VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

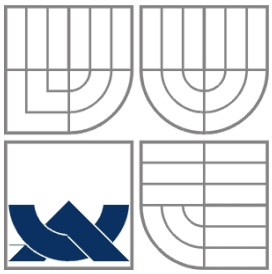
DYNAMICKÉ VYTVÁŘENÍ GRAFICKÝCH UŽIVATELSKÝCH ROZHRANÍ PRO INTERNETOVÉ APLIKACE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

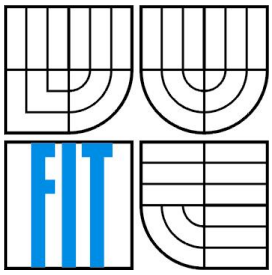
AUTOR PRÁCE
AUTHOR

LADISLAV NAVRÁTIL

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DYNAMICKÉ VYTVÁŘENÍ GRAFICKÝCH UŽIVATELSKÝCH ROZHRANÍ PRO INTERNETOVÉ APLIKACE

DYNAMIC CREATION OF GRAPHICS USER INTERFACE FOR INTERNET APPLICATIONS.

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LADISLAV NAVRÁTIL
AUTHOR

VEDOUCÍ PRÁCE — ING. RUDOLF KAJAN
SUPERVISOR

BRNO 2011

Abstrakt

Hlavním námětem této bakalářské práce jsou Rich Internet Applications (RIA), webové aplikace simulující desktopové programy. V práci je popsán komplexní úvod do dané problematiky, možnosti a rozdělení těchto aplikací. Pozornost je převážně věnována nástrojům a vývojovým prostředím, které se při tvorbě RIA aplikací využívají a dále vlastnímu konceptu vývojového prostředí. Zkoumané vlastnosti daného konceptu jsou uplatněny v návrhu a implementaci aplikace, která slouží k dynamickému vytváření grafických uživatelských rozhraní.

Abstract

The main topic of this thesis are Rich Internet Applications (RIA), web applications simulating desktop programs. The thesis contains complex introduction into the field, its possibilities and division. General focus is set on tools and frameworks used to create RIA applications and on architecture of author's own RIA framework. Attributes analyzed in the thesis are applied in the concept and implementation of an application that offers dynamic creation of GUI.

Klíčová slova

rich internet applications, RIA, grafické uživatelské rozhraní, GUI, Flash Builder

Keywords

rich internet applications, RIA, graphicaluser interface, GUI, Flash Builder

Citace

Navrátil Ladislav: Dynamické vytváření grafických uživatelských rozhraní pro internetové aplikace, bakalářská práce, Brno, FIT VUT v Brně, 2011

Dynamické Vytváření Grafických Uživatelských Rozhraní Pro Internetové Aplikace

Prohlášení

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

……
Ladislav Navrátil
18. 5. 2011

Poděkování

Tímto bych chtěl poděkovat vedoucímu své práce Ing. Rudolfu Kajanovi, za včasnou a bezproblémovou komunikaci a za veškeré nápady, poznámky a připomínky, které pomohly udělat toto dílo lepší.

Obsah

1 Úvod

Čím dál častěji je při tvorbě webové aplikace kladen důraz na interaktivní zpracování, multimédia a další z mnoha funkcí, které nám Rich Internet Applications (RIA) přinášejí. Možností jak vytvořit takovou aplikaci je v dnešní době poměrně dost a tím samozřejmě vzrůstají požadavky na vývojáře, který se musí orientovat v mnoha technologiích a vývojových prostředích.

Podle mého názoru, žádné dnešní vývojové prostředí neposkytuje dostatečnou podporu vývojáři, ale spíše se zaměřuje na podporu programovacího jazyka, se kterým se v daném prostředí pracuje. Jaké by to bylo udělat prostředí, kde programovací jazyk není potřeba? Prostředí, ve kterém by si svou aplikaci mohl udělat každý a dostatečně rychle, tedy prostředí, které by mohl ovládat i uživatel, nejen vývojář. Hlavní náplní této práce je vytvořit vývojové prostředí tak trochu odlišné od ostatních variant.

V kapitole 2 – Rich Internet Application, je popsán úvod do RIA aplikací. Od historie, přes dělení (využití nástrojů, typy vývojových prostředí a programovacích jazyků) až po možnosti a využití Rich Internet Applications.

Kapitola 3 – Analýza existujících řešení, uvádí některá srovnání s jinými variantami tvorby webových aplikací. Převážně se porovnávají vývojová prostředí, která slouží k tvorbě RIA aplikací.

Kaptiola 4 – Návrh Aplikace a Kapitola 5 – Implementace, popisuje samotnou aplikaci, tedy od výběru technologií, popisu konceptu a rozdělení aplikace, až po její samotnou implementaci.

Kaptiola 6 – Testování, ukazuje postup tvorby testovací aplikace, který je popsán krok za krokem.

Neméně důležitou částí je závěr, ve kterém jsou uvedeny možnosti pokračování.

2 Rich Internet Application

Rich Internet Application [1], je webová aplikace, která se chová podobně jako klasická desktopová aplikace, tedy rozšiřuje funkčnost klasických technologií, které jsou použity pro zobrazování webových stránek a obecně obsahu, který je sdílen internetovou sítí. Nejčastěji je tato funkční nástavba realizována formou pluginu, tedy nějakého programového modulu, ve webovém prohlížeči [2].

2.1 Historie

Pojem Rich Internet Application byl stanoven firmou Macromedia v roce 2002. Byl to sice nový termín, ale vycházel z již existujících technologií jako Remote Scripting (termín firmy Microsoft z roku 1999) [3], X Internet (termín firmy Forrester Research z roku 2000) [4] a dále Rich Clients [5] a Rich Web Application. Všechny tyto technologie měly za cíl udělat webový obsah více interaktivní s velkým počtem funkčních nástaveb, zkrátka udělat technologii, která by smazala rozdíl mezi internetovou aplikací a desktopovou aplikací. Jak následující roky ukážou, firmě Macromedia se to s její technologií, kterou nazvali Flash [6] (a v rámci prohlížeče, se tento plugin jmenuje Flash Player [7]), podařilo ze všech nejlépe a již po krátké době se tato technologie stala nejrozšířenější RIA platformou a první pozici hájí dodnes. V roce 2005 firma Adobe, mezi jejíž nejznámější produkty patřil třeba Photoshop, koupila firmu Macromedia za 3.4 miliardy USD a tím získala produkty jako Fireworks, Dreamweaver, FreeHand, Breeze, ale hlavně Flash, tedy majoritní technoogii používanou pro tvorbu RIA aplikací.

