Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta

Katedra informatiky

Tvorba internetových aplikací pomocí Rich Internet Application Adobe Flex

Bakalářská práce

Karel Peka

Vedoucí práce

PaedDr. Petr Pexa

Duben 2011

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
P e d a g o g i c k á f a k u l t a
Akademický rok: 2009/2010

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	Karel PEKA
Osobní číslo:	P07246
Studijní program:	B1802 Aplikovaná informatika
Studijní obor:	Výpočetní technika
Název tématu:	Tvorba internetových aplikací pomocí Rich Internet Application Adobe Flex
Zadávající katedra:	Katedra informatiky

Z á s a d y p r o v y p r a c o v á n í :

V bakalářské práci bude komplexně zpracována perspektivní problematika tvorby webových interaktivních aplikací v RIA Adobe Flex z pohledu webmastera-profesionála, bude provedeno seznámení s jeho funkcemi a možnostmi a porovnání s obdobnými moderními webovými technologiemi (Flash, AJAX). Součástí práce bude řada ukázkových příkladů a také aplikace většího rozsahu a otestována její podpora v aktuálních verzích prohlížečů Firefox, Opera, Explorer, Chrome a Safari.

Rozsah grafických prací:

Rozsah pracovní zprávy: 60

Forma zpracování bakalářské práce: tištěná

Seznam odborné literatury: viz příloha

Vedoucí bakalářské práce: PaedDr. Petr Pexa

Katedra informatiky

Datum zadání bakalářské práce: 7. dubna 2010

Termín odevzdání bakalářské práce: 30. dubna 2011

doc. PhDr. Alena Hošpesová, Ph.D.
děkanka

L.S.

PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 7. dubna 2010

Příloha zadání bakalářské práce

Seznam odborné literatury:

1. Adobe Systems Incorporated. Adobe Flex [online]. 2003, 2010 [cit. 2010- 03-22]. Dostupné z WWW: <http://flex.org/>.

2. Avoka Technologies. Avoka [online]. 2009 [cit. 2010-03-22]. Dostupné z WWW: <http://www.avoka.com/ad_flex/flex_home_pradeep.shtml>.

3. Developer Shed. Tutorialized [online]. 2003, 2010-01-05 [cit. 2010-03- 22]. Dostupné z WWW: <http://www.tutorialized.com/tutorials/Adobe-Flex/1>.

4. Developer Shed. Dev Shed™ [online]. 2003, 2010-01-05 [cit. 2010-03- 22]. Dostupné z WWW: <http://www.devshed.com/c/a/PHP/Rich-Internet-Applications-Introduction-to-Adobe-Flex-and-PHP/>.

5. FREITAG, Pete. Simple Flex Tutorial [online]. November 07, 2005, December 21, 2007 [cit. 2010-03-22]. Dostupné z WWW: <http://www.petefreitag.com/item/490.cfm>.

6. KIRKPATRIK, Andrew. Accessible Rich Internet Applications with Flash, Flex, and AIR. [s.l.] : [s.n.], September 18, 2009. 33 s.

7. KOENIG, Kai. Rich Internet Applications with Adobe Flex and Java [s.l.] : [s.n.], 21/02/2007. 21s.

8. LI, Bryan. FlexTutorial [online]. February 24, 2009 [cit. 2010-03-22]. Dostupné z WWW: <http://flextutorial.org/>.

Prohlášení

Prohlašuji, že jsem předloženou bakalářskou práci vypracoval samostatně, s použitím citovaných literárních pramenů. Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne 26. dubna 2011

Karel Peka

Anotace

Bakalářská práce se zaměří na vysvětlení funkcí a možností tvorby interaktivních aplikací v RIA Adobe Flex a taktéž na srovnání s obdobnými webovými technologiemi, jako například AJAX, Microsoft Silverlight nebo Adobe Flash. Vysvětlí rozdíl mezi „obyčejnými“ weby a Rich Internet Application (RIA) a tento rozdíl ukáže na sérii ukázkových příkladů zpracovaných v programu Adobe Flash Builder (prostředí pro tvorbu Flex aplikací). Dále bude zpracována aplikace většího rozsahu pro komplexní porozumnění možností a vlastností jednotlivých částí, zároveň též kvůli otestování podpory v aktuálních verzích prohlížečů Firefox, Opera a Chrome. Výsledky tohoto testování budou také součástí práce.

Anotation

Bachelor work focuses on explaining the functions and development of interactive applications in Adobe Flex RIA also compared to similar web technologies such as AJAX, Microsoft Silverlight or Adobe Flash. Explain the difference between "ordinary" sites and Rich Internet Application (RIA) and the difference shows a series of demonstration examples were processed in Adobe Flash Builder (environment for building Flex applications). Also will be created large-scale application for comprehensive understanding of the features and capabilities of individual components, while also testing the support for current versions of Firefox, Opera and Chrome. The results of this testing will also be part of the job.

Obsah

1 ÚVOD....10

2 DEFINICE RIA....11

2.1 Co je a co není RIA....11
2.2 Vlastnosti které by měly RIA aplikace splňovat podle Jeremy Allairea....11
2.3 Vlastnosti RIA aplikací požadované uživateli....12

3 ROZDÍLY MEZI „KLASICKÝM“ WEBEM A RIA APLIKACÍ....14

3.1 Statické webové dokumenty....14
3.2 Dynamické webové dokumenty....14
3.3 Webové aplikace....14

4 ROZSAH RIA APLIKACÍ....15

5 VÝHODY A NEVÝHODY RIA APLIKACÍ....15

5.1 Interaktivita a bohatost....15
5.2 Méně čekání....15
5.3 Omezení bezpečnosti....15
5.4 Optimalizace SEO....16
5.5 Omezené ovládání....16
5.6 Instalace software....16

6 TECHNOLOGIE ADOBE FLEX....17

6.1 Instalace....17
6.2 Flex aplikace....17
6.3 Dva jazyky....18
6.4 Kompilace....18

PŘEDSTAVENÍ KONKURENČNÍCH TECHNOLOGIÍ ZABÝVAJÍCÍCH SE RIA....20

7 MICROSOFT SILVERLIGHT....20

7.1 Instalace....20
7.2 Jazyk....20
7.3 Silverlight architektura....21
7.4 Silverlight aplikace....22

8 OPENLASZLO....24

8.1 Instalace....24
8.2 Jazyk....24
8.3 Architektura....25
8.4 OpenLaszlo klient....25
8.5 OpenLaszlo server....25
8.6 Aplikace....26
8.7 OpenLaszlo bezpečnostní model....26

9 AJAX - ASYNCHRONOUS JAVASCRIPT AND XML....27

9.1 Instalace....27
9.2 Jazyk....27
9.3 DOM (document object model)....27
9.4 Aplikace....28

10 CURL....29

10.1 Instalace....29

10.2 Jazyk....29
10.3 Aplikace....30
10.4 Curl RIA technologie pro podniky....30

11 JAVAFX....31

11.1 Instalace....31
11.2 Jazyk....32
11.3 Aplikace....32

12 ADOBE FLASH....33

12.1 Instalace....33
12.2 Jazyk....33
12.3 Aplikace....33
12.4 Zaměření....34

13 POROVNÁNÍ RIA TECHNOLOGIÍ....35

13.1 Instalace běhového prostředí....35
13.2 Multiplatformovost....36
13.3 Možnosti technologie....37
13.4 Vývojové prostředí....38
13.5 Jazyk/y technologie....39
13.6 Kalkulace....41
13.7 Zhodnocení porovnávání....41

14 ADOBE FLASH BUILDER 4....42

14.1 FLASH BUILDER – POPIS ROZHRANÍ....42

14.1.1 Projekt....42
14.1.2 Vzhled - Kód....42
14.1.3 Vlastnosti....42
14.1.4 Stavy....43
14.1.5 Chybová hlášení....43
14.1.6 Třídy....43
14.1.7 Komponenty....43
14.1.8 Debugování....43

15 TVORBA APLIKACE FLEX....45

15.1 Založení projektu – ukázkový příklad Dialogové okno....45
15.2 Komponenty....46
15.3 Aplikační logika....46
15.4 Využití Designeru k tvorbě aplikace – ukázkový příklad Záložky....48
15.5 Vkládání obrázku do aplikace....50
15.6 OnClick event....51
15.7 Odstranění chyb....52
15.8 Datové služby – ukázkový příklad DataGrid....53
15.9 Odkazování na webové stránky....54
15.10 Omezení datových služeb....55
15.11 Ukázka cross-domain souboru....55
15.12 Efekty – ukázkový příklad Effect....56
15.13 Simple Motion Path....56
15.14 Parallel....57
15.15 Stavy a Radial Gradient....58
15.16 Plynulý přechod....59
15.17 Efekty zjednodušení....59
15.18 Vzhled kurzoru – ukázkový příklad Cursor....60
15.19 Vlastní vzhled....61
15.20 Drag & Drop – ukázkový příklad drag_drop....62
15.21 Posluchače dragEnter a dragDrop....64

15.22 Skinování - ukázkový příklad Styl....65
15.23 Vzhled specifikovaný CSS kódem....68
15.24 Themes....69
15.25 Práce se soubory – ukázkový příklad Files....69
15.26 Načítání....70
15.27 Ukládání....71
15.28 Zhodnocení práce se soubory....73
15.29 Item Renderer – ukázkový příklad Galerie....73

16 APLIKACE VĚTŠÍHO ROZSAHU – APLIKACE BRUSH....75

16.1 Načtení souboru....75
16.2 Otáčení a Roztahování....76
16.3 Ořez....77
16.4 Změna barevných hodnot a jasu....78
16.5 Filtry – Černobílý, Sépiový, Kontrast....78
16.6 Ukládání obrázku....79

17 TESTOVÁNÍ PODPORY PROHLÍŽEČŮ PRO APLIKACE FLEX....81

17.1 Testované prohlížeče....81
17.2 Metoda testování....82
17.3 Výsledky testování....82
17.4 Zhodnocení testu....83

18 ZÁVĚR....84

19 SEZNAM POUŽITÉ LITERATURY A ZDROJŮ....85

20 SEZNAM PŘÍLOH....88

1 Úvod

V dnešní době web neslouží pouze k prohlížení stránek a zjišťování informací. Proto je v mnoha odvětvích třeba vytvářet internetové stránky interaktivní, neboli reagující na uživatelovy aktuální podněty. Ve své práci se budu věnovat právě takovýmto webům, které se chovají obdobně jako desktopové aplikace a nabízí uživateli často intuitivní cesty práce s nimi.

RIA (Rich Internet Application) jsou webové aplikace, které se snaží překlenout rozdíly mezi klasickou webovou aplikací a desktopovou aplikací. RIA aplikace se snaží v rámci webového prohlížeče napodobovat desktopové aplikace svým vzhledem i chováním a poskytnout vyšší uživatelský komfort. [1]

Zlepšující se hardware našich počítačů umožňuje vytvářet graficky kvalitnější a uživatelsky příjemnější webové stránky, pročež se budu věnovat bohatému prostředí internetových aplikací (rich internet application). Vysvětlím, co termín znamená, jak se aplikace liší od „obyčejného“ webu a také jak ji můžeme vytvořit pomocí technologie Adobe Flex. Tvorbu aplikací přiblížím pomocí série ukázkových příkladů, a využitelnost Flex technologie předvedu na větší aplikaci. Technologii RIA se v dnešní době věnuje několik společností, jejich produkty jsou například Adobe Flex, Microsoft Silverlight, OpenLaszlo, Curl, JavaFX či Adobe Flash. Proto také provedu srovnání možností a funkcí těchto technologií. Se stále se zlepšujícími a měnícími weby přichází otázka: Jsou Prohlížeče vlastně schopné tyto nové technologie zobrazit? A pokud ano, budou zobrazeny, interpretovány a poskytnuty uživateli korektně? Na tyto otázky se pokusím odpovědět testem a srovnáním majoritních prohlížeču z hlediska jejich připravenosti na nové technologie bohatého prostředí.

2 Definice RIA

První výskyt pojmu RIA (rich internet application) můžeme nalézt v dokumentu popisujícím produkt Flash MX od firmy Macromedia (nyní je vlastníkem Adobe). Tento dokument napsal Jeremy Allaire v roce 2002. Výraz RIA použil ve smyslu bohaté aplikace oproti stávajícím webovým aplikacím. Neboli aplikace která má některé vlastnosti desktopových aplikací. Tyto vlastnosti jsou nejčastěji chápány ve smyslu grafického uživatelského rozhraní aplikací a způsobu jejich ovládání. Touto definicí tedy můžeme říci, že desktopové aplikace jsou „bohaté aplikace“. Tento termín vyústil z vazby mezi internetovými a desktopovými aplikacemi, kde ty internetové působí chudým dojmem, co se týče nabízených funkcí. Cílem bohatých internetových aplikací je tedy nabízet stejné možnosti práce jako má uživatel k dispozici u desktopových aplikací.

2.1 Co je a co není RIA

V dnešní době již několik RIA technologií nabízí možnost pracovat s aplikací mimo prohlížeč, RIA aplikace tedy nutně nemusí znamenat „interaktivní webová stránka“. Jak tedy poznat „obyčejnou“ desktopovou aplikaci od té Internetové? V zásadě můžeme říci, že RIA aplikace je taková, která má svůj smysl díky Internetu. Srozumitelněji řečeno, Word sice může pres Internet hledat nápovědu, či nové šablony, ale i bez internetu má své využití a smysl, proto není RIA aplikací. Jako příklad RIA aplikace mimo prohlížeč můžeme použít eBay Desktop, který sice může krátkodobě pracovat offline, avšak bez Internetu ztrácí svůj smysl. [2]

2.2 Vlastnosti které by měly RIA aplikace splňovat podle Jeremy Allairea

Jeremy Allaire ve svém dokumentu pojmenovaném „Rich clients and Rich Internet applications“ rozdělil nutné vlastnosti RIA aplikací do sedmi bodů. Zároveň také dokazuje, že všechny vlastnosti Flash MX splňuje.

2.2.1 Uveďme si tedy těchto sedm vlastností:

Poskytovat účinné, vysoce výkonné běhové prostředí pro spouštění kódu, obsahu a komunikace.

Zde je zdůrazněna potřeba dynamického generování obsahu. Dále prvek komunikace,

který je pro spojení serveru a klientu nutný.

Integrovat obsah, komunikaci a aplikační rozhraní do obecného prostředí.

V klasickém HTML se pro zpracování kódu používá samotný WWW prohlížeč, ale pro bohatý obsah se používají různé zásuvné moduly. Dle autora je nutné obsah integrovat do jednoho prostředí.

Nabízet výkonný a rozšiřitelný objektový model pro interaktivitu.

Přestože webové prohlížeče pokročily, ve smyslu podpory, prostřednictvím JavaScriptu atd., stále jim chybí bohatost potřebná pro budování seriozních aplikací

Umožňovat rapidní vývoj aplikací pomocí komponent a znovupoužití

RIA klienti by měli podporovat řízený vývoj, umožnující snadné znovupoužití vizuálních komponent. Čímž by se i nezkušenějším vývojářům poskytl přístup ke komplexnějším funkcím.

Umožňovat použití webových a datových služeb nabízených aplikačním serverem.

RIA klienti by měli poskytovat možnost čistě oddělit prezentační logiku a uživatelské rozhraní od aplikační logiky hostované na síti. Také by měli sloužit jako model pro snadné ovládání těchto vzdálených služeb.

Podporovat online i offline klienty.

RIA aplikace musí umožňovat obě možnosti, hlavně z důvodu používání PDA a laptopů, kde uživatel nemusí mít zajištěné trvalé spojení se serverovou částí aplikace

Umožňovat jednoduché zavádění aplikací na různých platformách a zařízeních.

RIA aplikace musí „běžet“ na různých platformách a zařízeních. Tato vlastnost je asi nejdůležitější a dává velikou výhodu RIA aplikacím oproti aplikacím desktopovým. [3]

2.3 Vlastnosti RIA aplikací požadované uživateli

2.3.1 Možnosti uživatelského rozhraní

Uživatel od aplikace očekává možnost pracovat s ní podobně jako s desktopovou, je tedy téměř nutností rychlá odezva, možnost ovládání aplikace zaběhlými způsoby, jako jsou drag&drop, klávesové zkratky apod.

2.3.2 Spouštění

Mnoho uživatelů může odradit nutnost instalace zásuvných modulů, běhových prostředí či přehrávačů před tím, než mohou spatřit obsah aplikace. Z uživatelského hlediska je tedy výhodou minimalizace nebo úplná absence těchto instalací. Tento bod souvisí také s multi-platformovostí, v případě, že je instalace software rychlá či žádná, ulehčuje to přístup k aplikaci z více zařízení a operačních systémů.

3 Rozdíly mezi „klasickým“ webem a RIA aplikací

Weby můžeme rozdělit na tři druhy, zaprvé statické webové dokumenty, zadruhé dynamické webové dokumenty a zatřetí webové aplikace.

3.1 Statické webové dokumenty

Jsou napsány ve formátu HTML a jejich obsah není žádným způsobem interaktivní, po uživatelově odeslání požadavku (kliknutí na odkaz atd.) je ze serveru zaslán kód cílené stránky, kterou prohlížeč vykreslí. Klientská část v tomto případě nepotřebuje žádnou aplikační logiku, prohlížeč stránku z kódu pouze vygeneruje.

3.2 Dynamické webové dokumenty

Již poskytují interaktivitu se serverem. Jejich tvorba začala být možná díky implementaci formulářů do formátu HTML2.0. Na první pohled uživatel tento rozdíl nemusí poznat, protože pro něj je stránka vlastně stále stejný typ dokumentu. Avšak po odeslání uživatelova požadavku se na serveru nenačte dokument z datového úložiště, jako tomu bylo u webových dokumentů. Na straně serveru se z uživatelových požadavků dokument nejprve poskládá a teprve potom se uživateli pošle. Mluvíme tedy o „Server-side Scripting“. V tomto případě je možné, aby se různým uživatelům zobrazil různý obsah. Když se nad výše zmíněným rozdílem zamyslíme, zjistíme, že stále nejde o novou technologii, pouze byla přidána logika na straně serveru, avšak komunikace typu Request-Reply mezi klientem a serverem jse stále stejná, posíla se celý dokument.

3.3 Webové aplikace

Jako počátek webových aplikací můžeme označit přechod skriptů ze serveru ke klientovi, tedy Client-side Scripting. Takovýto přechod byl umožněn pomocí technologie JavaScript či JScript. Přestože se část aplikační logiky dá takto převést ke klientovi, nelze převést celá. Z dané webové aplikace by se rázem stala aplikace kompletně na straně klienta. U takovýchto aplikací již potřebujeme na straně klienta přidaný software, ať to je, již implementovaný JScript, či Java Runtime Environment (JRE), či jiné běhové prostředí. [4]

4 Rozsah RIA aplikací

Z kapitoly „Vlastnosti které by měly RIA aplikace splňovat“ můžeme použít 7 bodů, které Jeremy Allaire uvedl jako vlastnosti RIA aplikací, a pomocí nich si definovat rozsah který pokrývají RIA aplikace. Můžeme říci, že statický webový dokument nemůže být RIA aplikace z důvodů absence komunikace (s vyjímkou základních požadavků na překreslení stránky) a nulového přínosu bohatých vlastností. RIA aplikace taktéž nemohou existovat bez serverové části a vazby na Internet. RIA aplikace se tedy nachází v pomyslném intervalu z jedné strany omezeném statickými webovými dokumenty a z druhé strany desktopovými aplikacemi (bez silné vazby na internet, viz kapitola: *Co je a co není RIA*).

5 Výhody a nevýhody RIA aplikací

5.1 Interaktivita a bohatost

Díky přenesení skriptů ke klientovi a jejich interpretaci zásuvným modulem v prohlížeči nebo samostaným běhovým prostředím se nabízí větší interaktivita a bohatost pro uživatele. RIA aplikace mohou poskytovat bohatší zvukové i grafické efekty, zároveň mohou uchovávat polohu a stav uživatelovy práce s aplikací.

5.2 Méně čekání

Tento typ aplikací také zkracuje čekání na „znovu-načtení“ stránky, aplikace totiž může od serverové strany vyžadovat konkretní data místo celé stránky. Tato data se zpracují na straně klienta a jako u desktopových aplikací, není potřeba překreslovat celou stránku, změní se pouze zpracovaná data od serveru. Z toho jasně vyplývá, že komunikace mezi klientem a serverem nebude tolik náročná a tudíž nejen urychlí načtení stránky, ale i odstraní zátěž ze serveru, který tedy ve stejném čase může zvládnout více požadavků.

5.3 Omezení bezpečnosti

Abychom nemluvili pouze o výhodách, RIA technologie má také své nevýhody. První z nich je problém bezpečnosti, velká část aplikace je interpretována u uživatele, což přináší své riziko. Avšak i s tímto problémem vývojáři některých RIA technologií

pracují, snaží se ho řešit pomocí zabezpečovacích protokolů a co nejčastěji vydávaných bezpečnostních záplat (security patches).

5.4 Optimalizace SEO

Další chybou RIA technologie je bezesporu špatná optimalizace pro vyhledávače (SEO). Dnes je stále většina vyhledávacích strojů založená na prohledávání textu, tudíž mají problém s indexováním kompilovaného souboru (jako například .swf). Na odstranění těchto problémů například společnost Adobe vytvořila aplikaci swf2html, která extrahuje odkazy a text z .swf souborů a data vrací např. Ve formě html dokumentu. [5]

5.5 Omezené ovládání

Jelikož je aplikace interpretována svým pluginem nebo běhovým prostředím, mívá prohlížeč problémy se získáním stavu aplikace. Například URL adresa zůstává stále stejná bez ohledu na změny obsahu, což způsobuje problémy funkcím jako Bookmark, Back, Page history atd.. I s tímto problémem se chtějí vývojáři RIA technologií „poprat“, spolupracují se světovými firmami (Google, Yahoo) a snaží se zlepšit výsledky vyhledávání. [6]

5.6 Instalace software

K většině RIA technologií je potřeba před spuštěním aplikace instalovat zásuvný modul do prohlížeče, běhové prostředí nebo přehrávač. Vyjímkou je AJAX, který využívá JavaScript, jež je podporován všemi majoritními prohlížeči. Jako nevýhoda také nemusí být brán Adobe Flash a Flex, a OpenLaszlo, neboť Flash player je mezi klienty velmi rozšířený a další požadavky na klienta nejsou. Například u technologie JavaFX je potřeba nejprve instalovat běhové prostředí Java a teprve poté běhové prostředí pro JavaFX a jakkoliv je instalační soubor malý, může uživatele od aplikace odradit.

