати при генерирането на файлове.

Изброените характеристики и функционалност на RAP впечатляват, но основната причина да се спрем на собствено разработена библиотека беше нуждата от надеждно и сравнително просто и олекотено решение, което да позволява генерирането на RDF файлове бързо и ефективно. Изискването към библиотеката е да може да генерира RDF файлове описващи съдържанието на даден WAP портал, като едновременно да се

съобразява с това, за какъв точно модел мобилен телефон е предназначено съдържанието.

От описанието на библиотеката RAP, можем да заключим, че тя представлява разработка, която може да бъде използвана за индексиране, изграждане на графи от RDF съдържание, както и търсене на съдържание.

След като разгледахме основните характеристики на горепосочените решения и след представните няколко теста на тези приложения, следва да обясним защо бе избран именно този вариант за разработка на текущата библиотека.

Предпоставките за това са няколко:

- лесен и удобен начин за генериране на RSS и RDF потоци
- възможност за бъдещо разширяване на функционалността в зависиомст от евентуални нови изисвкания

Първата предпоставка улеснява програмистите, които биха използвали разработваната библиотека при писането на код генериращ RSS и/или RDF потоци. Разполагайки с информационни потоци (входни данни от новинарски агенции), лесно програмистът може да генерира RSS и/или RDF поток. Това става чрез инстанциране на основния клас на библиотеката и извикване на няколко функции.

Добавянето на нов подпакет към съществуващата библиотека е възможно при появата на нови изисквания. Например появата на нов инфорамционен поток, който трябва да бъде представен в различен формат. Това е така благодарение на използвания модулен обектно-ориентиран подход при структурирането и дизайна на библиотеката.

4 ПРОЕКТИРАНЕ И РЕАЛИЗАЦИЯ

Основната цел на разработването на библиотеката е да се осигури надежден, бърз, лесен за употреба и разширяем софтуерен продукт, който да бъде използван от други приложения, чиято цел е да изградят RDF описание на данните/ съдържанието с които оперират. За да може всичко това да бъде реализирано е необходимо разработваната библиотека да бъде разделена логически на два модула, съответно единия да съдържа функционалността за генериране на RSS, а другият за генериране на RDF.

4.1 Техническа спецификация

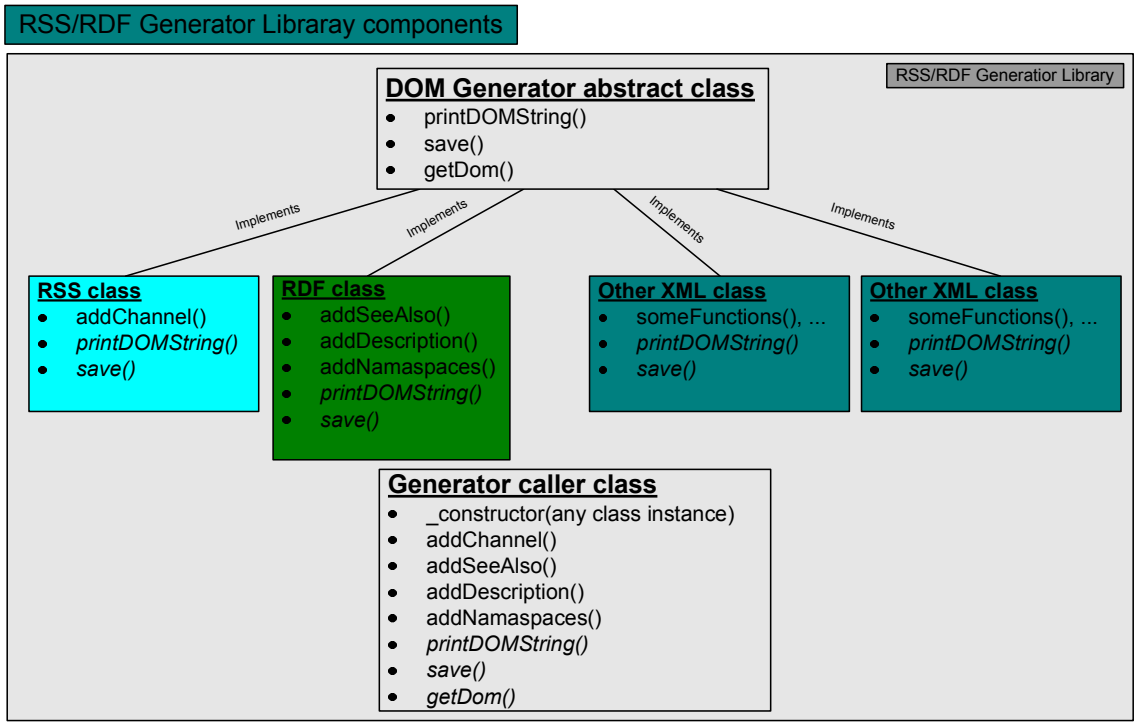
Този параграф съдържа кратка версия на техническа спецификация на разработваната библиотека.

4.1.1 Функционални изисквания

Основните функционални изисквания към библиотеката са:

1. Програмен интерфейс, който предоставя функции за:
    - Генериране на RSS или RDF съдържание
    - Записване на генерираното съдържание
    - Отпечатване на генерираното съдържание
    - Валидиране на генерираното съдържание спрямо XML Schema
2. Предоставяне на набор от класове, които да описват специфичните RSS и RDF компоненти, дефинирани в XML схемите на различни доставчици на RSS или RDF потоци от информация. Пълният набор от специфични за определени XML тагове и атрибути не са упоменати в текущото кратко изложение на техническата спецификация на библиотеката.
3. Възможност за разширяване на библиотеката – добавяне на функционалност за генериране на други XML-базирани формати.

На следващата фигура (Фиг. *2 Функционални компоненти на библиотеката*) са представени – основните компоненти на библиотеката, като са описани и основните функции на всеки един от нейните компоненти.



Фиг. 2 Функционални компоненти на библиотеката

4.1.2 Технически изисквания

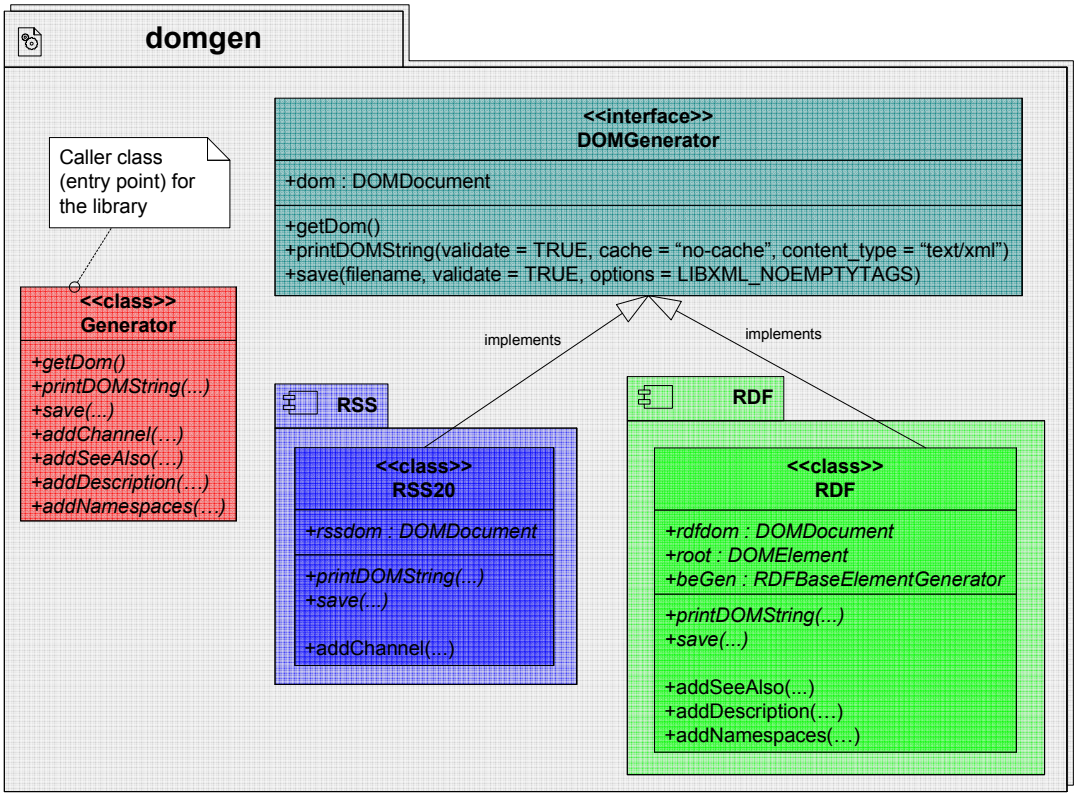
Основните технически изисквания към библиотеката са:

1. Обектно ориентиран модел разработен на базата на PHP 5
2. Документация на всички методи, класове и интерфейси на библиотеката
3. Тестови скриптове, които да гарантират/ тестват коректната работа на библиотеката, както и да служат за пример, как библиотеката трябва да бъде използвана от външни приложения.
4. Библиотеката трябва да работи на Apache Web Server, върху който е инсталирана дистрибуция на PHP 5.x.

4.2 Проектиране на библиотеката

За да удовлетвори всичките технически и функционални изисквания, библиотеката е разработена с помощта на обекно-ориентирания модел, който ни предоставя последната PHP 5.x дистрибуция. В настоящият параграф, ще опишем структурата на библиотеката, пакетите и основните класове които я изграждат.

Ще започнем с клас диаграма (*Фиг. 3 Клас диаграма на библиотеката*), описваща цялостната структура на библиотеката и нейните пакети.



Фиг. 3 Клас диаграма на библиотеката

На горната клас диаграма (*Фиг. 3 Клас диаграма на библиотеката*) са изобразени основните компоненти на разработваната библиотека. Пакетът *domgen* съдържа всички тези компоненти. От диаграмата (*Фиг. 3 Клас диаграма на библиотеката*) се вижда, че библиотеката е изградена от два пакета, а именно *rss* и *rdf*. Техните основни класове съответно имплементират интерфейса *DOMGenerator*, който дефинира основните функции, които библиотеката предоставя. Както се последната компонента от диаграмата е извикващия клас *Generator*. Той е разработен с цел да осигури лесен и централизиран достъп до библиотеката от клиентските приложения. Следва да опишем как е проектиран всеки един от компонентите, изобразени на диаграмата.

