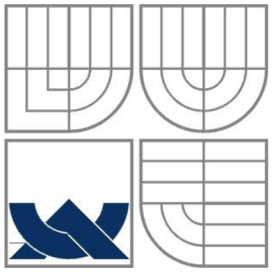
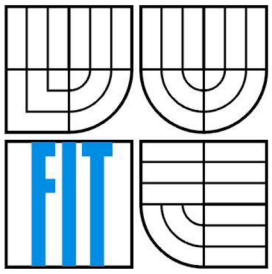


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

# INTELIGENTNÍ RSS ČTEČKA

INTELLIGENT RSS READER

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — MARTIN SEČKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. SVATOPLUK ŠPERKA
SUPERVISOR

BRNO 2008

# Abstrakt

RSS čtečka je aplikace sloužící k shromažďování a zobrazení syndikovaného webového obsahu. Tato práce se zabývá návrhem a implementací takovéto aplikace. Mimo to práce stručně popisuje formáty RSS a ATOM, které slouží jako prostředky syndikaci obsahu umožňující.



# Abstract

RSS reader is an application used for aggregation and viewing of syndicated web content. This thesis deals with design and implementation of such application. Besides that the thesis briefly describes RSS and ATOM formats which serve as means allowing the content syndication.



# Klíčová slova

RSS, ATOM, syndikace obsahu, inteligentní čtečka, webová aplikace



# Keywords

RSS, ATOM, content syndication, intelligent reader, web application



# Citace

Sečka Martin: Inteligentní RSS čtečka, bakalářská práce, Brno, FIT VUT v Brně, 2010

# Inteligentní RSS čtečka

## Prohlášení

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

.....
Martin Sečka
18. 5. 2010

## Poděkování

Tímto bych rád poděkoval vedoucímu své bakalářské práce, Ing. Svatopluku Šperkovi, za pomoc a cenné připomínky k mojí práci v průběhu jejího vzniku.

# Obsah

Obsah ..... 1
1 Úvod ..... 3
1.1 Úvod do syndikace obsahu webových stránek ..... 3
2 Standardy související se syndikací obsahu ..... 4
2.1 XML ..... 4
2.2 RDF ..... 5
3 RSS ..... 6
3.1 Historie RSS ..... 6
3.2 RSS 0.91 ..... 6
3.2.1 Verze společnosti Netscape ..... 6
3.2.2 Verze společnosti UserLand ..... 7
3.3 RSS 2.0 ..... 8
3.4 RSS 1.0 ..... 8
3.4.1 Element channel ..... 9
3.4.2 Element image ..... 9
3.4.3 Element item ..... 10
3.4.4 Element textinput ..... 10
4 ATOM ..... 11
4.1 Hlavní rozdíly oproti RSS ..... 11
4.2 Formát ATOM dokumentu ..... 11
5 Návrh aplikace ..... 13
5.1 Volba platformy ..... 13
5.2 Požadovaná funkcionalita ..... 13
5.2.1 Archiv zdrojů ..... 14
5.2.2 Aktualizace zdrojů ..... 14
5.3 Návrh uživatelského rozhraní ..... 14
5.3.1 Seznam kanálů a kategorií ..... 15
5.3.2 Seznam tagů ..... 15
5.3.3 Oblast zobrazení novinek ..... 15
5.3.4 Dialogy ..... 16
6 Implementace ..... 17
6.1 Framework ..... 17
6.1.1 Typy tříd a objektů frameworku ..... 17
6.1.2 AJAX ..... 18
6.1.3 Autentizace a přihlášení ..... 18
6.1.4 Uživatelská nastavení ..... 19
6.2 Databáze ..... 19
6.3 Analýza RSS a ATOM zdrojů ..... 21
6.3.1 Náhled článku ..... 22
6.4 Uživatelské rozhraní ..... 23
6.4.1 Hlavní stránka aplikace ..... 23
6.4.2 Kanály a kategorie ..... 23
6.4.3 Tagy ..... 24

6.4.4 Oblast zobrazení novinek ..... 24
6.4.5 Dialogy ..... 24
7 Závěr ..... 25

# 1 Úvod

Cílem této práce je návrh a implementace inteligentní RSS a ATOM čtečky, tedy aplikace sloužící k shromažďování syndikovaného webového obsahu poskytující pokročilé funkce pro organizaci a zobrazování tohoto obsahu.

Počáteční kapitola práce velmi stručně popisuje XML a RDF, což jsou standardy úzce související s publikováním webového obsahu a jeho metadat.

Kapitoly tři a čtyři se zabývají stručným popisem standardů RSS, rozdíly mezi jejich jednotlivými verzemi a také standardem ATOM coby nástupcem RSS.

Kapitola pět popisuje návrh cílové aplikace z hlediska výběru vhodné platformy a požadované funkcionality. Také se zabývá předpokládaným vzhledem, fungováním a rozmístěním uživatelského rozhraní a jeho jednotlivých prvků.

V kapitole šest je popisována samotná implementace čtečky, tj. struktura použitého frameworku, struktura databáze, způsob získávání agregovaného obsahu a způsob implementace uživatelského rozhraní.

## 1.1 Úvod do syndikace obsahu webových stránek

Syndikace (česky publikování) obsahu nebo také syndikace webu je metoda sdílení a distribuce obsahu webových stránek libovolnému množství jiných stránek, programů či uživatelů. Nejčastěji se pod tímto pojmem rozumí webové kanály, jinak také nazývané webové zdroje [1].

Webové kanály jsou dokumenty (většinou založené na XML), jejichž jednotlivé položky, představující většinou nejnovější publikované články, zahrnují mimo jiné odkaz na zdroj svého obsahu. Tímto zdrojem v nejvíce případech bývají novinkové portály a blogy, nicméně kanály mohou poskytovat i strukturované informace jako předpověď počasí nebo výsledky vyhledávání. Dvěma nejčastěji používanými formáty webových kanálů jsou v současné době RSS a ATOM [2].

# 2 Standardy související se syndikací obsahu

Standardy obou zmíněných formátů webových kanálů jsou založeny na standardu XML. Formát RSS 1.0 je dále založen na standardu RDF, který je nicméně jazykem z rodiny XML.

## 2.1 XML

*Pokud není uvedeno jinak, tato kapitola čerpá z [3].*

XML (Extensible Markup Language) je obecný značkovací jazyk vytvořený konsorciem W3C jako podmnožina jazyka SGML (Standard Generalized Markup Language). Jazyk je určený pro serializaci a výměnu dat mezi aplikacemi nebo pro publikaci dokumentů. XML dokumenty jsou obyčejné textové soubory sestávající výhradně ze znaků Unicode.

Znaky tvořící XML dokument představují buď značky (markup), nebo obsah (content). Jejich rozlišení je dosaženo použitím jednoduchých syntaktických pravidel. Veškeré řetězce tvořící značky začínají znakem < a končí >, nebo začínají znakem & a končí ;. Zbylé řetězce představují obsah.

V samotném obsahu dokumentu mohou s výjimkou několika řídících znaků (např. & nebo < [4]) vystupovat jakékoliv Unicodové znaky. Pro zahrnutí i těchto znaků do obsahu dokumentu poskytuje XML sadu předdefinovaných entit, kde např. *<* představuje znak <. Veškeré povolené znaky pak mohou být reprezentovány posloupností znaků obsahující číselný kód z UCS (Universal Character Set) Unicodu a to buď v šestnáctkovém nebo desítkovém tvaru, např. *&#x4e2d* nebo *中*.

Základními konstrukty jazyka jsou tagy (značky), elementy a atributy. Tagy jsou řetězce začínající znakem < a končící >. XML rozlišuje tři druhy tagů. Tagy počáteční (`<tag>`), dále tagy koncové (`</tag>`) a tagy představující prázdný element (`<tag/>`).

Elementy jsou logické části dokumentu ohraničené počátečním a koncovým tagem nebo sestávající pouze z tagu pro prázdný element. Řetězec mezi počátečním a koncovým tagem se nazývá obsahem elementu a může zahrnovat jiné, tzv. dceřiné, resp. synovské elementy (či prostě potomky).

Atributy jsou značkovací konstrukty tvořené párem jméno/hodnota existujícím v rámci počátečního tagu nebo tagu pro prázdný element. Např.

```
<položka pořadí="25">Obsah položky</položka>
```

představuje element „položka“ s atributem „pořadí“ s hodnotou 25 a obsahem „Obsah položky“.

XML je case-sensitive, což znamená, že např. elementy `položka` a `Položka` představují dva zcela odlišné elementy. Důležitý je také požadavek na nepřekrývání jednotlivých elementů, což znamená, že každý potomek musí být kompletně obsažen ve svém rodičovském elementu.

Součástí jazyka není definice konkrétních značek, ta je čistě v rukou autora dokumentu. Existují nicméně definiční jazyky pro XML, které umožňují automatickou validaci dokumentů. DTD (Document Type Definition) popisuje, jak mohou být značky navzájem uspořádány a vnořeny. Vymezuje atributy pro každou značku a typ těchto atributů. DTD je poměrně starý a málo expresivní jazyk, jehož další nevýhodou je, že sám není XML souborem. Mezi silnější jazyky pro popis XML patří např. XML Schema, které dovoluje mimo jiné definovat i datové typy elementů a jejich atributů.

K jednomu XML dokumentu lze připojit více než jeden definiční soubor, přičemž každému z těchto souborů lze přidělit vlastní jmenný prostor (namespace) a zabránit tak případným kolizím jmen.