6 Technologie Adobe Flex

Vlastník:	Adobe Systems
Licence:	open-source
Běhové prostředí:	Adobe Flash Player
Jazyk:	MXML, ActionScript
Platforma :	všechny s podporou Flash Playeru
Vývojářské prostředí:	Adobe Flash Builder

6.1 Instalace

Adobe Flex je vysoce produktivní open-source framework pro vytváření webových aplikací podporovaných všemi majoritními prohlížeči a operačními systémy. Využívá k tomu běhové prostředí Adobe Flash Player nyní ve verzi 10.2.153.1, o velikosti 2,70 MB(pro Windows), a Adobe AIR ve verzi 2.6, o velikosti 12.20 MB(pro Windows). Aplikace lze vytvářet v libovolném textovém editoru a s pomocí Flex SDK ve verzi 4, o velikosti 171 MB(pro všechny platformy), následně zkompilovat. Adobe nabízí placený software Flash Builder 4 o velikosti 388.2 MB pro Windows (s možností zkušební verze nebo studentské licence), který může pomoci zrychlit a usnadnit tvorbu aplikací díky možnostem jako například inteligentní kódování, interaktivní krokování při debugování, vzhledem uživatelského prostředí a integrovanou kompilací. Díky open-source projektu je možné nainstalovat plug-in do open-source vývojového prostředí Eclipse.[7]

6.2 Flex aplikace

Framework Adobe Flex vychází z technologie Adobe Flash a je přímo zaměřen na vytváření větších a komplexnějších RIA aplikací. Pomocí technologie Adobe Flash se samozřejmě dají RIA aplikace také tvořit, ale je to obtížnější a složitější. Technologie Adobe Flex taktéž poskytuje možnost kompilace aplikací do formátu .swf, tedy umožňuje kompilaci do stejného formátu jako Adobe Flash a následně využívá stejné běhové prostředí. Viz. obrázek 1. Adobe Flex si z technologie Adobe Flash bere také ActionScript. Tím však končí společné vlastnosti.

Zdrojové kódy aplikace se ukládají jako textové soubory a tudíž je můžeme na rozdíl

od souborů Adobe Flash (binární .fla) vytvářet či měnit i bez vývojového prostředí. Adobe Flex je zaměřený pro programátory, vývojář skládá uživatelské rozhraní ze znovu-použitelných komponent a má k dispozici oběktově orientovaný jazyk. [8]

Z open-source projektu je všechno dostupné v Adobe Flex Software Development Kit neboli Adobe Flex SDK. Součástí SDK je kompilátor, knihovna komponent a debugger. Avšak vřele doporučuji využít placeného vývojového prostředí Adobe Flash Builder, který usnadní tvorbu aplikací.[4] Knihovna komponent obsahuje řadu komponent v různých kategoriích, jako například ovládací komponenty, komponenty layoutu, navigační komponenty, formátovací komponenty, grafické efekty, grafické komponenty, validátory atd.[9]

6.3 Dva jazyky

Adobe Flex využívá ActionScriptu a MXML značkovacího jazyka. Ve zdrojovém souboru aplikace je tedy uživatelské rozhraní definováno v jazyce MXML, procedurální kód aplikace je pak v jazyce ActionScript, v souboru ho můžeme nalézt mezi tagy <script>.

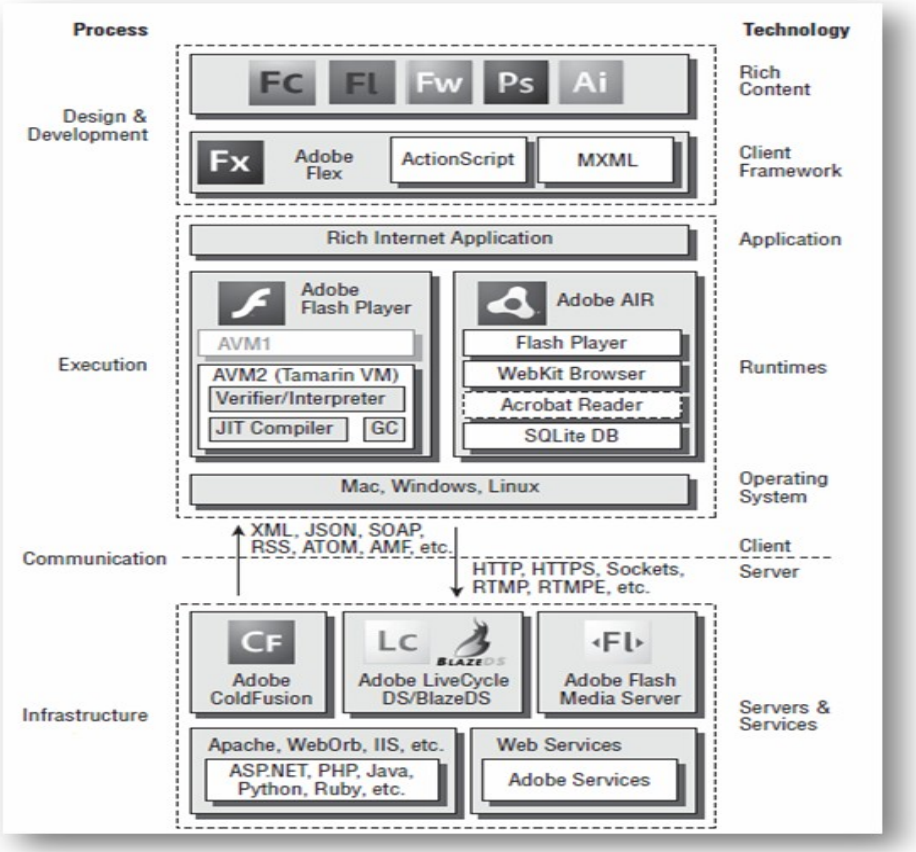
Jazyk MXML má silnou sémantiku pro tvorbu uživatelských rozhraní, používá elementy jako jsou Label, Button, Canvas, Image atd. Což ulehčuje tvorbu uživatelského rozhraní. Zároveň Flex podporuje podmnožinu CSS, takže můžeme styly definovat přímo v jednotlivých elementech nebo k projektu přiřadit samostatný soubor se styly (.css). Při kompilaci se komponenty v jazyce MXML převádí do ActionScriptu, kde všechny mají své ekvivalentní definice, a teprve poté se aplikace kompiluje do formátu .swf

6.4 Kompilace

Flex aplikace můžeme kompilovat několika způsoby, buďto využijeme command-line kompilátor přidaný k Flex SDK, nebo kompilátor ve vývojovém prostředí Adobe Flash Builder. Nebo můžeme použít například kompilátor implementovaný v Javě. Uživatel může kompilovat své komponenty do formátu .swc, při kompilaci aplikace je výstupním formátem výše zmíněný .swf a kompilátor vytvoří také .html formát ve kterém jsou soubory .swf umístěné. Stejně tak jako u Adobe Flash může aplikace ve formátu .swf obsahovat obrázky, hudbu, video a další, nebo mohou být přiřazeny externě.

Kompilací vzniká několik souborů, které jsou poté nutné ke spuštění aplikace.

Hlavním souborem je tedy „jmeno_aplikace.swf“, jako další zde je soubor se styly „jmeno_aplikace.css“, pak soubor html „jmeno_aplikace.html“, který je generován aby se postaral o načtení swf souboru. Dalším souborem našeho projektu je „playerProductInstall.swf“ který, pokud je to potřeba, zařizuje aktualizaci Flash Playeru. Flex aplikace také obsahuje složku s názvem „history“, která zajišťuje funkčnost tlačítka Zpět a zaznamenává historii aplikace, pro korektnost jeho používání. [8] Závěrem je zde několik dalších souborů, které jsou nutné pro běh aplikace na webu, avšak mezi aplikacemi mohou být sdíleny, to jsou soubory „framework, osmf_flex, rpc_*verseFlexu*, spark_*verseFlexu*, sparkskins_*verseFlexu*, textLayout“ a pokud jsme využili nějaké služby tak se vygeneruje i složka „services“.



Obrázek 1[26]

Představení konkurenčních technologií zabývajících se RIA

7 Microsoft Silverlight

Vlastník:	Microsoft
Licence:	open-source
Běhové prostředí:	Silverlight plug-in
Jazyk:	XAML, podmnožina .NET frameworku
Platforma(Silverlight) :	MS Windows, Mac OS X, Windows Phone 7, Symbian
Platforma(Moonlight):	Linux, FreeBSD
Vývojářské prostředí:	MS Visual Studio

7.1 Instalace

Pro spuštění Silverlight aplikací je nutné stáhnout plug-in o velikosti 6,0MB. Pro tvorbu aplikací budeme potřebovat Silverlight SDK, nyní ve verzi 4 o velikosti 13.1MB. Silverlight SDK obsahuje online dokumentaci, online vzorky aplikací, knihovny a nástroje pro tvorbu Silverlight aplikací. Také lze stáhnout plug-iny pro tvorbu aplikací přímo pro vývojové prostředí Visual Studio nebo Eclipse.

7.2 Jazyk

V aplikacích Silverlight se k deklaraci uživatelského rozhraní používá značkovacího jazyka XAML (Extensible Application Markup Language), a programovanou část aplikace je možné tvořit v jazycích C# a VisualBasic, mimo to díky Dynamic Language Runtime (DLR) i v jazycích Ruby a Python a dalších.

7.3 Silverlight architektura

Silverlight umožňuje dva modely pro tvorbu aplikací:

7.3.1 The managed API for Silverlight

Aplikace používající managed API mají přístup k Silverlight .NET frameworku, tudíž mohou být napsány v ve výše zmíněných jazycích.

.NET Framework for Silverlight

Je podmnožinou .NET frameworku která obsahuje komponenty a knihovny, včetně integrace dat, rozšiřitelných prvků pro ovládání Windows, práce se sítí, knihovny základních tříd a běhové prostředí běžných jazyků.

7.3.2 The JavaScript API for Silverlight

Aplikace používající JavaScript API mají přístup pouze k Silverlight presentation core a JavaScriptovému engine v prohlížeči.

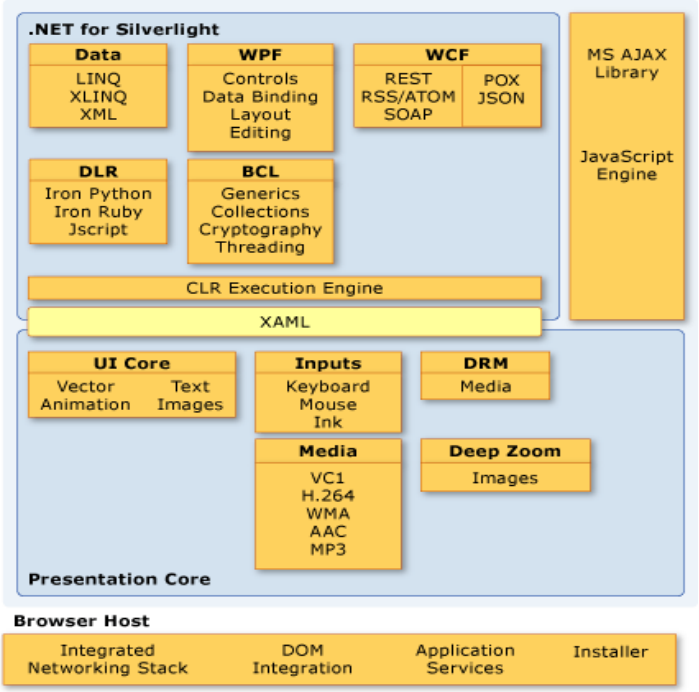
Core presentation framework

Obsahuje komponenty a služby orientované na uživatelské rozhraní a interakci, včetně uživatelských vstupů, dále ovládací prvky uživatelského rozhraní pro použití ve web aplikacích, přehrávání médií, správu digitálních práv, data binding a prezentační funkce, včetně vektorové grafiky, textů, animací a obrázků. Také zahrnuje Extensible Application Markup Language (XAML) pro určení rozložení aplikace.

V obou těchto API lze použít značkovacího jazyka XAML k tvorbě uživatelského rozhraní, avšak managed API umožňuje využití více než jednoho XAML souboru.
Silverlight také poskytuje několik funkcí pro ulehčení programování. Poskytuje bezpečný přístup z Silverlight klienta k lokálnímu souborovému systému. Povoluje lokální ukládání a načítaní dat. Asynchronní programování, souborový manažer. Umožňuje programátorům v .NET přímo manipulovat s elementy uživatelského rozhraní webové stránky. Viz obrázek 2. Weboví vývojáři mohou také využít JavaScript k přímému volání kódu a přístupu ke skriptovatelným objektům, vlastnostem, událostem a metodám. Poskytuje podporu pro serializaci typů CLR na JSON a XML. Poskytuje třídu Application a nástroje pro tvorbu .xap balíčků. Balíčky .xap obsahují aplikační a vstupní body pro běh Silverlight pluginů. Třídy XmlReader a XmlWriter usnadňují práci s xml daty z webových služeb. Funkce Xlinq umožňuje vývojářům

žádat data přímo v programovacích jazycích .NET frameworku.

Silverlight SDK také obsahuje instaler a updater, které usnadňují první instalace aplikací a následně řeší automatické aktualizace. [10]



Obrázek 2 [27]

7.4 Silverlight aplikace

Protože běhové prostředí Silverlight se instaluje jako plug-in do prohlížeče, jsou Silverlight aplikace závislé na prohlížečích, což sebou přináší omezení bezpečnostního „sandbox“ prohlížeče. S verzí Silverlight 4 byla přidána nová funkce out-of-browser, která se do jisté míry s těmito omezeními vyrovnává, umožňuje aplikaci stáhnout a pracovat s ní i v offline režimu. Aplikace pak sama zjišťuje zda-li je uživatel online a synchronizuje se se serverem, může také přistupovat na bezpečné uložiště v počítači stanovené uživatelem. Dalším omezením je podpora pouze některých prohlížečů, svšak s novějšími verzemi Silverlightu přibývají zásuvné moduly pro další prohlížeče a operační systémy.

Silverlight aplikace je zkompilována do zip formátu s příponou .xap, tento formát obsahuje soubor s aplikací zkompilovanou do formátu .dll, dále obsahuje knihovny pro využité komponenty (např. System.Windows.Controls.dll) a také soubor se značkovacím jazykem .xaml, závěrem jsou přidány zdroje (např. Obrázky). Další části

aplikace lze dynamicky stahovat ze serveru. Takto zkomprimovaná aplikace se vkládá do HTML dokumentu podobně jako aplikace Flash. Aplikace však může díky JavaScriptu a DOM HTML měnit parametry a rozložení WWW stránky a naopak WWW stránka může aplikaci předávat parametry a řídit její provádění.

8 OpenLaszlo

Vlastník:	Laszlo Systems
Licence:	open-source
Běhové prostředí:	Flash Player, prohlížeč podporující JavaScript
Jazyk:	LZX, JavaScript
Platforma:	všechny s podporou Flash Playeru, nebo JavaScriptu
Vývojářské prostředí:	OpenLaszlo DevKit (nutno taky mít Java SDK)

8.1 Instalace

Pro spuštění aplikací stačí mít nainstalován Adobe Flash Player, nebo pouze prohlížeč s podporou JavaScriptu, který podporují všechny majoritní prohlížeče. Pro tvorbu aplikací je třeba stáhnout OpenLaszlo Development Kits ve verzi 4.9.0 o velikosti 56 MB (pro Windows), avšak tento software nám nebude stačit, k jeho správnému běhu musíme mít také Java SDK, například ve versi 6 od firmy Oracle (společnost, která koupila Sun Microsystems). K dispozici pro pohodlnější tvorbu aplikací jsou také zásuvne moduly do vývojových prostředí NetBeans a Eclipse. Pokud chceme aplikace OpenLaszlo kompilovat až podle požadavků klienta, je možné stáhnout také OpenLaszlo servlet package, který se bude o serverovou kompilaci starat.

8.2 Jazyk

Aplikace OpenLaszlo jsou psány v jazyce JavaScript a značkovacím jazyku LZX. Jazyk LZX kombinuje deklarativní XML syntaxi a standartní CSS styly k popisu vzhledu, rozvržení a hierarchie komponent obsažených v aplikaci. JavaScript je použit k specifikování chování aplikace.

8.3 Architektura

Platforma OpenLaszlo se skládá z kompileru, volitelného server-side servletu, souboru služeb a tříd na straně klienta, které jsou dostupné pro všechny LZX aplikace. Jádro na straně klienta se jmenuje LFC (Laszlo Foundation Clases).

8.4 OpenLaszlo klient

LFC obsahuje věci jako vizualizace renderování aplikace, systém událostí, manažer dat, časovač, focus manažer, atd. Event system (systém událostí) rozpoznává a řeší události jako jsou uživatelská kliknutí na myš, či příchozí data ze serveru. OpenLaszlo snižuje zátěž na straně serveru zpracováním úloh jako jsou řazení, validace atd. na straně klienta.

Data Loader/Binder (manažer dat) slouží jako řízení provozu, přijímá toky dat z OpenLaszlo serveru a přiřazuje data k daným vizuálním elementům (text, pole, formuláře a položky menu). [11] Layout and Animation System (systém dispozice a animací) poskytuje změny stavů OpenLaszlo aplikacím, rozložení prvků rozhraní a algoritmy řízených animací. Dynamic layout mechanisms (mechanizmy dynamického rozložení) umožňuje modifikace vlastností jako je velikost aplikace, aby se inteligentně přizpůsobila platformě. Což zjednodušuje práci na zařízeních různých velikostí. Podobně upřednostňuje u animací definice časových indexů místo snímků, čímž vyřadí rozdíly rychlostí procesorů klientských zařízení. Services component (komponenta služeb) poskytuje řadu podpůrných možností, včetně časovačů, modálních dialogů, atd.

8.5 OpenLaszlo server

Serverová část OpenLaszlo Server se nemusí používat, avšak nabízí další funkce a především větší kompatibilitu aplikací. Může překódovat data z formátu do jiného přímo za běhu, deleguje požadavky pro vzdálené části serveru, umožňuje přístup k SOAP (Simple Object Access Protocol) a XML-RPC (Remote Procedure Call) služeb a mapuje OpenLaszlo objekty jako Java objekty. OpenLaszlo Server (dříve Laszlo Presentation Server) je implementován jako Java servlet a běží uvnitř servletového kontejneru nebo standartní J2EE aplikace, díky čemuž může běžet v prostředích operačních systémů Windows, Solaris, Linux a Mac OS X. [11]

8.6 Aplikace

OpenLaszlo dokáže kompilovat .swf aplikace a také DHTML (dynamické HTML). Kompilační technologie OpenLaszlo je zaměřená na kompatibilitu mezi prohlížeči, stará se tedy o jednotlivé rozdíly a nekompatibility mezi nimi a vývojářům umožňuje koncentraci na vzhled a chování aplikace. Aplikace napsané v OpenLaszlo mohou být „solo“ aplikace, tedy zkompilované do formátu .swf nebo DHTML, ve smyslu HTML+JavaScript+AJAX. Nebo mohou být uloženy na serveru ve formátu .LZX a podle klienta distribuovat z cache požadovanou verzi aplikace.[11]

Díky implementaci funkcí různých běhových prostředí se otevírá vývojářům cesta k psaní široce kompatibilních aplikací, avšak nese to s sebou omezení, vývojáři mohou použít pouze funkce, které jsou pro tyto formáty společné. OpenLaszlo aplikace jsou psány tak, aby se přizpůsobily velikostem a rychlostem zařízení, což usnadňuje vývojářům tvorbu dynamických aplikací s minimem programování a umožňuje využít jak relativní tak absolutní pozicování.

8.7 OpenLaszlo bezpečnostní model

Platforma OpenLaszlo aplikací podporuje prověřený bezpečnostní model SSL (Secure Sockets Layer). Data posílaná Internetem mohou být šifrována SSL a posílána skrze HTTPS. Aplikace ve formátu .swf jsou zabezpečeny „sandbox” prostředím Flash Playeru, tudíž nemohou zapisovat do lokálního souborového systému a přistupovat k nativnímu uživatelskému prostředí. Webové služby a databáze mohou být také zabezpečeny použitím uživatelského autentifikačního modelu. Tento mechanizmus je použit proti zneužití OpenLaszlo serveru jako proxy, nebo vstupní brány k chráněným službám či datům. [11]

9 AJAX - Asynchronous JavaScript and XML

Vlastník:	-
Licence:	-
Běhové prostředí:	prohlížeč podporující JavaScript
Jazyk:	XML, JavaScript, CSS, HTML
Platforma:	všechny s podporou JavaScriptu
Vývojářské prostředí:	libovolný textový editor

9.1 Instalace

Technologie AJAX je značně rozšířená, neboť všechny relevantní prohlížeče mají podporu JavaScriptu a proto uživatel nepotřebuje žádné další zásuvné moduly, nebo běhová prostředí. Není tedy třeba nic instalovat, na Internetu se ovšem dá najít množství vývojových prostředí, ať už online (např. Cloud9 IDE [http://www.ajax.org/]) nebo nebo offline textové editory, nebo přímo zaměřené na JavaScript.

