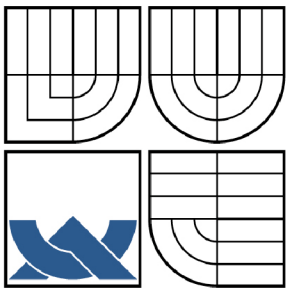
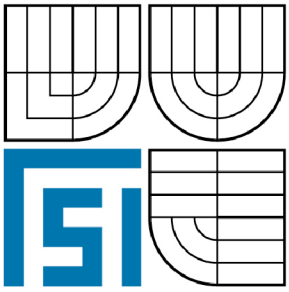


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

TVORBA INTERNETOVÝCH APLIKACÍ POMOCÍ RICH INTERNET APPLICATION ADOBE AIR

DEVELOPMENT OF INTERNET APPLICATION USING RICH INTERNET APPLICATION ADOBE AIR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — ADAM ŠIMÍK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN ROUPEC, Ph.D.
SUPERVISOR

BRNO 2009

Abstrakt

Tato práce se zabývá problematikou tvorby desktopových aplikací pomocí technologie Adobe AIR (Adobe Integrated Runtime). Tyto aplikace jsou vytvořeny na základě webových technologií, avšak nemají tak omezený přístup k prostředkům operačního systému, jako je tomu v případě aplikací webových. Navíc je možno tyto aplikace instalovat na různé operační systémy bez nutnosti jejich kompilace pro konkrétní operační platformu. Kromě popisu samotných technologií používaných při tvorbě AIR aplikací se práce věnuje realizaci vlastní aplikace, pomocí které jsou pak testovány některé z možností technologie AIR.

Abstract

This thesis explores the development of desktop applications using Adobe AIR (Adobe Itegrated Runtime) technology. Applications like these are based on web technologies but they do not have such restricted access to resources of operating system as web applications. Moreover, it is possible to install these applications on various operating systems without the need of their previous compilation for a specific operating platform. Apart from describing technologies used for developing AIR applications, this thesis also presents an application which is used for testing the possibilities and features that AIR technology offers.

Klíčová slova

webová aplikace, desktopová aplikace, webová technologie, Adobe AIR.

Keywords

web application, desktop application, web technology, Adobe AIR.

Poděkování

Chtěl bych touto formou poděkovat vedoucímu diplomové práce Ing. Janu Roupcovi, Ph.D. za věcné připomínky a rady ohledně tvorby této práce.

Obsah

1 Úvod

V dnešní době je možné najít na internetu čím dál více webových aplikací, které se svým vzhledem a vlastnostmi téměř rovnají podobným desktopovým aplikacím. Tyto webové aplikace jsou postaveny na technologiích, které se ve webovém světě používají již nějakou dobu. Mezi ně neodmyslitelně patří HTML, CSS, JavaScript, v další řadě jsou to pak technologie o něco mladší jako např. AJAX, Flash a Flex. Nejenže se tyto technologie vyvíjí daleko rychleji než klasické desktopové technologie, ale i počet lidí zabývajících se těmito technologiemi je pravděpodobně vyšší než počet programátorů desktopových aplikací.

Grafická podoba a funkčnost webových aplikací využívajících již zmiňovaných technologií je na různých operačních platformách téměř stále stejná. To je možné díky tomu, že webový prohlížeč řeší rozdíly mezi operačními systémy, namísto aby je řešila sama webová aplikace. Aby bylo dosaženo stejné funkčnosti a grafické podoby i v případě desktopových aplikací, je nutné vždy vytvořit zcela novou aplikaci pro konkrétní operační systém.

Technologie AIR (více viz kap. 2) dovoluje použití klasických webových technologií při tvorbě plnohodnotných desktopových aplikací, tzn. že daná aplikace může fungovat jak ve webovém prohlížeči, tak i jako nativní desktopová aplikace. Tato nativní aplikace je však daleko méně omezena v přístupu k systémovým prostředkům konkrétního operačního systému, než je tomu v případě webové aplikace.

2 Technologie AIR

AIR (Adobe Integrated Runtime) je běhový modul umožňující aplikacím, které by normálně fungovaly jen ve webovém prohlížeči, běžet v prostředí operačního systému. Tím dává vývojářům webových aplikací možnost využít svých dosavadních zkušeností z tvorby webových aplikací k psaní aplikací určených pro desktopové platformy. Při tvorbě těchto aplikací lze použít různé webové technologie, jako je např. HTML, CSS, JavaScript, AJAX, Flash a Flex, přičemž mezi operační platformy, na kterých tyto aplikace poběží zatím patří Windows, Mac OS X a Linux.

2.1 Běhový modul

Běhový modul AIR je sám o sobě knihovna, která běží nativně na daném operačním systému [1]. Jeho úkolem je spravovat instalace nových AIR aplikací, stejně tak i historii sama sebe. Tím je zaručena zpětná kompatibilita se staršími AIR aplikacemi v případě aktualizace běhového modulu. Běhový modul musít být také schopen zajistit, aby operace prováděné AIR aplikacemi probíhaly takovým způsobem, jakým by normálně tyto operace prováděl daný operační systém (např. vytvoření nového souboru či složky).

Běhový modul je k dispozici jako instalační balíček pro daný operační systém nebo může být zkompilován a distribuován společně s AIR aplikací.

2.2 Bezpečnost

AIR aplikace jsou desktopové aplikace sice založené na webových technologiích, ale oproti webovým aplikacím mají mnohem větší přístup k systémovým prostředkům počítače. Tato možnost AIR aplikací však skrývá potenciální riziko v podobě většího počtu škod, jež můžou samy aplikace napáchat operačnímu systému, než je tomu u aplikací webových. To je také jedním z důvodů, proč musí být AIR aplikace digitálně podepsány, aby vůbec mohly být nainstalovány na danou operační platformu.

2.2.1 Digitální certifikáty

S digitálním podpisem úzce souvisejí digitální certifikáty [1]. Pro AIR aplikace existují dva základní typy certifikátů. Jedná se o tzv. *self-signed* certifikát a certifikát, který lze získat od autorizovaných společností. Každý z těchto certifikátů má své výhody i nevýhody. Certifikáty self-signed se dají velice snadno vytvořit, avšak nezaručují věrohodnost autora aplikace, zatímco certifikáty od autorizovaných společností tuto autentičnost garantují. Na druhou stranu obstarání takového certifikátu trvá déle a většinou je i zpoplatněno.

2.2.2 AIR Sandboxes

Bezpečnostní moded AIR definuje karantény s omezeným oprávněním tzv. *Sandboxes*, ve kterých se nachází veškerý obsah (HTML, JavaScript, Flash atd.), se kterým AIR aplikace pracuje [2]. Existují dva základní typy těchto karantén. Jedná se o tzv. *Application Sandbox* a *Non-Application Sandbox*. Application Sandbox je karanténa obsahující všechny soubory instalované společně s AIR aplikací [3]. Tyto soubory mají neomezený přístup k AIR API (Application Programming Interface), takže můžou provádět operace jako např. čtení a zápis do lokálního souborového systému apod. Taktéž mají plný přístup k lokální síťi i internetu, a to bez doménových omezení.

Non-Application Sandbox je karanténa, ve které se nacházejí soubory, s nimiž AIR aplikace pracuje, ale tyto soubory přitom nejsou nainstalovány spolu s aplikací. Jedná se o soubory načtené z oblastí, které nespadají pod Application Sandbox (např. soubory načtené z internetu apod.). Následující seznam popisuje několik typů Non-Application Sandbox karantén.

- ***remote*** (soubory získané z internetu)
- ***local-trusted*** (možnost čtení z lokálních datových zdrojů a komunikace přes internet, omezený přístup k AIR API)
- ***local-with-networking*** (možnost komunikace přes internet, zákaz přístupu k lokálnímu souborovému systému)
- ***local-with-filesystem*** (možnost čtení lokálních dat, zákaz komunikace přes internet)

2.3 Možnosti AIR aplikací

Aplikace běžící ve webovém prohlížeči nedokážou vždy poskytnout stejnou interakci mezi koncovým uživatelem a samotnou webovou aplikací, jako je tomu u aplikací desktopových [4]. Jelikož tato interakce není ta, na kterou je uživatel zvyklý, aplikace se pro něj stává obtížně ovladatelnou.

AIR aplikace poskytují stejný typ interakce mezi aplikací a uživatelem jako aplikace desktopové. Tento druh funkčnosti mimo jiné zahrnuje:

- Odpovídající instalační a odinstalační zvyklosti
- Bohatá podpora pro tzv. *drag-and-drop* (mezi operačním systémem a AIR aplikacemi, mezi AIR aplikacemi, mezi nativními aplikacemi a AIR aplikacemi)
- Bohatá podpora pro *clipboard*
- Systémová hlášení
- Nativní ikony

Aby bylo možné poskytnout uživateli takový způsob chování aplikace, nabízí Adobe AIR množství API a přídavných vlastností. API je tzv. aplikační programové rozhraní, jež programátor využívá pro komunikaci jedné technologie s druhou. Mezi dvě hlavní skupiny API patří AIR API a Flash API. Obě tyto API umožňují přístup k systémovým prostředkům, které nejsou běžně k dispozici v případě webové aplikace. Mezi některé z hlavních API a dalších unikátních vlastností technologie AIR patří:

- API pro práci se soubory
- API pro práci s nativními okny
- API pro nativní menu
- Online/offline API pro detekci přístupnosti služeb
- API pro aktualizaci aplikace
- Jádro webového prohlížeče WebKit
- Integrovaná SQLite databáze

3 Desktopové a RIA aplikace

Webové aplikace jsou programy, které běží ve webovém prohlížeči, zatímco existuje aktivní připojení k internetu [5]. Na druhou stranu desktopové aplikace se spouští přímo z lokálního počítače, kde není potřeba připojení k internetu. Každá z těchto typů aplikací má určité vlastnosti, z nichž některé jsou pozitivní a jiné zase negativní. Tabulka 3.1 porovnává několik charakteristických rysů webových a desktopových aplikací.

Vlastnost	Webové aplikace	Desktopové aplikace
Internet	Požadováno	Obvykle není potřeba
Instalace	Žádná	Většinou nezbytná
Správa aplikace	Snadná aktualizace	Aktualizace
Podpora OS	Různé platformy	Různé platformy
Podpora I/O	Žádná	Přístup k systému souborů
Přístup k OS	Omezený	Přístup k lokálně uloženým souborům

Tabulka 3.1 Porovnání některých vlastností webových a desktopových aplikací.

3.1 Tradiční desktopové aplikace

Desktopové aplikace jsou v povědomí veřejnosti již nějakou dobu. Většina programovacích jazyků používaných při tvorbě těchto aplikací je závislá na operačním systému nebo se aplikace pro konkrétní operační platformu musí kompilovat [6]. Každý z těchto jazyků má své specifické vlastnosti. I když jsou možnosti konkrétního jazyka téměř neomezené, neznamená to, že je vhodné jej použít úplně na všechno. Některé jazyky jsou vhodnější např. pro psaní aplikací, zatímco jiné se hodí spíše k programování ovladačů zařízení.

C

Jedná se o procedurální programovací jazyk původně určený pro operační systém Unix. Má dostatek prostředků pro použití v systémovém programování (ovladače zařízení, jádro OS, apod.). C je psán jako člověkem čitelný jazyk obsahující struktury. Aplikace psané v tomto jazyce se musí kompilovat do binárního kódu, přičemž jsou snadno přenositelné na různé operační platformy. Na rozdíl od jazyka *C++* a *Java* je C slabě typovým jazykem.

C++
Je rozšířením jazyka C. Jedná se o silně typový objektově orientovaný programovací jazyk, tzn. že poskytuje zapouzdření a umožňuje polymorfismus a dědičnost. Kromě objektově orientovaného programování podporuje i další programovací styly, jakými jsou procedurální a generické programování. C++ dovoluje programátorům psát objekty, jež mají určité vlastnosti, obsahují metody, atributy atd. Je vhodný pro psaní větších projektů, neboť třídy umožňují znovupoužití kódu a lepší operativnost než je tomu v případě hůře organizovaného jazyka C. Aplikace napsané v C++ se taktéž musí kompilovat.

Java
Je objektově orientovaný programovací jazyk, jež zachovává mnoho principů z C a C++. V objektových principech je však Java důraznější než C a C++. V dnešní době považována za jeden z nejpopulárnějších programovacích jazyků. Kompiluje se do bytového kódu, se kterým pak pracuje *Java Virtual Machine (JVM)*. JVM je běhové prostředí specifické pro danou operační platformu, které stejně jako AIR zastává funkci prostředníka mezi operačním systémem a Java aplikacemi.

3.2 RIA aplikace

RIA (Rich Internet Application) je aplikace, která běží v klasickém webovém prohlížeči, ale navíc využívá určité mezivrstvy, pomocí níž dokáže obejít standardní znovunačtení stránky [6]. Mezi technologie, jež dokáží pracovat s touto mezivrstvou patří JavaScript použitý v AJAX (Asynchronous JavaScript and XML) aplikacích a Flex nebo Flash využívající Flash Player. K dalším z RIA řešení se pak řadí OpenLaszlo, XUL a Microsoft Silverlight.

3.2.1 Historie RIA

Pojem RIA poprvé uveden společností Macromedia v březnu roku 2002. Od té doby došlo k velkému pokroku mezi nástroji a jazyky používanými při tvorbě RIA aplikací. Nárůst počtu aplikací postavených na základech Flash a Flex technologií spolu s využitím AJAX přístupu společnostmi jako Google doslova katapultoval RIA do popředí webových technologií.

Adobe Flash
Před tím než jej společnost Macromedia přejmenovala na Macromedia Flash, byl Flash známý pod jménem FutureSplash. Flash umožňoval tvorbu vektorových animací, které se daly vložit do webové stránky. Ačkoliv jeho hlavní využití spočívalo ve tvorbě animací, vydání jazyka ActionScript 1.0 dovolilo vznik prvních RIA aplikací.