2.2 Možnosti

Možnosti těchto „chytrých“ aplikací jsou velice rozsáhlé. Jak jsem již zmínil, snaží se kopírovat funkce desktopových programů, další rozdíl oproti běžně používaných technologií je asynchronní komunikace s cross - side scripty, jako je třeba php, asp, coldfusion atd. Tuto funkci má rovněž technologie zvaná Ajax [8] (Asynchronní Javascript a Xml). Další nejvíce využívané funkce, typické pro realizaci aplikací RIA, je práce s videem, kde jde video progresivně stahovat a přehrávat třeba ve Flashovém přehrávači (www.youtube.com), nebo jde video streamovat. Stejně tak práce se zvukem a různé mp3 přehrávače, které jsou realizovány jako RIA. Hojně se RIA aplikace používají na tvorbu her a interaktivních prezentací. Další rozvoj této technologie, známý jako Adobe Air, je technologie, která umožňuje využívat RIA aplikace psané pro Flash Player, i jako klasické desktopové programy, spouštěné v rámci operačního systému a nejen jako aplikace uvnitř webového

prohlížeče. Tyto aplikace mohou používat programovou nástavbu, která přináší různou podporu funkcí operačního systému. Na závěr výpisu možností RIA aplikací, je neméně důležitá vlastnost a to, že aplikace psané pro Flash Player nebo jako Adobe Air aplikace jsou multiplatformní a nově i s podporou pro mobilní telefony, jako je iPhone a jeho operační systém iPhone Os a také pro Android, kde je kompletní podpora Adobe Air.

2.3 Rozdělení

Rozdělení RIA aplikací provedu podle nejčastěji využívaných platforem a zkráceně popíšu jejich výhody a nevýhody a další zajímavé věci. Jako RIA aplikaci, jde považovat i aplikaci, která používá AJAX, i když to není v pravém slova smyslu RIA, jak ji chápe většina publika. Z hlavních platforem, které mají procentem užití co říct, je s 54% (červenec 2010), jako třetí nejvyužívanější platforma Microsoft Silverlight [9]. Tento framework vznikl s cílem konkurovat nejrozšířenější platformě pro tvorbu RIA aplikací Adobe Flash. Je podporován ve většině webových prohlížečů v operačním systému Windows, MacOS X. Dále je podporován Windows Phone 7 a Symbian telefony. Ve spolupráci s firmou Novell, je také Silverlight podporován pomocí programu Moonlight na Linuxu a FreeBSD. Aplikace Silverlight může být psána v jakémkoliv .NET programovacím jazyku [10], ale místo .NET Framework CLR je používán Silverlight CoreCLR.

Další nejrozšířenější platforma pro RIA aplikace je Java [11], vyvinutá firmou Sun, kterou koupil Oracle Corporation. Je přístupná pro Linux, Mac OS X a Solaris. Posléze byla zpřístupněna i pro Microsoft Windows. Hlavně dříve byla podporována na většině mobilních zařízení, ovšem dnes, s nástupem chytrých telefonů, začíná být podle mého názoru opomíjená, protože tvůrci těchto mobilních platforem chtějí, aby programy byly vyvíjeny jejich formáty, které zaručují co největší optimalizaci a rychlost aplikace, a které chtějí prodávat přes jejich obchody s aplikacemi. Typickým příkladem a dozajista i průkopníkem je Apple Computers Inc. se svým AppStorem na platformě iPhone OS, který je použit v telefonech iPhone (generace 2.0, 3G, 3GS, 4), iPodech Touch a nově i na zařízení iPad (první a druhé generace).

Jako nejrozšířenější platformou pro RIA aplikace je Adobe Flash (dříve Macromedia Flash), která svou penetrací 99% je absolutně nejrozšířenější platformou v této kategorii. Jako ostatní konkurenti je multiplatformní i s exportem pro iPhone / iPod / iPad a další zařízení.

Jako, ne sice moc rozšířené pluginy, ale za to velice zajímavé svou funkčností jsou webové pluginy firem jako je Stonetrip [12] či Unity3D [13] a další. Tyto programy, jejichž pluginy sice nejsou moc rozšířené, ale na druhou stranu přináší zajímavější funkce, hlavně přehrávání 3D obsahu přímo v prohlížeči. Tyto programy podporují multiplatformní export pro různá zařízení a platformy (Pc, Mac Os, Linux, XboX 360, Wii, iPhone OS, Android,...) nabízí i export právě pro svůj webový plugin, tedy jedna aplikace, může být vyvíjena jedenkrát a prezentována prakticky všude.

Ve spojení s 3D obsahem bych podotknul ještě novinky od firmy Adobe – technologii Molehill [14]. Tato novinka je součástí Flash Playeru a přináší plnohodnotnou podporu 3D, podobně jak je tomu u výše zmíněných pluginů firmy Stonetrip či Unity3D.

2.4 Využití

Využití RIA aplikací je všude tam, kde chceme přinést větší míru interaktivity, než co nám nabízí klasické webové technologie. Dále pokud chceme využít na stránkách video (nejen v rámci přehrávače, ale třeba jako designový prvek) a stejně tak audio.

Mezi příklady, jak už bylo zmíněno patří hry, interaktivní prezentace a různé aplikace a doplňky webových stránek (video přehrávače, ...). Samozřejmě využití je mnohem širší, ale tohle jsou asi nejužívanější možnosti RIA platforem, na které jsou používány.