9.2 Jazyk

AJAX je spojení již známých technologií, XML, JavaScriptu, DHTML a HTTP. Vzhled a rozvržení aplikace je specifikováno CSS styly. Aplikace je uložená v HTML dokumentu, který obsahuje i JavaScriptový kód. Tento kód můžeme v HTML dokumentu najít mezi tagy <script>. Pomocí JavaScriptu lze posílat mezi klientem a serverem data ve formátu XML a následně je zobrazovat na stránce aniž by se musela celá posílat ze serveru. Stránka se nemusí obnovovat celá díky použití dynamického DOM modelu v HTML dokumentu, což v kombinaci s JavaScriptem a CSS styly nazýváme DHTML (dynamické HTML). [12]

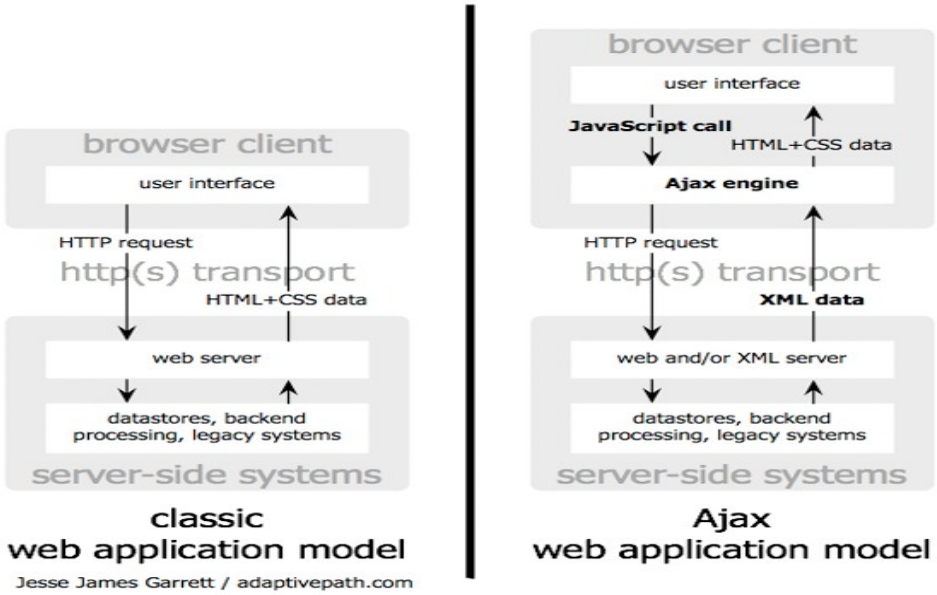
9.3 DOM (document object model)

objektový model dokumentu je aplikační programové rozhraní (API). Pomocí tohoto rozhraní můžeme pracovat se samotnou webovou stránkou. DOM umožňuje JavaScriptu přistupovat k jednotlivým objektům XML (XHTML) dokumentu a pracovat s nimi. Těmito objekty jsou elementy, atributy, text, komentáře atd. DOM umožňuje na přesně určené místo přidat jakýkoli obsah. [13]

9.4 Aplikace

AJAX aplikace může tedy na první pohled vypadat stejně jako HTML dokument. Je zcela jasné, že u této technologie se neprovádí žádná kompilace, neboť vše je, přímo v kódu který napsal tvůrce, interpretováno prohlížečem. Vytvořená aplikace bude určitě obsahovat jeden či několik .html dokumentů, v kterých je také obsažen JavaScript, dále soubor se styly .css. K psaní aplikace také můžeme využít ostatních jazyků slučitelných s HTML(např. PHP).

Aplikace poté komunikuje se serverem pomocí protokolu HTTP a jednotlivé požadavky posílá ve formátu XML. Takovouto komunikaci je možné uskutečnit skrze rozhraní XMLHttpRequest, není však nutné použít pouze XML, požadavky je možné posílat i ve formátu HTML, JSON nebo jako prostý text, viz obrázek 3. Využití protokolu HTTP, který není vhodný pro intenzivní komunikaci klienta se serverem může být nevýhodou AJAXu. Pro každý požadavek se musí vytvořit nové spojení a po vyřízení je spojení automaticky ukončeno, což zvětšuje prodlevy mezi jednotlivými požadavky. Zároveň server nemůže kontaktovat klienta například v případě zasílání nových příspěvků všem přihlášeným uživatelům chatu. Takovéto situace se dají řešit opakujícími výzvami klienta na aktualizaci dat.[12]



Obrázek 3[28]

10 Curl

Vlastník:	Curl inc.
Licence:	zdarma pro nekomerční využití
Běhové prostředí:	Curl RTE
Jazyk:	Curl
Platforma:	běh: Windows, Linux, Mac
	tvorba: Windows, Linux
Vývojářské prostředí:	Curl IDE, Curl/Pro IDE

10.1 Instalace

Technologie Curl je komerční platformou pro podnikové RIA aplikace, avšak Společnost Curl uvolnila pro nekomerční využití vývojové prostředí, Curl IDE ve verzi 7, jeho stažení je však potřebná registrace. Ke komerčnímu využití aplikací je třeba si koupit vývojové prostředí Curl/Pro IDE. Pro spouštění aplikací stačí stáhnout běhové prostředí Curl RTE7 o velikosti 7.8 MB. V případě aktualizování novějším RTE zůstává staré zachováno kvůli zpětné kompatibilitě starších aplikací. Dále lze, také po registraci, stáhnout Curl CDE, neboli zásuvný modul do vývojového prostředí Eclipse.

10.2 Jazyk

Aplikace je psaná v jazyce Curl, který kombinuje deskriptivní a aktivní elementy, které lze najít v tradičních vývojářských nástrojích, s plně funkčním objektově orientovaným programovacím jazykem. Jazyk Curl byl vytvořen za účelem sjednocení více technologií, pro které pak stačí jen jedno běhové prostředí. Jazyk Curl zahrnuje textové a vzhledové funkce (jako HTML a CSS), skriptovací funkce (jako JavaScript), objektově orientovaný jazyk (jako Java) a podporu 2D/3D grafiky a podporu multimédií (jako Flash).

10.3 Aplikace

Curl aplikace může běžet uvnitř HTML dokumentu, nebo uvnitř prohlížeče namísto HTML dokumentu a od verze Curl7 lze nainstalovat přímo do operačního systému a běžet nezávisle na prohlížeči.[14] Po inicializačním požadavku od klienta je ze serveru zaslán kód aplikace a související data v kompaktním souboru s příponou .curl ve formátu prostého textu nebo jako předzpracovaný kód ve formátu PCURL (pre-processed curl). Soubor může být velikostí srovnatelný s aplikacemi na bázi HTML. Od této doby běží aplikace krom zasílání požadovaných dat, na straně klienta. Tím pádem je server, podobně jako u MS Silverlight, Adobe Flex či OpenLaszlo, oproštěn od značného vytížení. Výhodou oproti ostatním technologiím je možnost práce dlouhobobě offline, kdy měněná data jsou ukládána a po opětovném připojení se klient synchronizuje se serverem.

10.4 Curl RIA technologie pro podniky

Jak bylo výše zmíněno, Curl je především technologie pro velké podniky. Tvůrci technologie si určili požadavky podniků na RIA aplikace a zároveň tvrdí, že technologie Curl je schopna je splnit.

Veliké soubory dat, které vyžadují velký výkon na straně klienta
Komplikované operace, které potřebují nadstandartní design uživatelského rozhraní
Vizualizace dat, která vyžaduje velký grafický výkon na straně klienta
Distribuce pro tisíce uživatelů, k čemuž je potřeba proměnlivá výkonnost
Komplexní aplikace, které potřebují výkonné vývojové prostředí
[14]

11 JavaFX

Vlastník:	Oracle Corporation
Licence:	open-source
Běhové prostředí:	JavaFX a Java JRE
Jazyk:	JavaFX Script
Platforma:	Windows, Mac, Linux, Solaris, Android, Windows Mobile
Vývojářské prostředí:	NetBeans

11.1 Instalace

Abychom mohli spustit JavaFX aplikace, musíme nainstalovat běhové prostředí JavaFX, které se ostatně samo nabídne v podobě odkazu, v případě, že nedojde ke spuštění aplikace. Avšak samotné prostředí JavaFX nestačí. Ke spuštění aplikace, v počítači již musíme mít nainstalované běhové prostředí Java (JRE) nyní ve verzi 6 update 24 o velikosti 15.3 MB. Nutnost JRE může být nepříjemná, avšak stažení a instalace prostředí není zdlouhavá.

Společnost Oracle se však neomezuje pouze na stolní počítače, JavaFX je dostupná i v mobilních telefonech, a s verzí 1.3 i v televizorech, a jiných zařízeních. K tvorbě JavaFX aplikací je potřeba stáhnout JavaFX SDK nyní ve verzi 1.3.1, nebo lze použít zásuvný modul do rozšířeného software pro tvorbu v mnoha jazycích, NetBeans také ve verzi 1.3.1. NetBeans IDE for JavaFX i JavaFX SDK jsou dostupné ve verzích pro Windows, Mac, Linux a Solaris. Dále k tvorbě můžeme využít dvou dalších částí, konkrétně JavaFX Mobile for Windows Mobile, ve verzi 1.2, pro tvorbu aplikací pro mobilní telefony a JavaFX Production Suite který obsahuje sadu nástrojů a pluginů usnadňujících kooperaci mezi programátorem a grafikem při tvorbě grafických prvků. [16]

11.2 Jazyk

Aplikace JavaFX jsou psány v objektovém jazyce JavaFX Script. JavaFX Script je staticky typovaný deklarativní skriptovací jazyk. Syntakticky je tento jazyk podobný s JavaScriptem, podporuje automatický data binding, 2D grafiku, swingovské komponenty a deklarativní animace. JavaFX staví na Java JRE (u mobilních zařízení Java ME) díky čemuž mohou vývojáři využívat jakékoliv knihovny z tohoto prostředí. Výhodou jsou možné zkušenosti se zmíněnými prvky a také mnohem bohatší možnosti tvorby aplikací.

11.3 Aplikace

Od Java SE verze 6 aktualizace 10 a vyšší je možné spouštět aplikace nezávisle na prohlížeči. JavaFX aplikace před kompilací obsahuje několik soubory typu .xf. Pro každou třídu naší aplikace jeden takový. Aplikaci můžeme šířit několika způsoby, zaprvé tedy offline. Zadruhé ve formě appletu, nebo přes Java Network Launching Protocol, neboli webstart. V případě využití appletové formy, bude naše aplikace v souboru .jar, a bude se na ní odkazovat z HTML dokumentu, v kterém bude obsažena.

V tomto HTML dokumentu jsou dvakrát použity tagy <script>, poprvé odkazují na soubor, vyskytující se na stránkach javy, který kontroluje, zda-li má klient aktualizovanou verzi JavaFX, a v druhém tagu <script> se již odkazuje na samotnou aplikaci. V případě spouštění aplikace přes webstart vlastně stahujeme soubor .jnlp, který odkazuje na Internetové umístění aplikace a nezávisí svým spuštěním na prohlížeči. Platforma JavaFX obsahuje soubor nástrojů a technologií, které umožňují vývojářům a designérům spolupracovat, vytvářet a šířit aplikace pro majoritní platformy a prohlížeče, aniž by aplikace měnily vzhled nebo svou funkčnost. Navíc vývojáři mohou použít emulátory (Mobile & TV Emulators) pro testování mobilních a televizních aplikací využívajících JavaFX technologii.

12 Adobe Flash

Vlastník:	Adobe Systems
Licence:	open-source
Běhové prostředí:	Adobe Flash Player
Jazyk:	ActionScript
Platforma:	všechny s podporou Flash Playeru
Vývojářské prostředí:	Flash Professional CS5

12.1 Instalace

Abychom mohli aplikace spouštět, musíme si stáhnout Flash Player, který je volně dostupný na stránkách firmy Adobe. Aktuální verze 10.2.153.1 o velikosti 2.7 MB. S tvorbou aplikací to již nebude takto snadné, oficiální software je placený a ačkoliv formát .swf byl společností Adobe uvolněn, nebude jednoduché najít alternativní software pro tvorbu aplikací.

12.2 Jazyk

Technologie Flash používá skriptovacího jazyka ActionScript, což je objektově orientovaný jazyk umožňující plné řízení pro navržené uživatelské rozhraní. Může být integrovaný do back-end technologií, které využívají ostatní jazyky a frameworky, jakými jsou například PHP, ASP. Disponuje velikou knihovnou tříd pro vývoj jednak online aplikací, ale i aplikací ryze desktopových.[17]

12.3 Aplikace

Aplikace Flash je kompletně v jednom souboru, nemá žádné externí soubory s actionScriptem nebo čímkoliv dalším, většinou obsahuje také obrázky a zdroje. Aplikace Flash je uložena v binárním souboru .fla a od verze CS4 i .xfl, neboli Extensible Flash. Tento formát obsahuje komprimované XML a značně zmenšuje velikost souboru a zjednudušuje práci s ním napříč platformami.

Zkompilovaná aplikace je ve formátu .swf a v této podobě se také umisťuje do HTML dokumentu. Nejčastěji mezi tagy <object>. Soubory ve formátu .swf můžeme spouštět i offline přímo v počítači.

12.4 Zaměření

Adobe Flash je zaměřen na tvorbu animací, bannerů a grafických prvků. Základním prvkem je časová osa, klíčové snímky a přechody. Prvotně tedy slouží především grafikům a designérům, spíše než programátorům a vývojářům webových aplikací. Nezamená to však, že by Flash nebyl použitelný při tvorbě celých aplikací. Přináší to s sebou ale nevýhody, jako je absence znovupoužitelných komponent, objektového jazyka atd. [8]

Technologie Flash má dobrou podporu multimédií, od obrázků (JPG, PNG, GIF), animací, zvuku (MP3), přes video (H.264 kodek), 3D animace až po vlastní protokol RTMP. [18] S rozšířením standartu HTML5 na Internet přestane být smysluplné také největší nynější využití Flashe, přehrávání videa a audia. HTML5 obsahuje tagy, které toto přehrávání uskuteční i bez flashového přehrávače. Avšak nemusíme se bát, že by technologie Flash zanikla, stále má své uplatnění, ať už v online hříčkách, offline aplikacích, bannerech, zpravodajstvích, překladačích, programech atd.

13 Porovnání RIA technologií

Porovnání technologií provedu na základě získaných bodů v ruzných kriteriích hodnocení, tyto body budou uděleny mnou, tudíž budou subjektivní, budu se však snažit hodnotit co nejvíce podle dostupných faktů a tím co nejvíce odstranit subjektivitu. Hodnocená kritéria a jejich váhy jsou: Instalace běhového prostředí (0.2), Multiplatformovost (0.25), Možnosti technologie (0.3), Vývojové prostředí (0.1) a Jazyk/y technologie (0.15). Technologie mohou v jednotlivých kritériích získat maximálně pět bodů, z těchto bodů vypočítám vážené skóre a následně ze všech hodnocených kritérií celkové skóre dané technologie.

13.1 Instalace běhového prostředí

13.1.1AJAX – 5 bodů

Všechny majoritní prohlížeče podporují JavaScript potažmo AJAX technologii, proto není potřeba instalovat žádné běhové prostředí

13.1.2 Adobe Flash – 4 body

Instalace běhového prostředí Flash Player či Shockwave Flash plug-inu do prohlížeče je jednoduchá a rychlá, s přihlédnutím k velké rozšířenosti Flash technologie je jisté, že velká část uživatelů má běhové prostředí již nainstalována.

13.1.3 Adobe Flex – 4 body

Technologie Flex využívá stejného běhového prostředí jako Flash, tudíž hodnocení vyplývá z technologie Flash.

13.1.4 Microsoft Silverlight – 4 body

Běhové prostředí Silverlight plug-in se instaluje společně pro všechny prohlížeče a jeho instalace je rychlá, avšak rozšířenost běhového prostředí je značně menší než například Flashové.

13.1.5 JavaFX – 2 body

Instalace běhového prostředí technologie JavaFX je časově náročnější, z důvodu nutnosti instalace běhového prostředí Java před samotnou instalací JavaFX.

13.1.6 OpenLaszlo – 5 bodů

Díky možnosti kompilovat aplikace do SWF formátu nebo jako HTML+JavaScript je instalace běhového prostředí snadná nebo žádná, zároveň je veliké rozšíření běhového prostředí.

13.1.7 Curl – 3 body

Instalace běhového prostředí Curl RTE je snadná, avšak jeho rozšíření mezi běžnými uživateli je téměř zanedbatelné. Ze samotné podstaty technologie (podniková řešení) vyplývá, že rozšířenost mezi běžnými uživateli není relevantní.

13.2 Multiplatformovost

13.2.1 AJAX – 5 bodů

Vzhledem k dlouhé historii JavaScriptu a rozsáhlé podpoře prohlížečů, můžeme říci, že technologie AJAX je podporována na největším počtu platforem.

13.2.2 Adobe Flash – 5 bodů

Podpora technologie Flash je velmi rozsáhlá, podporují ji nejen běžné operační systémy (Windows, Mac, Linux, Solaris), ale také další, včetně mobilních operačních systémů.

13.2.3 Adobe Flex – 5

Zde hodnocení taktéž vyplývá z technologie Flash.

13.2.4 Microsoft Silverlight – 4 body

S poslední vydanou verzí (4) se podporující platformy značně rozšířily, kromě Windows a Mac operačních systémů přibyli také mobilní platformy Windows phone a Symbian. Díky open-source komunitě a společnosti Novell je k dispozici také Moonlight pro platformy Linux a FreeBSD.

13.2.5 JavaFX – 5 bodů

Technologie JavaFX je také podporována širokým spektrem platforem (Windows, Mac, Linux, Solaris, Android, Windows Mobile).

13.2.6 OpenLaszlo – 5 bodů

Opět díky možnosti kompilovat aplikace do více formátů (SWF, HTML+JavaScript) je technologie OpenLaszlo podporována velkým množstvím platforem.

13.2.7 Curl – 3 body

Technologie Curl je podporována platformami Windows, Linux a Mac. Oproti ostatním technologiím se může zdát počet platforem malý, avšak je dostačující k pokrytí velkého množství uživatelů.

13.3 Možnosti technologie

13.3.1 AJAX – 3 body

Technologie AJAX je skvělé řešení pro změnu HTML dokumentu v RIA aplikaci. Pomocí této technologie vznikla a vzniká řada kvalitních webových aplikací, avšak vývoj větších aplikací může být těžkopádný a nepřehledný. Některé možnosti chybí, například podpora multimédií.

13.3.2 Adobe Flash – 2 body

Technologie Flash nabízí kvalitní grafické řešení aplikací, avšak je zaměřena na grafiky a například nenabízí žádné znovu-použitelné komponenty.

13.3.3 Adobe Flex – 4 body

Technologie Flex sice využívá stejného jazyka jako Flash, avšak díky jinému přístupu k tvorbě lze dosáhnout mnohem více použitelné aplikace.

13.3.4 Microsoft Silverlight – 4 body

Technologie Silverlight poskytuje možnost tvořit kvalitní RIA aplikace, částečně i díky úzké vazbě s .NET Frameworkem.

13.3.5 JavaFX – 5 bodů

Technologie JavaFX nabízí možnost tvorby kvalitních aplikací. Když přihlédneme k možnosti využití technologie Java, lze s touto technologií dosáhnout vyjímečných výsledků.

13.3.6 OpenLaszlo – 3 body

Technologie OpenLaszlo size nabízí možnost výstupní aplikace ve dvou formátech, avšak možnosti aplikací jsou omezeny na průnik těchto formátů (resp. Technologií).

13.3.7 Curl – 5 bodů

Možnosti technologie Curl jsou velmi široké. Dlouhodobý vývoj jazyka a knihovna obsahující mnoho ovládacích prvků poskytují možnost tvorby kvalitních aplikací.

13.4 Vývojové prostředí

13.4.1 AJAX – 3 body

Protože technologie AJAX není vyvinuta jednou společností, nemá také oficiální vývojové prostředí. Avšak aplikace můžeme vytvářet v libovolném textovém editoru či například v online nástroji WebToolKit od společnosti Google.

13.4.2 Adobe Flash – 2 body

Vývojové prostředí pro technologii Flash je Flash (profesional) CS5, toto prostředí je placené a zaměřené spíše na grafiky, než-li programátory. Tvorba aplikací bez tohoto vývojového prostředí je složitá (potřeba swf kompilátoru třetích stran).

13.4.3 Adobe Flex – 4 body

Vývojové prostředí technologie Flex je Adobe Flash Builder, tento software je placený. Tvořit aplikace však lze i bez Flash Builderu, protože zdrojové kódy lze psát, na rozdíl od Flash technologie, v libovolném textovém editoru a následně aplikaci zkompilovat pomocí volně dostupného Flex SDK.

13.4.4 Microsoft Silverlight – 2 body

Vývojové prostředí technologie Silverlight je Microsoft Visual Studio, tento software je placený. Dalším faktorem může být omezení Visual Studia pouze na platformu Windows.

13.4.5 JavaFX – 5 bodů

Technologie JavaFX poskytuje krom základního JavaFX SDK také zásuvný modul do vývojového prostředí NetBeans, které je zdarma a dostupné ve verzích pro Windows, Mac, Linux a Solaris.

13.4.6 OpenLaszlo – 4 body

Technologie OpenLaszlo využívá vývojového prostředí OpenLaszlo Development Kits, které je zdarma a dostupné ve verzích pro více platform, včetně kódů samotného vývojového prostředí. K tvorbě aplikací je však potřeba nainstalovat také JavaSDK. Je možné využít také vývojová prostředí NetBeans a Eclipse se zásuvným modulem OpenLaszlo. Pokud chceme kompilovat aplikace až podle běhového prostředí uživatele (Flash Player, JavaScript), budeme také potřebovat OpenLaszlo servlet package.

13.4.7 Curl – 3 body

Technologie Curl poskytuje vlastní vývojové prostředí (po registraci dostupné k nekomerčním účelům zdarma) Curl IDE, které je podporováno platformami Windows a Linux.