Adobe Flex
Flex 1.0 poprvé představen společností Macromedia v roce 2003. Na rozdíl od technologie Flash využívající časové osy, byl Flex programovací jazyk. V červenci roku 2006 byl vydán Flex 2.0 společně s bezplatnou vývojářskou sadou SDK (Software Development Kit). Tato verze zahrnovala ActionScript 3.0, který již byl objektově orientovaným skriptovacím jazykem. V roce 2007 byla představena třetí verze této technologie a navíc se Adobe Flex stává *open-source* projektem.

AJAX
Termín AJAX poprvé zveřejněn v dubnu roku 2005 v článku Jesse Jamese Garretta zvaném *Ajax: A New Approach to Web Applications* [7]. Jeho historie však sahá do roku 1996, kdy byl zaveden element *iframe* v prohlížeče Microsoft Internet Explorer 3.0. V roce 1997 byl zaveden v prohlížeči Netscape Navigator 4.0 element *layer*, od kterého se však na počátku vývoje Mozilly upustilo. Pátá verze prohlížeče Microsoft Internet Explorer již obsahovala objekt *XMLHttpRequest*, který je prakticky hlavním rysem dnešního AJAX přístupu.

3.2.2 Výhody RIA

V dnešním internetovém světě je velice obtížné získat si pozornost webového uživatele. Lidé jsou čím dál více náročnější, a proto obyčejná HTML stránka, postrádající určitý uživatelský komfort a flexibilitu, nemůže uspět. RIA aplikace jsou však schopny doplnit tyto nedostatky klasických webových aplikací a nabídnout tak uživateli graficky zajímavé, přístupné a interaktivní řešení [8]. Hlavní přínosy RIA aplikací popisuje následující seznam:

- Možnost spojení s existující nebo novou databází
- Uživatelský komfort bez nutnosti znovunačtení stránky
- Přístup k datům v reálném čase
- Možnost vzdálené správy dat přes webový systém
- Jednoduchá a efektivní práce s komplexními daty
- Okamžitá vizuální odezva uživateli
- Podpora různých operačních platforem

4 Technologie použité v AIR

Technologie použité v AIR se dají rozdělit do dvou rozdílných kategorií: technologie aplikace a technologie dokumentu.

4.1 Technologie aplikace

Jedná se o technologie, které je možno použít při psaní AIR aplikací. Dvě hlavní skupiny takových technologií tvoří HTML a Flash [4].

Pro potřebu aplikací založených na HTML obsahuje AIR plnohodnotné jádro webového prohlížeče WebKit (open-source projekt). Díky této integraci je možná uplná podpora pro:

- HTML
- JavaScript
- CSS
- XHTML
- DOM
- AJAX

Plnou podporu aplikacím postavených na Flash a Flex technologii zajišťuje Adobe Flash Player. Mezi některé funkce, jež Adobe Flash Player nabízí, patří:

- Jádro jazyka ActionScript (pro zaručení výkonu aplikace)
- Plná podpora pro práci se sítí (HTTP, RTMP, binární a XML sokety)
- Vektorové renderovací jádro
- Rozšířená podpora multimédií (bitmapy, vektory, audio a video)

Jelikož techologie Adobe Flex taktéž využívá Adobe Flash Player, je kromě podpory jazyka ActionScript 3 zaručena i podpora jazyka MXML.

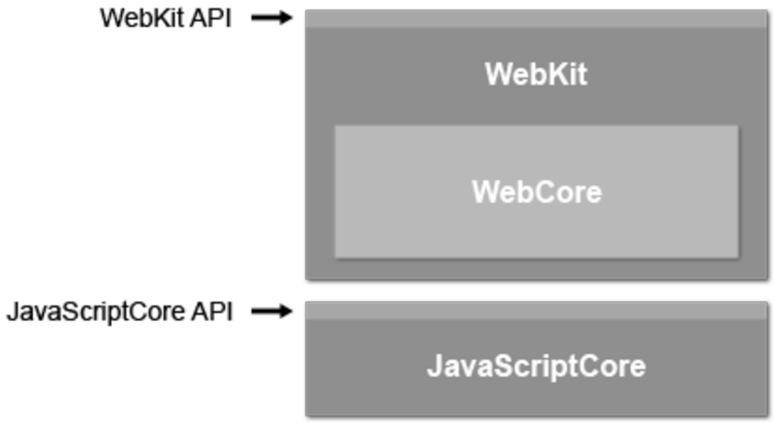
4.1.1 WebKit

Jedním z největších problémů v případě komplexní webové aplikace je zajištění její funkčnosti v různých webových prohlížečích [4]. Zatímco aplikace funguje správně v jednom webovém prohlížečí, v druhém může naopak kompletně selhat. To je také jeden z důvodů, proč společnost Adobe integrovala WebKit do běhového modulu AIR.

Hlavní důvody použití webového jádra WebKit:

- Aplikace pracují a vypadají identicky na všech operačních platformách (stejné renderovací jádro)
- Striktní dodržování webových standardů (na rozdíl od jiných webových prohlížečů)
- Osvědčená technologie, kterou weboví vývojáři již znají

WebKit je open-source aplikační framework, který poskytuje plnohodnotný základ, na kterém je možno postavit webový prohlížeč. [17] Samotný WebKit framework je tvořen dvěma hlavními jádry: *WebCore* a *JavaScriptCore* (viz Obr. 1).



Obr. 1 Struktura WebKit frameworku.

WebCore je na operační platformě nezávislé jádro, které poskytuje API pro danou operační platformu. Jeho hlavním účelem je zajištění následujících funkcí:

- HTML parser (převedení HTML kódu na DOM)
- CSS (Cascading Style Sheets)
- DOM tree (Document Object Model)
- Loader (práce se síťovou vrstvou)
- SVG (Scalable Vector Graphics)
- Rendering (renderovaní obsahu webových stránek)

JavaScriptCore je oddělená knihovna s vlastní API. Tato knihovná dokáže kompilovat JavaScript přímo do strojového kódu počítače [18]. Tím dochází ke značnému zrychlení ve vykonávání samotného kódu, protože není potřeba interpretovat bytový kód, jako je tomu v případě jiných webových prohlížečů. JavaScriptCore se skládá ze dvou hlavních částí:

- VM (Virtual Machine) - vykonává JavaScriptový kód
- Runtime - pracuje s regulárními výrazy, provádí matematické operace atd.

4.1.2 HTML

HTML (HyperText Markup Language) je značkovací jazyk umožňující publikaci dokumentů na Internetu formou webových stránek [9].

Jazyk HTML je charakterizován jako množina značek (tagů) a jejich atributů, které jsou definovány danou verzí jazyka HTML. Jednotlivé tagy mohou být párové nebo nepárové. Párové tagy jsou tvořeny otevírací a uzavírací značkou, tzn. že uvnitř těchto tagů může nebo nemusí být nějaký obsah (např. text, komentář nebo další tagy). Obsah (povolený nebo zakázaný) jednotlivých tagů je dán specifikací příslušné verze jazyka. Každý HTML tag dává textu, který je v něm obsažen určitý význam (např. nadpis, odstavec textu, odkaz atd.).

Názvy tagů se zapisují mezi úhlové zavorky např. <*p*> (značka odstavce), přičemž v případě uzavíracího tagu se před název ještě přidá dopředné lomítko </*p*>.

Příklad jednoduchého HTML dokumentu [6]:

```
<html>
    <head>
        <title>Titulek</title>
    </head>
    <body>
        <h1>Nadpis 1. úrovně (největší)</h1>
        <h2>Nadpis 2. úrovně (menší než h1)</h2>
        <h3>Nadpis 3. úrovně (menší než h2)</h3>
        <p>Odstavec
        <br/><br/>
        2x zalomení řádku nad, což vytvoří jeden prázdný
        řádek a 1x zalomení řádku pod.<br/>
        <a href="http://seznam.cz">odkaz</a></p>
    </body>
</html>
```

Kořenovým tagem je `<html>`, v něm je pak umístěna hlavička `<head>` a poté následuje samotné tělo dokumentu `<body>`.

4.1.3 JavaScript

Jedná se o slabě typový skriptovací jazyk, který je interpretován webovým prohlížečem. Syntakticky je velmi podobný programovacím jazykům jako C, C++ a Java. Na rozdíl od jiných interpretovaných programovacích jazyků jako PHP nebo ASP, jež pracují na straně serveru, se JavaScript spouští až po načtení webové stránky (tzv. na straně klienta) [10].

Kromě již vestavěných objektů jako např. *Date* nebo *Math* umožňuje JavaScript vytváření objektů vlastních. V dnešní době navíc existuje spousta tzv. *frameworků*, které nejenže práci s objekty zjednodušují, ale také přidávají další užitečné funkce. Jejich hlavním účelem je však odstranění rozdílů v interpretaci a podpoře JavaScriptu na ruzných webových prohlížečích. Mezi nejpoužívanější dnešní frameworky patří [2]:

- Yahoo! User Interface (YUI) Library (*http://developer.yahoo.com/yui/*)
- Dojo (*www.dojotoolkit.org*)
- Rico (*www.openrico.org*)
- gooxdoo (*www.qooxdoo.org*)
- Ext JS (*www.extjs.com*)
- mootools (*www.mootools.net*)
- script.aculo.us (*http://script.aculo.us*)

Příklad jednoduchého HTML souboru s použitím několika bloků JavaScriptu [6]:

```
<HTML>
    <HEAD>
        <SCRIPT type="text/javascript">
            function showAlert(){
                alert ("JavaScript hláška");
            }
        </SCRIPT>
        <TITLE>Titulek Stránky</TITLE>
    </HEAD>
    <BODY>
        <SCRIPT type="text/javascript">
            document.write("Vepsáno JavaScriptem");
        </SCRIPT>
        <a href onclick="showAlert()">Klikni</a>
    </BODY>
</HTML>
```

V prvním bloku je definována funkce `showAlert()`, která se zavolá v okamžiku, kdy uživatel klikne na odkaz s názvem "Klikni". V druhém bloku je použita funkce dokumentu, která vepíše v místě své definice předaný parametr "Vepsáno JavaScriptem".

4.1.4 CSS

CSS (Cascading Style Sheets) je jazyk určený ke grafickému formátování HTML a XHTML dokumentů. Dříve se veškerá grafická podoba HTML stránek definovala přímo uvnitř základního HTML kódu. To je však z hlediska zpracování dokumentů a vyhledávání informací nežádoucí [11]. Hlavním účelem CSS je proto oddělení vzhledu dokumentu od jeho obsahu a struktury. Kromě těchto hlavních vlastností má CSS i další výhody:

- Bohaté možnosti formátování
- Variablita (většina změn jen v CSS souboru)
- Styl pro různé typy výstupních zařízení (tiskárna, hlasový syntetizátor atd.)
- Možnost formátování XML dokumentů
- Kratší doba načítání webové stránky

Příklad jednoduchého HTML souboru graficky formátovaného pomocí CSS [6]:

```
<html>
    <head>
        <title>Příklad formátování HTML pomocí CSS</title>
        <style type="text/css">
            h1 {color: blue;}
            .bold {font-weight: bold;}
        </style>
    </head>
    <body>
        <h1>Modrý nadpis</h1>
        Text bez formátování
        <p class="bold">Tučný text odstavce</p>
    </body>
</html>
```

V tomto příkladu je CSS styl definován přímo uvniř HTML dokumentu, v případě komplexního stylu je však dobré umístit definici do externího souboru.

4.1.5 XHTML

XHTML (Extensible HyperText Markup Language) je značkovací jazyk pro tvorbu hypertextových dokumentů. Jedná se o přísnější implementaci HTML, jejímž hlavním účelem mělo být převedení staršího jazyka HTML tak, aby vyhovoval podmínkám tvorby XML dokumentů, avšak při zachování zpětné kompatibility [12].

Dnes se na webu nějčastěji používají tyto čtyři verze jazyka XHTML:

- XHTML 1.0 Strict (obsahuje minimum tagů určených k formátovaní textu)
- XHTML 1.0 Transitional (zachovává kompatibilitu se staršími prohlížeči)
- XHTML 1.0 Frameset (dovoluje použití rámců)
- XHTML 1.1 (vynechává téměř všechny tagy určené k formátování textu)

Kromě těchto existují i verze: XHTML Basic 1.1, XHTML Mobile Profile, XHTML-Print, XHTML 2.0 a XHTML 5.

Příklad Scrict verze jednoduchého XHTML dokumentu [6]:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
    <head>
        <title>Příklad XHMTL Strict souboru</title>
    </head>
    <body>
        Nějaký text
    </body>
</html>
```

Příklad Transitional verze jednoduchého XHTML dokumentu:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//
EN" "DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
    <head>
        <title>Příklad XHMTL Transitional souboru</title>
    </head>
    <body bgcolor="#0000FF">
        Some Text
    </body>
</html>
```

Zde je vidět použití atributu `bgcolor` pro nastavení barvy pozadí elementu `<body>`, to by však bylo v případě verze Strict nepřípustné.

4.1.6 DOM

DOM (Document Object Model) je objektově orientovaná reprezentace XML nebo HTML dokumentu [13]. Jedná se o specifikaci API (Application Programming Interface), která umožňuje přístupovat k dokumentu jako ke stromu. Samotný strom je tvořen uzly různých typů (element, atribut, text atd.). Každý tento uzel má své metody a vlastnosti. Prostředkem pro přístup k jednotlivým uzlům, a tedy i k jejich metodám a vlastnostem, je JavaScript. Pomocí něj je možná modifikace obsahu, struktury, stylu nebo určitých částí dokumentu.

Specifikace *W3C DOM* rozděluje DOM do několika úrovní (Level 0, Level 1, Level 2, Level 3). Každá úroveň má svá specifika, která musí splňovat. Pokud nějaká aplikace splňuje určitý *DOM level*, musí splňovat i specifikace úrovní nižších.

Kromě použití metod pro přístup k jednotlivým uzlům DOM stromu, je možné HTML dokument procházet přímo pomocí tečkové notace.

Příklad přístupu k DOM stromu přes tečkovou notaci [6]:

```
<html>
    <head>
    <title>Přístup k DOM objektu přes tečkovou notaci</title>
        <script language="javascript">
            function alertIt(){
                alert (document.myForm.textField.value);
            }
        </script>
    </head>
    <body>
        <form name="myForm">
<input type="text" name="textField" value="Hello World"/>
<input type="button" onclick="alertIt();"/>
        </form>
    </body>
</html>
```

V bloku JavaScript kódu je definována funkce, která přistupuje k hodnotě textového pole "Hello World" postupně přes několik uzlů DOM stromu.