Na závěr této kapitoly, bych se chtěl zmínit o vhodnosti využití RIA aplikací. Určitě není dobré používat RIA aplikace bezhlavě a všude. Tyto aplikace většinou potřebují ke svému běhu spuštění externího pluginu, který zatěžuje procesor (a většinou i ve velice značné míře) a tím třeba u notebooků a mobilních zařízení snižuje dobu výdrže počítače na baterie a obecně větším zatížením samozřejmě snižuje výkon počítače, který by mohl využívat jiný proces. Další důležitá věc, na kterou je třeba myslet, je možnost indexování obsahu pro vyhledávače. Dnes už nějaké postupy existují, ale stejně nejsou tak účinné, jako u klasických webových technologií. Neméně důležitým prvkem, který musíme při nasazení RIA zvážit, je na kterých zařízeních chceme aplikaci mít přístupnou, protože na mnoha mobilních zařízeních není třeba Flash Player dostupný (například iPhone OS,...) a tím se dostáváme do problému. Proto se často s RIA aplikací dodává i verze, která je napsána klasickými mobilními technologiemi a v případě nedostupnosti pluginu se zobrazí tato non-RIA verze, stejně tak, pokud není dostupná potřebná verze pluginu.

2.5 Pohled do budoucnosti

V poslední době a hlavně velkým nástupem mobilních zařízení firmy Apple Computers – a jejím odmítnutím podpory Flash Playeru v mobilní verzi prohlížeče Safari, či odmítnutí Adobe Air platformy, se stále více začíná hovořit o novém standardu HTML 5, který má rozšířit funkčnost klasických non-RIA technologií. HTML 5 je stavěn do role oponenta Flashové technologie, tedy nové technologie, která vytlačí využívání Flash Playeru. O to víc je do této role tlačen právě díky Applu, jehož spory s Adobe se stále více prohlubují. Nicméně v téhle věci jsem skeptický, protože tím, že HTML 5 není vyvíjen jednou firmou jako Flash, je schvalování nějakých standardů mnohem více komplikované a časově náročné. Další velký problém je pak podpora těchto standardů v prohlížečích, jak známe z dnešní doby, je problém i s interpretací standardů klasických webových technologií.

Můj názor je, že HTML 5 zatím, určitě ne v krátké době, nevytlačí Flash jako technologii, možná omezí její užití, jako třeba při přehrávání videa a podobně, ale nemůže nahradit celou platformu, která dnes přináší obrovské množství funkcí, včetně podpory velice kvalitního programovacího nástroje Action Script 3, který podporuje objektově orientované principy a další potřebné nástroje a funkce, které jsou nezbytné pro vytváření složitějších aplikací. Dále vývojové prostředí pro Flash aplikace, včetně rozrhraní pro tvorbu animací a využití spousty kvalitních grafických nástrojů. V každé verzi vývojářského programu je i větší podpora komunikace s dalšími nástroji od Adobe, jako je Adobe Photoshop, Adobe Illustrator a také Adobe Flash Builder, jako další alternativě pro tvorbu Flashových aplikací. Tak či tak, Adobe má obrovský náskok, který se bude těžko dohánět.

3 Analýza existujících řešení

Na začátek, bylo potřeba zjistit, zda existuje podobné řešení a projít alternativy tvorby RIA aplikací. Vybral jsem tyto aplikace, jako řešení, které stojí za to, abych se na nich poučil, vybral to dobré z nich a přenesl tyto kladné vlastnosti do mého návrhu.

Nutno podotknout, že jsem se nesetkal s řešením, které by více vystihovalo to moje – většina editorů je stále určena primárně pro vývojáře, než-li pro uživatele méně odborně zdatné. Dost editorů nabízí nějaké řešení WYSIWYG editoru, ale vždy je třeba funkčnost doprogramovat, nebo nabízí možnosti tak malé, že o uplatnění, jako v případě zde popsané aplikace, nelze uvažovat a vždy mohou tyto aplikace sloužit jako pomůcky, nikoliv řešení. Nejlepší varianty jsem nakonec našel u firmy Adobe, nebyl to záměr, ale aspoň nám to ukáže jednotlivé varianty:

- Adobe Flash Builder [16]
- Adobe Dreamweaver [17]
- Adobe Flash Catalyst [18]
- Tvorba pomocí šablon

3.1 Adobe Flash Builder

Zástupce Adobe Flash Builder jsem vybral z důvodu, že má aplikace je v něm tvořená, jak bude psáno dále v rámci tohoto dokumentu. Je to představitel řešení typu WYSIWYG editor + editor programovacího kódu. Takových řešení je spousty, mě na Flash Builderu zaujal ne jeho koncept programování, ale samotný program, ze kterého jsem se dost poučil (– výborná práce s panely a moduly atd.). Samotný Flash Builder jsem z konkurence typu Microsoft Visual Studio ši QT Creator vybral z důvodu, že umožňuje, stejně jako moje aplikace, export do Flashového formátu. Více o aplikaci Adobe Flash Builder později.

3.2 Adobe Dreaweaver

Řešení, které je už inspirativní v rámci toho, co bude ve skutečnosti dělat moje aplikace. Omezená je podpora v rámce HTML stránek, ale to je pochopitelné z pohledu portfolia firmy Adobe a proto bych se nesnažil aplikaci vytýkat toto omezení. Opět aplikace obsahuje WYSIWYG editor, ale tentokrát již lze teoreticky napsat webové stránky bez znalosti HTML jazyka a dalších potřebných náležitostí. Na tom moje aplikace stojí a dále to rozvíjí.

3.3 Adobe Flash Catalyst

Opět řešení od firmy Adobe. Tady vidím největší podobnost v hodně aspektech s mojí aplikací. Adobe Flash Catalyst sice dělá most mezi grafikem a programátorem, neboura úplně tuto hranici a ani se o to nesnaží, chce být zkrátka jen mostem. Co je zajímavé na této aplikaci, že umožňuje zpracování grafických podkladů do logiky výsledné animace a dokonce umožňuje aplikovat základní funkčnost pro vybrané prvky (navigace, posuvníky, reprezentace dat,...). Vždy je nakonec potřeba práce programátora, který daný produkt dokončí, a proto se Adobe Flash Catalyst nejvíce používá v prezentační fázi vývoje softwaru. Umožňuje po dokončení grafických návrhů ukázat klientovi funkční verzi programu, tedy funkční části (jak jsem zmiňoval navigaci program atd.).

3.4 Tvorba pomocí šablon

Dnes nejrozšířenější metoda vlastní editace vzhledu aplikace – a to nejčastěji webových aplikací. Uživatel si vybere šablonu i s grafickým návrhem, často má možnost nějakých velice omezených úprav a tuto šablonu s dodaným obsahem publikuje. Zatím se to zdá být jediné možné existující řešení, které bych chtěl překonat.