## 2.2 RDF

*Pokud není uvedeno jinak, tato kapitola čerpá z [5].*

RDF (Resource Description Framework) je obecný mechanismus pro zápis metadat poskytující interoperabilitu mezi aplikacemi, jež si na Webu vyměňují strojům srozumitelné informace. Datový model RDF je založen na myšlence přiřazení výrazu ve tvaru podmět-predikát-předmět zdrojům, především pak webovým. Tyto výrazy se v terminologii RDF nazývají *trojice* (triples). Např. jedním ze způsobů jak zapsat výrok „Tráva má zelenou barvu" je trojice, v níž podmětem je „tráva", predikátem „má barvu" a předmětem „zelená". Tento přístup je velmi podobný klasickému konceptuálnímu modelování, např. ERD nebo diagramu tříd [6].

RDF jako abstraktní model může pro svou realizaci a serializaci využívat různé formáty. Jedním z nich je i XML. Mimo něj však konsorcium W3C[1] představilo i formát Notation 3 (N3)[2].

Podmětem RDF výroku je buď URI (Uniform Resource Identifier) nebo prázdný uzel, přičemž obojí označuje zdroje. Zdroje představované prázdnými uzly jsou nazývány anonymní zdroje a nejsou z RDF výroku přímo identifikovatelné. Např. ve výroku „Jan má kamaráda narozeného 21. dubna.", který je možné rozdělit na dva jednodušší výroky „Jan má kamaráda." a „Tento kamarád se narodil 21. dubna." představuje „kamarád" anonymní zdroj spojující tyto dva výroky.

Predikátem je URI rovněž identifikující zdroj, který představuje vztah. Předmětem může být URI, prázdný uzel nebo Unicodový řetězcový literál.

[1] http://www.w3.org/

[2] http://www.w3.org/DesignIssues/Notation3

# 3 RSS

RSS je rodina webových kanálů specifikovaných pomocí XML. V současné době se jedná o pravděpodobně nejpoužívanější způsob syndikace obsahu na internetu.

## 3.1 Historie RSS

V březnu 1999 vydala společnost Netscape RSS 0.9, které bylo založeno na pracovní verzi systému RDF. V červenci téhož roku vydal Netscape verzi RSS 0.91, která však je díky vypuštění RDF prvků s verzí 0.9 nekompatibilní. Tato verze je již založena čistě na XML. V červnu 2000 uveřejnil Dave Winner a jeho společnost UserLand svou vlastní specifikaci RSS 0.91, která se v několika věcech odlišuje od stejně označované verze společnosti Netscape (viz 2.2.2). V prosinci 2000 vypustila RSS-DEV Working Group specifikaci RSS 1.0, která opět staví na RDF. V té samé době UserLand vydal specifikaci RSS 0.92, následovanou RSS 0.93 v dubnu 2001 a RSS 0.94 v srpnu 2002. RSS 2.0 byla UserLandem vydána v září 2002. V současnosti jsou nejvíce využívány verze 0.91, 1.0 a 2.0 [7][8].

## 3.2 RSS 0.91

### 3.2.1 Verze společnosti Netscape

*Tato kapitola čerpá z [9].*

RSS 0.91 je založeno na XML (stejně jako všechny ostatní verze), které je jazykem striktně case-sensitive. Názvy tagů proto musí být psány přesně tak, jak udává specifikace a dále musí být řádně ukončeny. Každý RSS 0.91 dokument by měl začínat udáním používané verze jazyka, popř. používaného kódování:

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

Následuje identifikátor typu dokumentu, který obsahuje adresu DTD souboru s definicí formátu. Je vyžadován pro zajištění validity dokumentu:

```
<!DOCTYPE rss PUBLIC "-//Netscape Communications//DTD RSS 0.91//EN" "">
```

Samotný obsah RSS dokumentu je uzavřen do elementu `rss`, který obsahuje atribut *version* udávající o jakou verzi RSS se jedná:

```
<rss version="0.91">
```

Element `rss` musí obsahovat element `channel` obsahující informace o konkrétním kanálu, jehož dceřiné elementy popisuje tab. 3.1.

| Název | Popis | Povinný |
|---|---|---|
| **description** | Prostý text s popisem kanálu. | Ano |
| **language** | Mezinárodní kód jazyka používaného v dokumentu. | Ano |
| **link** | Adresa serveru, na kterém je zdroj umístěn. | Ano |
| **title** | Titulek kanálu. | Ano |
| **copyright** | Informace o autorských právech. | Ne |
| **docs** | URL odkazující na popis kanálu. | Ne |
| **image** | Odkaz na obrázek představující logo kanálu. Vnořeny jsou povinné elementy `url`, `link`, `title` a nepovinné elementy `description`, `width` a `height`. | Ne |
| **item** | Položka kanálu; obsahuje povinné elementy `title` (titulek odkazu) a `link` (adresa odkazovaného dokumentu) a nepovinný element `description` (výtah, popř. popis nebo úryvek z odkazovaného dokumentu). Standard dovoluje maximálně 15 elementů `item` na jeden kanál. | Ne |
| **lastBuildDate** | Datum a čas poslední modifikace kanálu. | Ne |
| **managingEditor** | E-mail šéfredaktora webu, na kterém se kanál nachází. | Ne |
| **pubDate** | Datum publikace kanálu. | Ne |
| **rating** | PICS[3] hodnocení kanálu. | Ne |
| **skipDays** | Seznam dní v týdnu, během nichž kanál nebude aktualizován. Jednotlivé dny jsou uzavřeny v sub-elementech `day`. | Ne |
| **skipHours** | Seznam hodin ve formátu GMT, během nichž kanál nebude aktualizován. Jednotlivé hodiny jsou uzavřeny v sub-elementech `hour`. | Ne |
| **textinput** | Textové pole pro vstup. Povinně obsahuje element `title`, `description`, `name` a `link` (adresa CGI skriptu pro zpracování textového vstupu). | Ne |
| **webMaster** | Kontakt na webmastera. | Ne |

Tab. 3.1: Sub-elementy elementu `channel` u RSS 0.91; podrobněji v [9]

### 3.2.2 Verze společnosti UserLand

*Tato kapitola čerpá z [10] a [11].*

Verze společnosti UserLand se od verze společnosti Netscape liší pouze v několika detailech. RSS 0.91 společnosti Netscape udává jako povolené hodnoty elementu `hour` celá čísla od 0 do 23, zatímco verze společnosti UserLand používá celá čísla od 1 do 24. Verze Netscapu obsahuje element `textinput`, zatímco verze UserLandu obsahuje element `textInput`. Vzhledem k tomu, že XML je striktně case-sensitive, jedná se o dva odlišné elementy. DTD ve verzi společnosti Netscape dovoluje použití pojmenovaných znakových entit. UserLand ve své specifikaci DTD nepoužívá. A konečně

[3] Platform for Internet Content Selection popisuje mechanismus, s jehož pomocí mohou uživatelé vybírat a filtrovat příslušně označený webový obsah (například zabránit zobrazení nevhodného obsahu dětem).

verze UserLandu uvádí maximální délku textových řetězců, naproti tomu zde však neexistuje omezení maximálního počtu elementů `item`.

## 3.3 RSS 2.0

*Tato kapitola čerpá z [11] a [12].*
Oproti verzi 0.91 obsahuje element `channel` navíc nepovinné elementy uvedené v tab. 3.2.

| Název | Popis | Povinný |
|---|---|---|
| **`category`** | Specifikace jedné nebo více kategorií, do nichž kanál spadá. | Ne |
| **`generator`** | Řetězec identifikující program, kterým byl kanál vytvořen. | Ne |
| **`cloud`** | Specifikuje webovou službu podporující rozhraní rssCloud. | Ne |
| **`ttl`** | Počet minut, po který může být dokument v cache paměti. | Ne |

Tab. 3.2: Elementy elementu `channel` u RSS 2.0, které nebyly obsaženy již v RSS 0.91; podrobněji v [12]

Oproti RSS 0.91 zde existují rozdíly ve významu některých elementů. Hodnotou elementu `hour` vnořeného do elementu `skipHours` jsou celá čísla od 0 do 23, element `docs` zde neodkazuje na popis kanálu, ale obsahuje URL s dokumentací RSS.

Všechny sub-elementy elementu `item` jsou v této verzi nepovinné, nicméně je vyžadován alespoň jeden z elementů `title` nebo `description`. Oproti verzi 0.91 obsahuje `item` navíc elementy uvedené v tab. 3.3.

| Název | Popis | Povinný |
|---|---|---|
| **`author`** | E-mailová adresa autora článku. | Ne |
| **`category`** | Specifikuje jednu či více kategorií, do kterých položka patří. | Ne |
| **`comments`** | URL stránky s komentáři k položce. | Ne |
| **`enclosure`** | Příloha položky; obsahuje tři povinné atributy - *url* (adresa přílohy), *length* (velikost přílohy v bytech) a *type* (MIME typ přílohy). | Ne |
| **`guid`** | Globálně unikátní identifikátor. Používá se pro určení, zda položka existovala již dříve nebo je nová. | Ne |
| **`source`** | Jméno RSS kanálu, ze kterého položka pochází. Povinný atribut *url* obsahuje adresu odkazovaného zdroje. | Ne |

Tab. 3.3: Sub-elementy elementu `item` u RSS 2.0, které nebyly obsaženy již v RSS 0.91; podrobněji v [12]

Omezení délky textových řetězců nebo počtu elementů `item` zde neexistují.

## 3.4 RSS 1.0

*Tato kapitola čerpá z [13].*

Hlavním rozdílem mezi RSS 1.0 a verzemi odvozenými od RSS 0.91 je rozšiřitelnost pomocí XML jmenných prostorů a využití standardu RDF.

Moduly založené na jmenných prostorech umožňují, aby RSS bylo rozšířeno bez nutnosti opakovaného přepisování základní specifikace, bez nutnosti dohodnout se na každém jednom elementu,

bez znečištění RSS elementy, které většina aplikací nijak nevyužije, a bez kolizí jmen. RDF pak také umožňuje bohatší reprezentaci metadat, než jaká byla možná v dřívějších verzích RSS.