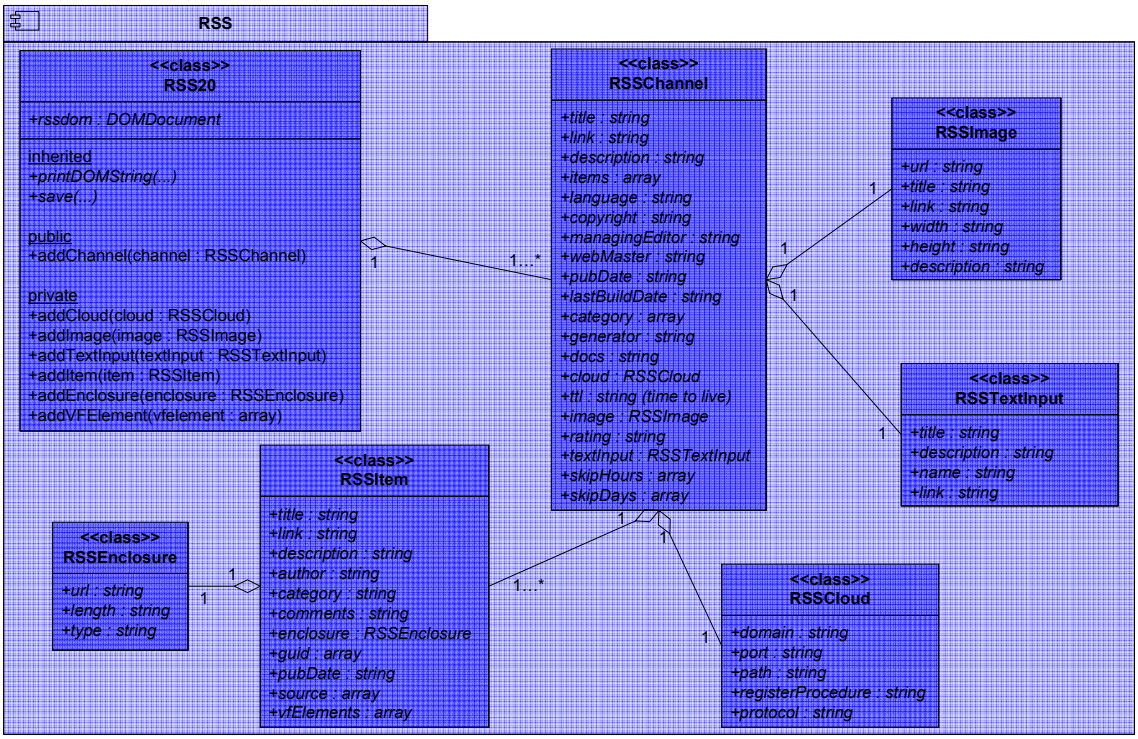
Ще започнем с интерфейсът *DOMGenerator*. Както споменахме той описва основните характеристики и функции, които библиотеката ще предоставя. Те са:

- *dom : DOMDocument* – инстанция от тип *DOMDocument*, която ще съдържа DOM обектното дърво, което ще бъде използвано в процеса на генериране на XML документа.

- *getDom()* – функция, която връща тази инстанция от тип *DOMDocument*.
- *printDOMString()* – функция, предоставяща възможността DOM обектното дърво да бъде отпечатено като стринг. Първият параметър на функцията определя дали XML документът да бъде валидиран спрямо съответна XML Schema. Следващите два параметъра определят типа на генерирания документ и chace header-а, който да бъде използван.
- *save()* – позволява XML документът, описан от DOM обектното дърво, да бъде записан във файл.

Преди да продължим с описанието на двата пакета *rss* и *rdf* да се спрем на извикващия клас *Generator*. Той представлява входна точка за всеки един клиент на библиотеката. За тази цел класът съдържа, както двете функции дефинирани в интерфейса *DOMGenerator*, така и функциите дефинирани в основните класове на пакетите *rss* и *rdf*. Тези функции ще разгледаме подробно в описанието на пакетите, по-нататък в дипломната работа.

Описанието на пакетите на библиотеката ще започнем с *rss* пакета. Основният клас в този пакет е наречен *RSS20*. Той имплементира главния интерфейс в библиотеката *DOMGenerator* и добавя специфични функции нужни при генерирането на XML документ с формата дефиниран от RSS 2.0 спецификацията. Преди да опишем подробно този клас и останалите класове нека разгледаме клас диаграмата (*Фиг. 3 Клас диаграма на rss пакета*) на пакета.



Фиг. 4 Клас диаграма на rss пакета

Както е видно от диаграмата (*Фиг. 4 Клас диаграма на rss пакета*) основният клас *RSS20* предоставя една основна функция. Това е функцията *addChannel(channel : RSSChannel)*. Чрез тази функция се извършва добавянето на RSS канал към XML документа. Описанието на канала и XML елементите и атрибутите, които го изграждат е реализирано, чрез останалите класове в пакета. Те представляват обектно описание на основните XML елементи и атрибути дефинирани в RSS 2.0 спецификацията. Елементите и атрибутите са представени, като публични променливи на класовете, които са достъпни директно от инстанцията на класа.

Описването на XML атрибутите и елементите по този начин, в класове, позволява добавяне на нови елементи и атрибути в случай на промяна или разширение на спецификацията. От диаграмата (*Фиг. 4 Клас диаграма на rss пакета)* също така се вижда, че някои от класовете съдържат инстанции от останалите класове (*aggregation*). Пример за това е основния описателен клас *RSSChannel*. Всъщност той може да съдържа в себе си инстанции от всички останали описателни класове. От своя страна пък класът *RSSItem* също така агрегира *RSSEnclosure*. Това решение в процеса на проектиране, позволява лесно добавяне на нови класове в бъдеще и свързването им със

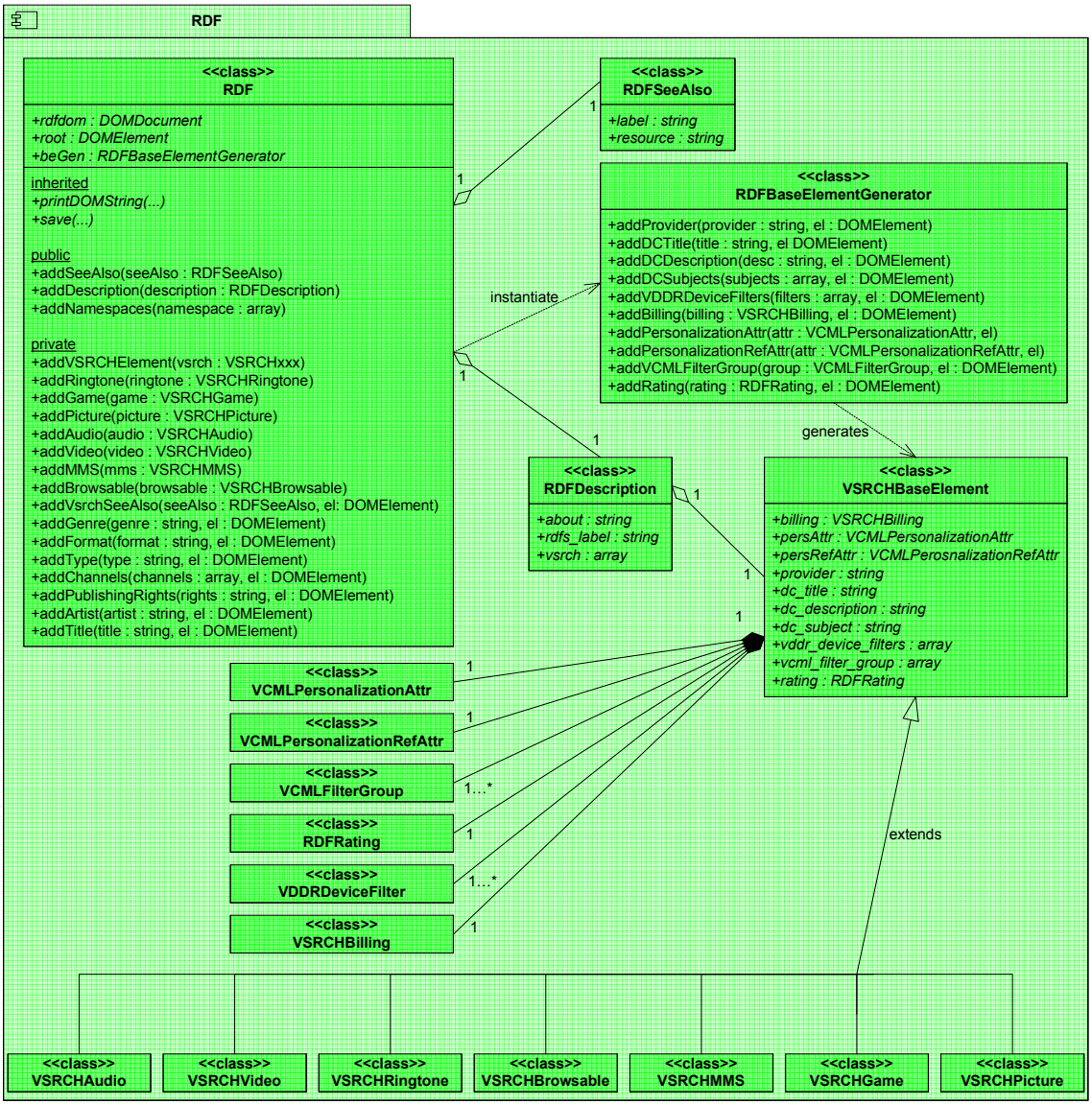
съществуващите вече такива. Така при поява на нови RSS елементи, те могат лесно да бъдат интегрирани в *rss* пакета.

Описаните класове, които се използват в процеса на генериране, са следните:

- *RSSChannel* – обектно представяне на RSS 2.0 <channel> елемента.
- *RSSImage* – обектно представяне на RSS 2.0 <image> елемента.
- *RSSItem* – обектно представяне на RSS 2.0 <item> елемента.
- *RSSTextInput* – обектно представяне на RSS 2.0 <textInput> елемента.
- *RSSCloud* – обектно представяне на RSS 2.0 <cloud> елемента.
- *RSSEnclosure* – обектно представяне на RSS 2.0 <enclosure> елемента.

Бизнес логиката (алгоритъмът) за генерирането на XML документи с RSS формат се базира на тези описателни класове. Тя е реализирана както споменахме в основния клас *RSS20* на пакета *rss*.

Освен описанието на този пакет от голямо значение е и проектирането на пакета *rdf*. Подходът при проектирането на този пакет е подобен на този, който беше използван и при *rss* пакета. Атрибутите и елементите на RDF XML документа са описани пак с помощта на класове, предоставящи стойностите на атрибутите и елементите, като публични променливи. Създаден е и основен клас *RDF*, в който са реализирани основните функции заложени в пакета. Преди да пристъпим към подробно описание на всички тези класове, нека разгледаме клас диаграмата (*Фиг. 5 Клас диаграма на rdf пакета*) на *rdf* пакета.