Příklad přístupu k DOM stromu pomocí jeho metody:

```
<html>
    <head>
    <title>Přístup k DOM objektu pomocí metody</title>
        <script language="javascript">
            function alertIt(){
        alert (document.getElementById('textField').value);
            }
        </script>
    </head>
    <body>
        <form name="myForm">
<input type="text" id="textField" value="Hello World"/>
<input type="button" onClick="alertIt()"/>
        </form>
    </body>
</html>
```

V bloku JavaScript kódu je definována funkce, která přistupuje k hodnotě textového pole "Hello World" přímo přes `id` daného uzlu.

4.1.7 AJAX

AJAX (Asynchronous JavaScript and XML) je programátorská technika, která využívá JavaScript ke komunikaci se serverem a DOM k manipulaci s určitou částí webové stránky, aniž by došlo k jejímu znovunačtení [6]. Data se získají pomocí objektu *XMLHttpRequest* jako XML, HTML, JSON (JavaScript Object Notation) nebo prostý text. Tato data jsou pak zpracována a na základě manipulace DOM stromu se určitá část stránky aktualizuje. Jelikož nedochází při každém kontaktu se serverem k překreslení celého HTML dokumentu, práce s dokumentem se jeví jako plynulá a navíc se šetří náčítání obrovského množství dat.

4.1.8 MXML

MXML je deklarativní značkovací jazyk vycházející z XML [3]. Slouží ke tvorbě uživatelského rozhraní a rozvržení jednotlivých komponent Flex aplikací. Podobně jako HTML se jedná o strukturovaný jazyk, který však nabízí větší množství tagů než již zmiňovaný jazyk HTML. Jednotlivé MXML komponenty se dají rozšiřovat nebo je také možnost vytvářet komponenty vlastní.

Funkční část kódu, tedy *ActionScript* se vkládá do MXML souboru pomocí tagu `<mx:Script>`. Blok kódu se dá přímo vložit mezi `<mx:Script>` tagy, anebo je možno vytvořit odkaz na externí ActionScript soubor pomocí příslušného atributu daného tagu. V případě bloku kódu se tento vkládá mezi `CDATA` tagy. Tím se řekne analyzátoru kódu, že tato část nemá být zpracována jako MXML, ale jako prostý text.

Příklad jednoduchého MXML souboru obsahující blok kódu jazyka ActionScript 3:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Canvas xmlns:mx="http://www.adobe.com/2006/mxml"
width="400" height="300">
   <mx:Script>
      <![CDATA[
         public function myFunction():void
         {
            trace('tohle je ActionScript uvnitř MXML'):
         }
      ]]>
    </mx:Script>
</mx:Canvas>
```

4.1.9 ActionScript 3

ActonScript 3 je objektově orientovaný programovací jazyk, jehož syntaxe je velice podobná jazykům, jako je C#, Java a Javascript [6]. Vychází totiž ze standardu ECMA (European Computer Manufacturers Association) jazyka JavaScript. ActionScript 3 využívají technologie Flash a Flex. Stejně jako MXML se i ActionScript 3 kompiluje do *SWF* formátu, se kterým potom pracuje samotný Flash Player.

Příklad syntaxe jednoduchého kódu jazyka ActionScript 3 [6]:

```
package
{
    import mx.controls.Button;
        public class AngleButton extends Button
        {
            public function AngleButton()
            {
                super();
                this.rotation = 45;
            }
        }
}
```

Příklad znázorňuje rozšíření třídy `mx.controls.Button` třídou `AngleButton`, která nastavuje rotaci komponenty na 45°.

4.2 Technologie dokumentu

Jedná se o technologie, které se v Adobe AIR používají pro renderování a interakci s elektronickými dokumenty. Mezi tyto technologie spadá PDF a HTML [4].

4.2.1 PDF

PDF (Portable Document Format) je souborový formát vyvinutý společností Adobe, jež umožňuje ukládání dokumentů do formátu nezávislého na softwaru i hardwaru, na kterém byly pořízeny [14]. Také proto se jedná o webový standard k publikaci elektronických dokumentů na internetu. Aby bylo možné využívat v AIR aplikacích funkce, které Adobe Acrobat Reader nabízí ve webovém prohlížeči, musí být přinejmenším nainstalována jeho verze 8.1 [4].

4.2.2 HTML

V AIR aplikacích se HTML používá ve smyslu technologie dokumentu převážně pro potřeby rozvržení a stylu obsahu.

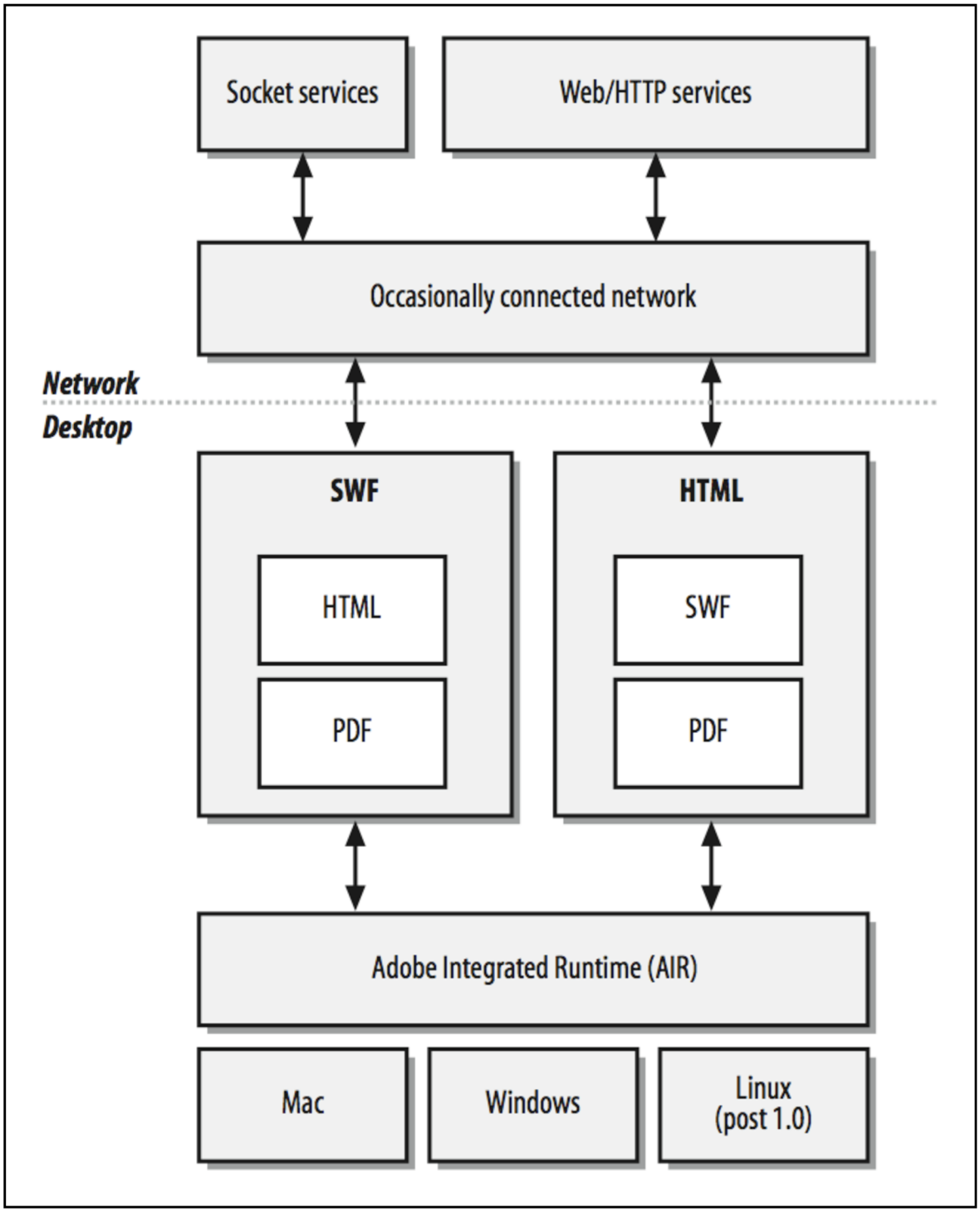
4.3 Kombinace technologií

Při tvorbě AIR aplikací lze vybírat z vélkého množství technologií. Vzhledem k tomu, že některé jazyky spolu mohou vzájemně komunikovat, je možné v AIR aplikacích používat kombinace těchto technologií [4]:

- HTML/JavaScript
- HTML/JavaScript + Flash obsah
- Flash (soubory s příponou .SWF)
- Flex (soubory s příponou .SWF)
- Flash + HTML obsah
- Ve všech kombinacích lze použít PDF

Jelikož běhový modul AIR obsahuje kromě jádra webového prohlížeče WebKit i jádro Adobe Flash Playeru, jsou obě tato jádra vzájemně integrována jen na velmi nízké úrovni. To dovoluje např. technologii Flash, která obsahuje část HTML obsahu, provádět s tímto obsahem operace, které by běžně provádět nešly. Jedná se na např. o rozmazání, rotace, transformace atd. Tato nízká úroveň integrace platí i v případě skriptovacích jazyků ActionScript a JavaScript. AIR totiž umožňuje tzv. *script bridging*, který dovoluje mezi jednotlivými skriptovacími jazyky provádět tyto operace:

- JavaScript může využívat Adobe AIR, Flash Player a ActionScript API
- ActionScript může využívat JavaScript API
- ActionScript může přímo manipulovat s HTML DOM stromem
- Registrace určité události je oboustranná (JavaScript i ActionScript)



Obr. 2 Struktura AIR aplikace.

5 Vývojové nástroje

Pro vývoj AIR aplikací je k dispozici hned několik vývojových nástrojů. Většinu z těchto nástrojů nabízí přímo sploločnost Adobe, avšak existují i jiné varianty. Mezi přední vývojové nástroje AIR aplikací patří:

- AIR SDK (všechny typy projektů)
- Flex Builder 3 (Flex AIR projekty)
- Flash CS3/CS4 (Flash AIR projekty)
- Dreamweaver CS3/CS4 (HTML/JavaScript AIR projekty)
- Aptana Studio (HTML/JavaScript AIR projekty)

5.1 AIR SDK

AIR SDK (Software Development Kit) je bezplatná softwarová vývojářská sada umožňující testovat a kompilovat AIR aplikace pomocí příkazové řádky operačního systému. K tomu se používají dva nástroje: *ADL* (AIR Debug Launcher) a *ADT* (AIR Developer Tool).

ADL
Dovoluje testovat AIR aplikace bez nutnosti vytvoření instalačního balíčku a jeho následné instalace.

ADT
Slouží k finální kompilaci aplikace, kterou je pak možno distribuovat a instalovat. Kromě toho jej lze také použít k vytvoření certifikátu potřebného ke kompilaci aplikace.

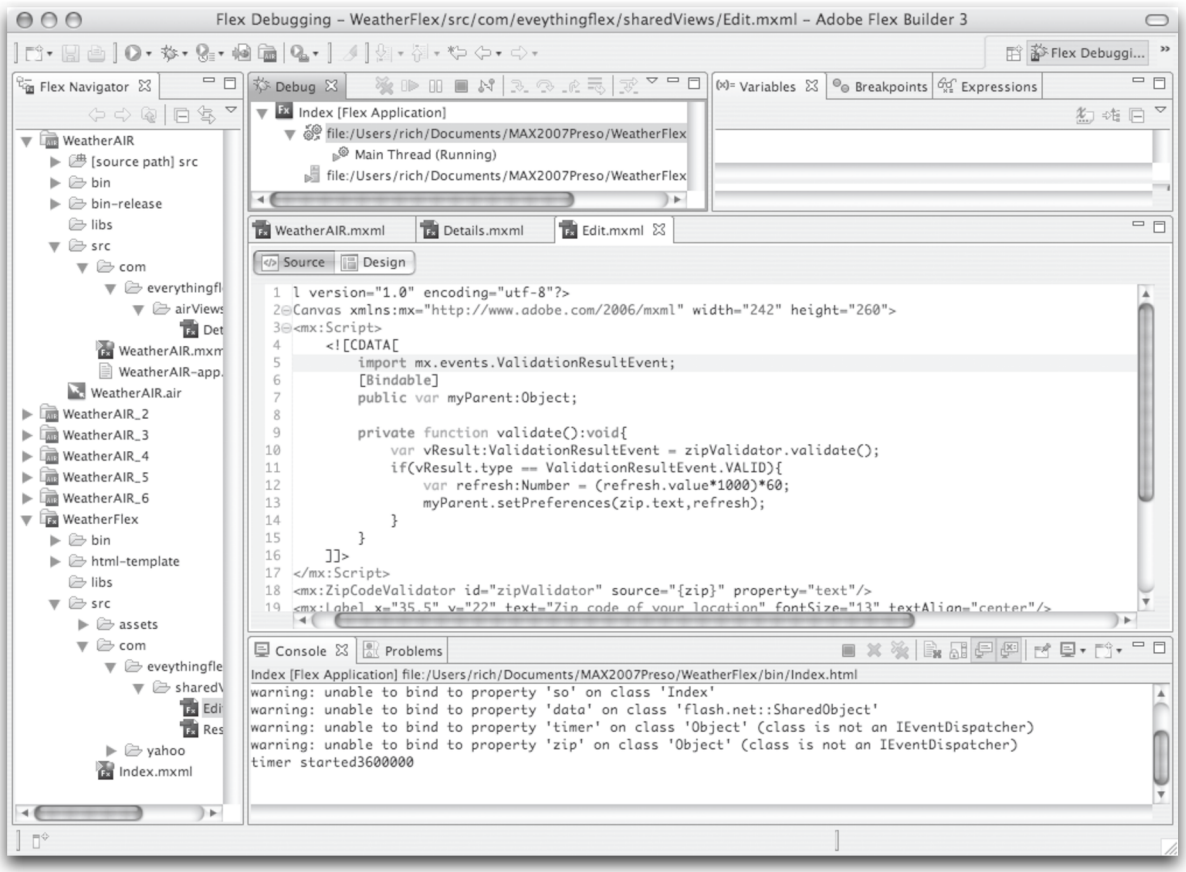
Kromě těchto dvou nástrojů obsahuje AIR SDK i šablony konfiguračního souboru, nástroje pro lokalizaci a aktualizaci aplikace a další. Více o použití AIR SDK viz kapitola 6.

5.2 Flex Builder 3

Jedná se o vývojové prostředí od společnosti Adobe, které slouží k vývoji Flex aplikací. Mimo jiné nabízí i podporu pro AIR aplikace postavené na této technologii. Flex Builder 3 je k dispozici ve dvou různých distribucích: *samostatná* a *plug-in* [6].

V samotném vývojovém prostředí je možno pracovat v několika různých režimech. K jednomu z nich patří *Mód zdrojového kódu*, který poskytuje automatickou nabídku kódu, kompletaci kódu, zvýraznění syntaxe a hlášení o chybách v reálném čase. Mezi další režimy patří *Návrhový mód* a *Mód ladění programu* (viz Obr. 3).

Návrhový mód slouží k rozmístění jednotlivých MXML komponent a tvorbě samotného uživatelského rozhraní aplikace. Mód ladění programu umožňuje zase sledování a editaci proměnných, krokování programu a práci s konzolí.

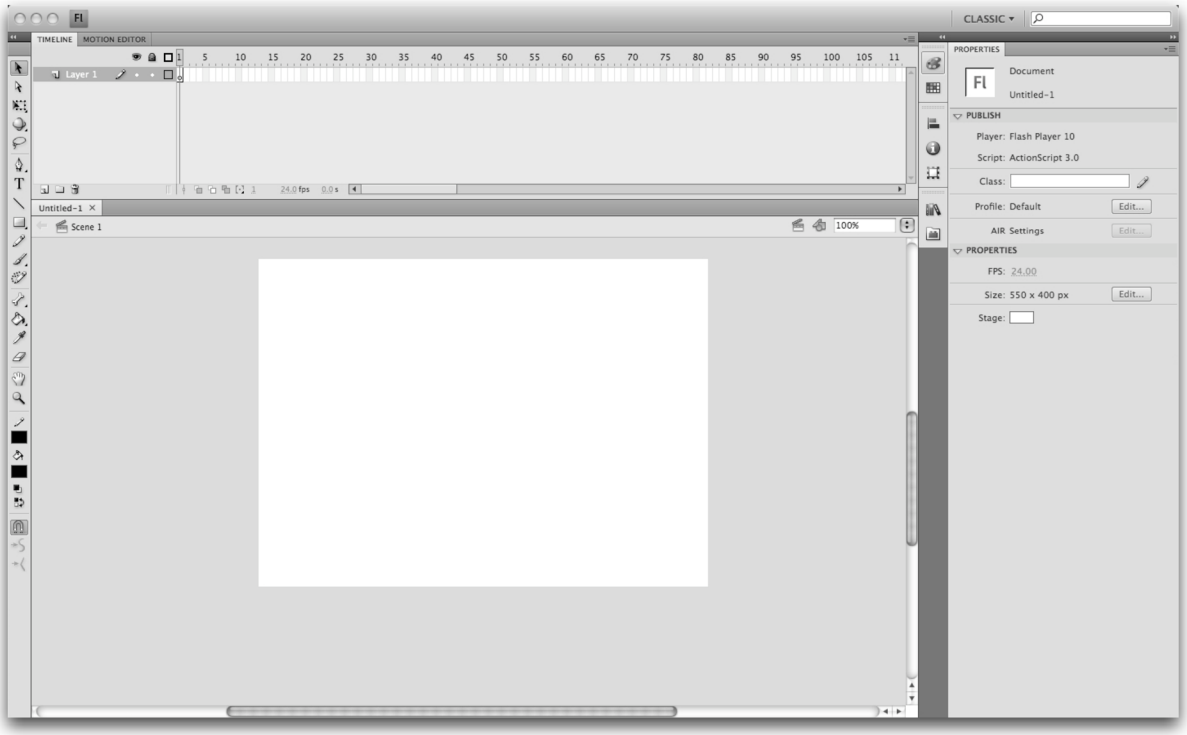


Obr. 3 Flex Builder 3 v ladícím módu.

5.3 Flash CS3/CS4

Jedná se o produkt společnosti Adobe určený k tvorbě Flash animací a aplikací. Podobně jako Flex Builder 3 umožňuje Flash CS3/CS4 IDE (Integrated Development Environment) práci v různých vývojových prostředích [6]. K nejpoužívanějším z nich patří *Základní pracovní prostředí* a *Prostředí pro ladění programu*. Jelikož technologie Flash slouží převážně k tvorbě animovaných scén, osahuje Základní pracovní prostředí ve výchozím stavu časovou osu (viz Obr. 4). Prostředí pro ladění programu obsahuje naopak komponenty určené k ladění a sledování průběhu programu.

Aby byl možný vývoj AIR aplikací v prostředí Flash CS3, je nutné nainstalovat příslušné rozšíření programu. Toto rozšíření je však dostupné bezplatně na stránkách společnosti Adobe. Verze Flash CS4 již obsahuje všechny potřebné nástroje.



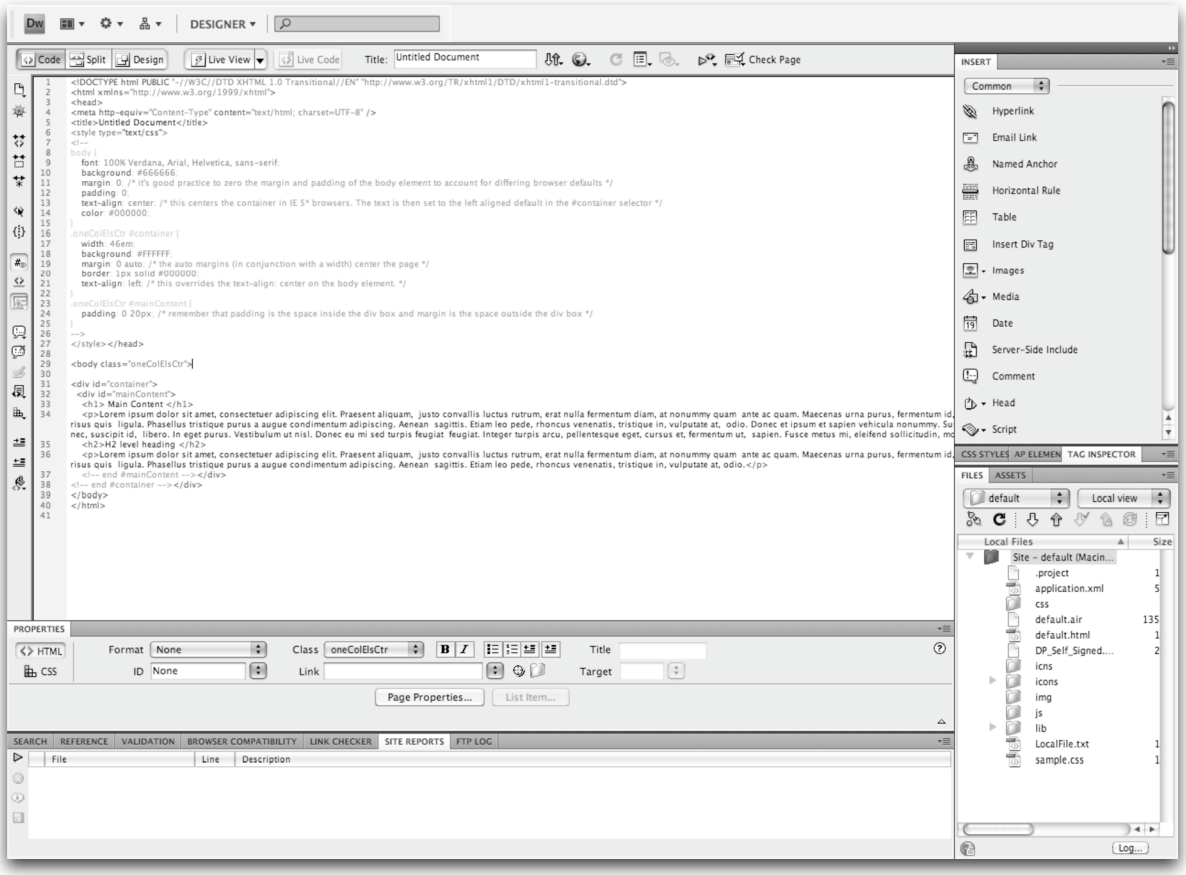
Obr. 4 Základní pracovní prostředí programu Flash CS4.

5.4 Dreamweaver CS3/CS4

Dreamweaver je produktem společnosti Adobe určený pro komplexní tvorbu webových stránek a aplikací (viz Obr. 5). Mimo to nabízí Dreamweaver i nástroje, které souvisí s vývojem AIR aplikací. [6] Jedná se především o nástroje pracující s tzv. *Spry frameworkem*, což je AJAX knihovna umožňující tvorbu bohatých uživatelských rozhraní. Mezi hlavní nástroje, jež využívají Spry framework patří:

- *Spry efekty* (grow, shrink, fade, highlight)
- *Spry widgety* (menu, seznamy, tabulky atd.)
- *Spry data* (integrace XML nebo mash-ups s RSS)

Použití všech nástrojů nutných pro tvorbu AIR aplikací v obou verzích (CS3 i CS4) je podmíněno instalací příslušného rozšíření programu. Toto rozšíření je opět dostupné bezplatně na stránkách společnosti Adobe.

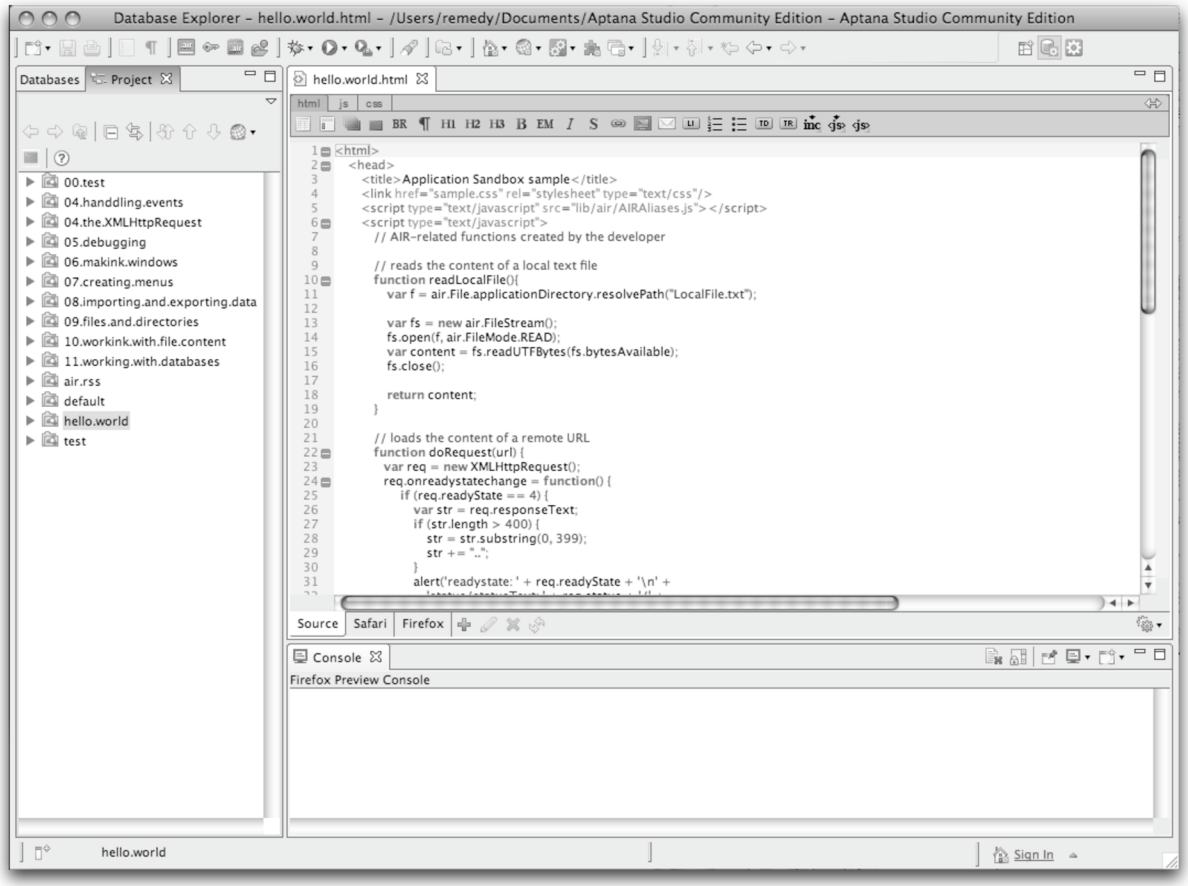


Obr. 5 Základní pracovní prostředí programu Dreamweaver CS4.

5.5 Aptana Studio

Kromě produktů společnoti Adobe jsou pro tvoru AIR aplikací k dispozici i jiná vývojová řešení. K jedním z nich patří Aptana Studio (viz Obr. 6), které umožňuje vývoj AIR aplikací založených na kombinaci technologií HTML a JavaScript. Mimo podpory pro AIR aplikace disponuje Aptana Studio i prostředky pro podporu širokého spektra různých programovacích jazyků.

Aptana Studio existuje ve dvou verzích: *Community Edition* a *Pro Edition*, přičemž Community Edition se dá využívat bezplatně. Pro podporu AIR aplikací je však nejprve nutné nainstalovat příslušnou aktualizaci.



Obr. 6 Vývojové prostředí programu Aptana Studio.

6 Tvorba instalačního balíčku

K tomu, aby byla možná distribuce a instalace AIR aplikace, je nutno nejprve vytvořit instalační balíček. Jedná se o soubor s příponou .air, který obsahuje všechna data potřebná k vlastní instalaci, konfiguraci a chodu samotné aplikace.

6.1 Konfigurační soubor AIR

Konfigurační soubor AIR je soubor typu XML, který obsahuje veškerá nastavení nezbytná k distribuci dané aplikace [6]. Samotný soubor je obsažen v AIR SDK, anebo je vytvořen příslušným vývojový nástrojem (Flex Builder 3, Flash CS3/CS4, Dreamweaver CS3/CS4, Aptana Studio apod.) Jelikož se jedná o XML soubor, jsou jednotlivá nastavení řešena prostřednictvím tagů s odpovídajícím názvem, z nichž některé jsou povinné a jiné nikoliv.

Seznam tagů, které nesmí chybět uvnitř hlavního tagu `<application>` konfiguračního souboru AIR je následující: `<id>`, `<filename>`, `<version>`, `<initialWindow>` a `<content>` uvnitř `<initialWindow>`.

Konfigurační soubor obsahuje následující tagy:

<application/>
Je kořenovým tagem konfiguračního souboru. Jeho jedinou vlastností je atribut `xmlns`, jež definuje verzi běhového modulu AIR, která je potřeba k instalaci a spuštění dané aplikace.