Deklarace verze XML zde není povinná, nicméně se doporučuje. Kořenovým elementem každého RSS 1.0 dokumentu je element `rdf:RDF`. Ten asociuje předponu jmenného prostoru `rdf:` (může být použita jakákoliv jiná, nicméně kvůli zachování normy se to nedoporučuje) se syntaktickým schématem RDF a zavádí schéma RSS 1.0 jako výchozí jmenný prostor dokumentu:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://purl.org/rss/1.0/">
```

Kořenový element obsahuje elementy `channel`, `image`, `item` a `textinput`.

### 3.4.1 Element channel

Obsahuje metadata o kanálu samotném. Jeho atribut *rdf:about* musí obsahovat v rámci všech těchto atributů v dokumentu unikátní URL, které je URI identifikují kanál. Nejčastěji jím bývá URL popisované stránky nebo URL, na kterém se nachází samotný soubor s RSS, např.:

```
<channel rdf:about="http://www.xml.com/xml/news.rss">
```

Element `channel` obsahuje elementy popsané v tab. 3.4.

| Název | Popis | Povinný |
|---|---|---|
| **`title`** | Titulek odkazu. | Ano |
| **`link`** | Adresa odkazovaného dokumentu. | Ano |
| **`description`** | Výtah, popř. popis nebo úryvek z odkazovaného dokumentu. | Ano |
| **`items`** | RDF obsah asociující položky dokumentu s konkrétním kanálem. | Ano |
| **`textinput`** | RDF asociace mezi elementem `textinput` a konkrétním kanálem | Ne |
| **`image`** | RDF asociace mezi elementem `image` a konkrétním kanálem | Ne |

Tab. 3.4: Sub-elementy elementu `channel` u RSS 1.0; podrobněji v [13]

### 3.4.2 Element image

Jde o nepovinný element představující obrázek asociovaný ke kanálu. Většinou je zobrazován v HTML vygenerovaném z RSS dokumentu. Obsahuje tři povinné elementy (za předpokladu, že je element `image` v dokumentu zahrnut) popsané v tab. 3.5.

| Název | Popis | Povinný |
|---|---|---|
| `title` | Alternativní text obrázku, resp. atribut „alt“ u HTML značky `img`. | Ano |
| `link` | URL, na které bude obrázek odkazovat. | Ano |
| `url` | URL obrázku, resp. atribut *src* u HTML značky `img`. | Ano |

Tab. 3.5: Sub-elementy elementu `image`; podrobněji v [13]

### 3.4.3 Element item

Představuje položku kanálu, což v nejvíce případech bývá titulek článku, nicméně díky modularitě jím může v podstatě cokoliv. V dokumentu musí být obsažen alespoň jedenkrát, maximální množství není určeno, nicméně kvůli zpětné kompatibilitě se doporučuje maximálně 15 položek. Obsahuje elementy popsané v tab. 3.6.

| Název | Popis | Povinný |
|---|---|---|
| `title` | Titulek položky. | Ano |
| `link` | URL položky. | Ano |
| `description` | Stručný popis/výtah položky. | Ne |

Tab. 3.6: Sub-elementy elementu `item`; podrobněji v [13]

### 3.4.4 Element textinput

Umožňuje odeslání formulářových dat metodou GET na libovolné URL. Obvyklé je využití např. jako vyhledávací pole. Za předpokladu, že je tento element v dokumentu obsažen, má čtyři povinné elementy popsané v tab. 3.7.

| Název | Popis | Povinný |
|---|---|---|
| `title` | Popisek pole. | Ano |
| `description` | Stručný popis účelu pole. | Ano |
| `name` | Název pole, resp. odeslané proměnné. | Ano |
| `link` | URL, na které budou odeslaná data směřována. | Ano |

Tab. 3.7: Sub-elementy elementu `textinput`; podrobněji v [13]

# 4 ATOM

ATOM je webový standard pro syndikaci obsahu vzniklý jako nástupce, případně alternativa formátu RSS. Stejně jako RSS je i ATOM založen na jazyce XML. Oproti RSS je však jeho specifikace důslednější, díky čemuž se vyhýbá některým problémům, jež mohou při použití RSS nastat. Formát ATOM byl v prosinci 2005 akceptován IETF[4] a vydán jako RFC 4287 [16].

## 4.1 Hlavní rozdíly oproti RSS

ATOM vyžaduje, aby každý kanál a každá jeho položka (element `entry`) obsahovala tři elementy [15]:

- `id` - unikátní identifikátor, kterým může být jakýkoliv řetězec, nejčastěji však URI položky
- `title` - nadpis nebo titulek
- `updated` - časové razítko indikující čas poslední změny

Na rozdíl od RSS, které je schopno jako obsah položek poskytovat pouze obyčejný text nebo escapované HTML, je ATOM schopen navíc pojmout dobře zformátované XHTML, XML, binární obsah kódovaný pomocí base-64 nebo URI ukazatele na obsah, který není v kanálu obsažen přímo. ATOM také poskytuje dobře definovaný model pro své rozšíření, čímž umožňuje přidávat nová metadata a obsah stejně, jako to umožňuje RSS, dělá to však způsobem, který pomáhá chránit interoperabilitu mezi jednotlivými implementacemi. Např. jasně stanovuje, kde v dokumentu se rozšiřující elementy mohou nebo nesmí objevit [16].

ATOM podporuje rozvoj podcastingu díky možnosti připojit ke každé položce více než jednu přílohu, což je užitečné např. pokud je podcast distribuován ve více různých formátech. Na rozdíl od RSS tak není potřeba vytvářet speciální kanál pro každý poskytovaný formát audia. Přílohy nicméně nejsou omezeny pouze na audio soubory, ale mohou odkazovat na jakýkoliv soubor [15].

Kromě příloh umožňuje ATOM také vytvářet položky, jejichž obsah není obsažen přímo v dokumentu kanálu, ale jsou odkazovány pomocí URI. Tímto poskytuje prostředek pro syndikaci nejrůznějších druhů obsahu aniž by bylo nutné využití jakýchkoliv rozšíření [15].

## 4.2 Formát ATOM dokumentu

*Není-li uvedeno jinak, čerpá tato kapitola a její podkapitoly z [14].*

Kořenovým elementem dokumentu je element `feed`. Ten obsahuje několik elementů s metadaty o kanálu samotném popsaných v tab. 4.1.

[4] Internet Engineering Task Force (Komise techniky Internetu) vyvíjí a podporuje internetové standardy a úzce spolupracuje s konsorciem W3C a s orgány ISO/IEC. Zabývá se především standardy TCP/IP a soubory internetových protokolů.

| Název | Popis | Povinnost |
|---|---|---|
| `author` | Autor položek kanálu. | 1 a víc |
| `category` | Kategorie, do které kanál spadá. | 0 a víc |
| `contributor` | Osoba, která se podílela na vzniku kanálu nebo jeho položek. | 0 a víc |
| `generator` | Identifikátor aplikace, která kanál vytvořila. | Max. 1 |
| `icon` | Obrázek s ikonou kanálu. Strany by měly být v poměru 1:1. | Max. 1 |
| `logo` | Obrázek představující logo kanálu. | Max. 1 |
| `id` | Permanentní unikátní identifikátor kanálu. | Přesně 1 |
| `link` | Odkaz na webový zdroj kanálu. | Doporučen 1 |
| `rights` | Informace o právech držených nad kanálem. | Max. 1 |
| `subtitle` | Podtitul kanálu. | Max. 1 |
| `title` | Titulek kanálu. | Přesně 1 |
| `updated` | Datum poslední změny kanálu. | Přesně 1 |

Tab. 4.1: Sub-elementy elementu `feed`; podrobněji v [14]

Kromě těchto metadat obsahuje element `feed` libovolné množství elementů `entry`, které představují položky kanálu. Element `entry` může či musí obsahovat elementy popsané v tab. 4.2.

| Název | Popis | Povinnost |
|---|---|---|
| `author` | Autor položky; povinný v případě, že neexistuje stejnojmenný sub-element elementu `feed` ani sub-element `source` stejného el. `entry`. | ← |
| `category` | Kategorie, do které položka spadá. | 0 a víc |
| `content` | Obsahuje nebo odkazuje na obsah položky. | Max. 1 |
| `contributor` | Osoba, jež se podílela na vzniku položky. | 0 a víc |
| `id` | Permanentní unikátní identifikátor položky. | Přesně 1 |
| `link` | Odkaz na webový zdroj položky. Položka může obsahovat max. jeden tento element s atributem *rel* s hodnotou „alternate" (alternativní verze položky, např. webová stránka) a libovolné množství těchto elementů, pokud je hodnota jejich atributu *rel* jiná. | ← |
| `published` | Datum publikace položky. | Max. 1 |
| `rights` | Informace o právech držených nad položkou. | Max. 1 |
| `source` | Informace o zdroji položky v případě, že byla položka zkopírována z jiného kanálu. | Max. 1 |
| `summary` | Stručné shrnutí nebo výtah z položky. Musí v položce existovat v případě, že je `content` odkazem nebo obsahuje binární data. | ← |
| `title` | Titul položky. | Přesně 1 |
| `updated` | Datum poslední změny položky. | Přesně 1 |

Tab. 4.2: Sub-elementy elementu `entry`; podrobněji v [14]

# 5 Návrh aplikace

## 5.1 Volba platformy

Vzhledem k tomu, že funkčnost čtečky úzce souvisí s dostupností zdrojů na webu, bude čtečka vytvořena jako webová aplikace. Při její tvorbě budou použity technologie, standardy a jazyky XHTML, CSS, PHP, JavaScript s použitím knihovny jQuery a některých jejích rozšíření, AJAX a relační databáze MySQL.

Tato volba umožní vytváření a správu uživatelských účtů a nastavení, popř. sdílení dat mezi uživateli. Veškerá data včetně archivu novinek mohou být (a budou) ukládána do databáze na serveru. Aplikace bude zároveň uživatelům přístupná odkudkoliv, kde je dostupné internetové připojení, aniž by byli nuceni používat konkrétní operační systém či aplikaci jakýmkoliv způsobem instalovat/kompilovat před jejím prvním použitím.

Na druhou stranu z povahy protokolu http[5] vyplývá problém obtížného udržování kontextu v aplikaci. Navíc nutnost přecházet v kódu ze strany klienta na stranu serveru a naopak (zvláště pak při použití asynchronních požadavků na server) zabraňuje snadnému debugování. Dále hrozí, že aplikace nebude přístupná v případě potíží se serverem, a ačkoliv je nezávislá na operačním systému, její funkčnost a správné zobrazení se stále bude odvíjet od typu a verze použitého prohlížeče. Čtečka by měla být bez problémů použitelná alespoň v prohlížečích Firefox 3, Opera 9 a 10, IE 8 a Chrome.

## 5.2 Požadovaná funkcionalita

Základní funkcí čtečky je schopnost rozpoznat a syntakticky analyzovat dokumenty kanálů RSS a ATOM. Ze skutečnosti, že čtečka bude implementována jako webová aplikace, vyplývá nutnost poskytnutí možnosti registrace nových uživatelů a jejich následné autentizace pomocí uživatelského jména a hesla.

Organizace zdrojů a jejich obsahu bude v základu představována možností organizovat kanály do kategorií, přičemž kategorie bude možné libovolně zanořovat. K umístění zdrojů do kategorií dojde již při přidávání nového kanálu, poté budou k dispozici další možnosti organizace, jako je označování jednotlivých novinek tagy (štítky), čímž mohou uživatelé vytvářet skupiny novinek s podobnou tématikou, aniž by tyto novinky museli pocházet ze stejného zdroje. Posledním způsobem organizace novinek bude označení novinky za „oblíbenou“.

Čtečka bude obsahovat také jednoduché vyhledávání s možností vyhledávat novinky obsahující či neobsahující jednotlivá slova nebo celé fráze. Vyhledávací dotaz bude mít jednoduchou syntaxi, kde fráze jsou uzavírány do uvozovek (") a slova či fráze, jenž nemají být ve výsledných novinkách obsaženy, předchází znak minus (-). Přitom části dotazu, jenž obsaženy být mají, budou ve výsledném SQL dotazu v disjunkci a části, které obsaženy být nemají v konjunkci. Např. dotaz