4 Návrh Aplikace

Tato kapitola se zabývá návrhem aplikace. Od popisu aplikace, přes výběr technologií až po detailní popis řešení návrhu.

4.1 Obecný popis aplikace

Jako hlavní požadavek na aplikaci je přinést nový nástroj na tvorbu RIA aplikací, který umožní nejenom vytvářet uživatelské rozhraní, ale rovněž přinese uživateli možnost uvést je do kontextu a nastavit funkčnost jednotlivých prvků. Uživatel si tedy z grafických podkladů jednoduše sestaví aplikaci, přiřadí daným prvkům události a akce, které se po vyvolání událostí provedou a danou aplikaci si vyexportuje do požadovaného formátu. V návrhu této aplikace je export do formátu swf (Small Web Format - dříve ShockWave Flash) [19]. Daný soubor se vyexportuje současně jako komplexní řešení spolu s webovou stránkou, do které bude automaticky vložen.

Jak již je snad patrné, program bude užitečný hlavně jako úspora času programátora či samotným grafikům, kteří budou soběstační v tvorbě jednoduchých aplikací. Zároveň podstatně klesne pořizovací cena těchto aplikací.

4.2 Výběr vývojového prostředí a technologie

Aplikace bude vyvíjena v programu Adobe Flash Builder (dříve Adobe Flex) jako Adobe Air aplikace. Toto řešení skýtá několik důležitých výhod. Tou hlavní je bezesporu multiplatformní výstup na všechny nejrozšířenější operační systémy – Microsoft Windows, Mac OsX firmy Apple a běžně rozšířené Linuxové distribuce. Adobe Air technologie však nezůstává jen u multiplatformního pojetí v rámci desktopových operačních systémů, ale přináší podporu i pro další zařízení a to jak mobilní zařízení, PDA či tablety. Podporované jsou operační systémy Android, BlackBerry Tablet OS, iPhone OS (iOS) a dokonce některé televize či příslušenství k televizím podporují rovněž tuto platformu. V tuto chvíli firma Adobe chystá rozšíření i na další populární operační systém Palm OS.

Další výhodou kromě univerzálnosti aplikace je možnost distribuce. Pominuli možnost aplikaci distribuovat přes Adobe Marketplace, což je po vzoru App Store konceptu firmy Apple cesta, jak snáze propagovat a distribuovat aplikaci širšímu okruhu lidí, velkou výhodou od ostatních platforem na tvorbu desktopových aplikací je instalace přímo z webového prohlížeče, kdy uživatel jedním kliknutím zahájí instalaci bez nutnosti stahování instalačního balíčku a ručního spuštění tohoto instalátoru. Tato možnost sice nebude v této fázi využita u navrhnuté aplikace, ale určitě je dobré myslet dopředu a je to další dostupný krok, jak uživateli zpříjemnit práci s aplikací.

V neposlední řadě je návrh aplikace pojat globálně, aby aplikace mohla běžet i jako webová služba, tedy bez nutnosti nějaké instalace programu. Více o tomto řešení později.

4.3 Základní koncept aplikace

Aplikace je navrhnuta tak, aby nebyla tvořená aplikace závislá na jedné vybrané platformě, ale bylo možné exportovat jednou vytvořenou aplikaci na více platforem – nejenom jako *.swf soubor, ale v budoucnu třeba standardních webových technologií (HTML + CSS + Javascript) či jako aplikace pro mobilní zařízení. Tyto verze aplikace by pouze upravovaly možnosti daného programu dle podporované platformy.

Tedy základní koncept je vytvořit univerzální vývojové prostředí, s vlastní strukturou kompozice aplikace, systémem přiřazování událostí objektům a dalších potřebných podsystémů, který bude umožňovat tvořit aplikace různých technologií a platforem. Tento fakt je brán hlavně v pohledu do budoucnosti, v aktuální verzi softwaru podporuje pouze export do *.swf, jak bylo výše uvedeno.

4.4 Rozdělení aplikace

Samotná aplikace se skládá z několika hlavních vrstev. Vrstvou se rozumí souhrn knihoven (knihovny budou podrobněji popsány v kapitole Implementace), které mají společnou systémovou úroveň, typ funkčnosti a další aspekty, které jsou typické pro jednotlivé vrstvy. Každá z těchto vrstev je oddělena a komunikace mezi vrstvami probíhá přes speciální komunikační rozhraní (speciální komunikační vrstvu).

Tento systém vrstev je ještě důležitý z jednoho hlavního hlediska a to, že všechny vrstvy jsou nezbytné k chodu aplikace, ale jednotlivé knihovny dané vrstvy nezbytné nejsou. Toto bylo v návrhu bráno jako velice důležitý aspekt, protože to umožňuje libovolně měnit verze aplikace, kdy každá verze může být velice jednoduše funkčně omezena a to pouhým odebráním knihovny. Nic dalšího není potřeba a celý systém bude bez chyby pracovat dále, pouze ochuzen o funkce chybějící knihovny. V praxi je aplikace připravena třeba na to, aby běžela jako kompletně webová služba, bez podpory Adobe Air funkcí (mezi které patří vlastnosti typické pro desktopové programy, jako je přístup k datům na disku, u mobilních zařízení nástavby těchto zařízení, jako jsou multidoteková gesta, události akcelerometrů v těchto zařízeních a mnoho dalšího, co bude blíže popsáno u dané knihovny v *Core Layer*) a to pouhým odebráním Adobe Air knihovny. Samozřejmě využití se najde i jinde – jak se psalo výše, na různé verze aplikace, kde některé verze toho budou umět více atd.

Vrstvy aplikace:

- Core Layer (jádro aplikace)
- Engine Layer (část, která se stará o vykreslování celého editoru)
- Plugin Layer (vrstva nástaveb aplikací)
- Application Layer (vrstva aplikace, do které je celý systém vložen)

Jak bylo uvedeno výše, vrstvy jsou implementačně zcela odděleny. Proto aby mohli mezi sebou komunikovat je zda navíc vrstva *Communication Layer*. K této vrstvě má přístup každá z uvedených vrstev a tedy komunikace napříč programem probíhá přes ní.