```
<application xmlns="http://ns.adobe.com/air/application/1.0">
```

Ostatní níže uvedené tagy se nachází mezi tímto otevíracím `<application>` a uzavíracím `<application/>` tagem.

<id/>
Jedná se o unikátní identifikátor dané aplikace a měl by tudíž být pro ni specifický. Doporučuje se proto použití doménového jmenného prostoru.

```
<id>cz.mojedomena.mojesubdomena.nakonfigurujto</id>
```

<version/>
Toto nastavení říká instalátoru, aby dal uživateli na vědomí, že verze právě instalované aplikace se liší od verze, kterou má uživatel již nainstalovanou v počítači. V žádném případě neříká nic o tom, zda je instalovaná aplikace novější nebo starší než již nainstalovaná aplikace.

```
<version>verze 1.0</version>
```

<name/>
Jako název aplikace může být zvoleno cokoliv. Název bude zobrazen během instalačního procesu a poté na příslušných místech daného operačního systému.

```
<name>NakonfigurujTo</name>
```

<filename/>
Název spustitelného souboru aplikace po instalaci aplikace. Tento název je taktéž zobrazen během instalačního procesu.

```
<filename>NakonfigurujTo</filename>
```

<description/>
Jedná se o stručný popis aplikace. Zobrazuje se v příslušné sekci dialogového okna během instalačního procesu.

```
<description>Ukázková aplikace demonstrující konfigurační
nastavení</description>
```

<copyright/>
Obsah tohoto tagu je zobrazen na odpovídajícím místě v menu o dané aplikaci.

```
<copyright>Copyright 2009 MojeDomena.com</copyright>
```

<icon/>
AIR aplikace může obsahovat vlastní ikonu, a to ve čtyřech velikostech (16x16, 32x32, 48x48, 128x128), přičemž formát souboru ikony musí být PNG.

```
<icon>
    <image16x16>icns/i16.png</image16x16>
    <image32x32>icns/i32.png</image32x32>
    <image48x48>icns/i48.png</image48x48>
    <image128x128>icns/i128.png</image128x128>
</icon>
```

Ikona potřebné velikosti se potom zobrazuje na příslušných místech daného operačního systému. V případě Windows je to např. záhlaví aplikace, lišta spuštěných aplikací, seznam nainstalovaných aplikací apod. V případě Mac OS X je to např. dock, informační okno aplikace, seznam spuštěných aplikací atd.

<installFolder/>
Umožňuje přenastavit výchozí umístění instalované aplikace. Například v Mac OS X bývá výchozí umístění instalace `/Applications`. Pokud se nastaví hodnota tagu na `AIRapp/`, bude aplikace nainstalována do složky `/Applications/AIRapp`.

```
<installFolder>AIRApp/</installFolder>
```

<customUpdateUI/>
Může obsahovat hodnoty `true` a `false`. Nastavení hodnoty true říká, že veškeré budoucí aktualizace dané aplikace si řeší aplikace sama. To znamená, že pokud se uživatel bude snažit nainstalovat nový instalační balíček dané aplikace, dojde ke spuštění již nainstalované aplikace, která se sama zaktualizuje.

```
<customUpdateUI>true</customUpdateUI>
```

<initialWindow>
Obsahuje hned několik parametrů, které se dají nastavit. Kromě tagu `<content>` jsou všechny další hodnoty nepovinné.

```
<initialWindow>
    <content></content>
    <title></title>
    <systmChrome></systemChrome>
    <transparent></transparent>
    <visible></visible>
    <minimizable></minimizable>
    <maximizable></maximizable>
    <resizable></resizable>
    <width></width>
    <height></height>
    <x></x>
    <y></y>
    <minSize></minSize>
    <MaxSize></maxSize>
</initialWindow>
```

<content/>
Slouží k nastavení kořenového souboru aplikace. Jedná se o hlavní zdrojový soubor, který může být typu `.swf` nebo `.html`. Tag `<content>` nesmí být prázdný v případě, že je aplikace kompilována pomocí AIR SDK, v ostatních případech jej doplní příslušný vývojový software.

```
<content>NakonfigurujTo.html</content>
```

<title/>
Hodnota tohoto tagu bude použita pro titulek hlavního okna aplikace.

```
<title>NakonfigurujTo</title>
```

<transparent/> a <systemChrome/>
Průhlednost aplikace lze nastavit pomocí tagu `<transparent>`. Povolené hodnoty jsou tedy `true` nebo `false`.

```
<transparent>true</transparent>
<systemChrome>none</systemChrome>
```

System Chrome je možno nastavit na `standard` nebo `none`. Jestliže je zvolena hodnota standard, AIR aplikace bude využívat nativních ovládacích prvků pro okno aplikace (zavřít, minimalizovat, maximalizovat). Pokud se však použije hodnota none, tak v případě HTML AIR aplikace, zde nebudou zádné ovládací prvky. V případě Flex aplikace bude mít okno alternativní ovládací prvky, které lze však jednoduše zakázat.

Průhlednost aplikace lze povolit pouze tehdy, je-li hodnota `<systemChrome>` nastavena na false.

<visible/>
Dovoluje nastavit viditelnost aplikace. Výchozí hodnotou je `true`, nicméně v případě použití `false` je tuto hodnotu možné změnit na true i po spuštění aplikace.

```
<visible>false</visible>
```

<minimizable/>, <maximizable/> a <resizable/>
Výchozí hodnotou těchto všech tagů je `true`, avšak uživatel má možnost zakázat tyto vlastnoti nastavením hodnoty na `false`.

```
<minimizable>false</minimizable>
<maximizable>false</maximizable>
<resizable>false</resizable>
```

<width/> a <height/>
Slouží k nastavení šířky a výšky hlavního okna aplikace.

```
<height>600</height>
<width>800</width>
```

<x/> a <y/>
Nastavují pozici `x` zleva a `y` shora hlavního okna aplikace.

```
<x>100</x>
<y>200</y>
```

<minSize/> a <maxSize/>
Slouží k omezení minimálních a maximálních rozměrů hlavního okna aplikace.

```
<minSize>640 480</minSize>
<maxSize>800 600</maxSize>
```

<allowBrowserInvocation/>
Hodnota nastavená na `true` povoluje možnost instalace aplikace z webového prohlížeče. Jedná se pak o konkrétní instalační balíček, který je předurčen pro tento způsob instalace.

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

<programMenuFolder/>
Říká instalátoru, že má vytvořit zástupce s daným názvem uvnitř Windows start menu.

```
<programMenuFolder>AIRApp</programMenuFolder/>
```

<fileTypes/>
Umožňuje definovat libovolný počet typů souboru, jež budou s danou aplikací asociovány. To znamená, že uvnitř tagu `<fileTypes>` může být jakýkoliv počet definic tagu `<fileType>` s příslušným obsahem.

```
<fileTypes>
    <fileType>
        <name>Test</name>
        <extension>test</extension>
        <description>Popis typu souboru</description>
        <contentType>text/plain</contentType>
        <icon>
            <image16x16>i16.png</image16x16>
            <image32x32>i32.png</image32x32>
            <image48x48>i48.png</image48x48>
            <image128x128>i128.png</image128x128>
        </icon>
    </fileType>
</fileTypes>
```

6.2 Kompilace aplikace pomocí AIR SDK

Je mnoho možností, jak vytvořit instalační soubor s příponou *.air*. Každé z již zmiňovaných vývojových prostředí (Flex Builder 3, Flash CS3/CS4, Dreamweaver CS3/CS4, Aptana Studio) poskytuje grafické rozhraní pro nastavení konfiguračního XML souboru a finální kompilaci aplikace. Kromě těchto možností je zde také varianta použití AIR SDK (Software Development Kit).

AIR SDK sice nemá žádné grafické rozhraní, nicméně je zcela zdarma. Samotná kompilace aplikace se děje z prostředí příkazové řádky operačního systému [6].

Podle typu AIR aplikace (Flex, Flash, HTML/JavaScript) je potřeba nastavit konfigurační XML soubor. Konkrétně se jedná o obsah tagu `<content>`, který se nachází uvnitř tagu `<initialWindow>`.

V případě Flex nebo Flash aplikace musí být použit soubor s příponou *.swf*, jako např. `<content>HelloWorld.swf</content>` a v případě HTML/JavaScript aplikace zase soubor s příponou *.html*, např. `<content>HelloWorld.html</content>`.

Dále je zapotřebí obstarat soubor s digitálním certifikátem. Ten je možno získat od konkrétních věrohodných společností, anebo si jej může uživatel vytvořit sám (při distribuci aplikace však tento typ certifikátu nepůsobí věrohodně).

Vzor příkazu pro vytvoření vlastního digitálního certifikátu:

```
adt -certificate -cn name [-ou org_unit][-o org_name]
[-c country] key_type pfx_file password
```

Konkrétní příklad příkazu pro vytvoření certifikátu:

```
adt -certificate -cn SelfSign -ou QE -o "NazevOrganizace"
-c US 1024-RSA mycert.pfx 1b2c3d
```

Jakmile je k dispozici soubor s certifikátem, je možná kompilace AIR aplikace. Vzor kompilačního příkazu vypadá následovně:

```
adt -package SIGNING_OPTIONS <air-file> <app-desc> FILE_ARGS
```

Konkrétní příklad kompilace HTML aplikace (přípona .html):

```
adt -package -storetype pkcs12 -keystore mycert.pfx
HelloWorld.air application.xml HelloWorld.html
```

Příklad kompilace Flex nebo Flash aplikace (přípona .swf):

```
adt -package -storetype pkcs12 -keystore mycert.pfx
HelloWorld.air application.xml HelloWorld.swf
```

Příklad kompilace zahrnující další soubory aplikace:

```
adt -package -storetype pkcs12 -keystore mycert.pfx
HelloWorld.air application.xml HelloWorld.swf
icons/i16.png icons/i32.png icons/i48.png icons/i128.png
```

Příklad kompilace s přidáním celé složky *icons*:

```
adt -package -storetype pkcs12 -keystore mycert.pfx
HelloWorld.air application.xml HelloWorld.swf /icons
```

Jelikož ADT je vlastně spustitelný soubor obsažený v AIR SDK, je potřeba zadávat celou cestu k tomuto souboru. Lze však provést konfigurace, která umožní použití příkazu ADT bez nutnosti poskytnutí úplné cesty. Tato konfigurace se liší s daným operačním systémem. Po konfiguraci pak stačí volat příkaz ADT ze složky, která obsahuje konfigurační XML soubor, jinak je také potřeba zadat úplnou cestu k tomuto souboru.

7 Vlastní aplikace

Aby bylo možné ověřit, že technologie AIR funguje odpovídajícím způsobem, byla vytvořena aplikace, která využívá zakladní, ale i některé z pokročilých možností této technologie. Ve vlastní aplikaci byla tedy snaha použít co největší množství ostatních technologií a přístupů, které jsou k dispozici při tvorbě AIR aplikací.

Jako vlastní aplikace byla proto zvolena RSS čtečka, která kromě technologií HTML, JavaScript, CSS, DOM a AJAX, využívá i pokročilých funkcí technologie AIR. Konkrétně se jedná o využití jádra WebKit webového prohlížeče, SQLite databáze a asynchronního přistupu při tvorbě dané aplikace.

7.1 RSS

RSS (Really Simple Syndication) je rodina tzv. *web feed* formátů (zpravidla XML) používaných k publikaci často aktualizovaného obsahu ve standardizovaném formátu [14].

Samotná publikace obsahu se většinou děje na základě umístění odkazu na RSS dokument na webovou stránku. Uživateli pak stačí poskytnout tento odkaz RSS čtečce, která poté umožní čtení aktualit z daného zdroje. Tento způsob odběru informací přináší určité výhody:

- Možnost generování RSS dokumentů
- Čtení novinek z různých zdrojů na jednom místě
- Možnost čtení jednou publikovaného obsahu pomocí různých typů RSS čteček

RSS dokument (také často zvaný *feed*, *web feed* nebo *channel*) obsahuje text plus metadata (např. datum zveřejnění, autor atd.). Jedná se soubor typu XML, který má určitou standardní strukturu. Struktura XML souboru se liší podle dané verze RSS dokumentu, přičemž většinou obsahuje povinné a nepovinné tagy. Dnes se nejčastěji používá RSS verze 2. V této verzi je jen 5 povinných tagů:

- <rss>
- <channel>
- <title>
- <link>
- <description>

Příklad jednoduchého RSS verze 2 dokumentu [15]:

```
<?xml version="1.0"?>
<rss version="2.0">
 <channel>
  <title>RSS dokument</title>
  <link>http://www.example.cz/rss/</link>
  <description>Příklad RSS dokumentu</description>
  <language>cs</language>
  <pubDate>Tue, 10 Jun 2003 04:00:00 GMT</pubDate>
  <lastBuildDate>Tue, 10 Jun 2003 09:41:01 GMT</astBuildDate>
  <docs>http://blogs.law.harvard.edu/tech/rss</docs>
  <generator>MujSystem 1.0</generator>
  <managingEditor>editor@example.com</managingEditor>
  <webMaster>webmaster@example.com</webMaster>

  <item>
   <title>Třetí položka</title>
   <link>http://example.com/rss/treti-polozka.htm</link>
   <description>Třetí položka RSS dokumentu.</description>
   <pubDate>Tue, 03 Jun 2003 09:39:21 GMT</pubDate>
   <guid>http://www.example.com/rss/example.html#3</guid>
  </item>