```
žlutý -pes "poslanecká sněmovna" "-vlakový spoj"
```

[5] http://www.w3.org/Protocols/

vyhledá novinky obsahující slovo „žlutý“ nebo frázi „poslanecká sněmovna“ a neobsahující jak slovo „pes“, tak frázi „vlakový spoj“.

### 5.2.1 Archiv zdrojů

Data získaná z XML dokumentů obsahujících zdroje budou průběžně ukládána do databáze. To umožní zobrazovat data, která již nejsou v souborech zdrojů dostupná. V případě, že si následně jiný uživatel přihlásí odběr stejného zdroje, budou mu poskytnuta veškerá data uložená od okamžiku prvního přihlášení tohoto zdroje. Pokud dojde k odhlášení zdroje u všech uživatelů, kteří jej kdy odebírali, data o tomto zdroji a jeho položkách v databázi zůstanou i nadále a budou k dispozici v případě budoucího přihlášení tohoto zdroje.

### 5.2.2 Aktualizace zdrojů

K aktualizaci zdrojů, tj. znovunačtení a analýze XML dokumentu obsahujícího RSS nebo ATOM, bude moci dojít při několika událostech v aplikaci. Těmi jsou obnovení aplikace, resp. stránky v prohlížeči, dále výběr kanálu, kategorie nebo všech novinek pro zobrazení, automatické obnovení asynchronním požadavkem na server po uplynutí časového intervalu nebo ruční asynchronní obnovení.

Pro zajištění stálé aktuálnosti a úplnosti archivu zdrojů i v případě, že je aplikace delší čas nevyužívaná, bude navíc periodicky spouštěn skript zajišťující aktualizaci veškerých zdrojů v databázi bez ohledu na uživatele, tj. i těch zdrojů, které v databázi existují, ale žádný uživatel je právě neodebírá.

Aby nedocházelo ke zbytečnému zatěžování serveru a prodlevám před zobrazením obsahu, bude aplikace poskytovat možnost nastavení způsobu a periody aktualizace. Nastavitelná bude doba, která musí uplynout od poslední aktualizace zdroje, aby došlo k jeho aktualizaci při obnovení stránky, dále interval mezi dvěma automatickými obnoveními nebo ručním obnovením a následujícím automatickým obnovením (z čehož vyplývá, že ruční obnovení resetuje odpočet do dalšího automatického obnovení). Dále bude nastavitelné, při kterých událostech výběru (kanál, kategorie, vše) dojde k obnově právě vybraných zdrojů. Obnovení při výběru, stejně jako automatické i ruční obnovení všech zdrojů proběhne pouze u zdrojů, které byly naposled aktualizovány před více než dvěma minutami (tato hodnota je pouhým odhadem ideálního časového intervalu a nezakládá se na žádných měřeních či statistických výsledcích).

Všechna tato nastavení budou globální v kontextu uživatele, tzn. že nebudou nastavitelná pro každý zdroj zvlášť, ale pouze pro všechny přihlášené zdroje uživatele najednou.

## 5.3 Návrh uživatelského rozhraní

Uživatelské rozhraní bude rozděleno na tři, resp. čtyři hlavní části. Bude se jednat o seznam odběrů a kategorií, do kterých jsou odběry organizovány, seznam tagů a největší oblast sloužící převážně k zobrazování novinek. Minimálně rozměry těchto částí by měly být nastavitelné s tím, že toto nastavení se musí ihned projevit v databázi.

Navíc bude aplikace obsahovat ještě jeden horní panel s informacemi o přihlášeném uživateli, tlačítkem pro odhlášení, tlačítky pro zobrazení oblíbených nebo všech novinek, a vyhledávání.

### 5.3.1 Seznam kanálů a kategorií

Seznam kanálů a kategorií bude zobrazen ve formě rozbalovacího stromu. U každého odběru bude zobrazen počet nepřečtených článků, stejně jako u každé kategorie bude zobrazena suma nepřečtených článků ze všech kanálů v podstromu této kategorie.

Umístění kursoru na odběr zobrazí tlačítka s možnými akcemi, tj. odstranit kanál, přejmenovat kanál a označit kanál za přečtený. Umístění kursoru na kategorii rovněž zobrazí tlačítka s akcemi – označit kategorii jako přečtenou a smazat kategorii včetně všech jejích kanálů a podkategorií.

Označení kategorie kliknutím způsobí zobrazení všech odběrů v oblasti pro zobrazení novinek a v rychlém seznamu novinek, analogicky označení kanálu způsobí zobrazení tohoto kanálu.

Změna hierarchie zdrojů i kanálů v rámci stromu bude probíhat klasicky přetažením jednotlivých položek, kvůli přehlednosti však bude zajištěno, že kategorie budou vždy řazeny před kanály. Jak hierarchie, tak i stav položek stromů (otevřeno / zavřeno u kategorie) se ihned promítne do databáze za pomoci asynchronního požadavku na server.

Nad samotným stromem budou umístěna tlačítka s ním související a ovlivňující jeho obsah, tj. tlačítka pro zobrazení dialogů *Přidat kanál* a *Přidat kategorii,* a také tlačítko pro ruční asynchronní obnovení všech zdrojů, které bude zároveň sloužit jako ukazatel aktuálně probíhajícího obnovování, ať už vyvolaného ručně nebo automaticky.

### 5.3.2 Seznam tagů

Podobně jako seznam odběrů budou tagy zobrazeny ve formě stromu, nicméně nebude možné je jakkoliv sdružovat či organizovat. Umístěním kurzoru na tag bude zobrazeno jediné tlačítko sloužící k odebrání tagu ze stromu a příslušných novinek.

### 5.3.3 Oblast zobrazení novinek

Oblast zobrazení novinek bude hlavní částí uživatelského rozhraní výsledné aplikace. Budou zde zobrazeny nejen informace o kanálu, kategorii či jiné zobrazitelné skupině novinek, ale i novinky samotné.

Novinky budou zobrazitelné ve zvoleném pořadí a to buď podle data, stavu přečtení nebo pořadí, ve kterém byly přidány do databáze. Aplikace bude umožňovat zvolit, jaká část novinek bude v jejich výpisu zobrazována, tj. zda zobrazovat pouze jejich nadpisy nebo všechny dostupné informace.

Nahrávání starších novinek bude probíhat asynchronně. Při zobrazení poslední načtené novinky ve viditelné oblasti aplikace bude zobrazeno nemodální upozornění o probíhajícím načítání dalších novinek, které budou následně připojeny za poslední naposled načtenou novinku. Takto se dá docílit prohlížení novinek, aniž by byl uživatel zbytečně obtěžován nutností neustále žádat o další informace nebo aniž by byl server zbytečně zatěžován načítáním dat, která uživatel nakonec nevyužije.

U jednotlivých novinek budou k dispozici ovládací prvky pro označení novinky za přečtenou / nepřečtenou, přidání / odebrání novinky do / z oblíbených, odeslání novinky mailem, sdílení novinky přes Facebook a přidání / odebrání tagu. Jakákoliv změna stavu novinky bude okamžitě promítnuta do databáze. Přečtené a oblíbené položky budou jasně visuálně odlišeny od ostatních, navíc bude existovat možnost zobrazování přečtených položek zcela vypnout.