Фиг. 5 Клас диаграма на rdf пакета

Описанието на пакета ще започнем с дефинициите на описателните класове. Бяха дефинирани два основни класа, следвайки RDF спецификацията. Това са класовете *RDFSeeAlso* и *RDFDescription*. Техните характеристики могат да бъдат открити в дадената по-горе клас диаграма (*Фиг. 5 Клас диаграма на rdf пакета*). Специфичното е, че класът *RDFDescription* съдържа масив (*vsrch : array*) от *VSRCHBaseElement* обекти. Тези обекти задават специфични атрибути и елементи дефинирани в XML Schema, описваща RDF формата, дефиниран в съответствие с нуждите на конкретен доставчик. Ще изброим тези класове, но няма да се спираме подробно на всеки един от тях. Важно е да се отбележи, че те всички наследяват *VSRCHBaseElement* класа, като добавят

допълнителни специфични характеристики към, него в зависимост от атрибутите и елементите, заложени в RDF XML Schema. Тези класове са: *VSRCHAudio, VSRCHVideo, VSRCHRingtone, VSRCHBrowsable, VSRCHMMS, VSRCHGame* и *VSRCHPicture*. Те всички описват съдържание, което реално WAP порталът предоставя на клиентите си. Това съдържание бива индексирано с цел лесно търсене и предоставяне на клиентите на портала.

Трябва да отбележим също така, че *VSRCHBaseElement* класът, събира в себе си инстанции на няколко други описателни класове, които задават други специфични характеристики. Това са класовете: *VCMLPersonalizationAttr, VCMLPersonalizationRefAttr, VCMLFilterGroup, RDFRating, VDDRDeviceFilter* и *VSRCHBilling*. Няма да описваме подробно всеки един от тези класове. Те дефинират специфични характеристики на съдържанието публикувано на портала, като например за какви мобилни устройства е предназначено то, персонализация на клиентите, както и класифициране на съдържанието, относно специфични характеристики на клиентите, като възраст, пол, религия и др.

На базата на тези описателни класове е изградена бизнес логиката за генериране на RDF XML документи. Тя е реализирана в два класа, а именно *RDF* и *RDFBaseElementGenerator*. Това разделяне на имплементацията в два класа е резултат от стремежа за по-ясно и лесно реализиране на логиката и механизма за генериране на RDF документи. Класът *RDFBaseElementGenerator* съдържа базови функции, които генерират основните XML атрибути и елементи, които трябва да съдържа RDF документът. От своя страна класът *RDF* съдържа инстанция на класа *RDFBaseElementGenerator*, чрез която извиква тези основни методи. Класът *RDFBaseElementGenerator* не се използва директно (няма директно извикване) от външни клиентски класове. Неговите функции се използват само в основния за пакета клас *RDF*, който от своя страна осъществява интерфейса *DOMGenerator* и всички негови основни функции.

С това завършва описанието на проектирането на разработваната библиотека. Реализацията на повечето класове и компоненти на библиотеката са разгледани в параграф 4.4 Реализация на основните компоненти на библиотеката.

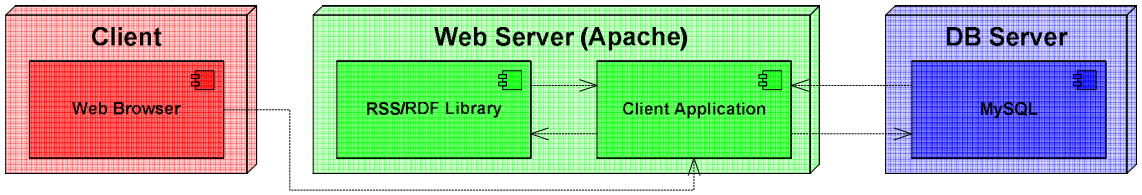
4.3 Проектиране на клиентско приложение

Следвайки техническите изисквания към библиотеката, стигаме до етапа на проектиране на клиентско приложение.

Разработването на клиентско приложение ще има няколко цели, а именно:

- да предостави скриптове, които да служат за пример при разработването на други клиентски приложения
- цялостно тестване на библиотеката, нейните пакети и компоненти
- демонстрация на функционалността която разработваната RSS/RDF библиотека предоставя

За постигането на тези цели са използвани два PHP скрипта, които в долната диаграма са обозначени като компонента с име Client Application. Реалният клиент в случая Web Browser чрез http протокола осъществява достъп до тези скриптове, които от своя страна извличат данни от базата и чрез помощта на RSS/RDF библиотеката генерират съответните RSS или RDF документи и ги принтират в уеб браузъра.



Фиг. 6 Пример за интегриране на библиотеката и клиентско приложение

На горната диаграма (*Фиг. 6 Пример за интегриране на библиотеката и клиентско приложение*) е представена точно тази последователност от действия, които ще ни послужат за постигане на всички изброени по-горе цели, поставени пред клиентското приложение. С постигането на тези цели не само удовлетворяваме техническите изисквания към библиотеката, но предоставяме и примерно приложение, което може да бъде много полезно за разработчиците на приложения, които биха използвали тази библиотека за генериране на RSS или RDF документи. Също така осигуряваме тестови скриптове, които могат да бъдат използвани и разширявани в последствие (например, ако библиотеката бъде разширена с нова функционалност).

Последното което трябва да отбележим, е че за целите на тази дипломна работа няма да използваме достъп до база данни, самите клиентски скриптове ще съдържат тестови данни, които ще бъдат използвани за генерирането на RSS или RDF документи.

4.4 Реализация на основните компоненти на библиотеката

В настоящият параграф ще разгледаме в детайли реализацията на всички по-важни класове, интерфейси и компоненти на библиотека.

За яснота ще следваме последователността на описание на компонентите, която използвахме при съставянето на параграф 4.2 Проектиране на библиотеката. Което означава, че ще започнем с основния интерфейс на библиотеката *DOMGenerator*, който дефинира основните функции, достъпни за клиентите на библиотеката. Както споменахме в 4.2, той предоставя следните характеристики и функции: *dom : DOMDocument, getDom()*, *printDOMString()* и *save()*. Основният интерфейс сме реализирали, чрез следния код:

Листинг 3

```
/**
 * @package domgen
 */

/**
 * Abstract class defining functionality for generation of DOM documents.
 *
 * @version 1.0
 * @author Vladimir Shtrakov
 * @copyright Copyright 2007
 * @package domgen
 */
abstract class DOMGenerator
{
        /** @var DOMDocument The DOM document instance. */
        protected $dom = NULL;

        /**
         * The DOMGenerator constructor.
         *
         * @param string namespaceURI The namespace URI of the document element to
         * create.
         * @param string $qualifiedName The qualified name of the document element to
         * create.
         * @param $doctype The type of document to create or NULL.
         */
        protected function __construct($namespaceURI, $qualifiedName, $doctype)
        {
                $impl = new DOMImplementation();
                if($doctype)
                {
                        $this->dom = $impl->createDocument($namespaceURI,
                                $qualifiedName, $doctype);
                }
                else
                {
                        $this->dom = $impl->createDocument($namespaceURI,
                                $qualifiedName);
                }
        }

        /**
         * Returns the DOMDocument instance.
         *
         * @return DOMDocument The DOMDocument instance.
         */
        public function getDom()
        {
```

```
            return $this->dom;
        }

        /**
         * Prints the DOM object.
         *
         * @param boolean $validate Flag for validation against XML Schema or DTD.
         * @param string $cache Flag for header cache directive.
         * Possible values:
         * "public" - Cache for all
         * "private" - Cache by MSISDN
         * "no-cache" - No cache
         * @return string The DOM object as XML string.
         */
        abstract public function printDOMString($validate = TRUE, $cache = "no-cache",
            $content_type = "text/xml");

        /**
         * Dumps the internal XML tree back into a file.
         *
         * @param string $filename The path to the saved XML document.
         * @param boolean $validate Flag for validation against XML Schema or DTD.
         * @param integer $options Additional Options. Currently only
         * LIBXML_NOEMPTYTAG is supported.
         */
        abstract public function save($filename, $validate = TRUE,
            $options = LIBXML_NOEMPTYTAG);
}
```

Основните класове на всеки един пакет на библиотеката е желателно да имплементират този интерфейс. По този начин ще бъде гарантирано, че какъвто и XML документ да бъде генериран от даден пакет, той ще предостави на клиентите основните методи, които са дефинирани на ниво функционалност на библиотеката.

Ще пропуснем разглеждането на кода дефиниращ извикващия клас *Generator*, защото той реално представлява обвивка на всички текущи публични функции предоставени от двата пакета на библиотеката. Неговия код може да бъде намерен в приложеното към дипломната работа CD.

Продължаваме с пакета *rss* и с това как е реализиран той. Описващите класове от този пакет както казахме се състоят от публични променливи, които описват специфичните за формата елементи и атрибути, поради което няма да се спираме подробно на кода им. По-голям интерес представлява основният клас в пакета, а именно *RSS20*. Освен наследените от интерфейса *DOMGenerator* методи, този клас предоставя един публичен метод наречен *addChannel()*. В тялото на този метод се намират извикванията към частни методи на класа, които предоставят логиката на генерирането на RSS документа. Следващият код съдържа дефиницията на този клас и на основните методи, произтичащи от него. Самите тела на методите не се съдържат в кода за да се постигне по-компактно представяне и яснота.