  <item>
   <description>Druhá položka RSS dokumentu.</description>
   <pubDate>Fri, 30 May 2003 11:06:42 GMT</pubDate>
   <guid>http://www.example.com/rss/example.html#2</guid>
  </item>

  <item>
    <title>První den</title>
    <link>http://example.com/rss/prvni-polozka.htm</link>
    <description>První položka RSS dokumentu.</description>
    <pubDate>Fri, 30 May 2003 9:16:28 GMT</pubDate>
    <guid>http://www.example.com/rss/example.html#1</guid>
  </item>
 </channel>
</rss>
```

Kdyby RSS dokument obsahoval pouze povinné tagy, nebyl by příliš užitečný. Proto je zde také celá řada nepovinných elementů (více viz [16]). V příkladu se konkrétně jedná o element <item>, který spolu s jeho příslušným obsahem tvoří jednu novinku RSS kanálu.

Kromě formátu RSS existují i jiné varianty jako např. ATOM nebo CDF.

7.3 SQLite

SQLite je tzv. *embedded* relační databáze, tzn. že neběží jako samostatný proces, ale je přímou součástí aplikace, která ji využívá [19]. Jedná se vlastně o takové soběstačné databázové jádro. Samotná SQLite databáze je pak v podobě souboru uložena v příslušné složce na disku počítače.

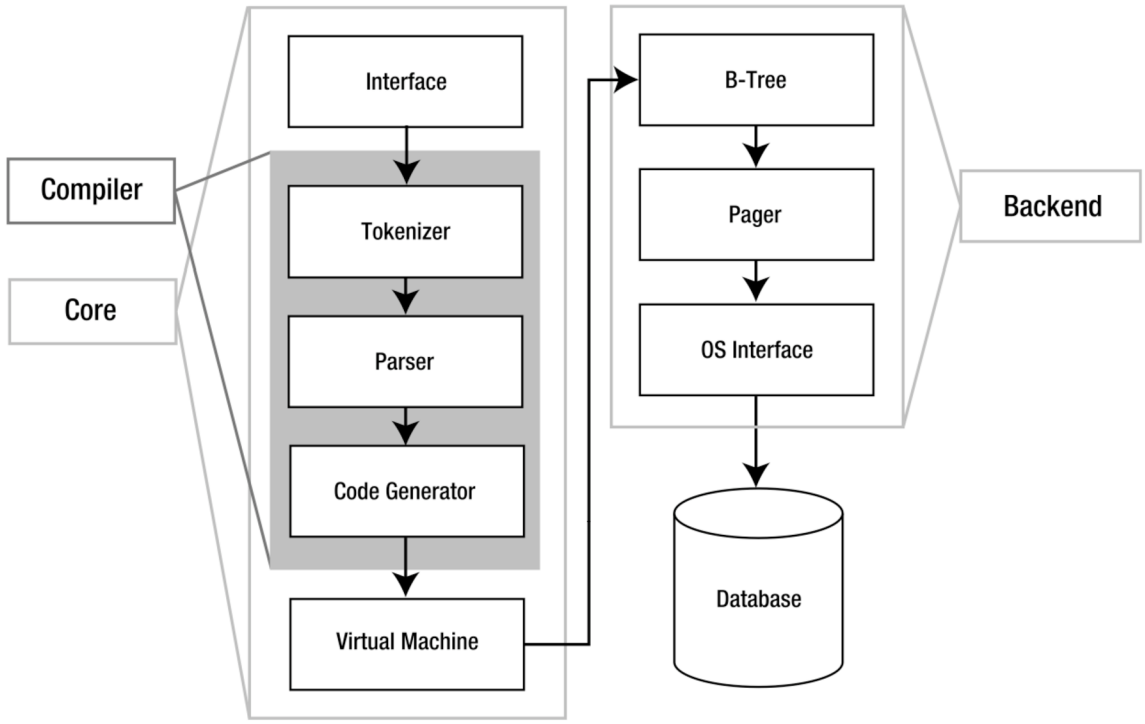
Hlavní výhodou vlastního databázového serveru, který je přímo součástí aplikace je:

- Žádná síťová konfigurace nebo administrace
- Klient a server běží společně jako jeden proces
- Redukce zatížení souvisejícího s neustálými požadavky na server
- Zjednodušení administrace samotné databáze
- Jednodušší distribuce dané aplikace

K dalším výhodám SQLite databáze patří:

- Jednoduchost (dobře zdokumentované a jednoduché API)
- Kompaktnost (velikost databázového souboru asi 250 kB)
- Přenositelnost (databáze vytvořená na jedné operační platformě funguje i na druhé operační platformě)
- Žádné poplatky ani omezení v používání

Modulární architektura SQLite databáze (viz Obr. 7) je tvořena osmi oddělenými moduly, které spadají pod tři hlavní subsystémy: core (jádro), compiler (kompilátor) a backend (paměťový blok). Zpracování dotazu je pak těmito moduly rozděleno na samostatné úlohy, které jsou vykonávány postupně.



Obr. 7 Struktura SQLite databáze.

Interface
Jedná se o SQLite C API rozhraní, pomocí kterého programy, skriptovací jazyky, knihovny apod. komunikují s SQLite databází.

Compiler
Tvoří jej *Tokenizer*, *Parser*, *Code Generator*. Tokenizer a Parser v podstatě pracují společně, jejich úkolem je validace syntaxe a převedení SQL dotazu v textové formě na určitou datovou strukturu, se kterou jsou schopny pracovat nižší vrstvy. Code Generator překládá pak tato data do jazyka specifického pro SQLite, jehož instrukce vykonává *Virtual Machine*.

Virtual Machine
Podobně jako Java Virtual Machine nebo interpret skriptovacího jazyka pracuje Virtual Machine s bytovým kódem. Obsahuje určitou instrukční sadu, na základě které se provádí dané databázové operace (SELECT, INSERT, DELETE, UPDATE atd.).

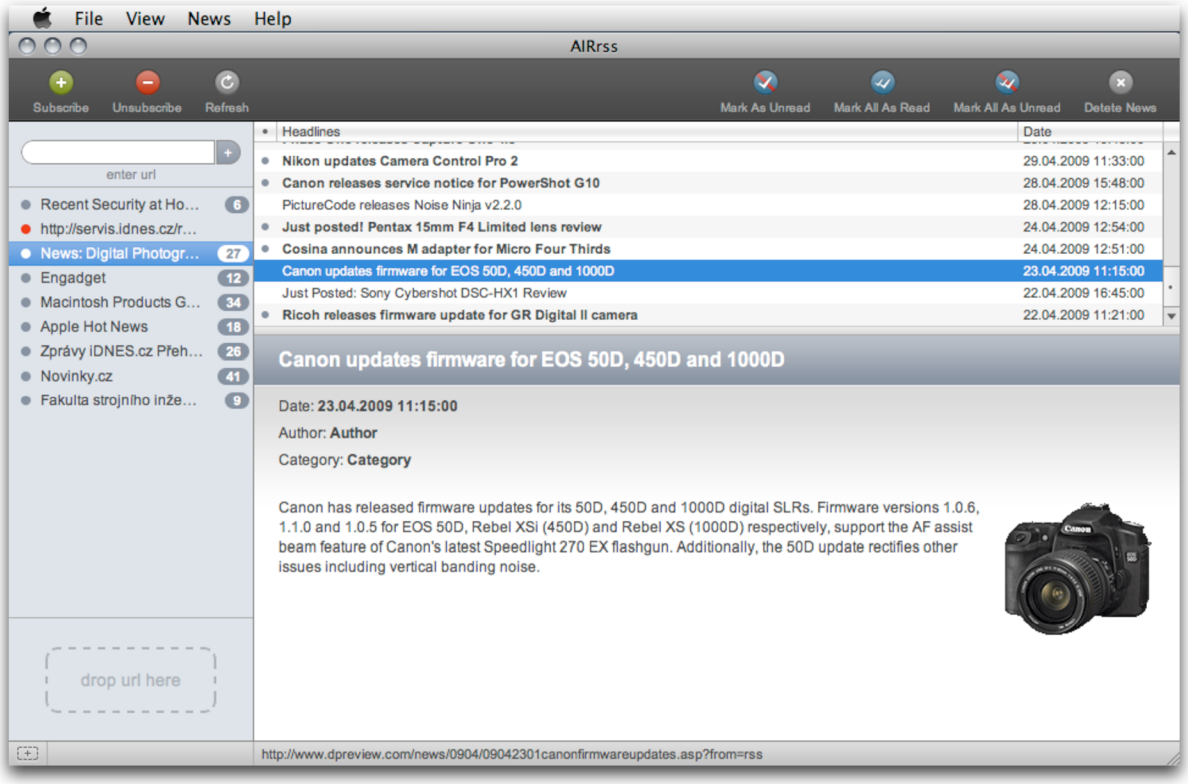
Backend
Jedná se o paměťový blok SQLite databáze složený ze stromových datových struktur, které jsou vysoce optimalizovány k prohledávání. Organizaci těchto struktur zajišťuje *B-Tree*, zatímco *Pager* zprostředkovává přenos dat mezi diskem počítače a samotným paměťovým blokem SQLite.

7.4 RSS čtečka

RSS čtečka je program sloužící ke čtení zpráv většinou z více zdrojů na jednom místě. Uživatel poskytne čtečce odkaz na určitý RSS kanál, ze kterého chce odebírat zprávy. Tento odkaz směřuje na XML soubor, který obsahuje nejen data o samotném RSS kanálu, ale i data o všech zprávách, které publikuje. RSS čtečka poté načte potřebná data z tohoto souboru a umožní uživateli přístup k těmto datům pomocí vhodného grafického rozhraní. Uživatel má pak kromě samotného čtení zpráv k dispozici i další funkce jako např. odebrání stávajících RSS kanálů, aktualizace zpráv daného RSS kanálu, mazání zpráv atd.

Jak již bylo zmíněno dříve, vlastní AIR RSS čtečka (viz Obr. 8) je postavena na těchto technologiích a přístupech:

- HTML (Hypertext Markup Language)
- JavaScript
- CSS (Cascading Style Sheets)
- DOM (Document Object Model)
- AJAX (Asynchronous JavaScript and XML)



Obr. 8 Vlastní AIR RSS čtečka na operační platformě Mac OS X.

HTML spolu s **CSS** tvoří grafické rozhraní dané aplikace (viz Obr. 8). Použité HTML značky slouží jako základní stavební prvky jednotlivých částí aplikace (lišta s funkčními tlačítky, seznam RSS kanálů, seznam zpráv vybraného RSS kanálu, stavový řádek apod.), přičemž jejich veškerá grafická podoba (rozměry, barvy, formát textu, obrázky atd.) je řešena pomocí CSS.

JavaScript a **DOM** zajišťuje funkční stránku aplikace. Jedná se např. o stažení obsahu RSS ze vzdáleného serveru, generování seznamu RSS kanálů, zpráv a obsahu dané zprávy, aktualizaci zpráv apod.

Aby bylo ovládání samotné aplikace plynulé, je třeba použít určitých programátorských technik při tvorbě dané aplikace. Proto také vlastní AIR aplikace kromě **AJAX** přístupu pro stahování dat ze serveru využívá i asynchronního spojení s lokální SQLite databází.

Výhodou asynchronního přístupu je to, že během právě probíhající programové operace je možno provádět další operace, aniž by se muselo čekat na dokončení předchozí operace, jako je tomu v případě synchronního přístupu. Jednotlivé operace totiž běží v pozadí právě prováděné operace, čímž je dosažena plynulá práce s daným programem.

Zatímco WebKit se stará o renderování HTML a CSS kódu a vykonávání JavaScriptu, SQLite databáze umožňuje kromě rychlého a sofistikovaného přístupu k datům i offline čtení samotných RSS zpráv.

Základem každé použitelné aplikace je grafické rozhraní, pomocí kterého je uživatel schopen jednoduše a intuitivně využívat jednotlivých funkcí dané aplikace. Popis hlavních částí grafického rozhraní vlastní AIR RSS čtečky (viz. Obr. 9):