Oblast zobrazení novinek bude obsahovat také informační a nástrojový panel s počtem načtených a celkovým počtem novinek, možností označit celou zobrazovanou množinu novinek (kanál, kategorii atd.) za přečtenou, popř. manuálně vynutit načtení dalších novinek v případě, že by automatické načítání selhalo.

### 5.3.4 Dialogy

#### 5.3.4.1 Přidat kanál

Dialog *Přidat kanál* bude modálním dialogem umožňujícím přidání kanálu se zvoleným nebo aplikací navrženým jménem přímo do vybrané kategorie. Jeho zobrazení bude vyvoláno kliknutím na tlačítko umístěné nad seznamem kanálů.

Výběr cílové kategorie bude probíhat označením položky stromu. V případě, že žádná položka označena nebude, bude nový kanál přidán do kořenové kategorie (tj. jako kořenový prvek v seznamu kanálů a kategorií v hlavní části aplikace). Dialog nesmí povolit přidání URL, která směřuje jinam než na zdroj RSS nebo ATOM, do databáze. Proto po zadání URL bude nutné nejprve ji ověřit (výsledek ověření bude uživateli oznámen včetně typu nalezeného kanálu), přičemž bude možné ze zdroje rovnou získat jméno kanálu, které bude uživateli před samotným přidáním kanálu nabídnuto. Přidání kanálu se musí okamžitě projevit v seznamu odběrů.

#### 5.3.4.2 Přidat kategorii

Přidávání kategorií bude probíhat v modálním dialogu podobném dialogu *Přidat kanál*. Jeho zobrazení bude vyvoláno kliknutím na tlačítko umístěné nad seznamem kanálů. Výběr cílové kategorie zůstane beze změny, jediným dalším požadovaným uživatelským vstupem bude název kategorie.

#### 5.3.4.3 Přidat tag

*Přidat* tag bude modální dialog vyvolaný tlačítkem nacházejícím se u každé novinky. Přidávané tagy budou zadávány do jednoduchého pole pro textový vstup, přičemž najednou půjde přidat více než jeden tag. Jednotlivé přidávané tagy budou oddělovány čárkou (,). V případě, že některý z přidávaných tagů již u novinky existuje, nebude tento tag přidán a oznámení o výsledku akce zohlední pouze přidané tagy („Tagy … byly úspěšně přidány.“). V případě, že u novinky již existují všechny zadané tagy, je tato skutečnost uživateli oznámena.

#### 5.3.4.4 Nastavení

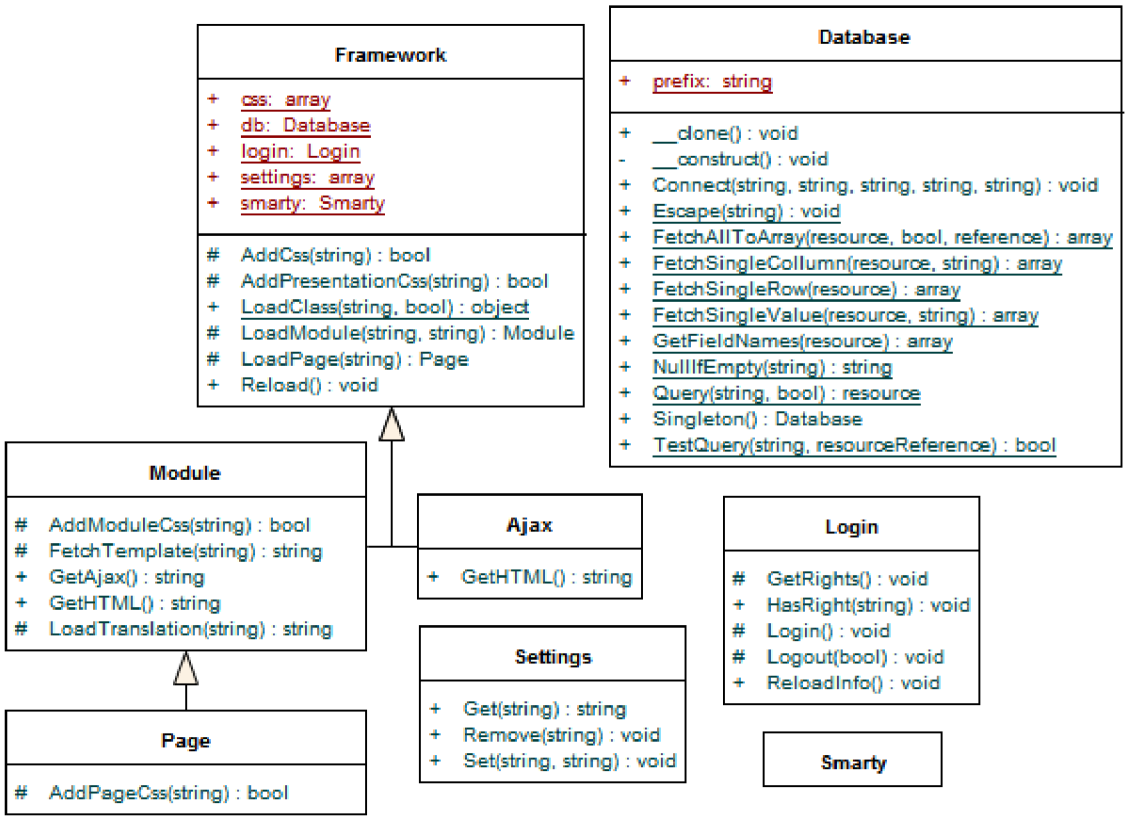
*Nastavení* bude nemodální dialog umožňující upravit některé parametry aplikace týkající se především způsobu obnovování zdrojů (viz 5.2.2).

# 6 Implementace

Aplikace je implementována v jazycích PHP[6] s použitím šablonovacího systému Smarty[7] na straně serveru a JavaScript s použitím knihovny jQuery[8] a jQuery UI[9] na straně klienta. Veškerá data, která aplikace využívá nebo vytváří, jsou uložena v relační databázi MySQL[10].

## 6.1 Framework

Aplikace na straně serveru využívá jednoduchý Framework starající se o autentizaci uživatelů, práci s databází (pouze MySQL), uživatelská nastavení, vkládání CSS souborů do stránky, zpracování asynchronních požadavků atd. Dále usnadňuje načítání tříd a to pouze těch, které vytvářená aplikace skutečně využívá. Základní třídy tohoto frameworku můžete vidět na obr. 6.1.




Obr. 6.1: Diagram základních tříd použitého frameworku (neobsahuje všechny metody a atributy).

### 6.1.1 Typy tříd a objektů frameworku

Framework rozlišuje tři základní typy tříd - stránky (odvozené od třídy *Page*), moduly (odvozené od třídy *Module*) a klasické třídy (u nichž dědění z jiné třídy není podmínkou). Moduly představují vět-

[6] http://php.net/
[7] http://www.smarty.net/
[8] http://jquery.com/
[9] http://jqueryui.com/
[10] http://mysql.com/?bydis_dis_index=1

šinou samostatně fungující části stránek (např. diskusní fórum nebo výpis databázové tabulky), není však vyloučena vzájemná závislost mezi moduly. Moduly jsou načítány metodou *LoadModule()*, která se pokusí vyhledat třídu modulu v předem daných umístěních, kterými jsou adresáře *modules* a *framework/modules* (vyhledávání v adresářích probíhá v tomto pořadí, aby bylo možné znovu implementovat modul, který již je obsažen v modulech frameworku). Soubor modulu musí být pojmenován podle adresáře, v němž se nachází a mít předponu *module*. Např. pokud je modul uložen v adresáři *modules/priklad-modulu*, musí jméno souboru být *module.priklad-modulu.php*. Název třídy modulu se řídí rovněž podle názvu adresáře, nicméně je zapsán notací camelCase s prvním písmenem názvu velkým (kvůli zvyklosti psát názvy tříd s velkým počátečním písmenem), např. PrikladModulu.

Analogicky jsou načítány, umístěny a pojmenovány i stránky a klasické třídy - stránky jsou uloženy v adresářích *pages* a *framework/pages* a jejich soubory mají předponu *page*, klasické třídy jsou v adresářích *classes* a *framework/classes* s předponou *class*.

Stránky a moduly by měly implementovat alespoň metodu *GetHTML*, která vrací (X)HTML kód stránky nebo modulu. Pro vytvoření tohoto kódu lze využít Smarty, jehož instance je vytvořena jako statická proměnná přímo ve třídě *framework*. Metoda *fetchTemplate* automaticky zpracuje šablonu uloženou v adresáři *_templates* ve stejném adresáři jako je umístěna stránka nebo modul a vrátí vygenerovaný kód. Pokud metodě není předán parametr, je zpracována šablona se stejným jménem, jako je název adresáře modulu nebo stránky, a příponou *.tpl*, např. pro modul v adresáři *priklad-modulu* je to soubor *priklad-modulu.tpl*.

### 6.1.2 AJAX

Ajaxové požadavky je možné zpracovávat pomocí třídy *Ajax* v souboru *framework/class.ajax.php*, která přijímá dva různé parametry předávané HTTP metodou GET a to parametry *page* a *module*. Pokud jsou přítomny oba tyto parametry, je parametr *page* ignorován. Na základě parametru je vytvořen objekt třídy nebo modulu a je zavolána jeho metoda *GetAjax()*, která v závislosti na své implementaci může vracet HTML kód, prostý text, XML, JSON nebo také nic. Vrácená hodnota je vytištěna na výstup. V kořenovém adresáři aplikace je umístěn soubor *ajax.php*, který do sebe zahrnuje soubor *framework/class.ajax.php* a umožňuje tak zkrátit adresu souboru, kterému jsou asynchronní požadavky předávány. Např. pokud by byl požadavek předán na URL