Листинг 4

```
/**
 * Implementation of the DOMGenerator for Really Simple Syndication (RSS)
 * documents generation.
 * The RSS 2.0 spec is released through Harvard under a Creative Commons
 * license. 7/15/03.
 *
 * @version 1.0
 * @author Vladimir Shtrakov
 * @copyright Copyright 2007
 * @package domgen
 * @subpackage rss
 */
class RSS20 extends DOMGenerator
{
        private $rssdom = null;
        private $root = null;

        /**
         * The RSS20 constructor.
         */
  function __construct()
  {
  }

        /**
         * Prints the DOM object.
         *
         * @param boolean $validate Flag for validation against XML Schema.
         * @param string $cache Flag for header cache directive.
         * Possible values:
         * "public" - Cache for all
         * "private" - Cache by MSISDN
         * "no-cache" - No cache
         * @return string The DOM object as XML string.
         */
        public function printDOMString($validate = TRUE, $cache = "no-cache",
                $content_type = "text/xml")
        {
        }

        /**
         * Dumps the internal XML tree back into a file.
         *
         * @param string $filename The path to the saved XML document.
         * @param boolean $validate Flag for validation against XML Schema or DTD.
         * @param integer $options Additional Options. Currently only
         * LIBXML_NOEMPTYTAG is supported.
         */
        public function save($filename, $validate = TRUE, $options = NULL)
        {
        }

        /**
         * Adds channel element into the RSS 2.0 document.
         *
         * @param RSSChannel The RSSChannel instance.
         */
        public function addChannel($channel)
        {
        }

        /**
         * Adds 'cloud' element into the RSS 2.0 document.
         *
         * @param RSSCloud The RSSCloud instance
         * @return DOMElement The DOMElement to be added into the RSS 2.0 document.
         */
        private function addCloud($cloud)
        {
        }

        /**
         * Adds 'image' element into the RSS 2.0 document.
         *
```

```
     * @param RSSImage The RSSImage instance
     * @return DOMElement The DOMElement to be added into the RSS 2.0 document.
     */
    private function addImage($image)
    {
    }

    /**
     * Adds 'textInput' element into the RSS 2.0 document.
     *
     * @param RSSTextInput The RSSTextInput instance
     * @return DOMElement The DOMElement to be added into the RSS 2.0 document.
     */
    private function addTextInput($textInput)
    {
    }

    /**
     * Adds 'item' element into the RSS 2.0 document.
     *
     * @param RSSItem The RSSItem instance
     * @return DOMElement The DOMElement to be added into the RSS 2.0 document.
     */
    private function addItem($item)
    {
    }

    /**
     * Adds 'enclosure' element into the RSS 2.0 document.
     *
     * @param RSSEnclosure The RSSEnclosure instance
     * @return DOMElement The DOMElement to be added into the RSS 2.0 document.
     */
    private function addEnclosure($enclosure)
    {
    }

    /**
     * Adds element that extend the RSS 2.0 Specification.
     *
     * @param array $vfElement The element to be added.
     * vfElement:
     * array
     * (
     *             "element" => array(element_name => element_value),
     *             "attributes" => array(attribute_name => attribute_value),
     *             "children" => array(index => vfElement)
     * )
     * Children array will be added recursively to the DOMElement.
     *
     * @return DOMElement The DOMElement to be added into the RSS 2.0 document.
     */
    private function addVFElement($vfElement)
    {
    }

}
```

Както личи от имената на *private* методите на класа, всеки един от тях се грижи за генерирането на XML елемент, описан от съответен клас. Вижда се, че параметрите на тези методи са обекти от типа на съответния описващ клас, съдържащи конкретни данни, от които да бъде генериран съответния XML елемент.

Така реализиран класът *RSS20* може да бъде инициализиран директно от дадено клиентско приложение. Това е възможен подход, но изисква клиентският код да включва референция към (чрез require_once например) всички класове използвани в *RSS20*. Това неудобство е една от причините за появяването на споменатия по-горе

извикващ клас *Generator*. В кода си той включва референции към всички необходими класове от библиотеката и така облекчава разработчика на клиентското приложение използващо описаната RSS/RDF библиотека.

Остава да разгледаме начина по който е реализиран другият пакет от библиотеката. Отново няма да се спираме подробно на кода на описващите класове включени в *rdf* пакета, защото те по подобие на *rss* класовете са обикновено изображение между XML атрибути/ елементи и променливи на класа. Кодът на тези класове, както и на цялата библиотека могат да бъдат намерени в приложеното към дипломната работа CD.

Сега да пристъпим към реализацията на основните два класа от *rdf* пакета. Ще започнем с класа *RDFBaseElementGenerator*. Както беше споменато в параграфа 4.2. Проектиране на библиотеката този клас съдържа методи които се грижат за генерирането на основните RDF XML елементи. За да представим нагледно как точно са дефинирани тези методи да разгледаме разпечатката на кода на този клас, дадена по долу. Отново телата на методите са премахнати, с цел яснота и краткост на представянето на класа.

Листинг 5

```
/**
 * Provides basic functions for adding the common elements for each RDF
 * document.
 *
 * @version 1.0
 * @author Vladimir Shtrakov
 * @copyright Copyright 2007
 * @package domgen
 * @subpackage rdf
 */
class RDFBaseElementGenerator
{
        private $rdfdom = null;

  function __construct($rdfDocument)
  {
        $this->rdfdom = $rdfDocument;
  }

  /**
   * Adds the 'vsrch:provider' element.
   *
   * @param string $provider The provider.
   * @param DOMElement $vsrchElement The DOMElement in which basic elements
   * will be added.
   */
  public function addProvider($provider, $vsrchElement)
  {
  }

  /**
   * Adds the 'dc:title' element.
   *
   * @param string $dc_title The dc:title string.
   * @param DOMElement $vsrchElement The DOMElement in which basic elements
   * will be added.
```

```
     */
    public function addDCTitle($dc_title, $vsrchElement)
    {
    }

    /**
     * Adds the 'dc:description' element.
     *
     * @param string $dc_description The dc:description string.
     * @param DOMElement $vsrchElement The DOMElement in which basic elements
     * will be added.
     */
    public function addDCDescription($dc_description, $vsrchElement)
    {
    }

    /**
     * Adds the 'dc:subject' elements.
     *
     * @param string|array $dc_subject The dc:subject string or array of strings.
     * @param DOMElement $vsrchElement The DOMElement in which basic elements
     * will be added.
     */
    public function addDCSubjects($dc_subjects, $vsrchElement)
    {
    }

    /**
     * Adds the 'vddr:device' or 'vddr:ignore-device' or 'vddr:class' elements.
     *
     * @param VDDRDeviceFilter|array $vddr_device_filters VDDRDeviceFilter
     * instance or array of such VDDRDeviceFilter instances.
     * @param DOMElement $vsrchElement The DOMElement in which basic elements
     * will be added.
     */
    public function addVDDRDeviceFilters($vddr_device_filters, $vsrchElement)
    {
    }

    /**
     * Adds the billing attributes.
     *
     * @param VSRCHBilling $billing VSRCHBilling instance.
     * @param DOMElement $vsrchElement The DOMElement in which basic elements
     * will be added.
     */
    public function addBilling($billing, $vsrchElement)
    {
    }

    /**
     * Adds the personalizationAttr attributes.
     *
     * @param VCMLPersonalizationAttr $personalizationAttr VCMLPersonalizationAttr
     * instance.
     * @param DOMElement $vsrchElement The DOMElement in which basic elements
     * will be added.
     */
    public function addPersonalizationAttr($personalizationAttr, $vsrchElement)
    {
    }

    /**
     * Adds the personalizationRefAttr attributes.
     *
     * @param VCMLPersonalizationRefAttr $personalizationRefAttr
     * VCMLPersonalizationRefAttr instance.
     * @param DOMElement $vsrchElement The DOMElement in which basic elements
     * will be added.
     */
    public function addPersonalizationRefAttr($personalizationRefAttr,
          $vsrchElement)
    {
    }

    /**
     * Adds the 'filter' elements.
```

```
   *
   * @param VCMLFilterGroup $filter_group VCMLFilterGroup instance.
   * @param DOMElement $vsrchElement The DOMElement in which basic elements
   * will be added.
   */
  public function addVCMLFilterGroup($filter_group, $vsrchElement)
  {
  }

  /**
   * Adds the 'rating' elements.
   *
   * @param RDFRating|PMLContainerRestriction $rating
   * RDFRating|PMLContainerRestriction instance.
   * @param DOMElement $vsrchElement The DOMElement in which basic elements
   * will be added.
   */
  public function addRating($rating, $vsrchElement)
  {
  }

  /**
   * Adds the 'vddr:device' or 'vddr:ignore-device' or 'vddr:class' elements.
   *
   * @param VDDRDeviceFilter $vddr_device_filter VDDRDeviceFilter instance.
        */
  private function addVDDRDeviceFilter($vddr_device_filter)
  {
  }
}
```

Така описаните методи са основата върху която е реализиран основният клас *RDF*. Той освен наследените от интерфейса методи предоставя още три публични метода. Тяхната дефиниция, както и дефиницията на *private* методите на класа са представени на следващия листинг:

Листинг 6

```
/**
 * Implementation of the DOMGenerator for Resource Description Framework (RDF)
 * documents generation.
 *
 * @version 1.0
 * @author Vladimir Shtrakov
 * @copyright Copyright 2007
 * @package domgen
 * @subpackage rdf
 */
class RDF extends DOMGenerator
{
   private $rdfdom = null;
   private $root = null;

   private $beGen = null;


  /**
   * The RDF constructor
   */
   function __construct()
   {
   }

  /**
   * Prints the DOM object.
   *
   * @param boolean $validate Flag for validation against XML Schema.
   * @param string $cache Flag for header cache directive.
   * Possible values:
```

```
     * "public" - Cache for all
     * "private" - Cache by MSISDN
     * "no-cache" - No cache
     * @return string The DOM object as XML string.
     */
     public function printDOMString($validate = TRUE, $cache = "no-cache",
                $content_type = "text/xml")
     {
     }

    /**
     * Dumps the internal XML tree back into a file.
     *
     * @param string $filename The path to the saved XML document.
     * @param boolean $validate Flag for validation against XML Schema or DTD.
     * @param integer $options Additional Options. Currently only
     * LIBXML_NOEMPTYTAG is supported.
     */
     public function save($filename, $validate = TRUE, $options = NULL)
     {
     }

    /**
     * Adds rdfs:seeAlso element into the RDF document.
     *
     * @param RDFSSeeAlso $seeAlso The RDFSSeeAlso instance.
     */
     public function addSeeAlso($seeAlso)
     {
     }

    /**
     * Adds rdf:Description element into the RDF document.
     *
     * @param RDFDescription The RDFDescription instance.
     */
     public function addDescription($description)
     {
     }

    /**
     * Adds all necessary XML namespace definitions.
     *
     * @param array $namespace All namespaces to be added in the RDF document.
     */
     public function addNamespaces($namespaces)
     {
     }

    /**
     * Adds vsrch element into the RDF document.
     *
     * @param VSRCH-prefixed $vsrch VSRCH-prefixed instance.
     */
     private function addVSRCHElement($vsrch)
     {
     }

    /**
     * Adds vsrch:ringtone element into the RDF document.
     *
     * @param VSRCHRingtone $vsrch The VSRCHRingtone instance.
     */
     private function addRingtone($vsrch)
     {
     }

    /**
     * Adds vsrch:game element into the RDF document.
     *
     * @param VSRCHGame $vsrch The VSRCHGame instance.
     */
     private function addGame($vsrch)
     {
     }

    /**
```

```
     * Adds vsrch:picture element into the RDF document.
     *
     * @param VSRCHPicture $vsrch The VSRCHPicture instance.
     */
     private function addPicture($vsrch)
     {
     }

    /**
     * Adds vsrch:audio element into the RDF document.
     *
     * @param VSRCHAudio $vsrch The VSRCHAudio instance.
     */
     private function addAudio($vsrch)
     {
     }

    /**
     * Adds vsrch:video element into the RDF document.
     *
     * @param VSRCHVideo $vsrch The VSRCHVideo instance.
     */
     private function addVideo($vsrch)
     {
     }

    /**
     * Adds vsrch:MMS element into the RDF document.
     *
     * @param VSRCHMMS $vsrch The VSRCHMMS instance.
     */
     private function addMMS($vsrch)
     {
     }

    /**
     * Adds vsrch:browseable element into the RDF document.
     *
     * @param VSRCHBrowseable $vsrch The VSRCHBrowsealbe instance.
     */
     private function addBrowseable($vsrch)
     {
     }

    /**
     * Adds 'rdfs:seeAlso' element into the vsrch element.
     *
     * @param RDFSSeeAlso $rdfs_seeAlso The RDFSSeeAlso instance.
     * @param DOMElement $vsrchElement The DOMElement in which element will be
     * added.
     */
     private function addVsrchSeeAlso($rdfs_seeAlso, $vsrchElement)
     {
     }

    /**
     * Adds 'vsrch:genre' element into the vsrch element.
     *
     * @param string $genre The genre.
     * @param DOMElement $vsrchElement The DOMElement in which element will be
     * added.
     */
     private function addGenre($genre, $vsrchElement)
     {
    }

    /**
     * Adds 'vsrch:format' element into the vsrch element.
     *
     * @param string $format The genre.
     * @param DOMElement $vsrchElement The DOMElement in which element will be
     * added.
     */
     private function addFormat($format, $vsrchElement)
     {
     }
```

```
    /**
     * Adds 'vsrch:type' element into the vsrch element.
     *
     * @param string $type The type.
     * @param DOMElement $vsrchElement The DOMElement in which element will be
     * added.
     */
    private function addType($type, $vsrchElement)
    {
    }

    /**
     * Adds 'vsrch:channel' element(s) into the vsrch element.
     *
     * @param string|array $channels The channels.
     * @param DOMElement $vsrchElement The DOMElement in which element will be
     * added.
     */
    private function addChannels($channels, $vsrchElement)
    {
    }

    /**
     * Adds 'vsrch:publishing-rights' element(s) into the vsrch element.
     *
     * @param string $publishing_rights The publishing rights expiration date.
     * @param DOMElement $vsrchElement The DOMElement in which element will be
     * added.
     */
    private function addPublishingRights($publishing_rights, $vsrchElement)
    {
    }

    /**
     * Adds 'vsrch:artist' element into the vsrch element.
     *
     * @param string $artist The artist.
     * @param DOMElement $vsrchElement The DOMElement in which element will be
     * added.
     */
    private function addArtist($artist, $vsrchElement)
    {
    }

    /**
     * Adds 'vsrch:title' element into the vsrch element.
     *
     * @param string $title The title.
     * @param DOMElement $vsrchElement The DOMElement in which element will be
     * added.
     */
    private function addTitle($title, $vsrchElement)
    {
    }
}
```