13.5 Jazyk/y technologie

13.5.1 AJAX – 3 body

Technologie AJAX využívá primárně jazyka JavaScript, kombinovaného s XML pro komunikaci, HTML pro rozložení aplikace a CSS pro vzhled. Pro vývojáře může být složité zorientovat se mezi množstvím kódu především u větších aplikací, naopak jazyky HTML, CSS a XML ovládá obrovské množství lidí.

13.5.2 Adobe Flash – 3 body

Technologie Flash využívá jazyka ActionScript, v kterém je psána celá aplikace. Jazyk ActionScript má na Internetu mnoho vývojářských komunit a jeho naučení není příliš složité.

13.5.3 Adobe Flex – 4 bodů

Jazykem technologie Flex je také ActionScript, ačkoliv můžeme částečně psát v značkovacím jazyce MXML, který celou aplikaci zpřehlední. Následná kompilace převede zdrojové kódy do jazyka ActionScript.

13.5.4 Microsoft Silverlight – 5 bodů

Technologie Silverlight umožňuje využití .NET jazyků (C#, VisualBasic), mimo to také Python nebo Ruby. Vývojáři mají tedy na výběr širokou skálu a mohou využít jazyka který jim nejvíce vyhovuje. Jako značkovací jazyk Silverlight využívá XAML.

13.5.5 JavaFX – 3 body

Technologie JavaFX využívá jazyka JavaFX script, výhodou může být možnost využití knihoven z jazyka Java a znalost (u vývojářů zabývajících se Javou) syntaxe.

13.5.6 OpenLaszlo – 4 body

Technologie OpenLaszlo využívá kombinace jazyka JavaScript a značkovacího jazyka LZX.

13.5.7 Curl – 3 body

Technologie Curl využívá jazyka Curl, který zahrnuje vzhled, skriptování, objektovou orientaci i podporu multimédií a grafiky.

13.6 Kalkulace

Kritérium	Váha	AJAX		Flash		Flex		Silverlight		JavaFX		Laszlo		Curl	
Instalace běh. prostředí	20%	5	1	4	0.8	4	0.8	4	0.8	2	0.4	5	1	3	0.6
Multiplatformovost	25%	5	1.25	5	1.25	5	1.25	4	1	5	1.25	5	1.25	3	0.75
Možnosti technologie	30%	3	0.9	2	0.6	4	1.2	4	1.2	5	1.5	3	0.9	5	1.5
Vývojové prostředí	10%	3	0.3	2	0.2	4	0.4	2	0.2	5	0.5	4	0.4	3	0.3
Jazyk/y technologie	15%	3	0.45	3	0.45	4	0.6	5	0.75	3	0.45	4	0.6	3	0.45
Vážené skóre			**3.9**		**3.3**		**4.25**		**3.95**		**4.1**		**4.15**		**3.6**

Tabulka 1

Legenda k Tabulce 1: Světle šedé sloupce značí vypočítanou váhu bodů získaných z jednotlivých kritérií. Tmavě šedé buňky udávají celkové vážené skóre jednotlivých technologií. Hodnoty jsou udány v bodech.

13.7 Zhodnocení porovnávání

V Tabulce 1 můžeme vidět, že několik technologií překonalo hodnotu 4 bodů (vážených). Neznamená to ovšem, že tyto technologie jsou nejlepší. Musíme brát v potaz, že každému vyhovuje něco jiného a každý má jiné preference. Při eventuelním výběru hraje spíše roli popis technologií, znalost jazyka, vyžitá platforma, způsob tvorby aplikací a další. Proč tedy hodnotit? Především kvůli ukázání (a očíslování) možností jednotlivých technologií a doporučení případným začínajícím tvůrcům. Odebrané body také v některých případech znamenají nedostatky technologie, které by mohl vybírající člověk přehlédnout.

14 Adobe Flash Builder 4

Adobe Flash Builder 4 je placené vývojové prostředí pro Flex framework. Poskytuje uživateli usnadnění pro tvorbu aplikací a urychluje práci. Z jeho výhod můžeme jmenovat barevné zvýrazňování kódu ve formátech MXML, ActionScript a CSS, doplňování kódu (našeptávač), interaktivní krokové debuggování, automatické generování běžných částí kódu, přejmenování všech referencí na danou třídu, metodu či proměnnou, a kromě dalších výhod také umožňuje kompletní práci s Adobe AIR aplikacemi. [25]

14.1 Flash Builder – popis rozhraní

Pro účely bakalářské práce mi společnost Adobe poskytla studentskou licenci na tento produkt, pojďme si tedy přiblížit základní orientaci v tomto software. Následující vysvětlení jsou číselně zobrazeny na obrázku 4.

14.1.1 Projekt

Pod číslem jedna vidíme všechny otevřené projekty, třídy projektů (.mxml soubory), stylové soubory, vložené animace a komponenty. Zde můžeme také všechny tyto soubory otevírat pro následnou práci s nimi.

14.1.2 Vzhled - Kód

Pod číslem dva vidíme přepínání mezi tlačítky „Design“ a „Source“, pomocí nichž můžeme přepínat obsah hlavní části, v kterém se nám zobrazí vzhled aplikace (jako na obrázku 4) nebo zdrojový kód.

14.1.3 Vlastnosti

Pod číslem tři vidíme kompletní nastavení vzhledu, uspořádání, zdroje dat a posluchačů k právě aktivní komponentě. Aktivní komponenta je ta, která je označená ve vzhledové části rozhraní (viz. bod dvě). Panel s nastavením je viditelný pouze při definování vzhledu aplikace, pokud jsme v režimu psaní kódu, panel není zobrazen.

14.1.4 Stavy

Pod číslem čtyři můžeme vidět okno pro stavy aktivní komponenty. Pomocí stavů lze u komponenty v závislosti na uživatelově interakci měnit jak po stránce vzhledu, tak funkcí. Okno se stavy je viditelné pouze při definování vzhledu aplikace, pokud jsme v režimu psaní kódu, okno není zobrazeno.

14.1.5 Chybová hlášení

Pod číslem pět vidíme přepínací tlačítka Problems, Data/Services, Network Monitor, Console, Properties, Debug, Progress. Těmito tlačítky přepínáme obsah okna pod nimi, kde se mohou zobrazit chyby v projektu a jejich popis, použité datové služby a jejich vlastnosti, výpis do konzoly, výpis probíhající kompilace aplikace a další.

14.1.6 Třídy

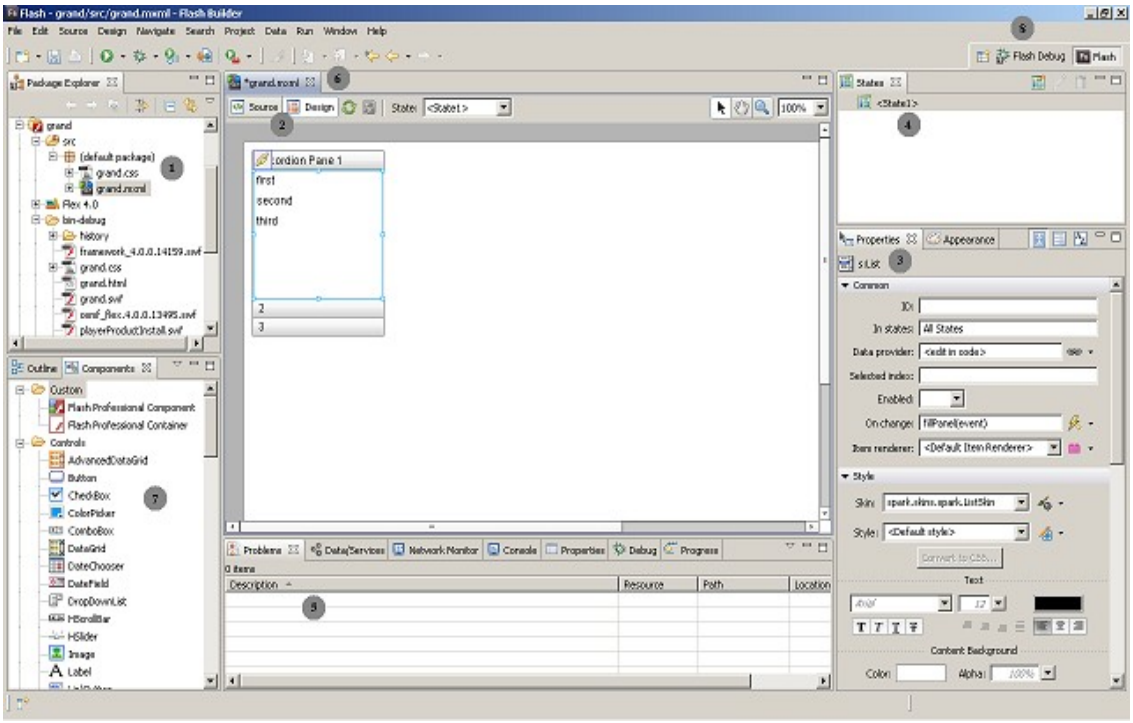
Pod číslem šest vidíme lištu, na které se zobrazují všechny otevřené soubory v našem projektu, nejčastěji třídy, zde také mezi nimi můžeme přepínat.

14.1.7 Komponenty

Pod číslem sedm vidíme přepínání mezi tlačítky Outline a Components. Pomocí těchto tlačítek přepínáme obsah okna pod nimi, kde můžeme vidět všechny komponenty použité v našem projektu, seřazené dle jejich vnoření do ostatních komponent. Po přepnutí na Komponenty se nám zobrazí kompletní seznam využitelných komponent (jako na obrázku 4), odkud je můžeme táhnutím použít v našem projektu.

14.1.8 Debugování

Pod číslem osm vidíme přepínání mezi tvorbou aplikace, debugovací perspektivou a dalšími. Debugovací perspektiva nám nabízí prohlížení proměnných, vláken aplikace, metod a dalšího během spuštěné aplikace. Také můžeme pomocí „BreakPoint“ značek aplikaci v určitém kroku zastavit a následně v debugovací perspektivě krokováním procházet řádek po řádku.


Flash - grand/src/grand.mxml - Flash Builder
File Edit Source Design Navigate Search Project Data Run Window Help
Flash Debug
Flash
Package Explorer
grand
src
(default package)
grand.css
grand.mxml
Flex 4.0
bin-debug
history
framework_4.0.0.14159.swf
grand.css
grand.html
grand.swf
playerProductInstall.swf
*grand.mxml
Source
Design
State:
<State 1>
100%
London Pane 1
first
second
third
2
3
States
<State 1>
Properties
Appearance
s List
Common
Id:
In states:
All States
Data provider:
<edit in code>
Selected index:
Enabled:
On change:
fillPanel(event)
Item renderer:
<Default Item Renderer>
Style
Skin:
spark.skins.spark.ListSkin
Style:
<Default style>
Convert to CSS...
Text
Arial
12
Content Background
Color:
Alpha:
100%
Outline
Components
Custom
Flash Professional Component
Flash Professional Container
Controls
AdvancedDataGrid
Button
CheckBox
ColorPicker
ComboBox
DataGrid
DateChooser
DateField
DropDownList
HScrollBar
HSlider
Image
Label
Problems
Data/Services
Network Monitor
Console
Properties
Debug
Progress
0 items
Description
Resource
Path
Location
1
2
3
4
5
6
7
8

Obrázek 4

15 Tvorba aplikace Flex

Téměř nutností před započetím tvorby aplikací je vytvořit bezpečné úložiště, odkud Flash Player bude moci načítat soubory a knihovny aplikací, pokud bychom toto úložiště nevytvořili, každá aplikace kterou bychom spouštěli (vyjma z Internetu) by hlásila neoprávněný přístup k souborovému systému a ukončila by se. Bezpečné úložiště můžeme vytvořit na stránkách adobe, kde je kompletní online nastavení Flash Playeru.[19]

15.1 Založení projektu – ukázkový příklad Dialogové okno

Před samotným založením nového projektu v programu Flash Builder musíme specifikovat, jestli se jedná o projekt určený na Internet, nebo o desktopovou aplikaci (Adobe AIR). Dále si také specifikujeme serverovou technologii naší aplikace. Na výběr jsou: Žádná, ASP .NET, ColdFusion, J2EE a PHP. Na závěr si určíme jakým způsobem mají být k aplikaci připojeny využité knihovny, zda budou přímo v souboru, nebo sdílené (Runtime Shared Library). Sdílení knihoven může být výhodné z hlediska velikosti u rozsáhlejších projektů. Flash Builder poté vygeneruje základní kód do námi pojmenovaného .mxml souboru. Vygenerovaný kód můžeme vidět zde:

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
                    xmlns:s="library://ns.adobe.com/flex/spark"
                    xmlns:mx="library://ns.adobe.com/flex/mx"
                    minWidth="955" minHeight="600">
      <fx:Declarations>
            <!-- Place non-visual elements (e.g., services, value
objects) here -->
      </fx:Declarations>
</s:Application>
```

Základním tagem vygenerovaného kódu je <s:Application> který má v atributech uvedené jmenné prostory objektových knihoven. Mezi tagy <fx:Declarations> lze vkládat ne-vizuální elementy, například služby RemoteObject, WebService a HTTPService.

15.2 Komponenty

Komponenty v projektu deklarujeme přímo mezi tagy <s:Application>. Příklad můžeme vidět zde:

```
<s:Button label="spusť dialogové okno" click="launch()"/>
```

V kódu je tedy komponenta z knihovny Spark, což značí „s“ před jménem komponenty. Vytvořená komponenta je tlačítko, které nese nápis definovaný atributem „label“ a posledním atributem tlačítka je „click“ kde je uvedeno, co se má stát po kliknutí na komponentu. V tomto případě se zavolá metoda „launch()“. Samozřejmě tlačítko může obsahovat více atributů (např. id, visible, enabled a také atributy měnící vzhled komponenty), avšak v tuto chvíli je nepotřebujeme. Co se tedy stane po zmáčknutí na tlačítko?

15.3 Aplikační logika

Veškerá aplikační logika, tedy i následná akce po zmáčknutí tlačítka je uvedena v aplikaci mezi tagy <fx:Script>, které se v aplikaci vyskytují mezi tagy <s:Application>. Jak můžeme vidět v kódu:

```
      <fx:Script>
            <![CDATA[


            ]]>
      </fx:Script>
```

Mezi tagy <fx:Script> se aplikační logika píše v jazyce ActionScript uvnitř bloku CDATA, právě zde se bude vyskytovat metoda, která se zavolá po kliknutí na tlačíko. Zmíněná metoda bude vypadat asi takto:

```
private function launch():void {


                  }
```

Všechny metody aplikace jsou označeny slovem „function“, statut „private“ definuje přístupnost k dané metodě ostatními částmi aplikace, zde může být také použit statut public, protected, internal a static. Při rozhodování který statut použít lze zjednodušeně

říci: s co nejmenší přístupností, která nám bude dostačovat. Název naší metody je „launch“ a závorky za názvem mohou obsahovat atributy, které se zasílají z místa volání, např. Při zobrazení konkrétní hodnoty uvedené v další komponentě. Slovo „void“ stanovuje zda a jaký formát bude metoda po skončení své funkce vracet. V tomto případě metoda nevrátí nic. Zde lze použít např: Boolean, Number, String nebo i * pro jakýkoliv formát. Všechny metody které vracejí nějakou proměnnou nebo soubor proměnných musí v těle obsahovat údaj „return“.

```
private function launch():void {

	var dialog:TitleWindow = new TitleWindow();
	dialog.title = "název okna";
	dialog.addEventListener(Event.CLOSE, zavrit);

	var textik:Label = new Label();
	textik.text = "obsah okna";

	dialog.addElement(textik);
	PopUpManager.addPopUp(dialog, this, true);
	PopUpManager.centerPopUp(dialog);

}
```

V kódu můžeme vidět obsah metody „launch“, prvním řádkem zakládáme proměnnou (var – variable) jménem „dialog“. Pomocí dvojtečky za jménem proměnné určíme její typ, v tomto případě „TitleWindow“. Zároveň zde proměnnou inicializujeme (.. = new TitleWindow()). Každý řádek psaný v ActionScriptu musí být uzavřen středníkem. Na dalším řádku dialogové okno pojmenujeme přiřazením řetězce k jeho atributu „title“. Následně vytvoříme posluchače, který kontroluje zavření aplikace. Od Flex verze 4 není třeba přidávat řádek definující viditelnost zavíracího tlačítka (křížek na horní liště okna), neboť defaultně je viditelné.

Další dva řádky vytvářejí, inicializují a naplňují proměnnou typu Label (jeden řádek textu, který uživatel nemůže měnit). Následně přidáváme do dialogového okna výše zmíněný řádek textu pomocí funkce addElement(). V závěru metody přiřazujeme dialogové okno k „PopUpManager“, který ho zobrazuje nad všemi ostatními komponentami a tyto komponenty také zablokuje dokud uživatel nezavře (potvrdí,

stornuje atd.) dialogové okno. Poslední řádek je čistě vzhledový, určuje umístění dialogového okna v aplikaci, v našem případě na střed.

V posluchači spouštěném při kliknutí na zavírací tlačítko se odkazujeme na metodu „zavrit“, její kód je zde:

```
private function zavrit(evt:CloseEvent):void {
      PopUpManager.removePopUp(TitleWindow(evt.target));
}
```

Metoda „zavrit“ s sebou nese parametr evt:CloseEvent, který udává událost zavírání a také obsahuje informace o zavírané komponentě. V jediném řádku metody odstraňujeme dialogové okno z PopUpManageru, což způsobí zmizení okna v aplikaci a odblokování ostatních komponent.

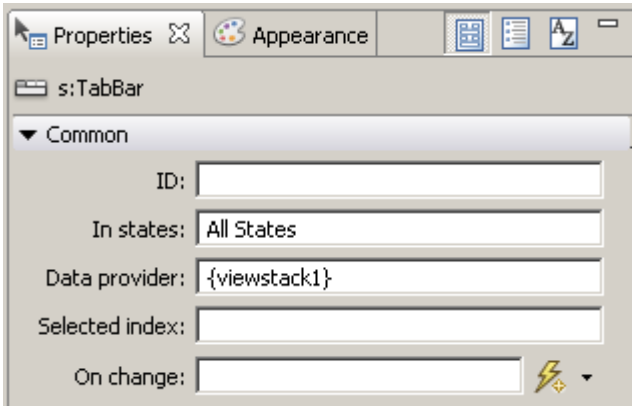
```
import mx.controls.Label;
import mx.events.CloseEvent;
import mx.managers.PopUpManager;
import spark.components.TitleWindow;
```

Zbývající kód naší aplikace se stará o import využitých knihoven a generuje se sám, při psaní aplikace, je umístěn jako první v bloku CDATA, aby při použití knihovny byly již navázány. Vytvořili jsme tedy první aplikaci která otevírá jednoduché dialogové okno s textem, po spuštění vidíme rovnou několik nedokonalostí, tlačítko je úplně vlevo nahoře nalepené na okraj, dialogové okno je sice ve prostřed ale je malé, a další. Zdrojový kód aplikace viz příloha I.

15.4 Využití Designeru k tvorbě aplikace – ukázkový příklad Záložky

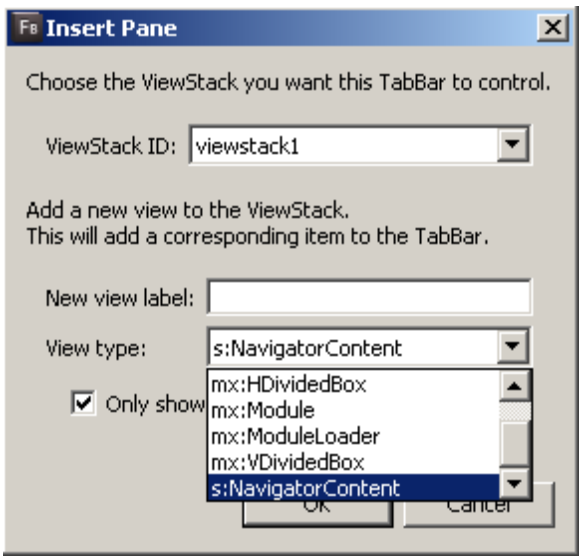
Po založení nového projektu nazvaného „zalozky“ se přepneme pomocí tlačítka Design do tvorby vzhledu aplikace. Ukážeme si, že aplikaci lze vytvořit s minimem programování a vše co budeme potřebovat je panel s komponentami, vzhledová stránka aplikace a nastavení jednotlivých komponent. Pro příklad záložky si nejprve vytvoříme dvě komponenty, první z nich bude TabBar z knihovny Spark, která nám poskytne tlačítka na přepínání záložek. Druhou komponentou bude ViewStack, tato komponenta poskytuje rámec pro uskladnění několika ploch přes sebe, což přesně splňuje naše požadavky. Nyní máme dvě komponenty které spolu zatím nejsou nijak provázány.

Propojení komponent se vytvoří, když v komponentě TabBar uvedeme do vlastnosti DataProvider jméno komponenty ViewStack, viz obrázek 5.



Obrázek 5

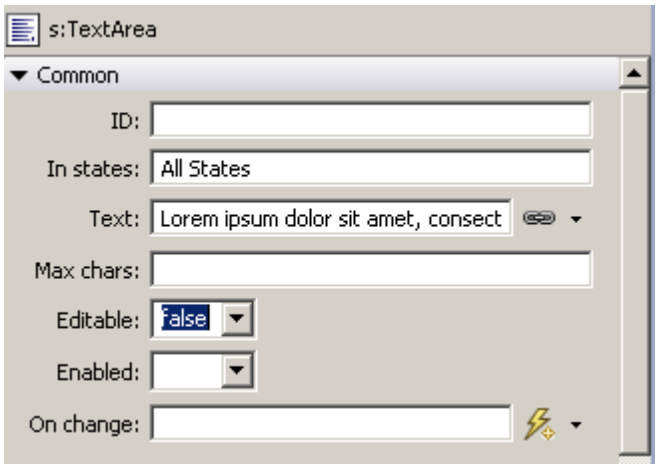
Nyní jsou komponenty provázány a pomocí klikání na tlačítko „+“ můžeme přidávat jednotlivé záložky. U těchto záložek je nutné určit typ vkládaného obsahu, v našem případě NavigatorContent z knihovny Spark, lze použít několik dalších komponent pro vložení do záložky, např. Form, DividedBox, Modul, viz obrázek 6.



Obrázek 6

Vytvořili jsme tedy několik záložek a k nim odpovídající počet tlačítek v komponentě TabBar, nyní je třeba záložky naplnit, je možné použít mnoho komponent. Do první záložky, respektive komponenty NavigatorContent vložíme například komponentu TextArea. Komponenta TextArea je kontejner který může obsahovat více řádek textu, tento text může uživatel měnit pokud nenastavíme jinak. Následně v nastavení komponenty do políčka text vložíme text, lze ho vložit dvěma způsoby, buďto přímo

vepíšeme konkrétní text, nebo můžeme použít službu, pomocí které získáme požadovaný text, nyní tedy pouze vepíšeme text. V nastavení komponenty TextArea změníme vlastnost editable na „false“, aby uživatel tento text nemohl měnit. Protože budeme k této komponentě přistupovat z metody v aplikaci, nastavíme jí identifikátor na který se budeme odkazovat. Identifikátor lze nastavit v políčku ID, viz obrázek 7.



Obrázek 7

15.5 Vkládání obrázku do aplikace

Do druhé tabulky vložíme obrázek, tedy komponentu Image. U této komponenty musíme nastavit hodnotu source, jinak řečeno vložit do ní obrázek. Pokud bychom aplikaci nevkládali na Internet, mohli bychom použít tlačítko Browse, kterým bychom vybrali náš obrázek. Avšak pokud aplikace bude použita na internetu, při kliknutí na obrázek by se objevilo chybové hlášení, že aplikace nemůže přistupovat do lokálního souborového systému, nehledě na fakt, že ostatní uživatelé tento obrázek v souborovém systému nemají. Máme tedy dva způsoby jak obrázek vložit, aniž by bylo nutné odkazovat na souborový systém. První variantou je, vložit na stránky obrázek, který chceme použít a při tvorbě aplikace na něj odkázat přimo na Internet (např. Http://webove-stranky.cz/obrázek.jpg).

Takto odkazovat na obrázek přináší několik výhod, určitě tím zmenšíme velikost aplikace, zrychlíme její načítání a daný obrázek můžeme měnit (dokud zachováme jeho jméno, příponu a umístění). Tato metoda má ovšem také omezení, musíme počítat s možnou změnou velikosti obrázku a upravit podle toho aplikaci. Posledním způsobem jak připojit obrázek je vložit ho přímo do aplikace. Připojení obrázku lze udělat několika způsoby, například vložením meta tagu Embed do ActionScriptové části

aplikace.