4.4.1 Core Layer

Vrstva *Core Layer* v sobě obsahuje knihovny, které patří mezi nejzákladnější systémové funkce aplikace. Tím se rozumí funkce, které jsou využívány napříč celým spektrem aplikace a tedy jsou potřeba v jakoukoliv chvíli běhu aplikace a v jakémkoliv stavu aplikace. Veškeré knihovny vrstvy *Core Layer* neobsahují žádné grafické prvky uživatelského rozhraní aplikace a přináší pouze programovou funkčnost.

Knihovny obsažené v *Core Layer*:

- Core Library
- Air Library
- Drag Library
- Loaders Library
- Parsers Library
- Validators Library

Jednotlivé knihovny vrstvy *Core Layer* budou popsány v části implementace, kde si je podrobněji probereme.

4.4.2 Engine Layer

Vrstva *Engine Layer* představuje jádro aplikace. V návrhu je tvořena pouze jednou částí – jedinou knihovnou *Engine*. Tato vrstva je navržena jako univerzální vykreslovací jádro, tedy část, která se stará o rozložení celé aplikace a o její hlavní ovládání.

Rozložení sekcí aplikace bylo navrženo takto:

- Editor
- Preview
- Export

Jednotlivé sekce budou reprezentovat moduly, které jsou součástí vrstvy *Plugin Layer*. Jedná se tedy o jejich vykreslení a o samotných modulech si toho řekneme více později.

V návrhu je rovněž kladen důraz na modulárnost celé aplikace, proto bude veškeré vykreslování řízeno konfiguračními soubory, o kterých si povíme více v sekci implementace.

4.4.3 Plugin Layer

Vrstva *Plugin Layer* je navržena jako nástavba aplikace, zde budou obsaženy všechny panely a moduly, které jednotlivé sekce programu budou využívat. Z toho vyplývá název *Plugin* – vrstva je navrhnuta tak, aby bylo velice jednoduché do aplikace přidat nebo odebrat daný modul nebo panel. Rozdíl, který je zde vhodné uvést, mezi modulem a panelem je ten, že panel ovládá nastavení nějakého prvku modulu, to znamená, že panel se váže ke konkrétnímu modulu. Jejich vykreslení jak jsme si uvedli, řídí vrstva *Core Layer*.

Aplikace v návrhu obsahuje tyto moduly:

- Editor
- Preview
- Export

Jak si můžete všimnout, názvy modulů korespondují s názvy sekcí v *Engine Layer*. Většinou totiž sekce má pouze jeden modul (není to podmínka, může jich mít i více, ale zatím to nebylo potřeba). Proto se hlavní moduly jmenují stejně jako sekce, která je obsahuje, aby bylo jasné, kam modul patří.

Editor

Základní sekce *Editor*, je navržena jako vývojové prostředí pro uživatele. V této části bude uživatel kompletně sestavovat aplikaci, tedy pracovat s grafickými podklady a bude nastavovat funkčnost daným prvkům.

Preview

Sekce *Preview* je navržena jako živý náhled na aplikaci. Uživatel si bude moci libovolně během vývoje prohlédnout aplikaci, aniž by bylo nutno jakoukoliv část programu kompilovat. Aplikace se spustí ve virtuálním prostředí programu a uživatel si může rychle vyzkoušet jak se jeho aplikace chová a může se vrátit se zpět k úpravám, nebo přejít do další sekce.

Export

Sekce *Export* bude sloužit k vyexportování meziformátu, který využívá aplikace do cílového formátu. V prvních fázích vývoje je navržen export do formátu *.swf – to je Flashový formát a tedy výstupem celé aplikace je Flashová aplikace, která je automaticky vložena do webové stránky.

Podrobněji si jednotlivé moduly, včetně panelů, které k nim patří, popíšeme v implementaci vrstvy *Plugin Layer*.

4.4.4 Application Layer

Vrstva *Application Layer* je nevyšší vrstvou zapouzdření programu. Jak jsme si výše uvedli, jádrem programu je vrstva *Engine Layer*, která obstarává veškeré funkce aplikace. Vrstva *Application Layer* je její nástavbou, tedy *Engine Layer* je vložena do *Application Layer*.

V této vrstvě máme velice jednoduchou možnost vložení naši aplikace do cílové platformy. V aktuální verzi návrhu jsou předpokládány dva podporované formáty, které technologie Flash podporuje, tedy jako Adobe Air aplikace nebo Flashová aplikace, která by běžela v pluginu Flash Player webového prohlížeče. Více v implementaci této vrstvy.

4.4.5 Communication Layer

V této práci již mnohokrát vzpomínaná vrstva *Communication Layer*. Rozhodl jsem se ji zařadit až na konec, protože je to taková globální vrstva, která slouží jako brána pro komunikaci napříč celou aplikací. Tato vrstva byla navržena jako univerzální komunikační rozhraní, které bude využíváno v případě potřeby komunikovat napříč vrstvami programu, nebo při komunikaci mezi knihovnami. Vše se bude odehrávat s důrazem na bezpečnost komunikace, aby aplikace neskončila v nepovoleném stavu a veškeré požadavky byly správně cíleny.

Více podrobností v části Implementace.

5 Implementace

Tato část práce popisuje konkrétní vybrané technologie a technickou implementaci návrhu aplikace. Dále také popisuje podrobněji jednotlivé vrstvy programu, tedy jejich jednotlivé knihovny z pohledu implementace.

5.1 Technologie

Jak již bylo mnohokrát zmíněno a naznačeno, samotná implementace programu probíhala v programu Adobe Flash Builder. Programovacím jazykm byl Action Script 3 [20] a ostatní potřebná data (konfigurační soubory,...) byla ukládána do formátu XML [21].

Adobe Flash Builder

Tento program, dříve známý jako Adobe Flex, jsem si vybral kvůli podpoře Adobe Air, programovacímu jazyku Action Script 3.0 a skvělému rozrhraní (postavenému na Eclipse), které toto vývojové prostředí přináší. Mým záměrem bylo, jak jsem několikrát uvedl, vyvíjet aplikaci za pomoci Adobe Air technologie a proto ve výběru vývojového prostředí byly pouze dvě potencionální volby. Adobe Flash, a nebo Adobe Flash Builder. Pro psaní aplikací, které nevyžadují animace, přílišné grafické efekty a podobně, je Adobe Flash Builder lepším nástrojem a co se týče podpory v programování v editorech, kterými oba editory disponují, je ten Flash Builderu mnohonásobně lepším. Další důležitým aspektem, který hrál důležitou roli, je práce s okny a objekty, co se týče pozicování a dostupných nastavení, včetně předpřipravených komponent. Adobe Flash Builder má více hotových předvoleb v tomto požadavku, které by bylo v Adobe Flash prostředí nutno naprogramovat. K tomu jako překladač do koncového formátu *.swf, který naše aplikace používá, byl zvolen Adobe Flex SDK, které je volně dostupné.