Така реализираните методи на *RDF* класа, удовлетворяват функционалните изисквания към библиотеката. Добавянето на нови елементи и нова функционалност става лесно. Както се вижда от реализацията е необходимо просто добавяне на описващ клас и след това реализиране на функция, която да генерира XML елемента, описан от този клас. С това приключва описанието на реализацията на основните компоненти изграждащи разработваната библиотека.

От всички кодове поместени в дипломната работа се вижда, че всички променливи, методи, класове и интерфейси са документирани. Използвана е нотация която е основа за създаване на необходимата документация на библиотеката чрез

phpDocumentor. Тази техническа документация може да бъде намерена също така в приложеното CD към настоящата дипломна работа.

4.5 Реализация на клиентско приложение

Разработването на клиентско приложение е не по-малко важно от самото разработване на библиотеката. Както беше споменато по-горе в дипломната работа, целите пред това клиентско приложение са няколко и затова ще разгледаме начина по който е реализирано.

Поради фактът, че разработваната библиотеката предоставя възможност за генериране на два формата XML документи клиентското приложение се състои от два PHP скрипта. Всеки един от тях реализира целите поставени пред клиентското приложение, съответно за определения XML формат. Двата скрипта са наречени, съответно *rss20.php* и *rdf.php*. Намират се в директория `test` от проектната директория на библиотеката.

Реализацията на двата скрипта се осъществява, чрез следните стъпки:

- инициализиране на извикващия клас *Generator*
  - `$generator = new Generator(new RSS20());` - за RSS генерирането
  - `$generator = new Generator(new RDF());` - за RDF генерирането
- създаване на обекти на описващите класове, инициализирани с тестови данни
- извикване на функции за генериране на XML елементи, дефинирани в основните класове на двата пакета (*RSS20* и *RDF*)
- извеждане на резултат

Така реализираното клиентско приложение изпълнява всички условия дефинирани за него във функционалната и техническата спецификация на библиотеката. Кодовете на двата клиентски скрипта могат да послужат, като отправна точка на всеки разработчик, който трябва да използва разработваната библиотека. Също така с редовете:

```
* @example ../../test/rss20.php How to use the Generator for RSS 2.0.
* @example ../../test/rdf.php How to use the Generator for RDF.
```

добавени в класа *Generator* се указва на phpDocumentor да добави кодовете на двата скрипта към документацията на библиотеката, както и да генерира линкове към тези

кодове в документацията на класа *Generator*. Кодовете на двата скрипта са дадени в приложеното към дипломната работа CD.

5 ТЕСТВАНЕ И ИНТЕГРАЦИЯ

След като описахме проектирането и реализацията на основните компоненти на разработваната библиотека следва да дефинираме процесите на тестване и интеграция на библиотеката.

5.1 Тестване

Тестването е основен процес от разработката на даден проект, бил той голям или сравнително малък като настоящата библиотека. При тестването могат да се открият и своевременно да се отстраняват появилите се грешки, на различни етапи – проектиране или разработка.

Голяма част от грешките проявили се след разработването на библиотеката бяха отстранени след използването на клиентското приложение описано в параграфи 4.3 Проектиране на клиентско приложение и 4.5 Реализация на клиентско приложение.

Поради спецификата на употребата на библиотеката и на клиентските приложения, в процеса на разработката й, се наложи да се използват софтуерни емулатори, които до голяма степен да имитират дейността на мобилните устройства и да улеснят процеса на разработка. Конкретно за тестването на тази библиотека беше използван web browser Mozilla Firefox с инсталирани набор от pluggins предоставящи основните характеристики на едно мобилно устройство. Този набор от плъгин приложения за Mozilla Firefox се състои от:

- DOM Inspector
- Modify Headers
- User Agent Switcher
- Web Developer
- Force Content-Type

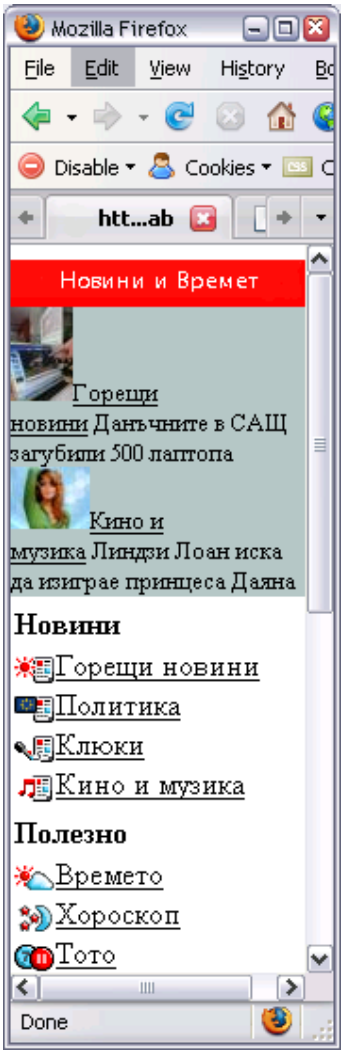
Повече информация относно всяка една от изброените надстройки на Mozilla Firefox може да бъде намерена на сайта на браузъра: http://www.mozilla.com/en-US/firefox/

Може да се каже, че за разработването и тестването на библиотеката сме използвали по-силни софтуерни и хардуерни средства от тези при мобилните телефони, с който се симулират и моделират реалните клиентски приложения. Този подход има

предимство с това, че по време на разработката и тестването разполагаме с по-големи ресурси във всяко едно отношение.

Процесът на тестване на реални клиентски приложения е свързан с използването на множество тестови данни и сценарии. Поради това за целите на дипломната работа ще дадем само отделни екрани, показващи резултатите от няколко направени тестове.

Ще започнем с екран показващ резултата от тестването на реален RSS канал, който съдържа последните невинни от деня. Преди това обаче ще разгледаме RSS XML документа, който библиотеката е генерирала:




```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
  <channel>
    <title> Горещи новини</title>
    <link/>
    <description>BGNES RSS Feed</description>
    <item>
      <title> Горещи новини</title>
      <link/>
      <description>BGNES RSS Feed</description>
      <pubDate>2007-04-09 10:21:10</pubDate>
      <vf:teaser> Данъчните в САЩ загубили 500
лаптопа</vf:teaser>
      <vf:article-
link>index.php?cat_id=6&id=82982</vf:article-link>
      <vf:image href="image.php/82982/3213/img"
caption="rss" mime-
          type="image/jpeg"/>
    </item>
  </channel>
</rss>
```

Така генерираният XML документ може да бъде интегриран в WAP портал и резултатът от това изглежда по следния начин. На екрана в дясно можем да различим представянето на генерирания от библиотеката RSS XML документ. Както се вижда той е най-отгоре на страницата и е оцветен в сив фон. Елементът *title* със съдържание „Горещи новини” е използван от рендъринг система за създаване на линк, който сочи към адреса описан в елемента *vf:article-link*. След което е изписан текстът, съдържащ се в елемента *vf:teaser*. И последният елемент обработен и визуализиран от генерирания RSS XML документ е *vf:image*. Стойността на неговия атрибут *href* се подава на специален имидж сървер, който в зависимост от модела на мобилното устройство, което има достъп до портала,

обработва картинката и я визуализира по възможно най-добрия начин за това устройство. В теста Mozilla Firefox емулира, чрез User Agen Swithcer – SonyEricsson Z1010.

След като видяхме реален тест на частта от библиотеката, което генерира RSS XML документи е интересно разглеждането на резултат и от функционалността предоставяща ни възможност за генериране на RDF XML документи. Преди да пристъпим към самите резултати, ще разясним целта на индексирането на съдържанието достъпно в WAP портала. Тъй като множество клиенти посещават въпросният портал с множество различни мобилни устройства е логично съдържанието да бъде филтрирано по този параметър. Това означава, че клиент използващ например устройство с марката Nokia не би трябвало да може да има достъп до съдържание което е специфично примерно за устройства с марката SonyEricsson. Красноречив пример за това са игрите. Както знаем производителите на игри за мобилни устройства предоставят отделни версии на игрите си съответно съобразени с възможностите на специфичните устройства, върху които те ще бъдат инсталирани.