(1) Hlavní menu aplikace

(2) Lišta s funkčními tlačítky

(3) Pole pro přidání RSS kanálu

(4) Seznam RSS kanálů

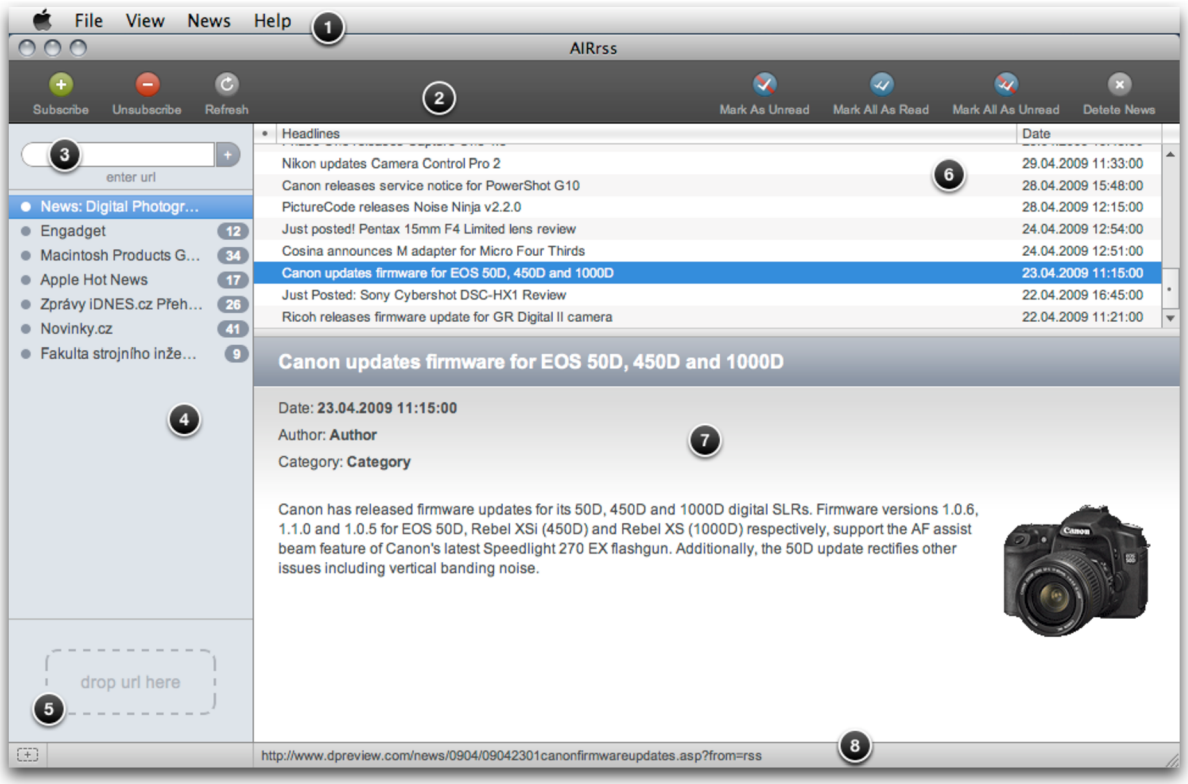
(5) Pole pro přidání RSS kanálu

(6) Seznam zpráv vybraného RSS kanálu

(7) Detail vybrané zprávy

(8) Stavový řádek

Hlavní menu aplikace (1)
Jedná se o nativní menu dané aplikace. Obsahuje kromě funkcí přístupných z lišty s funkčními tlačítky (2) i některé další funkce jako např. ukončení aplikace, zobrazení okna aplikace přes celou obrazovky nebo změnu rozlišení okna aplikace. Často používané funkce jsou navíc přístupné přes klávesové zkratky.

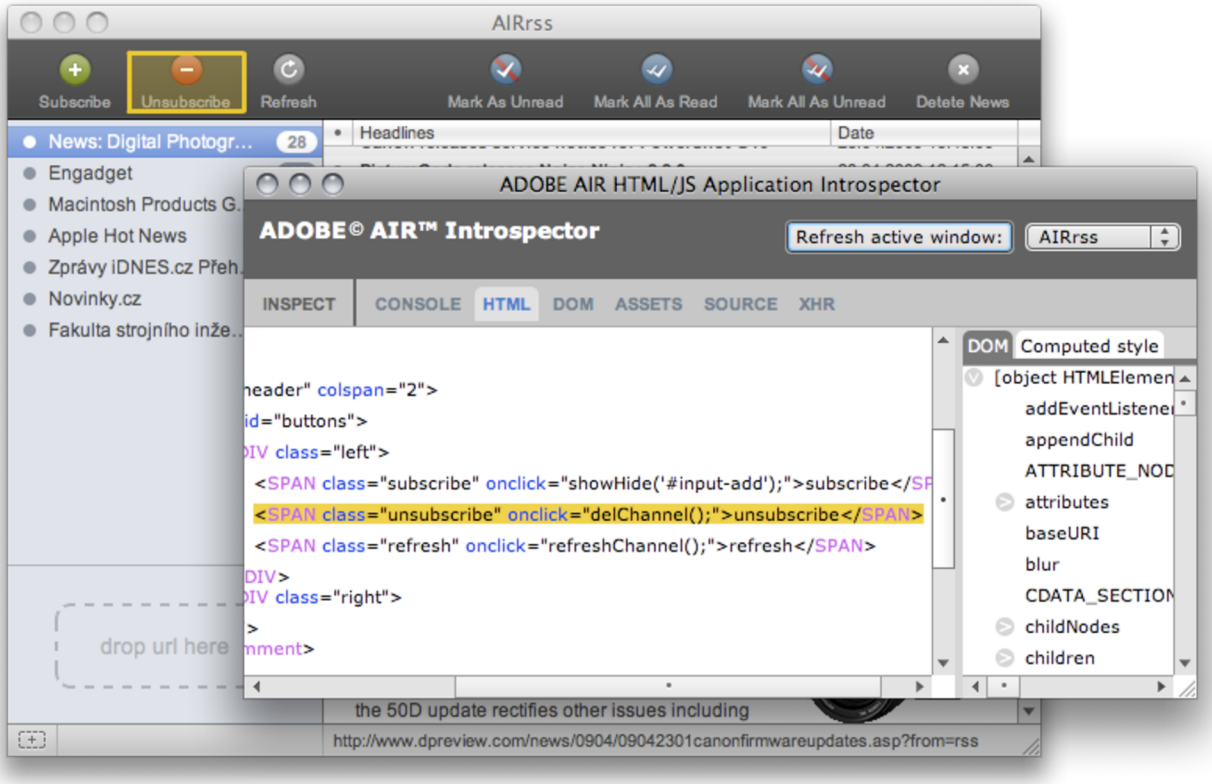


Obr. 9 Popis jednotlivých částí vlastní AIR RSS čtečky.

Každý operační systém má své určité zvyklosti v popisu názvů základních funkcí a jejich klávesových zkratek v nativním menu aplikace. Kromě toho jsou zde i rozdíly v tom, jak konkrétní operační systém vytváří samotné nativní menu. Například operační platforma Mac OS X používá tzv. *Application menu* (viz. Obr. 8), které není součástí okna aplikace, naopak Windows používá tzv. *Windows menu*, které je součástí každého okna aplikace.

Aby tedy byly zachovány zvyklosti nativního menu na dané operační platformě, je potřeba podniknout příslušná opatření při samotné tvorbě nativního menu aplikace. Vlastní AIR aplikace tyto zvyklosti zachovává v případě operačních systémů Mac OS X a Windows.

Lišta s funkčními tlačítky (2)
Jednotlivá tlačítka jsou zastoupena vhodnými HTML tagy, které obsahují příslušný název tlačítka. Každý z těchto tagů má nastaven atribut *onclick* na hodnotu funkce, která se zavolá v případě kliknutí na dané tlačítko (viz Obr. 10).



Obr. 10 Zdrojový HTML kód funkčního tlačítka.

Pole pro přidání RSS kanálu (3)
Je tvořeno pomocí klasických HTML formulářových prvků. Po vložení url RSS kanálu a kliknutí na tlačítko "+" se provede kontrola na dostupnost daného url. Jestliže není url k dispozici (uživatel offline), vloží se url do SQLite databáze pro možnost pozdější aktualizace. V seznamu přidaných kanálů (4) je pak takové url označeno červeným puntíkem (viz Obr. 8). V případě, že dané url k dispozici je, dojde ke stažení příslušného XML souboru a potřebná data se pak uloží do SQLite databáze.

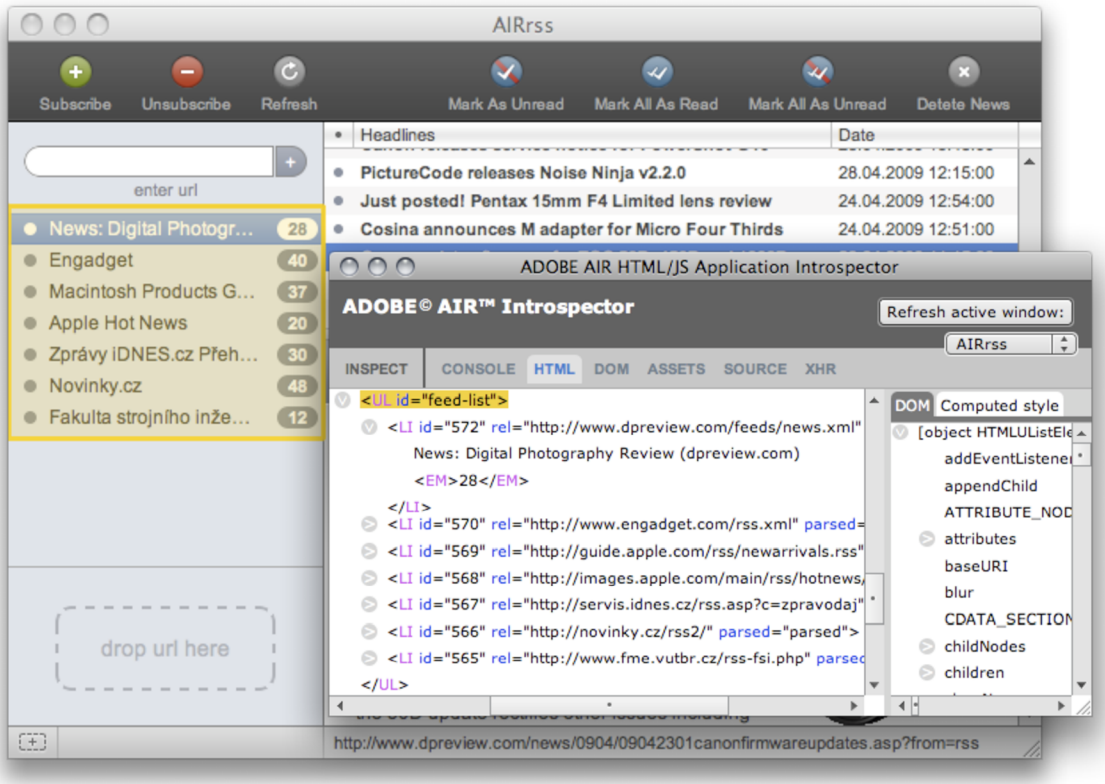
Kontrola dostupnosti url se provádí pomocí objektu *URLMonitor*, který je definován uvnitř *servicemonitor.swf* souboru, jež je součástí AIR SDK. Použití objektu URLMonitor je tedy podmíněno deklarací příslušného souboru v hlavičce HTML souboru, jež tvoří danou aplikaci.

Stažení samotného XML souboru probíhá na základě *XMLHttpRequest* dotazu, což je objekt JavaScriptu používaný pro výměnu dat mezi webovou stránkou a vzdáleným serverem, přičemž data můžou být ve formátu XML, HTML nebo prostý text. Mimo jiné je XMLHttpRequest základem AJAX přístupu.

Zajímavostí technologie AIR je, že lze tento XMLHttpRequest dotaz provádět i mezi různými doménami, což v případě webových stránek není zatím možné z bezpečnostních důvodů.

Seznam RSS kanálů (4)
Jedná se o klasický HTML netříděný seznam, který se generuje JavaScriptem při každé potřebné aktualizaci seznamu (přidání nového RSS kanálu, odebrání stávajícího RSS kanálu, přečtení zprávy, atd.) na základě manipulace s DOM stromem. Tento přístup byl zvolen proto, protože je mnohem jednodušší vygenerovat celý seznam znovu než aktualizovat jen některou jeho část.

Aby bylo možné jednoduše provádět vesměs základní operace nad SQLite databází (INSERT, SELECT, UPDATE, DELETE), jsou režijní informace (např. primární klíč RSS kanálu v databázi, url RSS kanálu apod.) zapsány jako hodnoty odpovídajících atributů jednotlivých HTML tagů netříděného seznamu (viz Obr. 11).

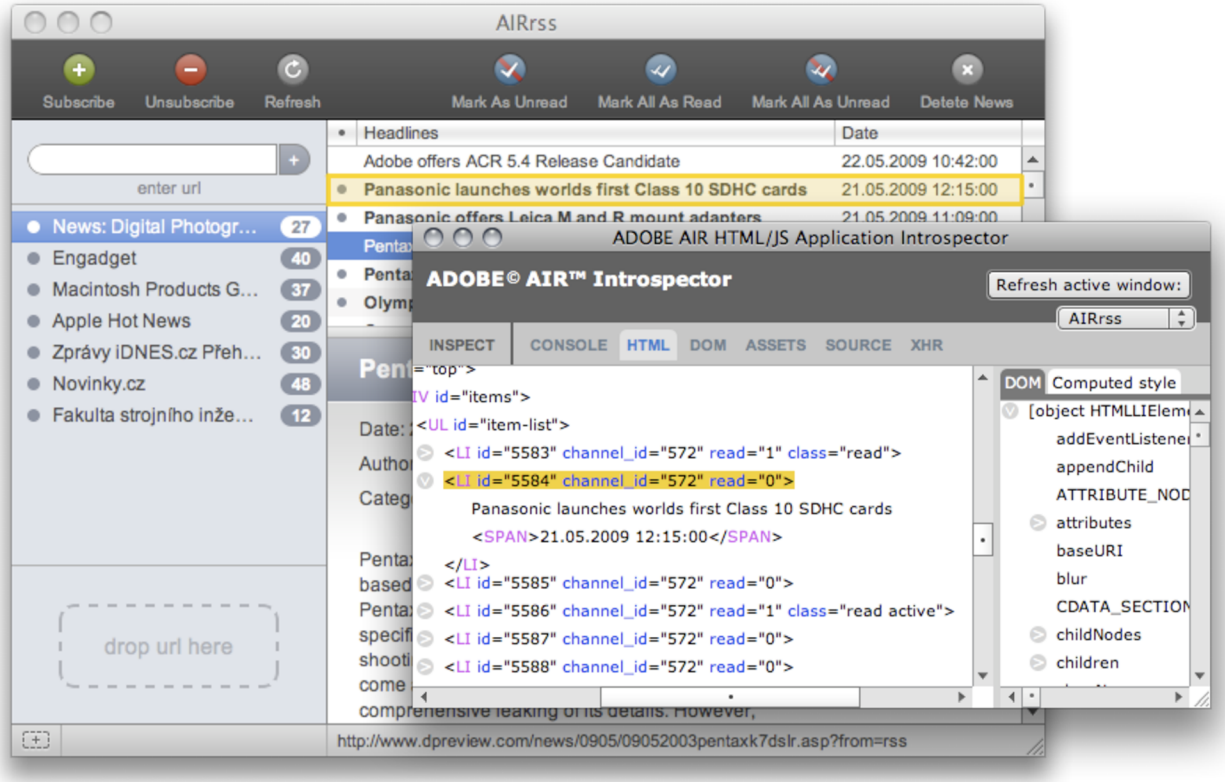


Obr. 11 Zdrojový HTML kód seznamu RSS kanálu.

Pole pro přidání RSS kanálu (5)
Plní stejnou funkci jako pole (3), rozdíl je pouze ve způsobu vložení url RSS kanálu. Zatímco v případě pole (3), kdy je RSS kanál přidán na základě vložení url do textového pole a následným kliknutím na tlačítko "+", v případě pole (5) je RSS kanál přidán přetažením daného url myší nad toto pole (5) a následným uvolněním levého tlačítka myši (tzv. *drag-and-drop* způsob).

Seznam zpráv vybraného RSS kanálu (6)
Stejně jako seznam RSS kanálů (4) je i seznam RSS zpráv (6) generován JavaScriptem jako HTML netříděný seznam (viz Obr. 12). Veškeré režijní informace jsou opět řešeny jako hodnoty odpovídajících atributů jednotlivých HTML tagů netříděného seznamu.

Jelikož informace týkající se data, roku a času zveřejnění jednotlivých RSS zpráv jsou v XML souboru zapsány v různých formátech (např. *Fri, 22 May 2009 07:43:19 PDT*), byl nutný převod času do našeho časového pásma. K tomu slouží v JavaScriptu objekt *Date*, který obsahuje všechny potřebné metody pro práci s datovým formátem.

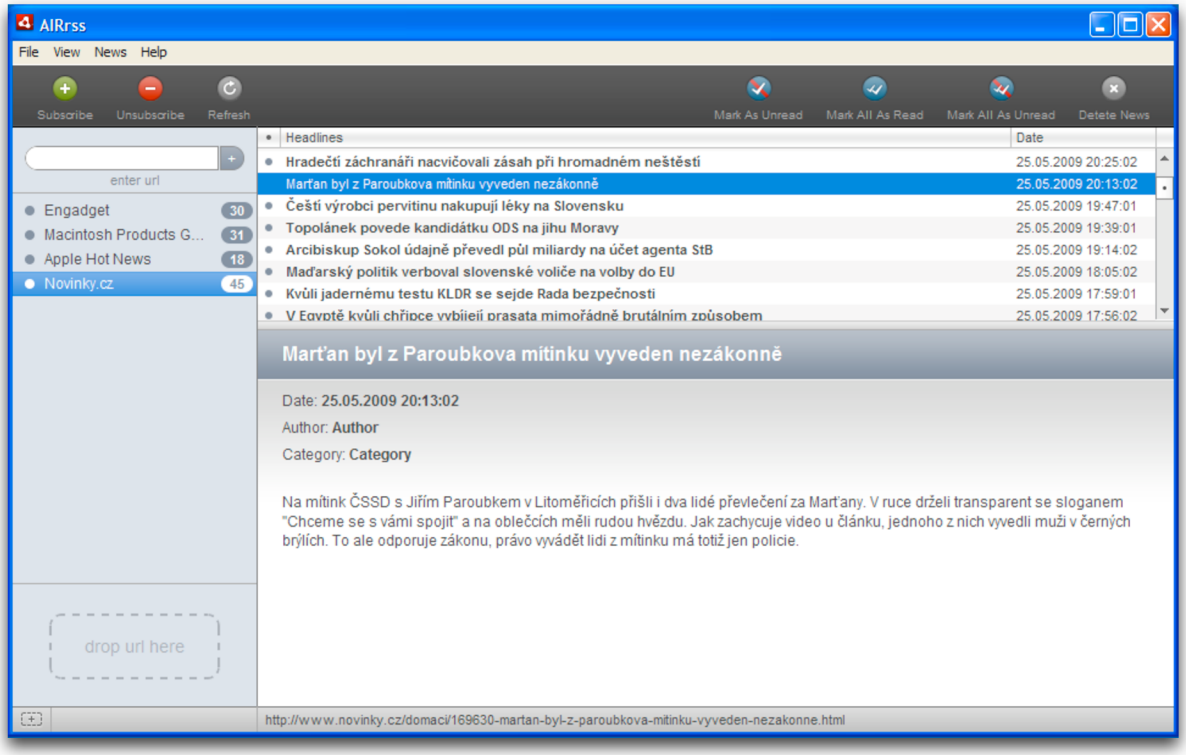


Obr. 12 Zdrojový HTML kód seznamu zpráv vybraného RSS kanálu.

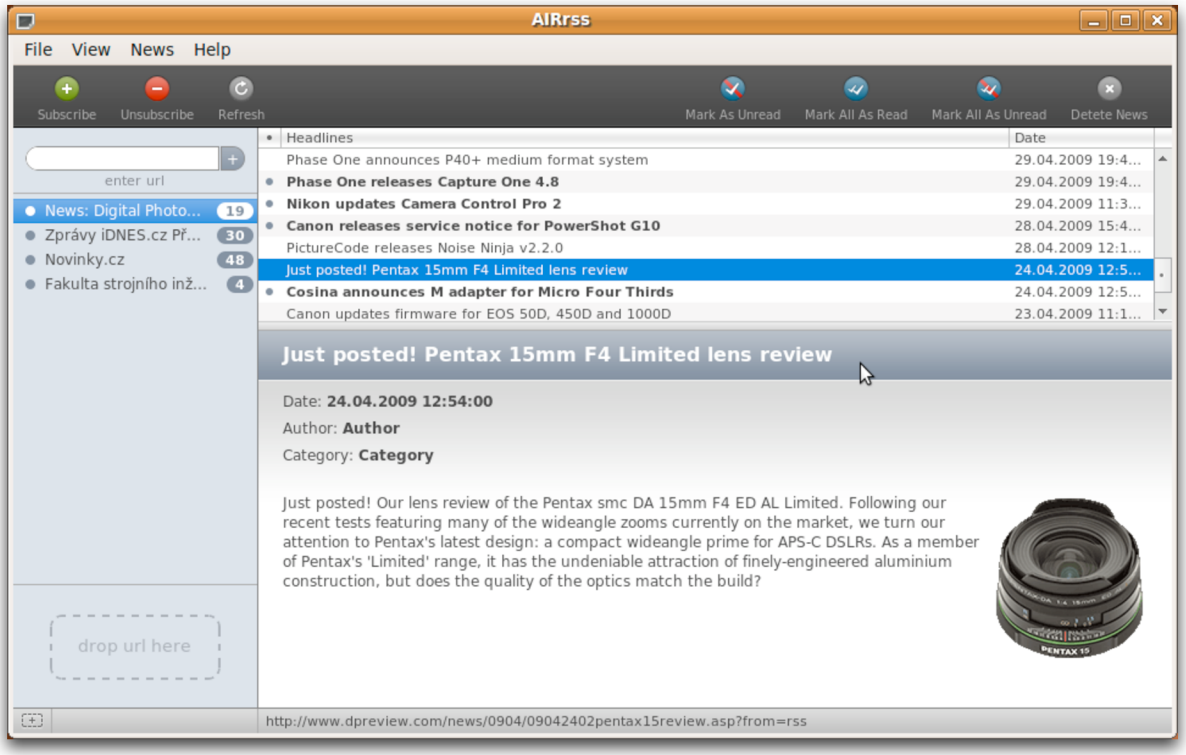
Detail vybrané zprávy (7)
Opět řešen jako HTML netříděný seznam, který se generuje JavaScriptem pro právě zvolenou zprávu. Kromě obsahu samotné zprávy obsahuje i některé další informace. Jedná se o titulek dané zprávy, který je zároveň odkazem na odpovídající článek na webu, datum publikace, autora zprávy apod. Url odkazu titulku zprávy se zobrazuje ve stavovém řádku (8).

Stavový řádek (8)
Slouží k zobrazení url odkazu, na který uživatel najel myší, přičemž ve výchozím stavu je zobrazeno url odkazu titulku vybrané zprávy. Všechny odkazy, na které uživatel klikne myší, se otevírají ve výchozím webovém prohlížeči pro daný operační systém.