Action script 3.0

Veškerý kód byl psán v programovacím jazyku Action Script 3. Ten je objektově orientovaný a navíc v něm mám velké zkušenosti, proto jsem se mohl pustit do tohoto náročnějšího projektu. Tento programovací jazyk je založen na mezinárodně standardizovaném programovacím jazyku pro skriptování ECMAScript. Action Script 3.0 je kompilován podle ECMAScript specifikace a je pouštěn v tzv. ActionScript Virutal Machine(AVM), který je součástí Flash Playeru. Nyní existuje druhá verze AVM, tedy AVM2. Ta umožňuje až 10 krát rychlejší puštění ActionScript kódů oproti první verzi. AVM2 je používán od Flash Playeru verze 9 – aktuální verze Flash Playeru je 10 (v Adobe Labs už je přístupná i verze 11, která mimo jiné přidává technologii Flash kompletní 3D prostředí a podporu hardwarové akcelerace celé aplikace).

XML

Formát XML netřeba zdlouhavě představovat. XML je rozšiřitelný značkovací jazyk, a proto jsem si ho vybral, jako formát pro uložení datových informací pro aplikaci. Právě jeho možnost postavit si vlastní struktury v poměrně přehledné podobě a zároveň dobrá podpora tohoto formátu v rámci Action Script 3.0 byly určující ve výběru formátu. Action Script obsahuje systémové funkce, které velice zjednodušují čtení a vytváření XML dokumentů, a jak bylo zmíněno, veškerá komunikace i meziformát aplikace je postaven na struktuře XML.

5.2 Implementace návrhu aplikace

Jak bylo popsáno v návrhu, celá aplikace se dělí na jednotlivé vrstvy a každá vrstva se dělí na jednotlivé knihovny. Toho bylo implementačně docíleno za použití Flex Libraries [22] – tedy samospustitelných knihoven, které pomáhají držet v celkovém projektu přehled, řád a ještě zkracují dobu vývoje tím, že při výsledné kompilaci se vždy překompilovávají jen knihovny, které byly upraveny a tím šetří čas, který by bylo potřeba vynaložit na testování aplikace. Dané knihovny mají další výhodu spíš v potencionálu budoucnosti než v aktuální verzi programu. Knihovny v této verzi jsou vloženy do výsledného programu – tedy výstupem je jediný soubor. To zcela nevyužívá potenciálu, jaký Flex Library mají, ale pro daný program – desktopovou aplikaci, je to nejlepší řešení. V budoucnu, by knihovny mohli být externí, což by mělo větší výhody ve webové aplikaci, kdy více aplikací může sdílet jednou načtenou knihovnu – tj. i když máme několik různých aplikací a více z nich využívá nějakou externí runtime knihovnu, stačí ji načíst jedenkrát. Knihovna je ukládána v mezipaměti buď prohlížeče, nebo přímo v mezipaměti Flash Playeru a v případě potřeby jinou aplikací již nemusí být znovu načítána z internetu. Další výhodou je aktualizace programu, kdy stačí aktualizovat jednu knihovnu a není třeba stahovat celý program znovu.

Jak bylo v návrhu popsáno, vrstvy se skládají alespoň z jedné knihovny – vyjímku tvoří *Application Layer*, což je nejvyšší vrstva implementace. Ta je dle požadavku výstupu aplikace tvořena jako Flexový projet s výstupem do Adobe Air nebo Adobe Flash platformy.

Nyní si popíšeme knihovny jednotlivých vrstev, které byly popsány v návrhu aplikace.

5.2.1 Core Layer

Vrstva *Core Layer* jako jediná obsahuje podporu Adobe Air technologie, tedy jako jediná, je závislá na funkcích, které Adobe Air technologie poskytuje aplikaci. To nám přináší výhodu rychlého přenosu aplikace do webového prostředí, kdy stačí upravit danou vrstvu – respektive pouze jednu z knihoven, které vrstva *Core Layer* má a zbytek aplikace můžeme ponechat beze změn.

Core Library

Základní knihovna vrstvy *Core Layer*. Tato knihovna se stará o příjem požadavků z komunikační vrstvy a rozesílá tyto požadavky do požadovaných knihoven. Pokud daná knihovna není dostupná, požadavek se zkrátka neprovede – to ukazuje možnost odebírat jednotlivé knihovny z vrstvy. V případě odebrání *Core Library* by nebylo možné využívat žádnou z dalších knihoven vrstvy *Core Layer*.

Air Library

Knihovna *Air Library* přináší celé aplikaci nástavbu technologie Adobe Air oproti běžné funkčnosti technologie Flash. Knihovna se například stará o práci se soubory, kam patří import souborů (grafické podklady a všechny soubory potřebné k tvorbě aplikace), managment celého projektu (tvorba složek a souborů v Application Storage – datové úložiště aplikace, kde se exportují potřebné soubory k tvorbě uživatelské aplikace) a export výsledné aplikace, včetně volání Adobe Flex SDK, které se stará o vytvoření výstupního *.swf souboru. Jak je jistě patrné, *Air Library* je svou podporou funkcí klíčovou knihovnou.

Drag Library

Knihovna *Drag Library* je vlastní řešení podpory funkcí Drag & Drop, tedy systému pro přetahování objektů myší, včetně posílání dat s touto událostí. Tato knihovna je například využita při přetahování grafických podkladů do scény. Knihovna musela být do systému přidána kvůli nedostatečné kontrole vestavěné Action Script třídy Drag Manager. Ve vlastním řešení byl podstatně rozšířen systém událostí a reakcí na události, jako například půjčování přetahovaných objektů do jiných objektů, které umějí reagovat na typ přetahovaných dat.