Този факт, както и много други подобни характеристики на съдържанието и това как да бъде предоставено възможно най-добре и коректно на клиента водят до нуждата от RDF документи за описването на съдържанието на даден WAP портал. За нуждите на нашия тест ще уточним, че отново ще използваме Mozilla Firefox емулираща, чрез плъгина User Agent Switcher мобилното устройство SonyEricsson Z1010.

Както в предишния тестов сценарии, ще разгледаме първо генерирания RDF XML документ и след това ще разгледаме резултата от визуализурането на този документ на портала.

Листинг 7

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rating="http://purl.org/vcml/rating/1.0/"
xmlns:srch="http://purl.org/vcml/search#"
xmlns:filter="http://purl.org/vcml/filter/1.0/" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:prism="http://prismstandard.org/namespaces/1.2/basic/"
xmlns:er="http://purl.org/vcml/sdp/er/1.0/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xsi:schemaLocation="http://www.somecompany.com/schema/VZV-PARTNER.xsd">

<rdfs:seeAlso rdfs:label="повече" rdf:resource="..."/>
<rdf:Description about="..." rdfs:label="повече">
         <vsrch:audio>
                  <vsrch:channel/>
                  <vsrch:provider>CDP</vsrch:provider>
                  <dc:title>Waste My Time</dc:title>
                  <vsrch:artist>The Brand New Heavies</vsrch:artist>
                  <vsrch:title>Waste My Time</vsrch:title>
                  <dc:description>Waste My Time</dc:description>
                  <vsrch:format>RealMusic</vsrch:format>
```

```
                    <rdfs:seeAlso rdfs:label="Купи" rdf:resource="..."/>
                    <vddr:device>Alcatel OT-S853</vddr:device>
                    <vddr:device>LG KE800</vddr:device>
                    ...
                    <vddr:device>Motorola C975</vddr:device>
                    ...
                    <vddr:device>Nokia 3250</vddr:device>
                    ...
                    <vddr:device>Panasonic VS6</vddr:device>
                    ...
                    <vddr:device>Sagem my800v</vddr:device>
                    <vddr:device>Samsung D500</vddr:device>
                    ...
                    <vddr:device>Sharp 902SH</vddr:device>
                    <vddr:device>SonyEricsson K320i</vddr:device>
                    ...
                    <vddr:device>SonyEricsson Z1010</vddr:device>
                    ...
                    <dc:rights>
                    <prism:expirationTime/>
                    </dc:rights>
          </vsrch:audio>
</rdf:Description>
</rdf:RDF>
```

Генерирания RDF XML документ съдържа множество изброени модели устройства и с цел компактно представяне на съдържанието, всички модели устройства с изключение на първото за всяка марка са заменени с три точки. Също така истинските URL адреси от атрибутите *rdf:resource* и *about* са заменени също така с три точки. От съдържанието на RDF XML документа ясно виждаме по какъв начин са описани метаданните за в случая аудио файл. Също така виждаме, че устройството, с което сме направили заявката е в списъкът: <vddr:device>SonyEricsson Z1010</vddr:device>

Това означава, че трябва да можем да открием на портала този аудио файл и да разгледаме съдържанието му. На екрана в ляво можем да открием описаното, в RDF XML документа, съдържание на четвърто място в секцията, съдържаща описание на файлове достъпни за устройството което използваме по време на теста.

Метаданните, описващи разглежданото от нас съдържание, са подредени и визуализрани коректно от рендъринг система. Ако последваме генерирания линк "Waste My Time" порталът ще зареди за нас описателна страница съдържаща повече информация за заявеното от нас съдържание. Това можем да видим на следващия екран.

В него вече виждаме подробното описание на заявеното съдържание. Предоставена е информация за:

- името на изпълнителят
- името на песента
- кратък номер с който съдържанието може да бъде заявено, чрез SMS
- формат на файла
- размер на файла
- цена която ще бъде начислена към сметката на потребителят, ако той реши да свали съдържанието на своето мобилно устройство.

Тази информация естествено варира в зависимост от вида на съдържанието, което клиентът завява. Именно затова разработваната библиотека предоставя всички разгледани в параграф 4.2 Проектиране на библиотеката описателни класове свързани с вида на съдържанието. Това са всички класове, започващи с *VSRCH* и наследяващи базовия описателен клас *VSRCHBaseElement*.

С това завършва описанието на процеса на тестване на основните методи, класове, интерфейси и компоненти на разработваната библиотека за генериране на RSS/RDF XML документи.

5.2 Интеграция на библиотеката

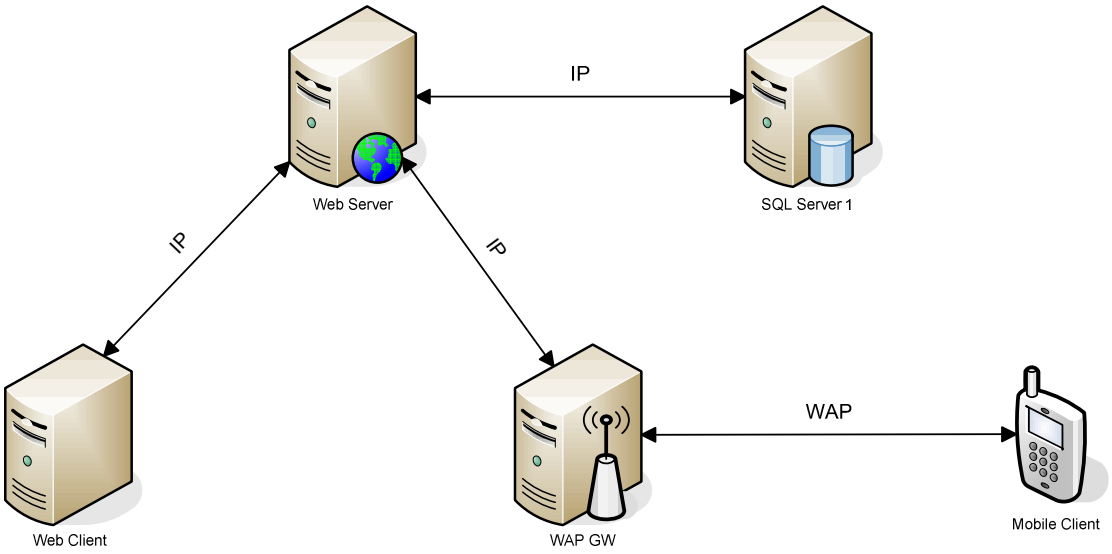
За да може да бъде използвана библиотеката в реална ситуация е необходимо да бъде инсталирана дистрибуция на PHP 5.x, като в конфигурацията е включен пакетът DOM. Класовете на библиотеката не се нуждаят от други библиотеки за да работят коректно.

Библиотеката може да бъде интегрирана и по начин подобен на този илюстриран на *Фиг. 6 Пример за интегриране на библиотеката и клиентско приложение* от параграф 4.3 Проектиране на клиентско приложение. Естествено компонентите от

илюстрацията и техните взаимовръзки могат да бъдат различни в зависимост от сценария описващ евентуалното използване на разработваната библиотека.

5.3 Внедряване на библиотеката

На диаграмата (*Фиг. 7 Физическо внедряване и използване на библиотеката*). е показана примерна физическа мрежа от компоненти за внедряване на разработваната библиотека. Както е показано мобилните устройства се свързват, чрез безжична мрежа до Web сървъра и използвайки WAP протокола, като преди това връзката минава през WAP Gateway. В ляво е изобразена и възможността Web клиент да се свърже директно към Web сървъра и да използва разработваната библиотека. Разглежданата схема изобразява реална постановка, която би могла да се използва като система за обслужване на множество клиенти (production). Идеята е по-скоро да се даде пример за система, в която разработваната библиотека може да бъде внедрена.



Фиг. 7 Физическо внедряване и използване на библиотеката

6 ЗАКЛЮЧЕНИЕ

В настоящата глава ще опишем евентуални възможности за бъдещо развитие и разширение на разработената библиотека, като в края ще предоставим и обобщение на темата на дипломната работа.

6.1 Заключение

В настоящата дипломна работа бяха разгледани етапите по проектирането, изграждането и реализацията на библиотека, предоставяща възможността за генериране на XML документи с RSS и RDF формат.

Реализираната библиотека дава възможност за лесно и коректно генериране на XML документи със споменатите два формата. Все по-бързо нарастващото използване на RSS и RDF потоците в глобалната мрежа, оправдава усилията вложени в разработката на подобна библиотека.

В дипломната работа основно се акцентира върху двата формата RSS и RDF. Както и върху начина по който лесно и удобно да бъде генериран документ реализиращ един от двата формата. Също така беше отделено внимание на проектирането и разработването на библиотека, базирана на PHP и DOM. Тези две технологии, както и обектно- ориентирания подход при разработването на библиотеката, спомогнаха за осъществяването на идеята за нея.

Тя остава отворена за бъдещо усъвършенстване, съобразно с възможностите на новите технологии и изискванията на потребителите.

Библиотеката е базирана на високопроизводителни съвременни технологии, които позволяват лесна поддръжка, възможности за бързо приспособяване към новите изисквания и разширяемост.

Обектно-ориентираната архитектура на библиотеката предполага лесно мигриране на кода към друг обектно-ориентиран език за програмиране. С не много услилия библиотеката може да бъде реализирана и с помощта на езика за програмиране Java. Това би се наложило в случай на изискване към платформата, на която библиотеката би следвало да работи.

6.2 Възможности за бъдещо развитие

Разработената от мен библиотека предполага бъдещо развитие в много и разнообразни аспекти. В настоящия параграф ще разгледаме няколко от тези аспекти.

1. Първият е разширяването на съществуващите пакети за генериране на файлове RSS и RDF формат. Това би се наложило, ако някоя от спецификациите на тези два формата бъде доразвита и обогатена с нови полезни XML елементи и атрибути. Изхождайки от изложените описания в параграф 4. Проектиране и реализация можем да твърдим, че това разширение няма да представлява трудност или проблем.

2. Вторият важен аспект за развитие на разработваната библиотека е нейното разширяване с нови пакети. Като всеки един от тези нови пакети би могъл да предоставя функционалност за генериране на документи със специфичен XML формат. Този аспект идва да покаже предимството на разработването на библиотеката, чрез използването на отделни модули, обединени от интерфейс, предоставящ основните функции на библиотеката.

3. Друг аспект за бъдещото развитие е разработването на версии на библиотеката на други програмни езици като Java, C++, C# и други, в зависимост от спецификата на приложенията и системите които биха използвали разработената библиотека. Това може да се постигне благодарение на ясните и коректно дефинирани класове и обекти на библиотеката. Естествено добре специфицираните формати RSS и RDF улесняват дизайна, проектирането и реализирането на подобни версии на библиотеката.

7 ИЗПОЛЗВАНА ЛИТЕРАТУРА

1. „Програмиране с XML”, Дейвид Хънтър, Кърт Кейгъл, Дейв Гибънс, Никола Озу, Джон Пинок, Пол Спенсър, 2001
2. Спецификация на RSS 2.0 - http://feedvalidator.org/docs/rss2.html
3. Спецификация на RSS 2.0 - http://cyber.law.harvard.edu/rss/rss.html
4. Спецификацията на RDF формата - http://www.w3.org/RDF/
5. RDF в сайта w3schools - http://www.w3schools.com/rdf/default.asp
6. PHP онлайн документация - http://bg2.php.net/manual/en/index.php
7. Спецификация на DOM - http://www.w3.org/DOM/
8. http://en.wikipedia.org/wiki/RSS_2.0
9. http://en.wikipedia.org/wiki/Resource_Description_Framework

8 РЕЧНИК

Really Simple Syndication (RSS) – семейство от web feed формати, използвани за публикуването на често обновяващо се дигитално съдържание

Resource Description Framework (RDF) – фамилия от спецификации, предоставени от World Wide Web Consortium (W3C), която първоначално е била замислена да служи за модел за метаданни, но в последствие прераства в метод за моделиране на информацията, чрез множество и разнообразни синтактични формати

Extensible Markup Language (XML) – текстов формат наследник на SGML, служещ за описание на разнообразни и разнородни данни

Document Object Model (DOM) – езиково и платформено независим интерфейс, който позволява на програми и скриптове динамично да достъпват и променят съдържанието на документи

Wireless Application Protocol (WAP) – приложна среда и пакет от протоколи, позволяващи незабавен достъп на потребителят до информация и услуги през безжични устройства като мобилни телефони

PHP – език за програмиране използван при разработката на библиотеката

9 ПРИЛОЖЕНИЯ

Кодът на настоящата дипломна работа е предоставен в прилежащо към нея CD. Неговото съдържание е:

1. Кодът на всеки един от класовете, описани в дипломната работа
2. Кодът на двата тестови скрипта използвани за тестване на функционалността на библиотеката
3. Пълна документация на класовете в HTML формат
4. Резюме на дипломната работа на български и английски език

В настоящата глава ще представим листинг с кода на основният интерфейс на библиотеката, както и на двата тестови скрипта използвани при тестването на библиотеката. Кодът на останалите класове изграждащи библиотеката могат да бъдат намерени на приложеното към дипломната работа CD.

Листинг 8 – Основнитя интерфейс на библиотеката DOMGenerator.class.php