```
[Embed("../slozka/obrazky/obrazek.jpg")]
public var tridaObrazku:Class;
```

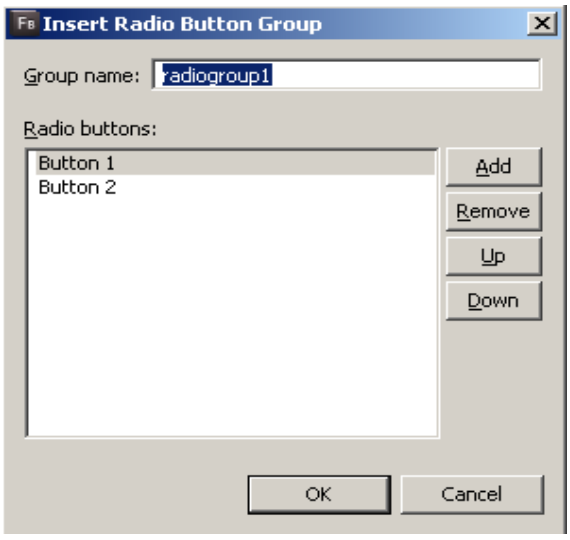
Mebo můžeme do parametru source u dané komponenty vložit obrázek následně:

```
@Embed(source='c:/slozka/obrazky/obrazek.jpg')
```

Ukládání obrázku přímo do zkompilované aplikace samozřejmě zvětší její velikost a zpomalí načítání (podle velikosti obrázku), ale zároveň to poskytuje výhodu, že můžeme aplikaci vytvořit přímo na míru tomuto obrázku a nemusíme vkládat žádný obrázek nikam na Internet. Takto vložený obrázek také nejde měnit jinak než znovu-zkompilováním aplikace. Komponentě Image také nastavíme identifikátor ID, kterým na ni budeme později odkazovat.

15.6 OnClick event

Nyní jsme vytvořili dvě záložky v kterých se zobrazuje text a obrázek. Do aplikace přidáme RadioButtony, které budou měnit obsah těchto záložek. Nejprve vytvoříme jejich vzhled, přetáhnutím komponenty RadioButtonGroup do aplikace. Po přetáhnutí se objeví dialogové okno kde můžeme pojmenovat skupinu RadioButtonů, vytvořit nové, zrušit existující a přejmenovat samotné RadioButtony, viz obrázek 8.



Obrázek 8

Po vytvoření a pojmenování RadioButtonů každému z nich vytvoříme metodu, která se spustí po zatrhnutí příslušného RadioButtonu. U vlastnosti „On click“ použijeme možnost Generate Event Handler, která tuto metodu vytvoří. Po vygenerování metody

se Flash Builder automaticky přepne do zobrazení kódu aplikace.

```
protected function radiobutton1_clickHandler(event:MouseEvent):void
                    {
                          // TODO Auto-generated method stub
                    }
```

V této metodě změníme atribut text u komponenty TextArea, která má identifikátor „popisek“, a atribut source u komponenty Image, která má identifikátor „obrazek“, aby se po kliknutí na daný RadioButton naplnily záložky odpovídajícími daty.

```
popisek.text="popisek obrázku";
obrazek.source="http://webove-stranky.cz/obrazek.jpg";
```

Následně obdobné metody vytvoříme všem RadioButtonům a tím bychom mohli předpokládat, že aplikace je kompletní.

15.7 Odstranění chyb

Jenže při spuštění by se po kliknutí na první RadioButton objevilo chybové hlášení, že nemužeme přistupovat k vlastnosti nulového objektu. Tato chyba znamená, že záložky, které nebyly před kliknutím na RadioButton zobrazené, nejsou ani instanciované a tudíž k nim nelze přistupovat. Když se při spouštění aplikace objeví chyba, Flash Builder nabídne debugovací perspektivu, kde lze zjistit jaká chyba to je, proč se objevila a kde v aplikaci je. Tuto chybu lze napravit přidáním atributu creationPolicy do komponenty ViewStack a nastavením hodnoty na „all“. Atribut CreationPolicy se stará o načítání obsahu komponent, v případě hodnoty all se vše načte hned při spuštění aplikace. Lze přiřadit hodnoty all, auto, non a queued, defaultní hodnotou je „auto“. Hodnota auto načte pouze komponenty zobrazené při spuštění aplikace, u větších komponent tedy může zrychlit načítání aplikace.

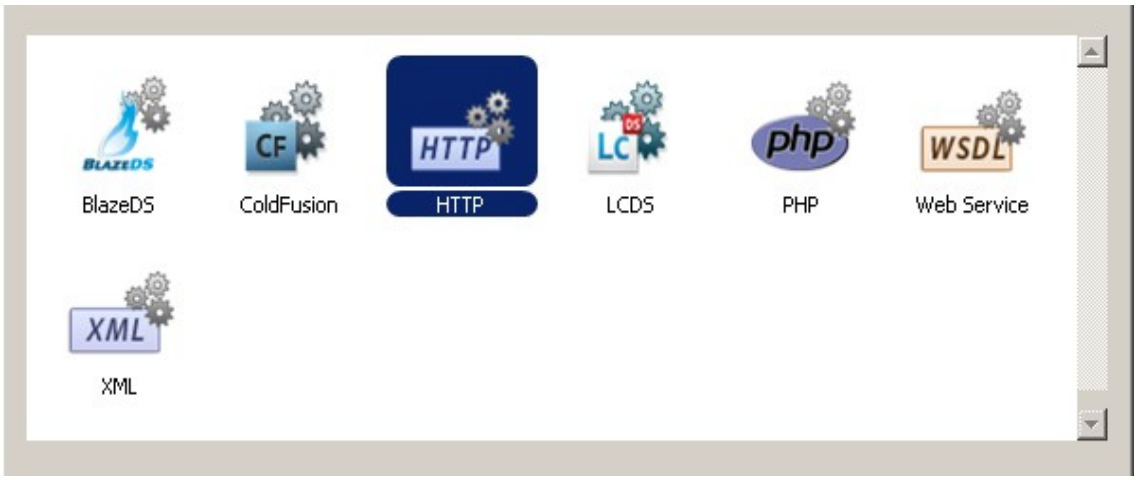
```
<mx:ViewStack x="154" y="121" id="viewstack1" width="212" height="200"
creationPolicy="all">
```

Vytvořili jsme aplikaci s minimálním využitím programování, pouze plnění proměnných v metodách a přidání jednoho atributu do kontejneru pro komponenty. Ostatní kód byl vygenerován automaticky, nebo podle atributů zadaných v nastavení jednotlivých komponent. Nedokonalostí této aplikace může být potřeba připojení na

Internet z důvodu načítání obrázků, avšak zároveň je samotná aplikace menší, než kdyby byly obrázky obsaženy v ní. Zdrojový kód aplikace viz příloha IX.

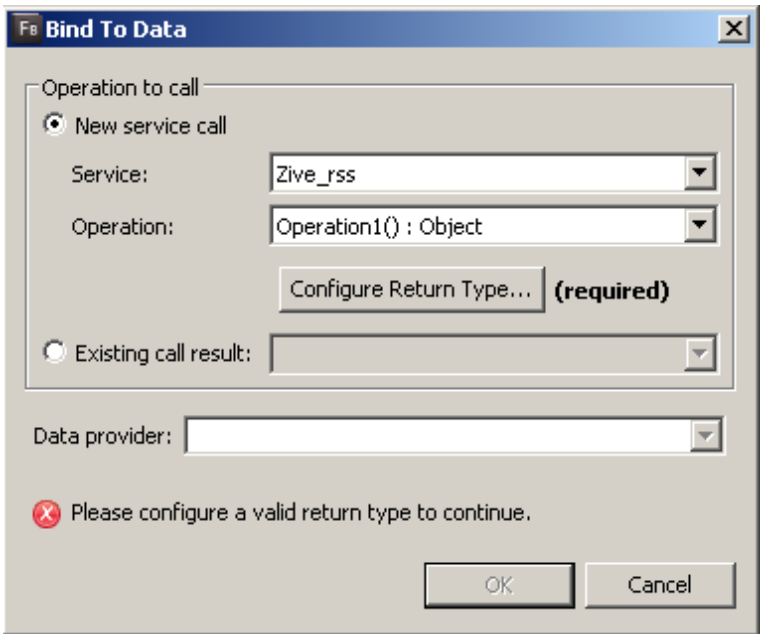
15.8 Datové služby – ukázkový příklad DataGrid

Po založení nového Flex projektu vytvoříme datovou službu pomocí menu Data – Connect to Data/Services, zobrazí se okno v kterém lze vybrat typ služby, pro naše potřeby bude optimální HTTP, která podporuje XML a JSON formát dat, jak můžeme vidět na obrázku 9. Po vybrání HTTP služby se zobrazí okno kde se specifikují její vlastnosti. Do kolonky URL vložíme adresu poskytovatele dat do služby (např. http://www.zive.cz/rss/sc-47/default.aspx), a do kolonky Service name vepíšeme jméno služby, zbytek se vygeneruje sám a pokud nechceme jinak, není třeba ho měnit.



Obrázek 9

Aby bylo kam vložit data, vytvoříme přetáhnutím komponentu DataGrid. Následně naplníme datovou mřížku daty ze služby. U komponenty DataGrid klikneme na tlačítko Bind To Data u atributu Data provider. Zobrazí se okno, kde lze přiřadit komponentě data ze služby či jiného zdroje, viz obrázek 10. Automaticky se vybere naše služba a jediné co musíme nastavit je výstupní formát, tedy co ze všech příchozích dat chceme zobrazit v komponentě. Nastavení formátu je přístupné pod tlačítkem Configure Return Type.



Obrázek 10

Při nastavování návratového typu vybereme která data chceme ze služby použít, v našem případě jako root vybereme data v hodnotě item, který zastupuje konkrétní články. Všechna data (title, link, guide, description, pubDate) se přiřadí do komponenty DataGrid, která se změní tak, aby obsahovala příslušný počet sloupců. Všechna tato data však zobrazit nepotřebujeme, proto ve vlastnostech komponenty najdeme tlačítko „Configure Columns …“, které otevírá okno, kde lze nastavit jména, počet, velikosti a další ohledně komponenty. Zrušíme všechna nepotřebná data a zachováme sloupce title (název článku), link (odkaz na článek) a description (úryvek článku). Následně zatrhneme u všech sloupců možnost word wrap (zalomení slov), aby se obsah jednotlivých kolonek zobrazil celý. Po zavření nastavení změníme vlastnosti datové mřížky, výšku a šířku nastavíme na 96%, aby se obsah komponenty zobrazil přes co největší plochu.

15.9 Odkazování na webové stránky

Na závěr příkladu necháme vygenerovat metodu, která se zavolá po změně řádky v komponentě DataGrid. Ve vlastnostech u parametru On change zmáčkneme na možnost „Generate Event Handler“, která nám vytvoří zmíněnou metodu.

```
protected function dataGrid_changeHandler(event:ListEvent):void
{
	var text:String = dataGrid.selectedItem.link;

	if (text != null) {
		var request:URLRequest = new URLRequest(text);
		navigateToURL(request, "_blank");
	}
}
```

V této metodě přistupujeme k hodnotě „link“, neboli odkazu na vybraný článek, který je umístěn v komponentě DataGrid na aktuální pozici. Podmínka „if“ je v metodě jako bezpečnostní opatření, kdyby chyběl zmíněný odkaz. Řetězec z datové mřížky se uvede jako parametr proměnné typu URLRequest, která se používá při odkazování na URL a poté se pomocí metody, z knihovny flash.net, navigateToURL otevře nové okno s cílenou webovou stránkou.

15.10 Omezení datových služeb

nasazení aplikací používajících datové služby na server je potřeba aby server akceptoval náš požadavek, respektive požadavek z domény, kde bude aplikace umístěna. Tento problém se dá řešit umístěním souboru crossdomain.xml na server se službou. Některé servery poskytují v tomto ohledu přístup z jakékoliv domény, avšak server živě.cz bohužel ne. Kvůli tomu tato aplikace nepůjde na Internetu spustit kvůli nemožnosti načíst data. Problém by šel řešit vytvořením externí datové služby pomocí jiné technologie, která by ukládala data na server, kde je umístěna aplikace. Aplikace by poté zpracovávala tato data a nemusela by se odkazovat na jiný server.

Problémy s tímto bezpečnostním omezením nemají pouze datové a webové služby, ale veškeré zdroje dat, obrázků, videí, komponent atd. mimo server na kterém je umístěná aplikace.

15.11 Ukázka cross-domain souboru

Soubor musí být umístěn v kořenovém adresáři serveru a musí se jmenovat crossdomain.xml. Může mít více podob, několik možností jak se povolují přístupy z určitých webů je v následující ukázce, více informací lze najít na stránce Cross-domain policy file specification [20]

```
<?xml version="1.0"?>
<cross-domain-policy>
    <site-control permitted-cross-domain-policies="by-content-type"/>
    <allow-access-from domain="www.webova-stranka.cz"/>
    <allow-access-from domain="*.cz"/>
    <allow-access-from domain="135.215.0.41"/>
</cross-domain-policy>
```

Zdrojový kód aplikace viz příloha III.

15.12 Efekty – ukázkový příklad Effect

Ve Flex aplikacích lze použít několik způsobů, jak animovat komponenty. Animace spadají mezi non-visual komponenty a proto je deklarujeme mezi tagy <fx:declarations>. V tomto ukázkovém příkladu jde především o přiblížení tvorby efektů, proto výsledná aplikace bude pouze ukázkou měnících se komponent.

15.13 Simple Motion Path

První animace bude typu SimpleMotionPath u níž se vytvářejí klíčové snímky (key frames) a animování mezi nimi. Animace jsou obsaženy v komponentě Animate, která musí mít parametr id, abychom na ni mohli odkazovat, také musí být definován parametr target, který určí jaká komponenta se bude animovat a jako poslední musíme určit parametr duration, jež stanovuje trvání každé obsažené animace, respektive celou animační dobu. Následně můžeme nastavit mnoho dalších možností, např. opakování, prodleva, více cílu animace, posluchače na různé události (začátek, konec, opakování atd.). Klíčové snímky lze definovat třemi způsoby, které jsou ukázány v následujícím kódu.

```
<s:Animate id="anim" target="{btn1}" duration="1100">
      <s:SimpleMotionPath property="x" valueFrom="50" valueTo="100"/>
      <s:SimpleMotionPath property="y" valueTo="100"/>
      <s:SimpleMotionPath property="width" valueBy="20"/>
</s:Animate>
```

V prvním případě animujeme v ose x a klíčové snímky se vytvoří na počáteční hodnotě 50 a koncové hodnotě 100, při využití tohoto animování se s vyvoláním každého dalšího cyklu vrátí komponenta na hodnotu 50. V druhém případě se komponenta animuje po ose y a udáváme pouze koncový klíčový snímek s hodnotou

100, počáteční se automaticky generuje při spuštění animace. Tato animace se z pohledu uživatele provede pouze jednou, protože v každém dalším cyklu se komponenta pohybuje z hodnoty 100 na stejnou hodnotu. Ve třetím případě měníme šířku komponenty a neudáváme žádný klíčový snímek, místo toho stanovíme o kolik se komponenta rozšíří. Klíčové snímky se při každém cyklu generují z aktuální šířky. V této animaci se komponenta bude při každém cyklu rozšiřovat. Hodnota property udává která vlastnost se bude animovat, může nabývat hodnot x, y, width, jako v kódu, ale i height, scaleX, scaleY, color, distance, angle a další.

15.14 Parallel

Jako další možnost lze použít komponentu parallel, která provádí několik konkrétních animací zároveň, a nedochází mezi nimi ke konfliktu. Parallel může obsahovat transformaci Move, Rotate a Scale, jak je ukázáno v následujícím kódu.

```
<s:Parallel id="anim2" duration="1100" target="{btn2}">
      <s:Scale scaleXFrom="1" scaleXTo="1.2" scaleYFrom="1"
            scaleYTo="1.2"/>
      <s:Rotate angleBy="360"/>
      <s:Move xBy="25" yBy="25" />
</s:Parallel>
```

Tyto transformace mají podobné atributy jako v minulém případě, udáváme parametry počátečního a koncového klíčového snímku, nebo hodnotu změny. U transformace scale udáváme roztažení, hodnota „1“ je normální velikost, menší hodnoty komponentu zmenšují a naopak. Nelze však použít hodnotu „0.0“.

Tyto tři efekty mají své obdoby s využitím třetí osy (Rotate3D, Move3D a Scale3D), kde lze roztahovat, otáčet či posunovat i v ose Z. V komponentě parallel můžeme také použít mnoho dalších transformací např. změny barev, stínu, lesku, rozmazání, elastičnost, různé třepání či průhlednost jako v následujícím kódu.

```
<s:Fade alphaFrom="1" alphaTo="0.3"/>
```

15.15 Stavy a Radial Gradient

Pro transformace můžeme využít i stavy komponent a následně vytvořit přechody mezi nimi. Je potřeba vytvořit aspoň dva stavy dané komponenty, vytvořit EventHandler který mezi nimi bude přepínat a nakonec plynulý přechod.

```
<s:Group mouseDown="currentState='stav2'"
      mouseUp="currentState='stav1'">
 <s:Ellipse width="100" height="100" x="35" y="300">
  <s:fill>
   <s:RadialGradient>
<s:GradientEntry id="stred" color="0x00ff00" color.stav2="0x003300"
      ratio="0"/>
<s:GradientEntry id="okraj" color="0x008800" ratio="1"/>
   </s:RadialGradient>
  </s:fill>
 </s:Ellipse>
</s:Group>
```

Měněná komponenta je v tomto případě elipsa, která je vyplněna barevným přechodem. Prvek GradientEntry a jeho hodnota ratio udávájí barvu výplně elipsy a procentuální vzdálenost od středu, kde bude barva výplně využita. V tomto případě udáváme dvě barvy, jedna je 0% vzdálena od středu, druhá 100% a mezi nimi se vytvoří barevný přechod. Nyní vytvoříme komponentu Group, která umožňuje vytvářet posluchače (mouseDown, mouseUp), u této komponenty vytvoříme a pojmenujeme dva stavy (viz. kapitola: Flash Builder – popis rozhraní, podkapitola: 4. Stavy), na které se později odkazujeme. Komponentě Group jsme přiřadili dva posluchače, jeden se aktivuje, když uživatel zmáčkne tlačítko myši a druhý když ho pustí. Právě tyto posluchače budou přepínat stavy. Aby se něco měnilo, musíme v jednom barevném vstupu přidat atribut color.*jméno_stavu* a jako hodnotu mu přiřadíme další barvu. Nyní při zmáčknutí tlačítka myši nad kružnicí, se změní její barva veprostřed a přepočítá se barevný přechod, při puštění tlačítka se vše vrátí.

15.16 Plynulý přechod

Aby byl přechod mezi jednotlivými barvami plynulý, přidáme do aplikace prvek typu Transition, v kterém lze přechod specifikovat.