Loaders Library

Knihovna *Loaders Library* slouží k nahrání obsahu souborů do aplikace. Například v situaci, kdy nahráváme do aplikace grafické podklady. Do knihovny přijde požadavek na obsah souboru, který je zadán cestou k tomuto souboru. Knihovna daný soubor načte a pošle nazpět data. Pokud již byl daný soubor načten, nemusí se načítat znovu, ale data se pouze zkopírují, což šetří paměť a čas, který by musel být vynaložen na práci se souborem. Veškeré typy souborů, které lze otevírat musí mít podporu v této knihovně. Tím ale není myšlena podpora pro každý datový typ (*.jpg, *.png, *.flv, ...), ale jen typ souboru, jakým je třeba obrázek, či video.

Parsers Library

Knihovna *Parsers Library* obsahuje třídy na parsování datových souborů – různých xml souborů, které jsou potřeba pro chod aplikace.

Knihovna obsahuje podporu pro parsování:

- konfiguračního souboru aplikace

- konfiguračního souboru modulů
- konfiguračního souboru panelů
- konfiguračního souboru projektu

Daný parser na požadavek odešle referenci na sebe, hned jak jsou data z požadovaného souboru připravena. Objekt, který parser volal, si voláním metod parseru (které parsují vždy nějakou část souboru) vybírá jen části, které potřebuje a není potřeba parsovat celý soubor, pokud to objekt nevyžaduje. Tím se samozřejmě šetří čas i paměť.

Validators Library

Knihovna *Validators Library* slouží k validaci hodnot ve vstupních formulářích a v případě nalezení chyby vrací chybné hlášení objektu, který dal na validátor požadavek. Mezi validátory aktuálně aplikované patří:

- Validátor hlídající nevyplněný vstup
  To slouží u polí formuláře, které jsou potřeba vyplnit.
- Validátor hlídající existující hodnotu ve vybraném poli
  To slouží při potřebě mít jedinečné hodnoty, například v názvech objektů, aby nevznikaly duplicity.

Další validátory je samozřejmě možno jednoduše přidat v případě potřeby.

5.2.2 Engine Layer

O vrstvě *Engine Layer* víme z návhru, že představuje jádro aplikace. Je tvořena pouze jednou knihovnou *Engine*, která obsahuje rozdělení uživatelského rozhraní a to na prostor pro navigaci a ostatní prostor okna, do kterého jsou vykreslovány moduly a panely. Tento prostor se nazývá *Base* a je to tedy ohraničený prostor pro vykreslování samotného programu. Navigační část se skládá z komponenty obsahující dva druhy menu. Základní horizotální menu a pak přepínací menu, které přepíná sekce programu (*Editor*, *Preview*, *Export*).

Aktuálně je toto přepínání sekcí ne zcela důležité, protože *Preview* není implementová a *Export* nemá uživatelské rozhraní - vše se děje na pozadí. Více o důležitosti tohoto rozdělení v části Závěr, kde jsou uvedeny možná další rozšíření. Části Editor a Export budou popsány v implementaci vrstvy *Plugin Layer*, kde jsou implementovány. V sekci *Engine Layer* však řešíme jejich vykreslení.

Za zmínku také stojí jednoduchost rozšíření těchto sekcí, jejich soupis je načítán z konfiguračního souboru aplikace, a tedy stačí přidat do tohoto souboru další sekci a nastavit ji

rozestavení jednotlivých elementů. O zobrazování, skládání a vykreslování těchto elementů (panelů a modulů) si řekneme nyní.

Nastavení aplikace

Celá kompozice aplikace je závislá na konfiguračním souboru. Tento soubor obsahuje jak rozložení hlavního menu, tak rozložení sekcí, včetně definici obsahu konkrétních sekcí. Veškeré informace jsou uloženy v přehledné XML struktuře, která, jak bylo uvedeno, nastavuje rozestavení všech elementů dané sekce.

Nastavení a vykreslení sekcí

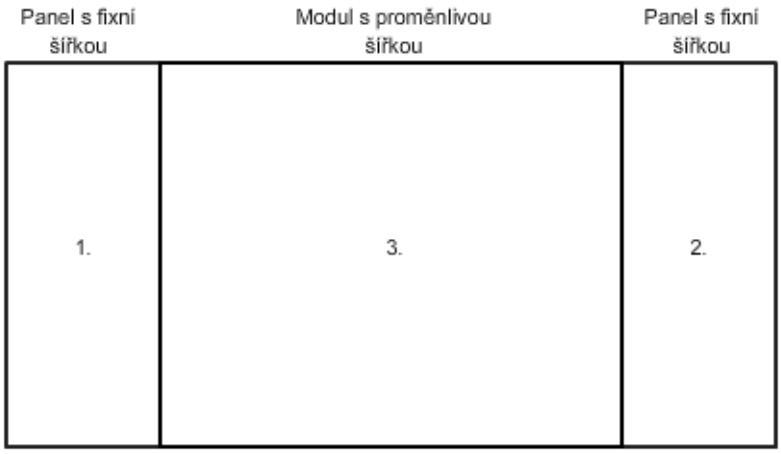
Každá sekce se zobrazí v navigační části aplikace – konkrétně v přepínání sekcí. Sekce je tvořena jednotlivými bloky (viz. níže). Kromě deklarace bloků obsahuje nastavení sekce taky rozestavení separátorů. Separátor je vlastně pomyslná hranice, mezi dvěma, či více bloky.

Separátory dělíme na:

- Vertikální separátory
- Horizontální separátory

Tyto dva typy slouží logicky buď jako vertikální, nebo horizontální oddělovače daných bloků. V závislosti na typu separátoru, má daný separátor vždy nastaveny sousední bloky a to buď vlevo a vpravo (u horizontálního separátoru) či nahoře a dole (u vertikálního separátoru). Tyto informace potom slouží k rozložení a zarovnání jednotlivých bloků.