```
/ajax.php?module=priklad-modulu
```

byl by výsledkem tohoto požadavku výstup odpovídající hodnotě vracené metodou *GetAjax()* objektu třídy *PrikladModulu* uložené v souboru *modules/priklad-modulu/module.priklad-modulu.php*.

### 6.1.3 Autentizace a přihlášení

Autentizace uživatelů probíhá ve třídě *Login*. Aby došlo k přihlášení uživatele, je nutné, aby metodou POST byly skriptu předány parametry *sentFormName=login*, *username* a *password*. Pro vytvoření cookie zajišťující automatické přihlášení po dobu jednoho týdne od posledního přihlášení je nutné předat také parametr *autologin=1*. Informace o uživateli jsou v případě, že je uživatel přihlášen, při každém obnovení stránky načítány z databáze a ukládány do globální proměnné *$_SESSION['login']*, aby byly snadno dostupné kdekoliv v aplikaci. V případě, že má být provedeno odhlášení uživatele, je třeba metodou POST zaslat parametr *sentFormName=logout*. K odhlášení dojde také v situaci, kdy uživatelský účet po obnovení stránky již neexistuje.

Třída *Database* (implementovaná návrhovým vzorem Singleton) představuje jednoduchou nadstavbu nad standardními funkcemi PHP pro práci s databází MySQL. Umožňuje snadné připojení

k databázi a její metody dokážou mimo jiné na základě SQL dotazu vracet pole všech vybraných řádků a sloupců (*FetchAllToArray()*), pole hodnot z jediného sloupce (*FetchSingleCollumn()*), pole obsahující hodnoty prvního řádku výsledku (*FetchSingleRow()*) nebo hodnotu jediného sloupce prvního řádku výsledku. Metoda *TestQuery()* ověřuje, jestli SQL dotaz nevrací prázdný výsledek a pokud ne, je schopna tento výsledek vrátit přes referenci.

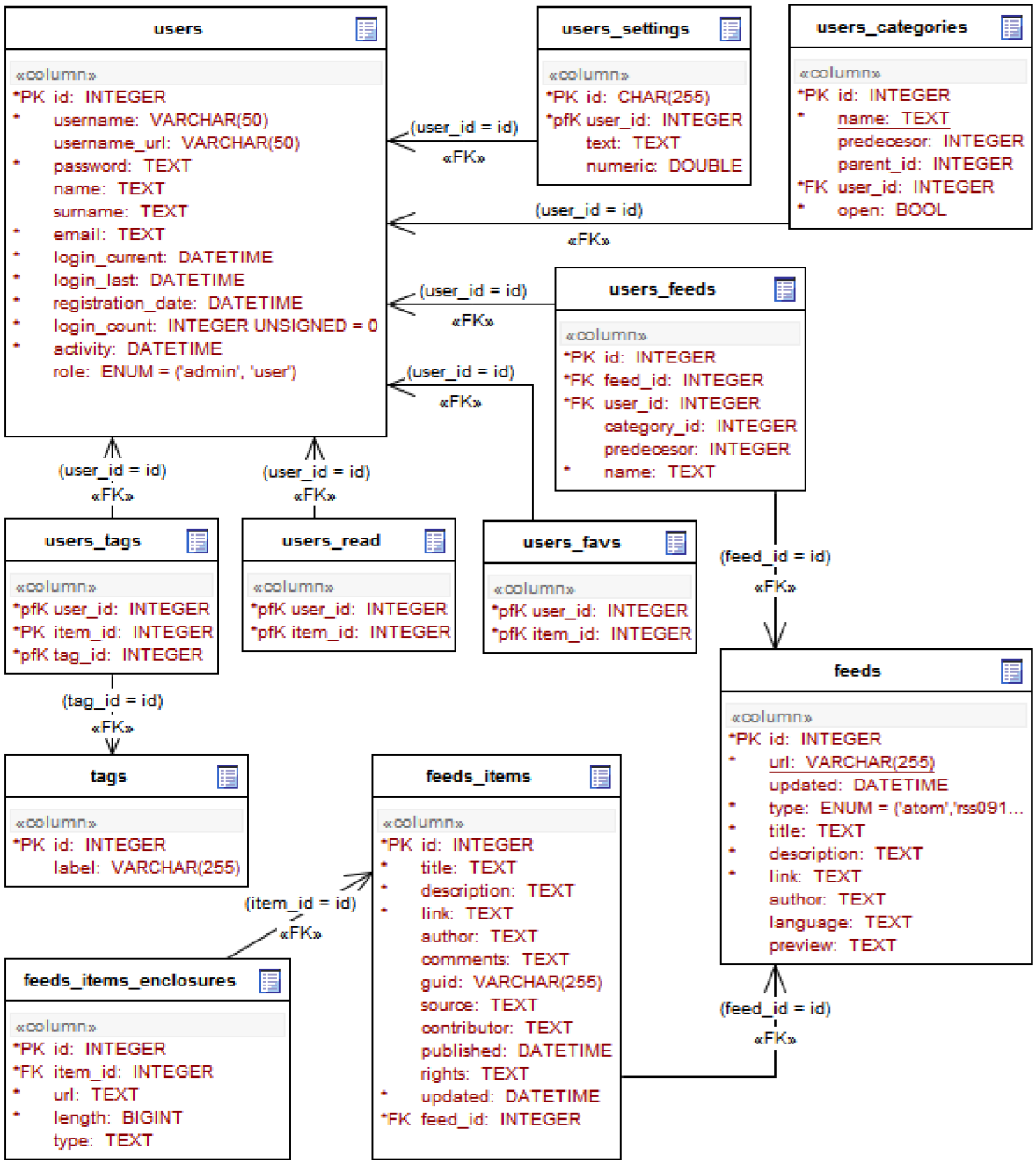
### 6.1.4 Uživatelská nastavení

Pomocí třídy *Settings* lze snadno ukládat (*Set()*), načítat(*Get()*) a mazat (*Remove()*) uživatelská nastavení. Framework obsahuje také modul *settings*, který obsluhuje asynchronní požadavky vyvolané JS funkcemi *setSetting(názevNastavení, hodnota, callback), getSetting(názevNastavení) a removeSetting(názevNastavení, callback)* uloženými v souboru *framework/scripts/functions.js*, který obsahuje i další užitečné funkce. Parametr *callback* umožňuje spustit jinou funkci při úspěšně provedené změně nastavení.

## 6.2 Databáze

Většina metod vyžadujících přístup k databázi se nachází ve třídě *Manager*. Na metodách této třídy závisí funkčnost všech modulů aplikace s výjimkou dialogu „Nastavení“, který využívá třídy *Settings* (viz kap. 6.1.4). Veškeré metody této třídy jsou statické a kromě třídy *AbstractFeed* (viz kap. 6.3) neexistují v aplikaci další třídy, které by zasahovaly do databázových tabulek, které nejsou běžnou součástí databáze použitého frameworku (tj. všech tabulek s výjimkou *users* a *users_settings*).

Schéma databáze, kterou aplikace využívá lze vidět na obr. 6.2 (omezení na cizí klíče znázorněná na obrázku ve skutečnosti databáze neobsahuje, nicméně zde pomáhají ilustrovat vztahy mezi daty aplikace).




Obr. 6.2: Schéma relační databáze výsledné aplikace

Tabulka *users* obsahuje informace o registrovaných uživatelích. Atribut pro přihlašovací heslo *password* je kvůli alespoň minimálnímu zabezpečení šifrován algoritmem MD5[11]. Atribut *role* není v aplikaci momentálně využíván, nicméně v budoucích verzích aplikace může sloužit k jednoduchému odlišení běžného uživatele od administrátora. Nastavení jednotlivých uživatelů jsou uložena v tabulce *users_settings*.

Tabulky *feeds* a *feeds_items* představují archiv kanálů a jejich položek (novinek). Zatímco konkrétní kanál může v databázi existovat pouze jednou díky požadavku na unikátnost atributu *url*, stejná novinka se může v databázi vyskytnout vícekrát v případě, že existuje ve více než jednom sou-

[11] Message-Digest algorithm je rozšířená rodina hašovacích funkcí, která vytváří ze vstupních dat výstup (otisk) fixní délky. Jeho hlavní vlastností je, že malá změna na vstupu vede k velké změně na výstupu, tj. k vytvoření zásadně odlišného otisku.

boru kanálu (např. v situaci, kdy jeden web poskytuje kanály s novinkami jednotlivých sekcí i kanál s novinkami z webu jako celku a uživatelé odebírají jak některou ze sekcí, tak i kanál celého webu). Tato situace by se dala vyřešit použitím vazební tabulky mezi kanály a jejich položkami, nicméně existence této situace není natolik častá, aby to bylo nezbytně nutné.

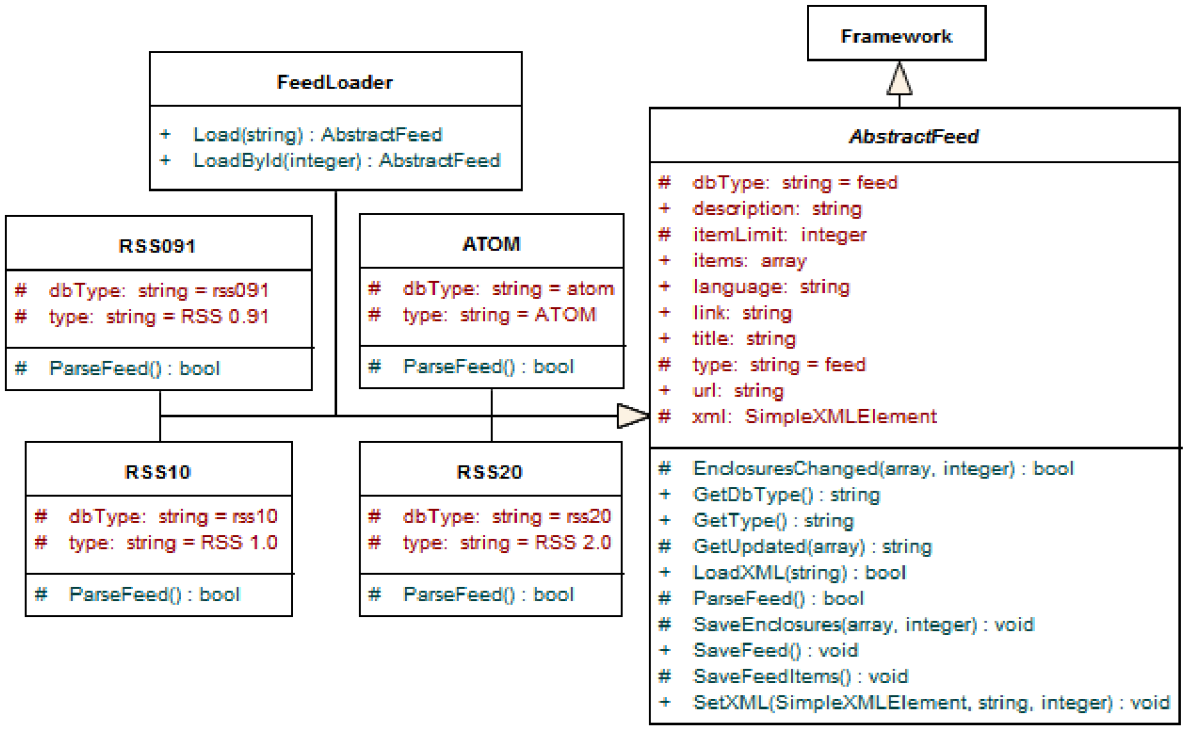
Hierarchie kanálů a kategorií uživatele zobrazovaná v aplikaci jako rozbalitelný strom je uložena v tabulkách *users_feeds* a *users_categories*. Atributy *predecesor* obou těchto tabulek určují pořadí v nadřazené kategorii, do které kanál nebo kategorie přísluší. Jeho hodnota je hodnotou atributu *id* kanálu nebo kategorie, která daný kanál nebo kategorii ve stromu předchází. V případě, že je kanál nebo kategorie na prvním místě, je jeho hodnota NULL. Atributy *parent_id*, resp. *category_id* naproti tomu ve stromě identifikují nadřazenou kategorii. Podobně pokud je kategorie nebo kanál umístěn ve stromě jako kořenový prvek, je hodnota těchto atributů NULL.

Konečně tabulky *users_tags*, *users_read* a *users_favs* představují množinu novinek, které uživatel označil tagem, za přečtené nebo za oblíbené. Tagy jsou uloženy v tabulce *tags*.

## 6.3 Analýza RSS a ATOM zdrojů

Parsování XML dokumentů probíhá za pomoci knihovny SimpleXML, která je součástí PHP od verze 5. Tato knihovna poskytuje jednoduché nástroje pro převedení XML dokumentu na objekty PHP, se kterými lze následně manipulovat použitím standardních struktur a operátorů jazyka.

Hierarchii tříd starajících se o analýzu a ukládání zdrojů lze vidět na obr. 6.3. Abstraktní třída *AbstractFeed* obstarává jak samotné načítání zdrojů z webu, tak ukládání informací o kanálech a jejich položkách do databáze. Třídy, jejichž názvy korespondují s názvem a verzí zdroje pak obstarávají zpracování informací ve zdroji obsažených.




Obr. 6.3: Diagram tříd pro analýzu RSS a ATOM zdrojů

Třída *FeedLoader* slouží ke snadnému načítání zdrojů, u nichž není předem známo, jakého typu jsou. Metoda *Load()* načte XML podle zadané URL, identifikuje typ zdroje podle kořenového elementu nebo podle použitého jmenného prostoru a vrátí objekt třídy odpovídající zjištěnému typu

zdroje. Metoda *LoadById()* slouží k načtení zdroje, jehož URL je uložena v databázi. Podle zadaného *id* zjistí URL a s tou pak zavolá metodu *Load()*.

Způsob, jakým jsou jednotlivé typy kanálů sjednoceny a uloženy do databáze, je patrný z tab. 6.1 a z tab. 6.2 (buňky tabulek obsahují výrazy XPath).

| *sloupec* | **RSS 0.91** | **RSS 1.0** | **RSS 2.0** | **ATOM** |
|---|---|---|---|---|
| **title** | channel/title | channel/title | channel/title | title |
| **description** | channel/description | channel/description | channel/description | subtitle |
| **link** | channel/link | channel/link | channel/link | link (alterna-te) |
| **language** | channel/language | --- | channel/language | --- |
| **updated** | channel/lastBuildDate nebo *Akt. datum a čas* | channel/lastBuildDate nebo *Akt. datum a čas* | *Akt. datum a čas* | updated |