```
<s:transitions>
      <s:Transition>
            <s:AnimateColor target="{stred}" duration="150"/>
      </s:Transition>
</s:transitions>
```

Prvek přechodu obsahuje efekt AnimateColor, který použijeme při změně barvy elipsy. Efektu nastavíme jaký barevný vstup se bude měnit a jak dlouho změna bude trvat. Všechny přechody v aplikaci musí být uvedeny mezi tagy <s:transitions>, které jsou v aplikaci umístěné stejně jako stavy. [21]

15.17 Efekty zjednodušení

Flex poskytuje třídu effects.easing z knihovny Spark. V této třídě můžeme najít efekty Bounce, Elastic, Linear, Power a Sine.

Objekt s efektem Bounce od výchozí hodnoty zrychluje k cílové a po jejím dosažení se od ní několikrát „odrazí“. Objekt s efektem Elastic od výchozí hodnoty zpomaluje k cílové a po jejím dosažení pokračuje za ní, následně osciluje kolem cílové hodnoty se zmenšující se vzdáleností dokud jí nedosáhne. Objekt s efektem Linear má tři fáze pohybu, zrychlení, konstantní rychlost a zpomalení. Trvání zrychlení z zpomalení můžeme specifikovat procentuálním podílem z času animace. Objekt s efektem Power má dvě fáze pohybu, zrychlení a zpomalení, které na sebe přímo navazují. Velikost zrychlení je nastavitelná, stejně jako podíl zrychlování z doby animace. Objekt s efektem Sine má dvě fáze, zrychlení a zpomalení, které na sebe přímo navazují. Podíl doby zrychlování z celkové doby animace je nastavitelný. [22] Zjednodušující efekty lze použít např. S efektem Move, Rotate, Resize, Scale, Move3D, Fade. V následujícím kódu je možné vidět jakým způsobem se přidávájí.

```
<s:Bounce id="bouncer"/>
<s:Move yBy="100" duration="300" easer="{bouncer}"
target="{elipsa2}"/>
```

Vytvořený ease-effect se přiřadí efektu Move v parametru easer ve složených závorkách, aby se hodnota nezpracovala jako řetězec, ale identifikátor objektu. Cílem efektu je druhá elipsa, uložená v další komponentě Group, avšak lepší je pohybovat s komponentou Group, jinak se bude kvůli pohybu elipsy rozšiřovat a mohla by překrývat (ve smyslu znemožňovat kliknutí a ostatní akce) jiné komponenty v aplikaci. Zdrojové kód aplikace viz příloha V.

15.18 Vzhled kurzoru – ukázkový příklad Cursor

V aplikaci Flex můžeme měnit vzhled kurzoru pomocí správy kurzoru (Cursor Manager). Správa kurzoru může poskytovat uživateli zpětnou vazbu, při načítání části aplikace, nebo rozpoznání typu vstupu dané komponenty. Pro práci se vzhledem kurzoru musíme nejprve v části <fx:Script> importovat třídu CursorManager

```
import mx.managers.CursorManager;
```

Některé komponenty mění vzhled kurzoru automaticky (např. TextArea), u jiných lze změnu povolit či zakázat (např. Image, Button, HTTPService), u ostatních komponent lze vzhled kurzoru měnit díky třídě Cursor Manager.

V novém projektu vytvoříme několik komponent, abychom ukázali jakým způsobem ze vzhled kurzoru mění. První vytvořenou komponentou bude TextArea. Po spuštění aplikace můžeme vidět, že kurzor se po najetí nad komponentu automaticky změnil ve svislítko. Tato komponenta tedy automaticky mění vzhled kurzoru. Další komponentou kterou vytvoříme bude tlačítko (Button), zde by však automaticky řádna změna neproběhla a proto musíme přepnout boolean hodnotu ButtonMode na true.

```
<s:Button x="296" y="10" useHandCursor="true" buttonMode="true"/>
```

Po přepnutí této hodnoty se nad tlačítkem bude uživateli zobrazovat kurzor zobrazující ruku s ukazujícím prstem. Použili jsme zde také boolean vlastnost useHandCursor, která určuje zda ukázat zmíněnou ruku, nebo normální vzhled kurzoru. Pokud se tato vlastnost nenastaví, je automaticky povolena, proto má smysl ji použít pouze s hodnotou „false“. Obdobným způsobem lze nastavit „busy“ kurzor komponentám které načítají nějaký obsah, ať už pres službu, nebo ze souboru.

```
<mx:Image x="296" y="39" id="obr" showBusyCursor="true"/>
```

U těchto komponent však změna kurzoru bude dočasná, do doby než se obsah načte bude zobrazen „busy“ kurzor, poté se vzhled vrátí zpět do normálního stavu.

Další možností je použít třídu MouseCursor, která poskytuje základní možnosti pro změnu vzhledu kurzoru, jsou jimy ARROW (klasická šipka), BUTTON (ukazující prst ruky), HAND (ruka), IBEAM (svislítko pro psaní textu) a AUTO. Možnost AUTO mění vzhled kurzoru podle komponenty nad níž uživatel najel myší.

```
<s:Button x="10" y="10" label="Button" mouseOver="Mouse.cursor=
MouseCursor.HAND" mouseOut="Mouse.cursor= MouseCursor.AUTO"/>
```

Když uživatel najede myší nad tuto komponentu, kurzor se změní na vzhled HAND, pokud uživatel opustí prostor tlačítka, kurzor se nastaví na AUTO, tedy normální vzhled.

15.19 Vlastní vzhled

Kurzoru lze přiřadit vlastní vzhled, který můžeme načíst z obrázku (JPEG, GIF, PNG, SVG) nebo jako Sprite objekt či SWF soubor. Nově vytvořenému kurzoru je potřeba přiřadit zmíněný soubor jako hodnotu cursorClass typu Class. Další vlastnosti jsou volitelné, a jsou jimi priorita (priority), odsazení po ose X a Y (xOffset, yOffset). Kurzor s největší prioritou je aktuálně zobrazen, lze použít buď číselné vyjádření priority, nebo hodnoty HIGH, MEDIUM, LOW.

```
[Embed(source="tribalblack.gif")]
private var cursorSymbol:Class;
private var cursorID:Number = 0;

private function zmenKurzor():void {
      cursorID = CursorManager.setCursor(cursorSymbol);

}

private function vratKurzor():void {
      CursorManager.removeCursor(cursorID);
}
```

V kódu přiřazujeme obrázek jako vzhled kurzoru a nastavujeme jeho identifikátor na „0“, abychom na něj později mohli odkazovat. V metodě zmenKurzor() přiráváme pod zmíněným identifikátorem kurzor mezi ostatní vzhledy do správy kurzorů a nastavujeme ho jako aktuální. V metodě vratKurzor naopak z kurzorových vzhledů tento vzhled odebíráme, právě pomocí identifikátoru jako parametru ve vestavěné metodě removeCursor(). Nyní stačí pouze specifikovat kdy se která z metod bude volat.

Vytvoříme proto tlačítko s eventHandlery při najetí a vyjetí z plochy tlačítka.

```
<s:Button label="Button" mouseOver="zmenKurzor()"
mouseOut="vratKurzor()"/>
```

Měnění vzhledu kurzoru je tedy relativně snadná záležitost, avšak musíme si uvědomit její využitelnost. Vzhledy kurzoru jako „Busy“ a „Button“ určitě uživateli usnadňují a zpřehledňují práci s aplikací, Díky tomu, že lze vzhled nahradit svými obrázky a animacemi se otevírá prostor pro kompletně nový vzhled kurzoru v aplikaci, například ve hře může tvůrce využít zaměřovacího kříže atd. Určitě bych tedy doporučil využít alespoň základního nastavení kurzorů, protože uživatel se bude více cítit jako v desktopové aplikaci. Zdrojový kód aplikace viz příloha II.

15.20 Drag & Drop – ukázkový příklad drag_drop

Ve Flex aplikaci lze drag-and-drop aplikovat na většinu komponent. U některých komponent stačí podporu pouze povolit (např. List, TileList), u jiných lze přidat vlastní drag-and-drop systém. U drag-and-drop rozlišujeme tři základní prvky, objekt z kterého lze přetahovat (drag initiator), objekt který přetahujeme (drag source) a objekt, který příjme tažený objekt, tedy cíl tažení (drop target). Drag initiator a drop target mohou být zároveň jedním objektem. Po založení nového projektu vytvoříme dvě komponenty List, a povolíme u nich drag-and-drop.

```
<s:List dragEnabled="true" dataProvider="{new
ArrayCollection(['první', 'druhá', 'třetí'])}"></s:List>
<s:List dropEnabled="true" dataProvider="{new
ArrayCollection()}"></s:List>
```

První seznam povoluje pomocí atributu dragEnabled začátek tažení objektu, pak obsahuje kolekci řetězců, které se budou přetahovat. Druhý seznam povoluje vhození taženého objektu a obsahuje prázdný seznam, do kterého se tažený objekt vloží. Po spuštění aplikace si můžeme všimnout, že můžeme přetahovat najednou pouze jeden objekt, nebo toho, že objekty se do druhého seznamu kopírují. Pro povolení označování více objektů přidáme prvnímu seznamu následující atribut.

```
allowMultipleSelection="true"
```

Kopírování prvků mezi seznamy nemusí být vždy vyhovující, protože můžeme přesouvat nejen řetězce ale i komplexnější objekty nesoucí více informací. Pokud bychom chtěli prvky přesouvat namísto kopírování, je třeba prvnímu seznamu přidat

následující atribut.

```
dragMoveEnabled="true"
```

Pro předvedení možného využití drag-and-drop systému vytvoříme dvě komponenty DataGrid. První bude sloužit jako nabídka produktů, do druhého budeme vkládat produkty, které chceme koupit. Pro naplnění produktů vytvoříme komponentu ComboBox.

```
<s:ComboBox dataProvider="{new ArrayCollection(['kazety', 'cd',
'dvd'])}" change="naplnDG()"/>
```

O plnění produktů do DataGridu, respektive vybrání kolekce která slouží jako dataProvider, se postará EventHandler reagující na změnu vybraného řetězce, který zavolá metodu „naplnDG()“.

```
private function naplnDG():void {
      if (combo.selectedItem == "kazety"){
            vyber.dataProvider = seznamKazety;
      } else if (combo.selectedItem == "cd") {
            vyber.dataProvider = seznamCd;
      } else if (combo.selectedItem == "dvd") {
            vyber.dataProvider = seznamDvd;
      }
}
```

V kódu vidíme, že metoda naplní komponentu vyber, tedy DataGrid s produkty, seznamem který uživatel vybral v ComboBoxu. Seznam kazet, cd a dvd musíme samozřejmě také vytvořit. Například podle následujícího kódu.

```
private var seznamKazety:ArrayCollection = new
ArrayCollection([{nazev:'Black Sabbath - Heaven&Hell', cena:199,
nosic:'kazeta'}]);
```

Nyní zbývá komponentám DataGrid přiřadit příslušné vlastnosti dragEnabled, dropEnabled, allowMultipleSelection. V tomto případě bude lepší, když se přesouvané prvky budou kopírovat, namísto přesunování. Vlastnost dragMoveEnabled nemusíme uvádět, protože defaultní hodnota je false. Následně můžeme využít EventHandler který bude reagovat na dokončené přetažení objektu, abychom mohli například spočítat cenu všech vybraných produktů.

```
dragComplete="spocitejCenu()"
```

V metodě „spocitejCenu()“ se jednoduchým cyklem zkontroluje, jestli přetažený

prvek přibyl do seznamu s vybranými produkty (tzn. jestli uživatel přetáhl objekt do komponenty DataGrid), a následně se do proměnné přičte cena přetaženého produktu, tato proměnná se přiřadí jako text komponentě Label (textové jednořádkové pole, které nemůže uživatel měnit), kód metody je zde:

```
private function spocitejCenu():void {
      if (nakupPocet < seznamNakup.length){
            nakupPocet += 1;
            soucet += vyber.selectedItem.col2;
            cena.text = soucet.toString();
      }
}
```

Přesunování produktů do druhého DataGridu tedy funguje, teoreticky nyní můžeme přidat dvě komponenty Button na resetování vybraných produktů a druhý na odesílání objednávky.

15.21 Posluchače dragEnter a dragDrop

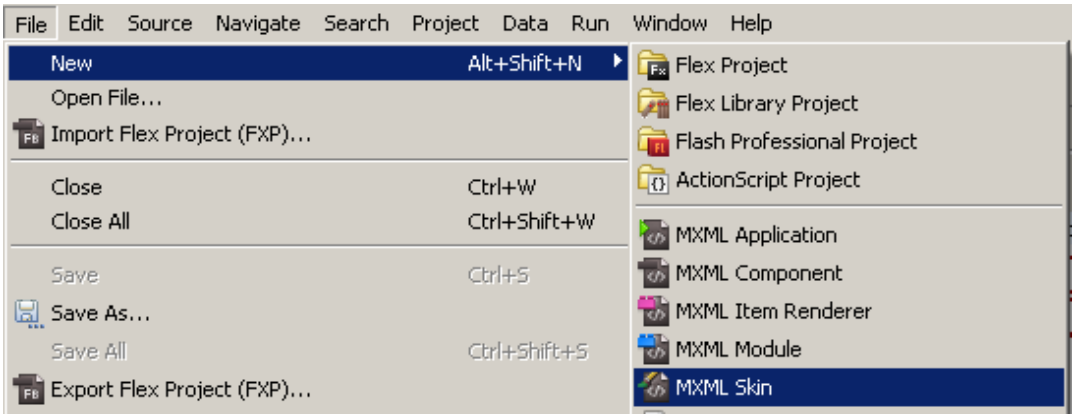
EventHandler dragEnter reaguje na přetažení objektu nad komponentu, která tento posluchač vlastní. Pomocí něj můžeme specifikovat přijímaný formát a povolit či zakázat příjem taženého objektu. [23]

```
private function dragEnterHandler(event:DragEvent):void {
      if (event.dragSource.hasFormat('color')) {
            var dropTarget:Canvas=Canvas(event.currentTarget);
            DragManager.acceptDragDrop(dropTarget);
      }
}
```

EventHandler dragDrop se spouští při puštění objektu nad komponentou, která umožňuje příjem pomocí systému drag-and-drop. Pomocí něj lze zpracovat vhozená data, podobně jako s výše zmíněným EventHandlerem dragComplete, avšak poskytuje výhodu, že samotné přidání dat do komponenty ještě neproběhlo. Data tedy můžeme přidat do jiné komponenty, nebo vymazat atd. Druhou výhodou oproti dragComplete je, že tento posluchač reaguje pouze na objekty vložené do cílového pole, dragComplete se spouští vždy, když skončí tažení objektu, bez ohledu na cíl, nebo úspěšnost tažení. Kód aplikace viz příloha IV.

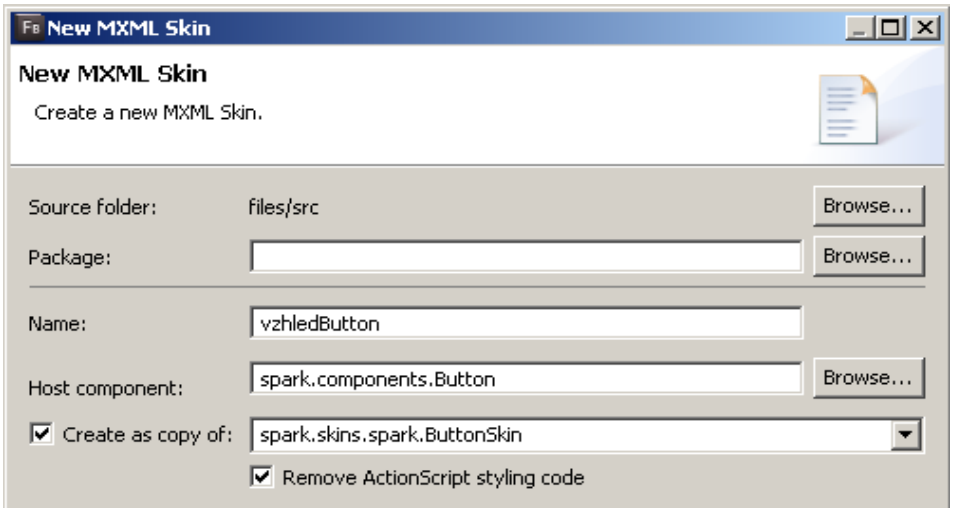
15.22 Skinování - ukázkový příklad Styl

Komponenty Flex aplikace můžeme vzhledově měnit několika způsoby, prvním je vytvoření třídy v aplikaci, kterou lze následně připojit jako skin komponenty. Pro ukázku vytvoříme v aplikaci novou třídu typu Skin, jako na obrázku 11.



Obrázek 11

Třída Skin musí mít jméno, abychom na ni mohli z aplikace odkazovat. Dalším nutným atributem je Host component, tedy typ komponent na které vzhled použijeme. Volitelná, avšak velmi nápomocná možnost je, že můžeme skin odvodit od existujícího vzhledu komponenty, nemusíme tedy začínat úplně od začátku, viz obrázek 12. Vytvoříme tedy třídu, která bude měnit vzhled tlačítka.



Obrázek 12

Na obrázku je vidět možnost „remove ActionScript styling code“, tato možnost je zatrhnuta kvůli zpřehlednění kódu. Po vytvoření třídy si všimneme několika důležitých částí v kódu. Mezi tagy „metadata“ je uvedena hostitelská komponenta.

```
<fx:Metadata>
        <![CDATA[
        [HostComponent("spark.components.Button")]
        ]]>
</fx:Metadata>
```

Jako další jsou zde uvedeny stavy komponenty, počet stavů a jejich názvy nesmíme měnit, protože komponenta by na ně nereagovala.

```
<s:states>
      <s:State name="up" />
      <s:State name="over" />
      <s:State name="down" />
      <s:State name="disabled" />
</s:states>
```

Zbytek kódu už je samotné nastavování vzhledu, rozdělené na osm částí. Všechny části okomentované podle funkce jako např. Shadow, Fill, Fill lowlight, Border, Text. Každá z těchto částí se skládá z grafických objektů, kterými jsou Rect (obdélník), Ellipse (ovál či kruh) a Path (křivka). Další částí je výplň těchto objektů, která může být SolidColor (jednobarevná výplň), LinearGradient a RadialGradient (lineární nebo kruhový barevný přechod) a BitmapFill (výplň obrázkem). Obdobně nastavujeme také orámování SolidColorStroke, LinearGradientStroke atd.

```
<s:Rect id="fill" left="1" right="1" top="1" bottom="1" radiusX="2">
        <s:fill>
            <s:LinearGradient rotation="90">
                <s:GradientEntry color="0xFFFFFF"
                                 color.over="0xBBBDBD"
                                 color.down="0xAAAAAA"
                                 alpha="0.85" />
                <s:GradientEntry color="0xD8D8D8"
                                 color.over="0x9FA0A1"
                                 color.down="0x929496"
                                 alpha="0.85" />
            </s:LinearGradient>
        </s:fill>
    </s:Rect>
```

V kódu je uvedena jedna z částí, konkrétněji část s výplní. Výplň je složena z jednoho obdélníku (s:Rect), který je z každé strany odsazen o „1“ kvůli orámování, zaoblení rohů má pro všechny rohy stejně tedy „2“. Barevně je obdélník vyplňen lineárním přechodem dvou barev otočeným o „90“ stupňů (po směru hod. ručiček). Oba barevné

vstupy přechodu mají nastaveny jinou barvu pro základní stav (up), stav over a down. Pro všechny stavy je však nastavena průhlednost stejná (alpha). V deklaraci Skinu si však můžeme všimnout nastavení průhlednosti pro stav disabled, který znamená, že na tlačítko nelze klikat.

```
<s:SparkSkin ... alpha.disabled="0.5">
```

Další vlastnosti, které můžeme ve vzhledech vidět jsou nastavování stavů. Pokud má komponenta více stavů, většinou chceme jiný vzhled pro každý, aby uživatel poznal v jakém stavu komponenta je, pokud tedy měníme některé vlastnosti pro některé stavy, následujícími příkazy můžeme stav „down“ vynechat, nebo naopak právě pro něj vzhled měnit.

```
excludeFrom="down"
includeIn="down"
```

Další vlastností, kterou můžeme vidět je hodnota ratio udávající u barevných přechodů v kterém procentě plochy bude zadaná barva přidána (v daném místě bude právě zadaná barva, teprve od udaného procenta začne barevný přechod). Hodnotu ratio uvádíme v rozmezí 0 a 1, tedy nula a sto procent.

```
<s:GradientEntry color="0x000000" alpha="0.07" ratio="0.3" />
```

V poslední části skinu je kód měnící vlastnosti textu, zde se nevyskytují žádné grafické objekty nebo barevné vyplně. Zde je uvedeno zarovnání textu horizontálně i vertikálně, odsazení od krajů, maximální počet řádek, velikost písma a jeho druh. Pokud neuvedeme velikost písma a jeho druh (nejsou automaticky generovány), defaultní bude velikost „11“ a písmo „Verdana“.

```
<s:Label id="labelDisplay"
      textAlign="center"
      verticalAlign="middle"
      maxDisplayedLines="1"
      horizontalCenter="0" verticalCenter="1"
      left="10" right="10" top="2" bottom="2"
      fontFamily="Courier">
</s:Label>
```

Napojení vzhledu na komponenty můžeme provést dvěma způsoby, prvním je udáním parametru skinClass u každé komponenty, které chceme přiřadit daný vzhled.

```
<s:Button label="tlacitko" skinClass="vzhledButton"/>
```

Druhým způsobem je přiřazení vzhledu komponentám v CSS bloku aplikace.

```
<fx:Style>
@namespace s "library://ns.adobe.com/flex/spark";
s|Button {
            skin-class: ClassReference('vzhledButton');
      }
s|Button.mujskin {
            skin-class: ClassReference('mujskin');
      }
</fx:Style>
```

V bloku Style definujeme vzhled pro všechna tlačítka (komponenty Button), tyto budou vypadat tak, jak jsme definovali v třídě „vzhledButton“. Tlačítka která budou vypadat podle třídy „mujskin“ musí obsahovat atribut styleName s hodnotou „mujskin“.