Vlastní aplikace byla testována na operačních platformách Mac OS X, Windows a Linux. Na všech zmiňovaných operačních systémech nebyly zaznamenány žádné funkční rozdíly, přičemž zůstala zachována i grafická podoba dané aplikace (viz Obr. 8, Obr. 13, Obr. 14).



Obr. 13 Vlastní AIR RSS čtečka na operační platformě Windows (verze XP).



Obr. 14 Vlastní AIR RSS čtečka na operační platformě Linux (distribuce Ubuntu).

8 Závěr

První webový prohlížeč byl představen před skoro sedmnácti lety. Za tu dobu se webové technologie i možnosti webových aplikací značně změnily. Kromě toho je poptávka po webových řešeních, které musí splňovat mnohdy velmi náročné požadavky, stále větší. Webové aplikace jsou tak nuceny používat různých technologií a přístupů, což však může vést k problémům s funkčností samotných aplikací v ostatních webových prohlížečích.

Řešením nekompatibilit aplikací s webovými prohlížeči je vytvoření odpovídající desktopové aplikace. To však často přináší potřebu vytvořit stejnou aplikaci, avšak pro více operačních systémů. S tím také souvisí i náklady na vývoj takové aplikace, což může být ve velkém množství případů problém.

Cílem této diplomové práce bylo proto představení nové technologie, která kombinuje výhody webových a desktopových aplikací. Dává tak možnost webovým vývojářům zúročit své dosavadní znalosti z tvorby webových aplikací v podobě aplikací, které však běží nativně v prostředí různých operačních systémů, a to bez nutnosti kompilace aplikace pro konkrétní operační platformu.

V diplomové práci byly tak kromě představení samotné technologie AIR poskytnuty i informace o některých dalších technologiích a přístupech, které nemusí být příliš rozšířeny. Jelikož jsem nenarazil na téměř žádnou česky psanou literaturu věnující se této problematice, dala tato práce možnost vzniku českého textu, který může sloužit nejen jako zdroj informací ohledně tvorby a technologiích použitých v AIR aplikacích, ale i jako inspirace potenciálním desktopovým programátorům AIR aplikací.

Pro potřeby testování možností technologie AIR byla vytvořena aplikace, která je však použitelná i jako alternativa jiné odpovídající desktopové aplikace. Jednotlivé funkce této aplikace byly ověřeny na všech technologií AIR podporovaných operačních platformách (Mac OS X, Windows a Linux). Během samotného testování nebyly zaznamény žádné rozdíly ve funkčnosti ani grafické podobě vlastní aplikace.

Osobně si myslím, že technologie AIR má budoucnost, ve které určitě obohatí pestrou paletu dosavadních desktopových aplikací.

Použité zdroje

[1] LOTT, J. *Adobe AIR in Action*. Greenwich (Connecticut): Manning, 2008. ISBN 1-933988-48-7.

[2] ULLMAN, L. *Adobe AIR with Ajax: Visual QuickPro Guide*. Berkeley (CA): Peachpit Press, 2008. ISBN 978-0-321-52461-4.

[3] GORTON, B. - TAYLOR, R. - YAMADA, J. *Adobe AIR Bible*. Indianapolis (Indiana): Wiley Publishing, 2008. ISBN: 978-0-470-28468-1.

[4] CHAMBERS, M., aj. *Adobe AIR for JavaScript Developers: Pocket Guide*. First Edition. Sebastopol (CA): O'Reilly Media, 2008. ISBN: 978-0-596-51837-0.

[5] CASARIO, M., aj. *The Essential Guide to Flash CS4 AIR Development*. Berkeley (CA): Apress, 2009. ISBN: 978-1-4302-1588-2.

[6] TRETOLA, R. *Beginning Adobe AIR: Building Applications for the Adobe Integrated Runtime*. Indianapolis (Indiana): Wiley Publishing, 2008. ISBN: 978-0-470-22904-0.

[7] *Asynchronous JavaScript and XML* [online]. Poslední revize 15.5.2009 [cit.2009-05-17]. Dostupné z: <http://cs.wikipedia.org/wiki/Asynchronous_JavaScript_and_XML>.

[8] *Rich Internet Applications: Benefits of RIA's* [online]. [cit.2009-05-18]. Dostupné z: <http://www.kalalau.org/solutions_ria_benefits.html>.

[9] *HyperText Markup Language* [online]. Poslední revize 27.4.2009 [cit.2009-05-18]. Dostupné z: <http://cs.wikipedia.org/wiki/HyperText_Markup_Language>.

[10] *JavaScript* [online]. Poslední revize 16.4.2009 [cit.2009-05-18]. Dostupné z: <http://cs.wikipedia.org/wiki/HyperText_Markup_Language>.

[11] *Cascading Style Sheet* [online]. Poslední revize 2.4.2009 [cit.2009-05-18]. Dostupné z: <http://cs.wikipedia.org/wiki/Cascading_Style_Sheets>.

[12] *Extensible HyperText Markup Language* [online]. Poslední revize 9.5.2009 [cit.2009-05-18]. Dostupné z: <http://cs.wikipedia.org/wiki/Extensible_HyperText_Markup_Language>.

[13] *Document Object Model* [online]. Poslední revize 3.4.2009 [cit.2009-05-19]. Dostupné z: <http://cs.wikipedia.org/wiki/Document_Object_Model>.

[14] *RSS* [online]. Poslední revize 20.5.2009 [cit.2009-05-21].
Dostupné z: <http://en.wikipedia.org/wiki/RSS_(file_format)>.

[15] *RSS 2.0* [online]. Poslední revize 16.9.2004 [cit.2009-05-21].
Dostupné z: <http://interval.cz/clanky/rss-2-0/>.

[16] *RSS 2.0 Specification* [online]. Poslední revize 29.4.2007 [cit.2009-05-21].
Dostupné z: <http://cyber.law.harvard.edu/rss/rss.html>.

[17] *Safari and WebKit Overview: Features, Enhancements, and Open Source Development* [online]. c2009, [cit.2009-05-21].
Dostupné z: <https://developer.apple.com/safari/index.php>.

[18] *WebKit* [online]. Poslední revize 7.5.2007 [cit.2009-05-21].
Dostupné z: <http://en.wikipedia.org/wiki/WebKit>.

[19] OWENS, M. *The Definitive Guide to SQLite*. Berkeley (CA): Apress, 2006.
ISBN: 978-1-59059-673-9.

Seznam příloh

Obsah přiloženého DVD:

- **aplikace**
  - air.rss.air
  - **behovy_modul**
    - **linux**
      - AdobeAIRInstaller.bin
    - **macintosh**
      - AdobeAIR.dmg
    - systemove.pozadavky.txt
    - **windows**
      - AdobeAIRInstaller.exe
  - instrukce.pro.instalaci.txt
  - **zdrojovy_kod**
    - air.rss.zip
- **dipl_prace_pdf**
  - 2009_DP_Simik_Adam_53929.pdf