```
<?php
/**
 * @package domgen
 */

/**
 * Abstract class defining functionality for generation of DOM documents.
 *
 * @version 1.0
 * @author Vladimir Shtrakov
 * @copyright Copyright 2007
 * @package domgen
 */
abstract class DOMGenerator
{
      /** @var DOMDocument The DOM document instance. */
      protected $dom = NULL;

      /**
       * The DOMGenerator constructor.
       *
       * @param string namespaceURI The namespace URI of the document
element to
       * create.
       * @param string $qualifiedName The qualified name of the document
element to
       * create.
       * @param $doctype The type of document to create or NULL.
       */
      protected function __construct($namespaceURI, $qualifiedName,
$doctype)
      {
            $impl = new DOMImplementation();
```

```
            if($doctype)
            {
                  $this->dom = $impl->createDocument($namespaceURI,
                        $qualifiedName, $doctype);
            }
            else
            {
                  $this->dom = $impl->createDocument($namespaceURI,
                        $qualifiedName);
            }
      }

      /**
       * Returns the DOMDocument instance.
       *
       * @return DOMDocument The DOMDocument instance.
       */
      public function getDom()
      {
            return $this->dom;
      }

      /**
       * Prints the DOM object.
       *
       * @param boolean $validate Flag for validation against XML Schema or
DTD.
       * @param string $cache Flag for header cache directive.
       * Possible values:
       * "public" - Cache for all
       * "private" - Cache by MSISDN
       * "no-cache" - No cache
       * @return string The DOM object as XML string.
       */
      abstract public function printDOMString($validate = TRUE, $cache =
"no-cache",
            $content_type = "text/xml");

      /**
       * Dumps the internal XML tree back into a file.
       *
       * @param string $filename The path to the saved XML document.
       * @param boolean $validate Flag for validation against XML Schema or
DTD.
       * @param integer $options Additional Options. Currently only
       * LIBXML_NOEMPTYTAG is supported.
       */
      abstract public function save($filename, $validate = TRUE,
            $options = LIBXML_NOEMPTYTAG);
}
?>
```

Листинг 9 – Тестов скрипт за RSS пакета

```
<?php
/**
 * Sample script for RSS 2.0 Generator usage.
 */
$inc_path = "../lib/";
require_once($inc_path . "domgen/Generator.class.php");
```

```
// Create the RSS20 Generator.
$generator = new Generator(new RSS20());

// Create RSS Cloud.
$cloud = new RSSCloud(
      "www.provider.bg", // domain
      "80", // port
      "/RPC2", // path
      "providerCloud.rssPleaseNotify", // registerProcedure
      "xml-rpc" // protocol
      );

// Create RSS Image.
$image = new RSSImage(
      "http://localhost/rss_rdf_generator/test/6_57ea2.jpg", // url
      "Vodafone live!", // title
      "http://www.provider.bg", // link
      "250", // width
      "160", // height
      "image description" // description
      );

// Create RSS TextInput.
$textInput = new RSSTextInput(
      "TextInput title", // title
      "TextInput description", // description
      "TextInput name", // name
      "TextInput link" // link
      );

// <vf:image href="images/beyonce.jpg" caption="Beyonce"/>
$vfImage = array(
      RSS_VFELEMENT_ELEMENT_KEY => array("vf:image"),
      RSS_VFELEMENT_ATTRIBUTES_KEY => array("href" => "images/beyonce.jpg",
            "caption" => "Beyonce")
);
// <vf:teaser>Read the latest news...</vf:teaser>
$vfTeaser = array(
      RSS_VFELEMENT_ELEMENT_KEY => array("vf:teaser", "Read the latest
news...")
);
// <vf:article-link>article2.pml</vf:article-link>
$vfArticleLink = array(
      RSS_VFELEMENT_ELEMENT_KEY => array("vf:article-link", "article2.pml")
);
//<vf:serviceid>12435</vf:serviceid>
$vfServiceid = array(
      RSS_VFELEMENT_ELEMENT_KEY => array("vf:serviceid", "12345")
);

// Creating RSS Item
$item = new RSSItem(
      "Хванаха поредния пиян шофьор да кара междуградски автобус", // title
      "http://dnes.dir.bg/2007/06/11/news1768153.html", // link
      "Пиян шофьор на автобус бе задържан от полицията в Търговище рано
сутринта.", // description
      "vladimir.shtrakov@gmail.com (Vladimir Shtrakov)", // author
      array("News", "dir.bg"), // category
      "http://dnes.dir.bg/2007/06/11/news1768153.html", // comments
      new RSSEnclosure(
```

```
            "http://dnes.dir.bg/2007/06/11/news1768153.html", // url
            "12216320", // length
            "audio/mpeg" // type
            ), // enclosure
      null, // guid
      "Mon 11 June 2007 12:12:12 GMT", // pubDate
      array("Хванаха поредния пиян шофьор да кара междуградски автобус",
            "http://dnes.dir.bg/2007/06/11/news1768153.html"), // source
      array($vfImage, $vfTeaser, $vfArticleLink, $vfServiceid) //
vfElements
      );

// Creating RSS Channel.
$channel = new RSSChannel(
      "Dir.bg", // title
      "http://dnes.dir.bg", // link
      "Dir.bg портал", // description
      array($item), // items
      "bg", // language
      "Copyright 2007, dir.bg", // copyright
      "editor@dir.bg (editor)", // managingEditor
      "webmaster@dir.bg (webmaster)", // webMaster
      "Mon 11 June 2007 12:12:12 GMT", // pubDate
      "Mon 11 June 2007 12:12:12 GMT", // lastBuildDate
      array("News", "dir.bg"), // category
      "Dir.bg RSS20 Generator", // generator
      "http://dnes.dir.bg/2007/06/11/news1768153.html", // docs
      $cloud, // cloud
      "60", // ttl
      $image, // image
      "rating", // rating
      $textInput, // textInput
      null, // skipHours
      null // skipDays
      );

// Adding the channel to the Generator.
$generator->addChannel($channel);

// Outputing the RSS content.
$xml = $generator->printDOMString();
?>
```

Листинг 10 – Тестов скрипт за RDF пакета

```
<?php
/**
 * Sample script for RDF Generator usage.
 */
$inc_path = "../lib/";
require_once($inc_path . "domgen/Generator.class.php");

////////////////////////////// Namespaces ////////////////////////////////
$namespaces = array(
      'xmlns:dc' => 'http://purl.org/dc/elements/1.1/',
      'xmlns:rdfs' =>   'http://www.w3.org/2000/01/rdf-schema#',
      'xmlns:filter' => 'http://purl.org/vcml/filter/1.0/',
      'xmlns:srch' =>   'http://purl.org/vcml/search#',
      'xmlns:rating' => 'http://purl.org/vcml/rating/1.0/',
      'xmlns:er' =>     'http://purl.org/vcml/sdp/er/1.0/',
```

```
      'xmlns:icra' =>   'http://www.icra.org/faq/decode',
      'xmlns:prism' => 'http://prismstandard.org/namespaces/1.2/basic/'
      );
////////////////////////////////////////////////////////////////////////////

//////////////////////////// Ringtone example ////////////////////////////////
// The RDF Generator.
$generator = new Generator(new RDF());
$generator->addNamespaces($namespaces);

$generator->addSeeAlso(new RDFSSeeAlso("More Music", "mserver/index.pml"));

$ringtoneBilling = new VSRCHBilling("12345", "ER content name", "12");
$ringtonePersonalizationAttr = new VCMLPersonalizationAttr(
      "http://headerlogo.com",
      "http://footerlogo.com", "true", "excludeDPAtitle",
"backgroundColorDPA",
      "true", "true");
$ringtonePersonalizationRefAttr = new VCMLPersonalizationRefAttr(
      "true", "true", "true");
$ringtoneSeeAlso = new RDFSSeeAlso("So What",
"mserver/metallica/so_what.pml");
$ringtoneDCSubjecs = array("music", "metal");
$ringtoneVDDRDeviceFilter = new VDDRDeviceFilter(
      array("SonyEricsson V800", "SonyEricsson P800"), // device
      //array("SonyEricsson T630", "SonyEricsson V630"), // ignore-device
      null,
//    "SonyEricsson V800", // device
//    "SonyEricsson T630", // ignore-device
      //"device class"
      null);
$ringtoneFilterGroup = new VCMLFilterGroup(
      array("SonyEricsson V800", "SonyEricsson P800"), null, "use
feature");