```
<s:Button label="tlacitko3" styleName="mujskin" x="0" y="58"/>
```

15.23 Vzhled specifikovaný CSS kódem

Druhým způsobem jak změnit vzhled komponent je definovat ho kompletně v bloku Style. Tento způsob umožňuje vynechat zakládání další třídy v aplikaci, potažmo dalšího souboru. V CSS stylech uvádíme prakticky stejné atributy a hodnoty, pouze syntakticky podle CSS namísto MXML. Společnost Adobe poskytuje na své doméně vynikající program využitelný pro změnu vzhledu komponent, v kterém stačí nastavit výběrem barev, pohybem posuvníků atd. vlastní vzhled a zkopírovat si CSS kód do své aplikace.[24] Tato aplikace je bohužel kompatibilní pouze s Flex2 a Flex3, využívá velmi odlišné syntaxe CSS. Pro využití vygenerovaných stylů bychom tedy museli naše aplikace kompilovat se zpětnou kompatibilitou minimálně pro Flex3. Ať už využijeme své nebo vygenerované stylování, je potřeba připojit ho k aplikaci, proto vytvoříme CSS soubor, který bude obsahovat styly. Tento soubor připojíme vytvořením dalšího bloku <fx:Style> v kterém bude uveden jako zdroj.

```
<fx:Style source="./styleButton.css"/>
```

Tento tag není párový, není k tomu žádný důvod, protože kromě napojení souboru nic neobsahuje. V CSS souboru můžeme použít třídy i identifikátory jako kdybychom stylovali HTML a podobně. V aplikaci tedy můžeme odkazovat na identifikátory obsažené v CSS souboru, nebo styly přiřadit všem komponentám určitého typu atd.

```
<s:Button label="tlacitko4" x="0" y="87" styleName="zeSouboru"/>
```

CSS soubor:

```
s|Button.zeSouboru {
      color: #0000FF;
      fontStyle: italic;
}
```

Kvůli měnění stylů nemusíme nutně zakládat externí CSS soubor, stejné identifikátory, atributy atd. můžeme uvést přímo v bloku Style v naší aplikaci. Tlačítka použitá v aplikaci mají atribut toolTip (zobrazí se při najetí nad komponentu), kde je u každého uvedeno jakým způsobem je stylováno.

15.24 Themes

Pokud bychom nechtěli přímo vytvářet zcela nový vzhled pro všechny komponenty, můžeme využít témat (themes). Flash Builder nabízí na výběr několik témat např. Spark(Default), Wireframe, Halo (kódový název pro Flex 3), a několik dalších. Pro použití tématu stačí otevřít nabídku Projects / Properties a v levém sloupci vybrat Flex Theme. Objeví se seznam dostupných témat s možností importace, či stažení dalších. Vybrané téma potvrdíme tlačítkem Ok a pro všechny komponenty u kterých jsme nenastavili jinak, se změní vzhled. Komponentám, kde jsme upravili například pouze barvu a styl textu, se téma nastaví také a následně se provede stylování textu, naopak komponenty, kterým jsme vytvořili vlastní stylovou třídu se nezmění s tématem. Využití témat v aplikaci se mi zdá výhodné, ušetříme práci se stylováním všech komponent a mezi nabídkou na Internetu jistě najdeme téma, které by nám mohlo vyhovovat. Zdrojový kód aplikace viz příloha VIII.

15.25 Práce se soubory – ukázkový příklad Files

Práce se soubory je v technologii flex do značné míry omezená kvůli bezpečnostnímu „sandboxu“ Flash Playeru. Toto bezpečnostní opatření nedovoluje například načítat soubory z jiné domény, než na které je aplikace umístěna, nebo přistupovat do souborového systému uživatele. S příchodem Flash Playeru verze 10 se některá bezpečnostní omezení změnila, tudíž lze s Flex aplikací udělat více. Následující metody jsou spouštěny komponentami Button.

15.26 Načítání

Soubory lze ve Flex aplikaci načíst několika způsoby, ať už přiřadit je přímo při kompilaci (viz. kapitola: Vkládání obrázku do aplikace), nebo udáním webové adresy souboru, necháním uživatele vybrat z možných adres souboru (v daném adresáři), či načtením souboru z uživatelova disku.

15.26.1 Z Internetu

Abychom mohli načíst soubor z Internetu, musíme dodržet bezpečnostní podmínky „sandboxu“, tedy soubor bude buďto ze stejné domény jako aplikace, nebo na adresované doméně bude „cross-domain.xml“ soubor, který povolí vstup z domény aplikace. Pokud budou tato pravidla splněna, využijeme proměnnou „req“ typu URLRequest a proměnnou „load“ typu URLLoader.

```
private var xemele:XML;
private var load:URLLoader = new URLLoader();

private function nacteni():void {
      var req:URLRequest = new URLRequest("./zdroje/soubor.xml");

      load.dataFormat = URLLoaderDataFormat.BINARY;
      load.load(req);
      cursorManager.setBusyCursor();
      load.addEventListener(Event.COMPLETE, nacteno);
}

private function nacteno(evtObj:Event):void {
      cursorManager.removeBusyCursor();
      xemele = XML(load.data);
      dg1.dataProvider =  xemele.game;
}
```

V kódu je vidět přiřazování Internetové adresy proměnné „req“, nemusíme ovšem přiřazovat pouze jednu adresu, můžeme nechat uživatele vybrat z komponenty daný soubor a jeho jméno potom zkonvertovat do formátu String a připojit ke zbytku adresy. Dále je vidět, že data budou načítána binárně, protože pokud bychom načetli formát XML jako String, nešlo by ho poté zkonvertovat zpět do XML. EventListener se spouští v okamžiku kdy se načetl celý soubor a volá metodu „nacteno()“, v které se data konvertují z binárních do XML objektu a přiřazují do proměnné „xemele“. Zbytek metody je plnění komponenty DataGrid (obdobné jako v kapitole Datové služby).

15.26.2 Z disku uživatele

Aby aplikace, potažmo Flash Player, mohla přistupovat k souborovému systému uživatele, je potřeba využívat Flash Player verze 10 a vyšší. Při načtení souboru z uživatelova disku bude použita proměnná „fr3“ typu FileReference.

```
private function nacteniOdUzivatele():void {

	fr3.browse();
	fr3.addEventListener(Event.SELECT, nacteniOdUzivatele2);
}

private function nacteniOdUzivatele2(evt:Event):void {
	fr3.addEventListener(Event.COMPLETE, nacteniOdUzivatele3);
	fr3.load();
}

private function nacteniOdUzivatele3(evt:Event):void {
	ta1.text = fr3.data.toString();
}
```

V kódu je vidět použití tří metod, ačkoliv se to může zdát nepřehledné, je to takřka nevyhnutelné. Druhé dvě metody jsou volány vždy po ukončení nějaké předchozí akce, kupříkladu aplikace nemůže začít načítat soubor v případě, že ho uživatel ještě nevybral. Metoda nacteniOdUzivatele() pomocí implementované funkce browse() otevírá okno, v kterém uživatele vybere soubor k načtení. Dále obsahuje posluchač, který se spustí v momentě, kdy uživatel vybere soubor a potvrdí jeho vybrání tlačítkem Ok, tento posluchač volá metodu nacteniOdUzivatele2. Zmíněná metoda se zabývá načtením vybraného souboru a kontrolou (posluchačem) zda-li je soubor už načten. Metoda nacteniOdUzivatele3 je volána po dokončení načítaní vybraného souboru a přiřazuje jeho obsah zkonvertovaný do formátu String komponentě ta1 (TextArea).

15.27 Ukládání

15.27.1 Na disk uživatele

Při stahování souboru je také třeba používat Flash Player verze 10 a vyšší, aby aplikace mohla ukládat na disk uživatele. Pro ukládání bude aplikace používat proměnnou „fr“ typu FileReference a proměnnou „ba“ typu ByteArray.

```
private function uloz():void {
      if(xemele != null) {
            var fr:FileReference = new FileReference();
            var ba:ByteArray = new ByteArray();
            ba.writeMultiByte(xemele, 'utf-8');
            fr.save(ba, 'xml-soubor.xml');
      }
}
```

Začátek metody „uloz()“ je jednoduchá kontrola podmínkou, zda proměnná „xemele“ (obsah, který chce uživatel stáhnout) není prázdná. Pokud je podmínka vpořádku, proměnná „xemele“ se načte kódováním utf-8 do proměnné „ba“. Následně se implementovanou funkcí save() otevře okno, v kterém uživatel vybere místo, kam se soubor uloží. Funkce save() obsahuje dva parametry, prvním je soubor v byte hodnotách, druhým je řetězec s doporučeným jménem souboru. Jméno souboru může uživatel změnit při výběru místa k uložení souboru.

15.27.2 Na server

Při ukládání na server je opět třeba dbát bezpečnostních pravidel „sandboxu“, pokud neukládáme na doménu shodnou s doménou aplikace. Ukládání souborů na server však s sebou nese další problém v podobě nutnosti využití server-side skriptu který samotné uložení provede. Tento skript by měl očekávat POST požadavek s elementy Content-Type, Content-Disposition a binární obsah souboru.

```
private var fr2:FileReference = new FileReference();
private var req2:URLRequest = new URLRequest("http://webove-
stranky.cz/");
private function nahraj():void {
      fr2.browse();
      fr2.addEventListener(Event.SELECT, nahraj2);
}
private function nahraj2(evt:Event):void {
      fr2.upload(req2, "Filedata");
}
```

V kódu je vidět, že aplikace opět používá proměnné typu FileReference (fr2) a typu URLRequest (req2). Proměnná „req2“ má při inicializaci přiřazenou webovou adresu, tedy místo, kam se soubor nahraje. Ve funkci nahraj() se implementovanou funkcí browse() otevírá okno, v kterém uživatel vybere soubor k nahrání na server. Funkce nahraj2() je volána posluchačem v okmažiku, kdy uživatel potvrdí vybrání souboru, po jejím zavolání započne uploadování na server, respektive se provede poslání dat server-side skriptu, který přijatý soubor uloží.

15.28 Zhodnocení práce se soubory

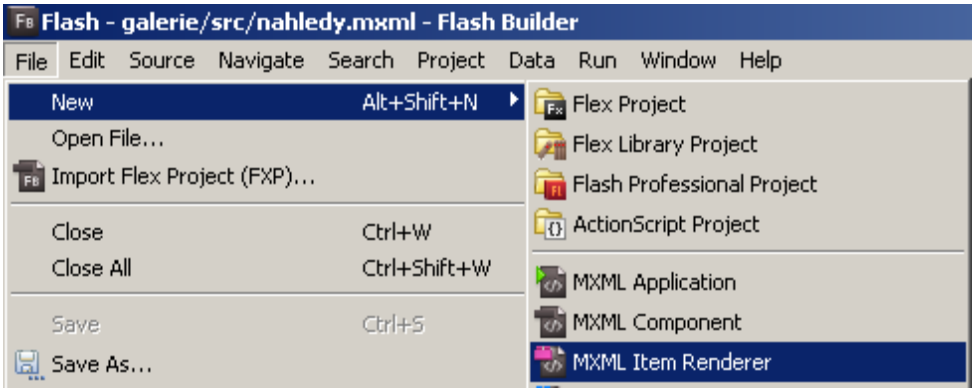
Je třeba říci, že technologie Flex je prvotně určena jako prezentační vrstva. Většinou tedy bude spolupracovat s dalšími technologiemi na straně serveru. Je však velmi nepříjemné, že aplikace neumožňuje samostatně ukládat alespoň základní data (např. registrace uživatelů a jejich vybraná barevná schémata, rozložení komponent v aplikaci a další), naopak Flex aplikace s Flash Playerem verze 10+, v které uživatel může svůj soubor načíst, pozměnit a uložit využití má (např. Aplikace stylu GoogleDocs). Zdrojový kód aplikace viz příloha VI.

15.29 Item Renderer – ukázkový příklad Galerie

V tomto ukázkovém příkladu vytvoříme dvě komponenty TitleWindow, jednu pro zobrazení náhledů obrázků, druhou na zobrazení velkého obrázku. Na zobrazování jednotlivých obrázku využijeme komponentu SWFLoader. Hlavní částí aplikace bude tvorba Item Rendereru, neboli vlastního zobrazení jednotlivých položek v komponentě List.

```
<mx:TitleWindow id="okno_nahledy" title="Náhledy">
      <mx:TileList change="setImg()" itemRenderer="nahledy"
columnCount="1"   dataProvider="{mainXML.obr}"
id="nahledy_foto"></mx:TileList>
</mx:TitleWindow>
```

V kódu lze vidět okno k zobrazení náhledů, jež obsahuje komponentu TileList. Tato komponenta bude mít jeden sloupec, zobrazí položky „obr“ z proměnné mainXML načtené z XML souboru označeného jako příloha XI. Při kliknutí na obrázek se zvětšený načte do druhého okna, což řeší metoda setImg(). Načítání souborů, události po uživatelově akci a plnění komponent jsem již popisoval v předchozích ukázkových příkladech. Důležitý je tedy atribut „itemRenderer“, který stanovuje vzhled jednotlivých položek, v tomto případě podle třídy „nahledy“. Item Renderer třídu můžeme snadno vytvořit přes nabídku Flash Builderu, jako na obrázku 13. Následně musíme kromě vyplnění jména stanovit komponentu podle které bude třída vypadat, pro naše potřeby to může být komponenta Group, Box, VGroup či VBox. Třída Item Rendereru místo počátečního tagu <application> bude začínat takovým, podle které komponenty je vytvořena.



Obrázek 13

```
<mx:VBox xmlns:mx="http://www.adobe.com/2006/mxml"
      horizontalAlign="center"
      verticalGap="0" borderStyle="none" height="150">
```

Vše co bude tato třída obsahovat, se následně zobrazí v každé z položek seznamu. Pokud v Komponentě uvedeme Item Renderer atribut, tak se z ní dědí proměnná „data“, která obsahuje položky dataProvideru. Tudíž v třídě Item Rendereru můžeme vzdáleně přistupovat k datům v komponentě kterou měníme. Každá položka seznamu bude obsahovat jméno obrázku, zmenšený obrázek a tlačítko na stáhnutí obrázku.

```
<mx:Label text="{data.@nazev}" textAlign="center" fontWeight="bold"/>
<mx:SWFLoader source="{data.@nahled}" id="pict_load" autoLoad="true"
                  scaleContent="true" height="100" width="100"/>
<mx:Button label="stáhnout" click="startDownload()"/>
```

Text zobrazující název obrázku je načten právě z děděné proměnné „data“, stejně tak jako adresa pro načtení zmenšeniny v komponentě SWFLoader. Třída Item Rendereru následně obsahuje metodu pro stažení obrázku po kliknutí na tlačítko. V této metodě ukládáme obrázky podobně jako v ukázkovém příkladě Files. Za zmínku však stojí získání jména souboru, jak lze vidět v následujícím kódu (úryvek metody), stanovujeme jméno souboru opět z děděné proměnné data.

```
downloadURL.url = "http://www.willow.wz.cz/test/"+data.@src;
```

Děděná proměnná „data“ obsahuje tedy XML objekt z načteného souboru a můžeme přistupovat k jednotlivým položkám načteného XML souboru. Zdrojový kód aplikace viz příloha VII.

16 Aplikace většího rozsahu – aplikace Brush

Tato aplikace umožňuje načíst obrázek v běžném formátu (JPG, GIF, PNG), následně obsahuje metody pro otáčení obrázku, roztahování a ořezávání. Dále také lze manipulovat s jednotlivými barvami, průhledností a jasem. Taktéž lze přidávat a ubírat filtry, které mění kontrast obrázku, nebo mění obrázek na černobílý, eventuelně sépiový. Na závěr lze obrázek uložit ve formátu PNG. Vzhled aplikace je sestaven z ovládací komponenty Accordion, která obsahuje tlačítka, posuvníky a pole pro čísla, jimiž lze obrázek měnit. Pro samotné zobrazení obrázku je v aplikaci komponenta SpriteVisualComponent, do níž se načítá obrázek v komponentě Sprite. Další částí aplikace je pop-up okno, které se aktivuje v případě zadání nesprávných hodnot (např. Oříznutí obrázku o větší rozměr než sám má).

16.1 Načtení souboru

Soubor se načítá obdobně jako v ukázkovém příkladě „files“, tedy pomocí proměnných typu FileReference a Loader. Aby uživatel nevybral soubor jiného typu než se může načíst do proměnných, je zde proměnná typu FileFilter, kterou lze vidět v následujícím kódu včetně použití.

```
var fileFilter:FileFilter = new FileFilter("Images: (*.jpeg, *.jpg,
*.gif,      *.png)", "*.jpeg; *.jpg; *.gif; *.png");
fileReference.browse([fileFilter]);
```

Po vybrání a načtení souboru je pomocí EventHandleru volána metoda „loadnutoS“, která soubor obsažený v proměnné „loader“ typu Loader načte do proměnné „bd“ typu BitmapData a následně vykreslí do proměnné „kontejner“ typu Sprite. Tato proměnná se přidává do komponenty „SVE“ typu SpriteVisualComponent.

```
private function loadnutoS(evt:Event):void {
      bd = Bitmap(loader.content).bitmapData;
      kontejner.graphics.beginBitmapFill(bd);
      kontejner.graphics.drawRect(0, 0, bd.width, bd.height);
      kontejner.graphics.endFill();
      SVE.addChild(kontejner);
}
```

16.2 Otáčení a Roztahování

Otáčení a roztahování obrázku provádím pomocí změn transformační matrice. Při otáčení posunuji matrici o polovinu šířky a výšky, aby se obrázek otáčel kolem svého středu, následně matrici vracím tak, aby opticky obrázek začínal na stejném místě. Při roztahování není potřeba posunovat, neboť počáteční bod obrázku se nemění.

```
private function rotate():void {
      otoceni += sliderO.value;
      m = kontejner.transform.matrix;
      m.translate(-kontejner.width/2,-kontejner.height/2);
      m.rotate(sliderO.value*(Math.PI/180));

      if (sliderO.value == 90 || sliderO.value == 270) {
            m.translate(kontejner.height/2, kontejner.width/2);
                        } else {
                  m.translate(kontejner.width/2,kontejner.height/2);
                        }
      kontejner.transform.matrix = m;
      vyreseniOtoceni();
}
```

Metoda pro otočení využívá globální proměnné „m“ typu Matrix, do které načítá matrici obrázku, mění ji a následně ji přiřadí jako matrici obrázku. Hlavní částí metody je samotná rotace, která se určuje v radiánech, proto se hodnota posuvníku (sliderO.value) násobí π/180. V metodě lze vidět další globální proměnnou „otoceni“, do které se ukládá aktuální otočení pro ostatní metody, s tím také souvisí volání metody „vyreseniOtoceni()“, která z této proměnné odebírá hodnotu 360, pokud je přesažena.

```
private function scale():void {
      if (sliderSX.value == 0 || sliderSY.value == 0) {
            errorWindow(" Chybně zadaná hodnota roztažení. \n Není
možné použít           hodnotu '0'");
      }else {
            m = kontejner.transform.matrix;
            m.scale(sliderSX.value, sliderSY.value);
            kontejner.transform.matrix = m;
      }
sx = sx*sliderSX.value;
sy = sy*sliderSY.value;
sliderSX.value = 1;
sliderSY.value = 1;
}
```

Metoda pro roztahování má ošetřenou možnost, že uživatel zadá na posuvníku hodnotu „0“, v tomto případě se uživateli ukáže chybové pop-up okno které bude obsahovat řetězec jež je v kódu červeně. Samotné roztahování se provádí příkazem

„scale(*po ose X*, *po ose Y*)“ aplikovaným opět na globální proměnnou „m“ typu Matrix, které se nejprve přiřadí matrice obrázku, poté se změní a přiřadí se zpět obrázku. Následně se v metodě uloží aktuální hodnota roztažení v globálních proměnných „sx“ a „sy“, a vyrovnají se posuvníky na hodnotu „1“.

16.3 Ořez

Metoda ořezávání se provádí jinak než dvě předešlé, celý obrázek se načte do proměnné typu BitmapData, vytvoří se druhá taková proměnná, která bude menší o tolik, kolik chceme oříznout a následně se do ní obrázek zkopíruje. Poté se vykreslí zpět do proměnné „kontejner“ typu Sprite.

```
----Výřez metody scale
var poX:int = Number(tiS.text);
var poY:int = Number(tiV.text);
var rect:Rectangle = new Rectangle(0,0, kontejner.width,
kontejner.height);
var bd2:BitmapData = new BitmapData(kontejner.width, kontejner.height,
true);
var p:Point = new Point(0,0);

rect.x = poY;
rect.height -= poX;
rect.width -= poY;
var bd3:BitmapData = new BitmapData(rect.width, rect.height, true);
bd2.draw(kontejner);
bd3.copyPixels(bd2, rect, p);

kontejner.graphics.clear();
kontejner.graphics.beginBitmapFill(bd3);
kontejner.graphics.drawRect(0, 0, bd3.width, bd3.height);
kontejner.graphics.endFill();
```

V kódu se načítají čísla z vstupních políček do proměnných „poX“ a „poY“. Následně se vytvoří obdélník „rect“ o velikosti obrázku a proměnná „bd2“ typu BitmapData, také o velikosti obrázku. Rozměry obdélníku se následně zmenší o hodnoty, které zadal uživatel. Podle změněné velikosti obdélníku se poté vytvoří další proměnná „bd3“ typu BitmapData. Předchozími kroky je provedena příprava k samotnému ořezu. Do proměnné „bd2“ se vykreslí obrázek a pomocí příkazu „copyPixels(*objekt, velikost, počáteční bod*)“ se obrázek bez oříznutých částí překopíruje do proměnné „bd3“. Zbytek metody je pouze vykreslení oříznutého obrázku do proměnné „kontejner“.

16.4 Změna barevných hodnot a jasu

Pro změnu barev se využívá transformace ColorTransform. Aby tedy šlo měnit hodnoty barev, vytvoříme proměnnou typu ColorTransform u které lze měnit barevné multiplikátory a vyvážení.