| **author** | --- | --- | --- | author/name |

Tab. 6.1: Vztah mezi XML elementy zdrojů a sloupci tabulky *feeds*

| *sloupec* | **RSS 0.91** | **RSS 1.0** | **RSS 2.0** | **ATOM** |
|---|---|---|---|---|
| **title** | title | title | title | title |
| **description** | description | description | description | summary |
| **link** | link | link | link | link[@rel="alternate"] |
| **author** | --- | --- | author | author/name |
| **comments** | --- | --- | comments | link[@rel="replies"] |
| **guid** | link | @rdf:about | guid \|\| link | id |
| **source** | --- | --- | source/@url | source/link[@rel="self"] |
| **contributor** | --- | --- | --- | contributor/name |
| **published** | --- | pubDate | pubDate | published |
| **rights** | --- | --- | --- | rights |
| **updated** | *GetUpdated()* | *GetUpdated()* | *GetUpdated()* | updated |

Tab. 6.2: Vztah mezi XML elementy položek zdrojů a sloupci tabulky *feeds_items* (cesty XPath jsou relativní k elementům *item* u RSS nebo *entry* u ATOM)

V případě, že datum publikace položky není v kanálu uvedeno, je použito datum prvního uložení položky do databáze.

### 6.3.1 Náhled článku

Tato funkce nebyla obsažena v původním návrhu a je pouze experimentální, tzn. že nemusí a nebude fungovat za všech okolností. Její implementace se nachází v modulu *article-loader*. V základu se jedná o načtení HTML dokumentu nahlížené stránky a nalezení HTML elementu obsahujícího text uložený v databázi v tabulce *feeds_items* ve sloupci *description*, který představuje abstrakt, souhrn nebo popis obsahu novinky. Zde se nicméně předpokládá, že většina webů nevyplňuje tento údaj

textem, který by pak nebyl obsažen v článku, ale například prvním odstavcem článku, jeho částí nebo perexem.

Dalším krokem je nalezení předka tohoto nalezeného elementu. Zde se spoléhá na předpoklad, že zbytek textu článku se bude nalézat v okolí vyhledaného článku. Jelikož se dokumenty různých webů mohou výrazně odlišovat, nelze univerzálně stanovit, o kolik úrovní nahoru při hledání předka jít. Toto je řešeno buď postupným zvyšováním počtu úrovní při opakovaném klikání na tlačítko pro zobrazení náhledu nebo zápisem XPath cesty k elementům obsahujícím text článku na daném webu do databáze (sloupce *preview* tabulky *feeds*). Cesta se zadává relativně k elementu `body`. Jelikož však aplikace zatím neposkytuje uživatelské rozhraní pro administrátorské funkce, je pro zadání této cesty nutné přistoupit k databázi buď přes konzoli, nebo přes jiný prostředek pro její správu.

## 6.4 Uživatelské rozhraní

### 6.4.1 Hlavní stránka aplikace

Hlavní stránka aplikace (stránka *home*) v sobě zahrnuje všechny moduly implementující prvky uživatelského rozhraní a udává jejich umístění v aplikaci. Pomoci jQuery UI (konkrétně pluginu *resizable*) je implementována možnost změnit šířku levého panelu aplikace, stejně jako poměr výšky stromu s kanály k výšce stromu s tagy.

Protože se mi čistě pomocí XHTML a CSS nepodařilo vytvořit stránku takovým způsobem, aby se přesně přizpůsobovala velikosti okna všech prohlížečů, na nichž má být aplikace použitelná (viz kap. 5.1), je při události změny velikosti okna volána JS funkce dopočítávající a měnící výšku roztažitelného levého panelu a prvků dokumentu stránky obsahujících stromy s kanály a tagy. Kvůli tomuto může být změna velikosti okna prohlížeče viditelně pomalejší než u jiných stránek.

Součástí hlavní stránky je také horní panel aplikace obsahující vyhledávání a tlačítka pro zobrazení všech nebo oblíbených novinek. Všechny tyto funkce jsou implementovány v modulu *feed-viewer*, hlavní stránka zajišťuje pouze jejich volání asynchronním požadavkem. Tlačítka ovlivňující způsob zobrazení novinek, která lze na panelu rovněž najít, asynchronním požadavkem mění uživatelská nastavení. V případě tlačítka *zobrazit pouze nepřečtené položky* je následně provedeno obnovení (opět asynchronní) oblasti pro zobrazení novinek (viz kap. 6.1.4). Tlačítko *skrývat obsah novinek* toto obnovení neprovádí, provede pouze rozbalení nebo sbalení všech aktuálně zobrazených novinek.

### 6.4.2 Kanály a kategorie

Strom s kanály a kategoriemi je implementován s využitím jQuery pluginu jsTree[12], který je schopný strom vygenerovat transformací z XML řetězce s přesně daným formátem. Toto XML je podle informací v databázi generováno v modulu *category-tree*. Ten umožňuje vytvoření stromu obsahujícího buď pouze kategorie, nebo i kanály, záleží na nastavení atributu *includeFeeds* (možné hodnoty jsou *true* nebo *false*). Modul dále obsahuje atributy umožňující povolení nebo zakázání přemístitelnosti položek (atribut *moveable)*, ukládání stavu kategorií otevřena/zavřena (*saveStates*) a zobrazování počtu nepřečtených novinek v závorce napravo od názvu položky stromu (*unread*).

Hlavní strom aplikace je implementován v modulu *category-tree/main-category-tree*, který výše zmiňovaný modul rozšiřuje, nikoliv však s využitím dědičnosti, pouze jej zaobaluje a implementu-

[12] http://www.jstree.com

je funkce volané při událostech výběru, přejmenování nebo smazání položek. Mimo to přidává ke stromu tlačítka sloužící k vyvolání posledních dvou jmenovaných událostí u konkrétních položek.

Počty nepřečtených novinek jsou načteny zároveň se zbytkem stromu a uchovávány v atributu *rel* elementu `a` obsaženém v každé položce stromu (`li`). Pomocí funkcí v souboru *scripts/functions.js* jsou tyto hodnoty upravovány a vkládány do stromu v zobrazitelné formě (jako obsah elementu `b` v elementu `a` položky stromu).

### 6.4.3 Tagy

Implementace stromu tagů je prakticky shodná se stromem kanálů a kategorií (kap. 6.4.2). Rozdíl je pouze v nastavení pluginu jsTree. zobrazovaných datech a funkcích použitých pro zobrazování počtu novinek obsahujících konkrétní tag (opět se nachází v souboru *scripts/functions.js*).

### 6.4.4 Oblast zobrazení novinek

Jedná se o nejsložitější část aplikace implementovanou v modulu *feed-viewer*. Ten obsluhuje asynchronní požadavky na zobrazení všech v aplikaci existujících množin novinek, tedy novinky kanálu, kategorie, tagu, oblíbené, vyhledané a všechny. Metody získávající z databáze příslušný výběr novinek jsou součástí třídy *Manager* (viz kap. 6.2), název těchto metod vždy začíná na *GetItems*. Metody stejné třídy začínající na *GetCount* slouží ke zjištění celkového počtu novinek vybrané množiny. O tom, zda do množiny zahrnout i přečtené novinky, rozhoduje parametr *unreadOnly* všech těchto metod, do nějž je předávána hodnota uživatelského nastavení *unreadOnly* (viz kap. 6.1.4). Dále zajišťuje obsluhu požadavku na označení/odznačení novinky jako přečtené a přidání/odebrání novinek z oblíbených.

Změna řazení novinek probíhá ve dvou krocích. Nejprve je asynchronně změněno nastavení řazení *orderBy* a poté dojde k znovunačtení celého modulu. Výchozím řazením při neexistenci hodnoty nastavení je řazení dle stavu přečtení a data. Datum podle nějž je řazení prováděno je kvůli možné absenci data publikace u novinky vybráno SQL výrazem:

```
IF(`published` IS NULL, `updated`, `published`) as `orderDate`
```

### 6.4.5 Dialogy

Dialogy jsou na straně klienta vytvořeny pomocí ovládacího prvku *Dialog* z jQuery UI. Ten poskytuje jednoduše nastavitelné, událostmi řízené rozhraní pro vytvoření dialogového okna. V případě, že není žádný dialog otevřen, obsahuje hlavní stránka pouze elementy, ze kterých mohou být dialogy generovány, tj. sadu prázdných elementů `div` s atributem *id* nastaveným podle toho, který dialog se z daného elementu vytváří, a skripty s nastavením jednotlivých dialogů. Samotný obsah dialogů je načítán asynchronně až při události jejich vytvoření. Po uzavření dialogu je tento obsah z dokumentu odstraněn a elementy `div` jsou uvedeny do původního stavu. Implementace dialogů na straně serveru se nachází v modulech *dialogs/…* .

V dialozích *Přidat kanál* (*dialogs/add-feed*) a *Přidat kategorii* (*dialogs/add-category*) je použit strom (modul *category-tree)* zobrazující pouze kategorie bez možnosti jejich přesunutí, ukládání jejich stavu nebo zobrazení informace o obsažených nepřečtených položkách.

# 7 Závěr

Výsledná aplikace odpovídá svému návrhu, je schopná zpracovat všechny dnes nejrozšířenější formáty webových kanálů (ačkoliv ignoruje např. obrázky a ikony kanálů) a je bez potíží zobrazitelná v prohlížečích IE8, Firefox, Opera i Chrome. Oproti návrhu obsahuje funkci pro zobrazení náhledu článků (ta je však pouze experimentální a nefunguje správně u všech kanálů).

Nicméně aby čtečka skutečně dostála přízvisku „inteligentní“, bylo by potřeba věnovat čas dalšímu jejímu vývoji. Jako první by bylo potřeba vytvořit UI pro administraci, rozšířit možnosti nastavení i na jednotlivé kanály a umožnit uživatelům měnit své registrační údaje. Co se uživatelského rozhraní dále týče, mělo by umožňovat skrývat své části, aby zůstávalo více prostoru na zobrazení novinek i při nižších rozlišeních. A ačkoliv současný způsob zobrazení novinek umožňuje pohodlné čtení v případě, že je uživatel chce prohlížet v zobrazeném pořadí, v situaci, kdy se uživatel chce dostat na konkrétní starší novinku, se tento přístup ukazuje jako silně nepraktický. Čtečka by proto měla být schopná buď rychle přecházet na zadanou stranu, nebo umožňovat složitější filtrování novinek, např. podle data. Kvůli lepší motivaci uživatelů k přechodu z jejich současné čtečky by měla čtečka poskytovat funkci pro import/export kanálů a kategorií ve formátu OPML, popř. být schopná automaticky importovat kanály a kategorie např. z aplikace Google Reader[13].

[13] http://www.google.cz/reader

# Literatura


[1] *Wikipedia, the free encyclopedia: Web syndication* [online]. c2010 [cit. 15. 03. 2010]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Web_syndication>

[2] *Wikipedia, the free encyclopedia: Web feed* [online]. c2010 [cit. 15.03.2010]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Web_feed>.

[3] *Wikipedia, the free encyclopedia: XML* [online]. c2010 [cit. 02.05.2010]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Xml>.

[4] W3C. *Extensible Markup Language (XML) 1.0 (Fifth Edition): XML* [online]. 26.11.2008 [cit. 02.05.2010]. Dostupný z WWW: <http://www.w3.org/TR/2008/REC-xml-20081126>.

[5] *Wikipedia, the free encyclopedia: Resource Description Framework* [online]. c2010 [cit. 02.05.2010]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Resource_Description_Framework>.

[6] W3C. *Resource Description Framework (RDF) Model and Syntax Specification* [online]. 05.01.1999 [cit. 02.05.2010]. Dostupný z WWW: <http://www.w3.org/TR/PR-rdf-syntax/>.

[7] *RSS Advisory Board: RSS History* [online]. [cit. 16.03.2010]. Dostupné z WWW: < http://www.rssboard.org/rss-history>.

[8] *Wikipedia, the free encyclopedia: RSS* [online]. c2010 [cit. 16.03.2010]. Dostupné z WWW: <http://en.wikipedia.org/wiki/RSS>.

[9] Netscape Communications. *RSS Advisory Board* [online]. July 10, 1999 [cit. 16.03.2010]. RSS 0.91 Specification (Netscape). Dostupné z WWW: <http://www.rssboard.org/rss-0-9-1-netscape>.

[10] UserLand Software. *RSS Advisory Board* [online]. June 9, 2000 [cit. 18.03.2010]. RSS 0.91 Specification (UserLand). Dostupné z WWW: <http://www.rssboard.org/rss-0-9-1>.

[11] PILGRIM, Mark. *Dive into mark* [online]. February 4, 2004 [cit. 18.03.2010]. The myth of RSS compatibility. Dostupné z WWW: <http://diveintomark.org/archives/2004/02/04/incompatible-rss>.

[12] UserLand Software. *RSS Advisory Board* [online]. podzim 2002 [cit. 19.03.2010]. RSS 2.0 Specification (UserLand). Dostupné z WWW: <http://www.rssboard.org/rss-2-0>.

[13] BEGED-DOV, Gabe, et al. *Web.resource.org: persistence for the people* [online]. 2000-12-09 [cit. 20.03.2010]. RDF Site Summary (RSS) 1.0. Dostupné z WWW: <http://web.resource.org/rss/1.0/spec>.

[14] NOTTINGHAM, Mark; SAYRE, Robert. *AtomEnabled* [online]. c2005 [cit. 21.03.2010]. Atom Syndication Format Spec. Dostupné z WWW: <http://www.atomenabled.org/developers/syndication/atom-format-spec.php>.

[15] SNELL, James. *IBM : United States* [online]. 02 Aug 2005 [cit. 21.03.2010]. An overview of the Atom 1.0 Syndication Format. Dostupné z WWW: <http://www.ibm.com/developerworks/xml/library/x-atom10.html>.

[16] CIMPRICH, Petr. Atom 1.0 : formát. *Root.cz* [online]. 26. 1. 2006, [cit. 21.03.2010]. Dostupný z WWW: <http://www.root.cz/clanky/atom-1-0-format/>. ISSN 1212-8309.