$ratings = array(array("BBFC", "18"), array("BBFC", "17"), array("BBFC",
"16"));
$icras = array(
      array(RDF_ICRA_NUDITY, array("na", "nb", "nc"))//,
      //array(RDF_ICRA_CHAT, array("ca"))//,
      //array("LANGUAGE", array("la", "lb", "lc"))
);
$ringtoneRating = new RDFRating($ratings, $icras,
      PML_RESTRICTION_CONTENT_CATEGORY_RINGTONE
      //array(RDF_ICRA_NUDITY, "nv")
      );

$ringtone = new VSRCHRingtone(
      "Metallica", // vsrch:artist
      "So What?", // vsrch:title
      VSRCH_RINGTONE_FORMAT_TYPE_MIDI, // vsrch:format
      "Heavy Metal", // vsrch:genre
      $ringtoneSeeAlso, // rdfs:seeAlso
      $ringtoneBilling, // billing attributes
      $ringtonePersonalizationRefAttr, // personalizationRefAttr
      $ringtonePersonalizationAttr, // personalizationAttr
      "musicprovider", // vsrch:provider
      "Metallica - So What?", // dc:title
      "The ringtone for your phone", // dc:description
      $ringtoneDCSubjecs, // dc:subject
      $ringtoneVDDRDeviceFilter, // vddr_device_filters
```

```
      $ringtoneFilterGroup, // filterGroup
      $ringtoneRating // rating
      );

$ringtoneDescription = new RDFDescription(
      "http://partner.com/ringtones/ringtone1.wav", //about
      $ringtone, //vsrch
      "Buy now" //rdfs:label
);

$generator->addDescription($ringtoneDescription);

// Outputing the RFD content.
$filename = "/usr/home/admin/lib/domgen/rdf/output/ringtoneRDF.xml";
$xml = $generator->save($filename);

print "File: " . $filename . " saved SUCCESSFULLY!<br/>"
      . "===== Content =====<br/>"
      . nl2br(htmlspecialchars(html_entity_decode($xml)));
////////////////////////////////////////////////////////////////////////////

////////////////////////////// Game example ////////////////////////////////
// The RDF Generator.
$generator = new Generator(new RDF());
$generator->addNamespaces($namespaces);

$game = new VSRCHGame(
      "shoot 'em up", // vsrch:genre
      new RDFSSeeAlso("Info", "arcade/raiders.pml"), // rdfs:seeAlso
      new VSRCHBilling("12345"), // billing attributes
      null, // personalizationRefAttr
      null, // personalizationAttr
      "javagamesinc", // vsrch:provider
      "Sushi Raider", // dc:title
      "New arcade thrills from Tokyo!", // dc:description
      array("atari", "classic", "anime"), // dc:subject
      null, // vddr_device_filters
      new VCMLFilterGroup(null, null, "SW/JavaSupportMIDPVersion2.0"), //
filterGroup
      null // rating
      );

$gameDescription = new RDFDescription(
      "http://partner.com/games/pacman.jad", //about
      $game, //vsrch
      "Buy now" //rdfs:label
);

$generator->addDescription($gameDescription);

// Outputing the RFD content.
$filename = "/usr/home/admin/lib/domgen/rdf/output/gameRDF.xml";
$xml = $generator->save($filename);

print "File: " . $filename . " saved SUCCESSFULLY!<br/>"
      . "===== Content =====<br/>"
      . nl2br(htmlspecialchars(html_entity_decode($xml)));
////////////////////////////////////////////////////////////////////////////

////////////////////////////// Picture example ////////////////////////////////
// The RDF Generator.
```

```
$generator = new Generator(new RDF());
$generator->addNamespaces($namespaces);

$picture = new VSRCHPicture(
      VSRCH_PICTURE_TYPE_WALLPAPER, // type
      new RDFSSeeAlso("Info", "pictures/index.pml"), // rdfs:seeAlso
      new VSRCHBilling("wallpaper service id"),
      null, // personalizationRefAttr
      null, // personalizationAttr
      "pictureprovider", // vsrch:provider
      "The wallpaper for your phone", // dc:title
      "The wallpaper description goes here!", // dc:description
      array("atari", "classic", "anime"), // dc:subject
      null, // vddr_device_filters
      new VCMLFilterGroup(array("SonyEricsson V800", "SonyEricsson P800")),
// filterGroup
      null // rating
      );

$pictureDescription = new RDFDescription(
      "http://partner.com/pictures/wallpaper1.jpg", //about
      $picture, //vsrch
      "Buy now" //rdfs:label
);

$generator->addDescription($pictureDescription);

// Outputing the RFD content.
$filename = "/usr/home/admin/lib/domgen/rdf/output/pictureRDF.xml";
$xml = $generator->save($filename);

print "File: " . $filename . " saved SUCCESSFULLY!<br/>"
      . "===== Content =====<br/>"
      . nl2br(htmlspecialchars(html_entity_decode($xml)));
////////////////////////////////////////////////////////////////////////////

////////////////////////////// Audio example ////////////////////////////////
// The RDF Generator.
$generator = new Generator(new RDF());
$generator->addNamespaces($namespaces);

$audio = new VSRCHAudio(
      $ringtone->artist,
      $ringtone->title,
      $ringtone->format,
      array(VSRCH_CHANNEL_MUSIC, VSRCH_CHANNEL_ENTERTAINMENT), // channels
      $ringtone->genre,
      $ringtone->rdfs_seeAlso,
      $ringtone->billing,
      $ringtone->personalizationRefAttr,
      $ringtone->personalizationAttr,
      $ringtone->provider,
      $ringtone->dc_title,
      $ringtone->dc_description,
      $ringtone->dc_subject,
      $ringtone->vddr_device_filters,
      $ringtone->filter_group,
      $ringtone->rating,
      "Date and time when the right to publish the resource expires." //
publishing_rights
      );
```

```
$audioDescription = new RDFDescription(
      "http://partner.com/audio/metallica_so_what.mp3", //about
      $audio, //vsrch
      "Buy now" //rdfs:label
);

$generator->addDescription($audioDescription);

// Outputing the RFD content.
$filename = "/usr/home/admin/lib/domgen/rdf/output/audioRDF.xml";
$xml = $generator->save($filename);

print "File: " . $filename . " saved SUCCESSFULLY!<br/>"
      . "===== Content =====<br/>"
      . nl2br(htmlspecialchars(html_entity_decode($xml)));
////////////////////////////////////////////////////////////////////////////

////////////////////////////// Video example ////////////////////////////////
// The RDF Generator.
$generator = new Generator(new RDF());
$generator->addNamespaces($namespaces);

$video = new VSRCHVideo(
      array(VSRCH_CHANNEL_NEWS, VSRCH_CHANNEL_ENTERTAINMENT), // channels
      new RDFSSeeAlso("Info", "video/index.pml"), // rdfs:seeAlso
      new VSRCHBilling("video service id"),
      null, // personalizationRefAttr
      null, // personalizationAttr
      "videoprovider", // vsrch:provider
      "Video of the day!", // dc:title
      "Be informed - watch the latest news", // dc:description
      array("atari", "classic", "anime"), // dc:subject
      null, // vddr_device_filters
      new VCMLFilterGroup(array("SonyEricsson V800", "SonyEricsson P800")),
// filterGroup
      null // rating
      );

$videoDescription = new RDFDescription(
      "http://partner.com/video/video1.mpg", //about
      $video, //vsrch
      "Buy now" //rdfs:label
);

$generator->addDescription($videoDescription);

// Outputing the RFD content.
$filename = "/usr/home/admin/lib/domgen/rdf/output/videoRDF.xml";
$xml = $generator->save($filename);

print "File: " . $filename . " saved SUCCESSFULLY!<br/>"
      . "===== Content =====<br/>"
      . nl2br(htmlspecialchars(html_entity_decode($xml)));
////////////////////////////////////////////////////////////////////////////

////////////////////////////// MMS example /////////////////////////////////
// The RDF Generator.
$generator = new Generator(new RDF());
$generator->addNamespaces($namespaces);
```

```
$mms = new VSRCHMMS(
      array(VSRCH_CHANNEL_MUSIC, VSRCH_CHANNEL_ENTERTAINMENT), // channels
      new VSRCHBilling("mms service id"),
      null, // personalizationRefAttr
      null, // personalizationAttr
      "mmsprovider", // vsrch:provider
      "The MMS dc:title goes here!", // dc:title
      "The MMS dc:description goes here!", // dc:description
      array("atari", "classic", "anime"), // dc:subject
      null, // vddr_device_filters
      new VCMLFilterGroup(array("SonyEricsson V800", "SonyEricsson P800")),
// filterGroup
      null // rating
      );

$mmsDescription = new RDFDescription(
      "http://partner.com/mms/mms.pml", //about
      $mms, //vsrch
      "Buy now" //rdfs:label
);

$generator->addDescription($mmsDescription);

// Outputing the RFD content.
$filename = "/usr/home/admin/lib/domgen/rdf/output/mmsRDF.xml";
$xml = $generator->save($filename);

print "File: " . $filename . " saved SUCCESSFULLY!<br/>"
      . "===== Content =====<br/>"
      . nl2br(htmlspecialchars(html_entity_decode($xml)));
////////////////////////////////////////////////////////////////////////////

//////////////////////////// Browseable example ////////////////////////////
// The RDF Generator.
$generator = new Generator(new RDF());
$generator->addNamespaces($namespaces);

$browseable = new VSRCHBrowseable(
      array(VSRCH_CHANNEL_MUSIC, VSRCH_CHANNEL_ENTERTAINMENT), // channels
      new VSRCHBilling("browseable service id"),
      null, // personalizationRefAttr
      null, // personalizationAttr
      "browseableprovider", // vsrch:provider
      "The browseable dc:title goes here!", // dc:title
      "The browseable dc:description goes here!", // dc:description
      array("atari", "classic", "anime"), // dc:subject
      null, // vddr_device_filters
      new VCMLFilterGroup(array("SonyEricsson V800", "SonyEricsson P800")),
// filterGroup
      null // rating
      );

$browseableDescription = new RDFDescription(
      "http://partner.com/browseable/browseable.pml", //about
      $browseable, //vsrch
      "Buy now" //rdfs:label
);

$generator->addDescription($browseableDescription);

// Outputing the RFD content.
```

```
$filename = "/usr/home/admin/lib/domgen/rdf/output/browseableRDF.xml";
$xml = $generator->save($filename);

print "File: " . $filename . " saved SUCCESSFULLY!<br/>"
      . "===== Content =====<br/>"
      . nl2br(htmlspecialchars(html_entity_decode($xml)));
////////////////////////////////////////////////////////////////////////////
?>
```