```
private function color():void {
      var red:Number = sliderCR.value;
      var green:Number = sliderCG.value;
      var blue:Number = sliderCB.value;
      var alpha:Number = sliderCA.value;
      var colorT:ColorTransform = kontejner.transform.colorTransform;
      colorT.redMultiplier = red;
      colorT.greenMultiplier = green;
      colorT.blueMultiplier = blue;
      colorT.alphaMultiplier = alpha;
      kontejner.transform.colorTransform = colorT;
}
```

Nejprve se v metodě načtou do proměnných hodnoty z posuvníků jednotlivých barev. Následně se do proměnné „colorT“, v které se budou barvy měnit, načte barevná transformace obrázku. Hodnoty z posuvníků se přiřadí jako multiplikátory jednotlivých barev a průhlednosti a následně se změněná proměnná „colorT“ přiřadí jako barevná transformace obrázku.

```
private function brightness():void {
      var rozdil:Number = sliderB.value - jasRozdil;
      jasRozdil += rozdil;

      sliderCR.value += rozdil;
      sliderCG.value += rozdil;
      sliderCB.value += rozdil;
      color();
}
```

Metoda měnící jas obrázku přidává či ubírá hodnotu, v proměnné „rozdil“, všem posuvníkům, měnícím barvy obrázku a následně volá metodu „color()“ která mění barvy. Přidáním všech barev se zvětší jas obrázku, odebráním se ubere.

16.5 Filtry – Černobílý, Sépiový, Kontrast

Obrázku můžeme přidat barevný filtr, který bude měnit jednotlivé barvy podle barevné matrice daného filtru. Všechny tři metody jsou postavené na stejném principu, mění se u nich hlavně matrice filtru a způsob aplikace filtru.

```
private function contrast():void {
      var filtry:Array = kontejner.filters;
      var a:Number = ( sliderK.value * 0.01 + 1 )
      var b:Number = 0x80 * ( 1 - a );
      var matrixC:Array = [
                              a   ,   0   ,   0   ,   0   ,   b,
                              0   ,   a   ,   0   ,   0   ,   b,
                              0   ,   0   ,   a   ,   0   ,   b,
                              0   ,   0   ,   0   ,   1   ,   0
                           ];
      var contrastFilter:ColorMatrixFilter = new
ColorMatrixFilter(matrixC);
      kontejner.filters = [contrastFilter];
}
```

V metodách vždy vytovříme seznam (Array), do kterého přičteme seznam filtru obrázku, následně vytvoříme proměnnou typu ColorMatrixFilter, která bude obsahovat matrici filtru a tuto přičteme do seznamu filtrů, ten na závěr přiřadíme jako seznam filtrů obrázku. V jednotlivých metodách se filtry přiřazují pokaždé jinak, například kontrast nemužeme stále přidávat, protože obrázek by více a více nabýval na kontrastu.

U černobílého a sépiového filtru je vytvořeno přepínání, které přidává či odebírá filtr (je bezvýznamné použít oba najednou). Ostatní filtry také nejsou nastavitelné jako kontrast, který můžeme měnit, stanovují se u nich přesné a vždy stejné hodnoty filtru, neovlivněné posuvníkem či jinou komponentou. Pro ukázku uvádím absolutně definované hodnoty pro černobílý filtr.

```
----matice černobílého filtru
[0.3  , 0.59 , 0.11 , 0    ,
 0    , 0.3  , 0.59 , 0.11 ,
 0    , 0    , 0.3  , 0.59 ,
 0.11 , 0    , 0    , 0    ,
 0    , 0    , 1    , 0    ]
```

16.6 Ukládání obrázku

Pro uložení obrázku se využijí proměnné typu FileReference, ByteArray a PNGEncoder a JPGEncoder. Abychom mohli obrázek převést do PNG či JPG formátu, respektive využít dekodéry musíme v skriptové části aplikace importovat PNGEncoder z knihovny mx.graphics.codec.

```
private function save():void {
      var kodovany:ByteArray;
      var jmenoSouboru:String = "untitled.png";
      var pngkoder:mx.graphics.codec.PNGEncoder = new
                        mx.graphics.codec.PNGEncoder;
      var bd5:BitmapData = new BitmapData(kontejner.width,
kontejner.height,             true);
      bd5.draw(kontejner, kontejner.transform.matrix,
                        kontejner.transform.colorTransform,
kontejner.blendMode);
      kodovany = pngkoder.encode(bd5);
      var saveFile:FileReference = new FileReference();
      saveFile.addEventListener(Event.SELECT, sele);
      saveFile.addEventListener(Event.COMPLETE, compl);
      saveFile.save(kodovany, jmenoSouboru);
}
```

Obrázek se v metodě „save()“ načte do proměnné „bd5“ typu BitmapData včetně tranformační matrice a barevné transformace. Následně se proměnná „bd5“ pomocí encoderu převede na formát PNG, nebo JPG do proměnné „kodovany“ typu ByteArray. Pomocí proměnné „saveFile“ typu FileReference se uživateli otevře okno pro uložení obrázku v proměnné „kodovany“ a defaultním názvem z proměnné „jmenoSouboru“.

V metodě jsou přidány také dva posluchače, první se spouští při uživatelově potvrzení jména a lokace souboru a volá metodu „sele()“, druhý se spouští při dokončení ukládání a volá metodu „compl()“. Tyto posluchače jsou zde, aby se v aplikaci změnil kurzor na *zaneprázdněný* a zobrazila se komponenta Label s textem, který uživateli oznamuje ukládání souboru, po dokončení akce se komponenta i kurzor vrátí do původního stavu. Zdrojový kód aplikace viz příloha X.

17 Testování podpory prohlížečů pro aplikace Flex

Před samotným testováním je nutné, aby byl v prohlížečích nainstalován a povolen Flash Player plug-in, který slouží jako běhové prostředí pro spouštění Flex aplikací. Pokud se aplikace nespustí kvůli chybějícímu plug-inu je nabídnuta instalace Flash Playeru. V každém z prohlížečů lze zjistit jinak, zda je Flash Player plug-in nainstalován a povolen.

17.1 Testované prohlížeče

17.1.1 Internet Explorer 8

V prohlížeči IE8 se používá plug-in Shockwave Flash, lze najít v Nástroje / Spravovat doplňky (je potřeba vybrat z combo-boxu možnost „všechny doplňky“. K plug-inům lze zobrazit podrobnější informace a také stránky pro které mají být použity (defaultní hodnota je * [pro všechny]). Pokud ve výpisu plug-in nenajdeme, nebo je zakázán, je potřeba ho stáhnout, respektive povolit.

17.1.2 Firefox 4

V prohlížeči FF4 se používá plug-in Shockwave Flash. Po klepnutí na Add-ons v Tools z horní lišty menu se objeví nová záložka pro spravování add-onů. Po kliknutí na možnost Plugins se zobrazí seznam všech nainstalovaných plug-inů. Opět pokud plug-in nenajdeme nebo je vypnutý, je třeba ho stáhnout nebo povolit.

17.1.3 Opera 11

V prohlížeči Opera se používá plug-in Shockwave Flash. Zobrazit lze v Tools / Advanced / Plug-Ins. Po vybrání této možnosti se otevře v nové záložce seznam nainstalovaných plug-inů. K plug-inům lze zobrazit podrobnější informace, ve smyslu kde je uložen, jaké typy souborů spouští a jaké verze daných aplikací podporuje. Pokud zde Shockwave Flash plug-in nenajdeme, nebo je zakázaný, je potřeba ho stáhnout či povolit.

17.1.4 Chrome 10

V prohlížeči Chrome se používá plug-in Shockwave Flash. Nejjednoduší cesta jak zobrazit nainstalované plug-iny je v nové záložce do adresního řádku vepsat „about:plugins“. K plug-inům lze zobrazit podrobnější informace ve smyslu kde jsou

uloženy a jaké typy souborů podporují. Prohlížeč Chrome při instalaci automaticky nainstaluje také plug-in Shockwave a v případě vydávání novějších verzí ho také automaticky aktualizuje. Pokud bude v zobrazeném seznamu plug-in zakázán, je potřeba ho povolit.

17.2 Metoda testování

Před samotným testováním jsem stáhl nejnovější verze prohlížečů, bohužel s vyjímkou prohlížeče Internet Explorer, kterého verze 9 vyžaduje operační systém Vista a mladší. Použil jsem tedy nejmladší verzi kompatibilní s operačním systémem Windows XP, verzi 8. Následně jsem postupně ve všech čtyřech prohlížečích spustil všechny ukázkové příklady z disku počítače. Poté jsem ukázkové příklady nahrál na server a spustil jsem je opět ve všech čtyřech prohlížečích z Internetu. Lokálně i z Internetu jsem u ukázkových příkladů vyzkoušel jejich funkčnost a správné zobrazení.

17.3 Výsledky testování

17.3.1 Funkčnost

Ve všech čtyřech prohlížečích se všechny ukázkové příklady podařilo spustit. Všechny aplikace prováděly funkce korektně a byly zobrazeny dle očekávání. Jedinou vyjímkou byla aplikace „datagrid_webservice“ (aplikace načítající údaje z RSS čtečky na stránce zive.cz do komponenty Datagrid) spouštěná z Internetu, která sice spustit šla, avšak data se do komponenty Datagrid nenačetla kvůli bezpečnostnímu omezení Flash Playeru „sandbox“ (tento problém je podrobněji popsán v kapitole: Omezení datových služeb).

17.3.2 Zobrazení

Při testování příkladů jsem vzal v potaz, že aplikace mohou být součástí HTML dokumentu, což se dá simulovat použitím vygenerovaných souborů ve formátu .html. Při zobrazení těchto souborů místo samotných aplikací může nastat problém se zobrazením při zvětšování či zmenšování stránky (zoom). Tyto problémy se netýkají aplikací, které mají rozměry nastaveny procentuálně, z ukázkových příkladů to je aplikace Datagrid_webservice. Aplikace s absolutními rozměry se při oddalování v prohlížečích Internet Explorer, Opera a Chrome vizuálně přibližují k levému hornímu rohu, což je projev jejich samotného zvětšování (aplikace má definovány minimální

rozměry). Naopak při přibližování aplikace „přetékají“ z jim vymezeného prostoru, tudíž nejsou částečně vidět. V prohlížeči Firefox se aplikace při zoomování vizuálně nemění a zůstávají celé zobrazeny. V prohlížeči Opera se při přibližování a oddalování aplikací s procentuálně definovanými rozměry po každé změně přiblížení přepočítá a zaokrouhlí jejich velikost na pixely a proto aplikace drobně mění rozměry. Tento problém však není příliš markantní, protože aplikace má vždy procentuálně správné rozměry a přechody mezi nimi nejsou pro chod aplikace ani pro uživatele zásadní.

17.3.3 JavaScript

Při spouštění Flex (potažmo .swf souborů) v HTML dokumentu je často využito JavaScriptu. Využití JavaScriptu můžeme vidět i v kódu Flash Builderem vygenerovaného HTML dokumentu, kde je využit k zprovoznění historie prohlížení, aktualizace Flash Playeru a využití správných tagů pro konkrétní prohlížeč. Pokud zakážeme zpracování JavaScriptu v prohlížečích Internet Explorer a Firefox, aplikace se načtou a fungují i spouštěné HTML souborem i samotné (SWF). V případě zakázání JavaScriptu v prohlížeči Opera se aplikace spouštěné souborem HTML nenačtou, avšak samotné ano. Pokud zakážeme provádění JavaScriptu v prohlížeči Chrome, aplikace spouštěné souborem HTML i samostatně načítané se nespustí vůbec.

17.4 Zhodnocení testu

Z testování vyplývá, že po stránce funkčnosti obstály všechny testované prohlížeče. V zobrazování aplikací nejvíce vyniká prohlížeč Firefox, který zobrazí vždy celou aplikaci bez ohledu na přiblížení.

Dalším aspektem je určitě snadnost instalace běhového prostředí, kde vyniká prohlížeč Chrome, který Flash Player plug-in stáhne automaticky po instalaci. Výhodou prohlížeče Internet Explorer může být možnost použít Flash Player plug-in pouze na některé stránky.

Pokud z jakéhokoliv důvodu máme v prohlížeči zakázaný JavaScript, vyřazuje to spouštění Flex aplikací v prohlížeči Chrome a ve většině reálných případů také v prohlížeči Opera (vyjma samostatných aplikací). Jako menší, ale také možná užitečné mohou být podrobné informace o plug-inech, které poskytují více či méně prohlížeče Internet Explorer, Opera a Chrome.

18 Závěr

V práci jsem se pokusil definovat pojem RIA aplikace a úspěšně se mi podařilo vymezit interval aplikací, které můžeme označit jako RIA. Taktéž jsem popsal výhody a nevýhody RIA aplikací, potažmo technologií zabývajících se jimi. Popsal jsem charakteristiky technologie Adobe Flex a několika konkurenčních technologií, z těchto charakteristik jsem následně vytvořil srovnání zmíněných technologií. Technologie Adobe Flex vyšla z tohoto srovnání jako jedna z nejlepších, avšak rozhodně je při výběru třeba zohlednit nároky vývojáře. Technologie Adobe Flex má svá omezení, kvůli kterým je v určitých případech moudré zvolit si alternativní technologii.

Pomocí software Adobe Flash Builder, jež je mimo jiné vývojovým prostředím pro technologii Adobe Flex, jsem vytvořil sérii ukázkových příkladů na které ukazuji způsob tvorby aplikací v této technologii. Ukázkové příklady jsem zaměřil na, podle mého úsudku, hlavní aspekty tvorby aplikací. Série ukázkových příkladů obsahuje 9 příkladů od jednodušších po obsahově složitější. Pomocí devíti příkladů jsem samozřejmě nemohl obsáhnout kompletní možnosti technologie Adobe Flex, avšak myslím si, že jsem jimi přiblížil hlavní rysy tvorby RIA aplikací.

Další součástí práce je aplikace většího rozsahu, kterou jsem taktéž vytvořil v software Adobe Flash Builder a využívá jak postupů ukázaných v sérii příkladů, tak nových technik, které jsem v práci taktéž popsal a vysvětlil. Tato aplikace, kromě přínosu nově použitých technik, se také vyznačuje svou použitelností pro uživatele. V aplikaci je možné načíst uživatelův obrázek, různými způsoby ho obměnit po stránce barev či velikosti a natočení a následně uložit změněný, aplikace tedy může být vhodná například pro úpravu fotografií.

Následně jsem v práci testoval podporu prohlížečů, jako testované prohlížeče jsem vybral čtyři nejrozšířenější. Prohlížeče jsem testoval pomocí série ukázkových příkladů a hodnotil jsem několik faktorů, od funkčnosti, přes korektní zobrazení až po instalaci a nastavení zásuvných modulů sloužících jako běhové prostředí pro technologii Adobe Flex. Testování prohlížečů zjistilo, že prohlížeče jsou na technologii Adobe Flex připraveny výtečně po stránce funkčnosti, a s rozdílnými výsledky v ostatních hodnocených kritériích.

19 Seznam použité literatury a zdrojů

[1] SYMBIO Digital, s. r. o. Rich Internet Application [online]. 1999–2011 [cit. 2011-01-19]. Dostupné z WWW: <http://www.symbio.cz/slovnik/rich-internet-application.html>.

[2] Interval [online]. 2008 [cit. 2011-04-25]. Rich Internet Applications v roce 2008. Dostupné z WWW: <http://interval.cz/clanky/rich-internet-applications-v-roce-2008/>.

[3] ALLAIRE, Jeremy. Macromedia Flash MX—A next-generation rich client [online]. 2002 [cit. 2011-04-25]. Dostupné z WWW: <http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf>.

[4] NOVÁK, Jiří. Analýza RIA metod a technik [online]. Brno, 2009. 128 s. Diplomová práce. Masarykova Univerzita.

[5] Digital Point [online]. 2005 [cit. 2011-04-25]. SEO of flash / swf files. Dostupné z WWW: <http://forums.digitalpoint.com/showthread.php?t=22338>.

[6] BRYAN , FlexTutorial [online]. Friday, July 31, 2009 [cit. 2011-04-25]. 7 Difference Between RIA and Traditional Web Application. Dostupné z WWW: <http://flextutorial.org/2009/07/31/7-difference-between-ria-and-traditional-web-application/>.

[7] Adobe [online]. 07-14-2009 [cit. 2011-04-25]. Adobe Flex. Dostupné z WWW: <http://www.adobe.com/products/flex/>.

[8] BERNARD, Borek. Interval [online]. 04. 06. 2008 [cit. 2011-04-25]. Adobe Flex - co je a co není. Dostupné z WWW: <http://interval.cz/clanky/adobe-flex-co-je-a-co-neni/>.

[9] Flex.org : Tour de Flex [online]. 07-14-2009 [cit. 2011-04-25]. Dostupné z WWW: <http://flex.org/tour>.

[10] Microsoft MSDN [online]. 2/24/2011 [cit. 2011-04-25]. Silverlight Architecture. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/bb404713(VS.95).aspx>.

[11] Laszlo Systems: Technology White Paper [online]. [s.l.] : [s.n.], November 2006 [cit. 2011-04-25]. Dostupné z WWW: <www.openlaszlo.org/whitepaper/LaszloWhitePaper.pdf>.

[12] Snizekweb [online]. 13.9.2005 [cit. 2011-04-25]. AJAX – kde jsou hranice?. Dostupné z WWW: <http://www.snizekweb.cz/clanky/ajax-kde-jsou-hranice/>.

[13] Tvorba-webu [online]. 2004 [cit. 2011-04-25]. DOM: Document Object Model. Dostupné z WWW: <http://www.tvorba-webu.cz/dom/>.

[14] Curl [online]. 2009 [cit. 2011-04-25]. Company News. Dostupné z WWW: <http://www.curl.com/company_news050709.php>.

[15] Curl [online]. 2008 [cit. 2011-04-25]. Enterprise RIA. Dostupné z WWW: <http://www.curl.com/products_enterprise_ria.php>.

[16] JavaFX [online]. 2010 [cit. 2011-04-25]. Develop Expressive Content with the JavaFX Platform. Dostupné z WWW: <http://javafx.com/about/overview/index.jsp>.

[17] NOVÁK, Milan. Technologie RIA - Flash, Silverlight. Voxcafe [online]. 17.06.2010, [cit. 2011-04-25]. Dostupný z WWW: <http://www.voxcafe.cz/clanky/vyvoj-aplikaci/technologie-ria-flash-silverlight.html>.

[18] Longtail [online]. 2011 [cit. 2011-04-25]. Supported Video and Audio Formats. Dostupné z WWW: <http://www.longtailvideo.com/support/jw-player/jw-player-for-flash-v5/12539/supported-video-and-audio-formats>.

[19] Adobe [online]. [cit. 2011-04-25]. Flash Player Settings Manager. Dostupné z WWW: <http://www.macromedia.com/support/documentation/en/flashplayer/help/settings_manager04.html#117502>.

[20] Adobe [online]. 2011 [cit. 2011-04-25]. Cross-domain policy file specification. Dostupné z WWW: <http://www.adobe.com/devnet/articles/crossdomain_policy_file_spec.html>.

[21] Adobe [online]. 2010 [cit. 2011-04-25]. Effects in Adobe Flex 4. Dostupné z WWW: <http://www.adobe.com/devnet/flex/articles/flex4_effects_pt1.html>.

[22] Adobe [online]. 2010 [cit. 2011-04-25]. Using Spark easing classes. Dostupné z WWW: <http://help.adobe.com/en_US/flex/using/WS91E85D63-A025-4c46-B758-A275D4D3B3FC.html>.

[23] Adobe : Help [online]. 2010 [cit. 2011-04-25]. Manually adding drag-and-drop support. Dostupné z WWW: <http://help.adobe.com/en_US/flex/using/WS2db454920e96a9e51e63e3d11c0bf69084-7cfe.html>.

[24]Adobe : Examples [online]. 2006 [cit. 2011-04-25]. Flex 2 Style Explorer. Dostupné z WWW: <http://examples.adobe.com/flex2/consulting/styleexplorer/Flex2StyleExplorer.html>.

[25] Adobe [online]. 2011 [cit. 2011-04-25]. Adobe Flash Builder 4. Dostupné z WWW: <http://www.adobe.com/products/flash-builder.html>.

[26] Sptechnolab [online]. January 4, 2011 [cit. 2011-04-27]. Flex Architecture. Dostupné z WWW: <http://blog.sptechnolab.com/2011/01/04/flash/flex-architecture/>.

[27] Microsoft : MSDN [online]. 2011 [cit. 2011-04-27]. Silverlight Architecture. Dostupné z WWW: <http://msdn.microsoft.com/en-s/library/bb404713(v=vs.96).aspx>.

[28] Adaptive path [online]. February 18, 2005 [cit. 2011-04-27]. Ajax: A New Approach to Web Applications. Dostupné z WWW: <http://www.adaptivepath.com/ideas/e000385>.

20 Seznam Příloh

Všechny přílohy jsou dostupné na přiloženém CD. Přílohy ve formě aplikací vytvořených v software Flash Builder jsou na CD včetně zdrojových kódů.

I. Ukázkový příklad Window
II. Ukázkový příklad Cursor
III. Ukázkový příklad Datagrid_webservice
IV. Ukázkový příklad Drag_drop
V. Ukázkový příklad Effect
VI. Ukázkový příklad Files
VII. Ukázkový příklad Galerie
VIII. Ukázkový příklad Styly
IX. Ukázkový příklad Zalozky
X. Aplikace Brush
XI. Soubor data.xml
XII. Naskenované zadání bakalářské práce – část 1
XIII. Naskenované zadání bakalářské práce – část 2
XIV. Naskenované zadání bakalářské práce – část 3
XIVI. Bakalářská práce ve formátu pdf