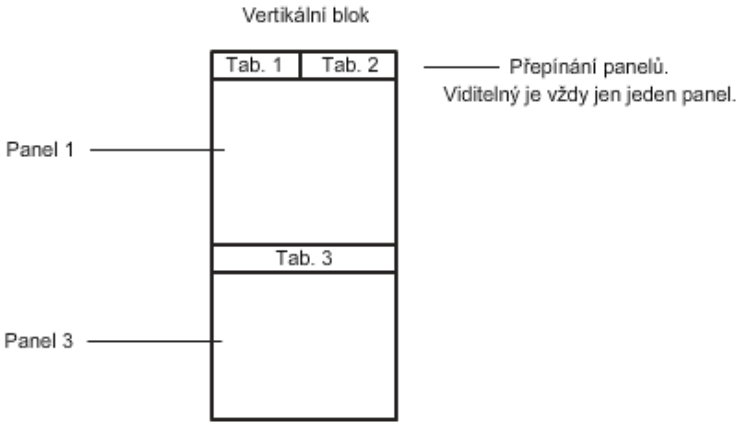
Vykreslovací jádro potom projde separátory zleva doprava (nebo z hora dolů u vertikálních separátorů) a dle nastavení bloků je vykresluje – každý blok má nastaveno, zda je jeho šířka, či výška fixní, nebo je proměnlivá dle prostoru, který na ní zbyl (viz. níže). Proto aplikace prochází separátor a vykresluje zleva doprava bloky, které mají fixní nastavení. V případě, že narazí na blok s proměnlivou hodnotou, přestane vykreslovat a začne vykreslovat z druhé strany, tedy v případě horizontálních separátorů zprava do leva. Opět vykresluje, dokud nenarazí na proměnlivou hodnotu. Nakonec tedy zbudou (pokud tedy má nějaký blok proměnlivou hodnotu velikosti) uprostřed bloky, mezi které se zbývající místo rozdělí. Algoritmus vykreslování je natolik sofistikovaný, že zvládne vykreslovat i možnosti, kdy mezi dvěma plovoucími bloky jsou bloky fixní a samozřejmě vše funguje při změně velikosti okna aplikace a dokonce bez nutnosti přepočítávání rozložení prvků. Krajní prvky jsou totiž připnuty na kraje a mají fixní velikosti, bloky proměnlivé mají hodnoty udány procenty, a tedy jsou expanzivní – proto vše funguje při změně velikosti obrazovky, a jak bylo uvedeno, bez dalších nutných výpočtů na straně aplikace.



Obrázek 5.1: Zobrazení pořadí vykreslovaných elementů na jednoduchém příkladě

Nastavení a vykreslení bloků

Nejprve je nutné říci, co je to vlastně blok v kontextu aplikace. Je to vlastně pás, ve kterém jsou naskládány jednotlivé prvky (panely, moduly). Tento pás je dle nastavení bloku vertikální (tj. prvky jsou v něm skládány shora dolů), nebo horizontální (tj. prvky jsou v něm skládány vedle sebe, zleva doprava). Dále, jak bylo u nastavení sekcí popsáno, mají bloky nastaveno, zda mají šířku, či výšku fixní, nebo proměnlivou (hodnoty expansion pro proměnlivou a implosion pro fixní). Další funkce, která posouvá vykreslování aplikace dále je ta, že každý prvek v bloku se může skládat z jednoho, či více elementů – to můžeme chápat, jako známé tabulátory/panely z různých programů (třeba i jako panely ve webovém prohlížeči). Tedy jsou to panely, či moduly programu, které sdílejí stejné nastavení velikosti a jsou umístěny na stejném místě, avšak viditelný je vždy jen jeden a uživatel si mezi nimi může přepínat.



Obrázek 5.2: Zobrazení jednoho bloku, který se skládá z více panelů.

Nastavení a vykreslení panelů a modulů

Popis funkčnosti a zaměření konkrétních panelů, či modulů, bude popsán později. V této části se jen zaměříme na vykreslování těchto prvků z pohledu jádra aplikace.

Panel je v kontextu této aplikace chápán jako modul aplikace, který slouží ke správě jedné, vybrané vlastnosti, či nastavení (například máme scene panel, event panel, component panel a mnoho dalších, které si popíšeme později). Tyto panely známe z různých vývojových prostředí a cíl nebyl tuto zvyklost nijak inovovat, ba naopak byla snaha se rozmístění standardních prvků držet, protože uživatelé jsou na ně zvyklí.

Modul je pak chápán jako větší – majoritní aplikace, která je vykreslena v rámci sekce (např. v sekci *Editor* je hlavní modul se stejným jménem *Editor modul*).

V rámci vykreslování je důležité zmínit nastavení konfiguračního souboru aplikace pro dané moduly a panely, kde je uvedeno, například nejen poslední rozestavení scény, ale také velikosti daných prvků – tedy v případě, že blok nemá nastavenu proměnlivou velikost, ale fixní. To umožňuje všechny změny nastavení vývojového prostředí ukládat a příště umožnit aplikaci vykreslit prostředí stejně.

5.2.3 Plugin Layer

Ve vrstvě *Plugin Layer* jsou implementovány jednotlivé moduly a jejich panely. Jak bylo stanoveno v návrhu, tato vrstva umožňuje jednoduché přidání, či odebrání modulu nebo panelu. To je umožněno díky navrhnutému komunikačnímu rozhraní vrstvy *Communication Layer*.

Editor

Knihovna - modul *Editor* je hlavní část uživatelského rozhraní. Uživateli nabízí kompletní možnost úpravy a nastavení všech potřebných funkcí a vlastností projektu. Dále v tomto panelu jsou držena všechna data právě otevřeného projektu, včetně všech datových struktur a podobně. Na tomto místě by bylo vhodné popsat rozvržení stavby celého projektu tak, jak byla navržena a implementována. Také zde uvedu popis panelů, které patří k modulu *Editor*.

Každý projekt, který je vytvářen v této aplikaci, má pevně danou strukturu. Na začátku je tvořen z objektu *Scenes*. Tento objekt obsahuje první úroveň scén – minimálně jednu. Scénou se zde myslí vlastně stav programu, aktuální jedno rozvržení uživatelského rozhraní, v přeneseném významu bych to vysvětlil na příkladu webové stránky. Máme web, kde máme 3 hlavní kategorie (info, zboží, kontakt). Tyto 3 hlavní sekce by v našem programu byly 3 hlavní scény – tedy místa, které jsou na stejném místě, ale vždy je zobrazena jen jedna. Do každé scény můžeme přidat libovolné množství objektů.

Mezi objekty, které jsou v aplikaci podporovány patří:

- Container
- Graphic
- Video

Další není díky důrazu na rozšiřitelnost aplikace problém přidat. Teď si popíšeme, k čemu slouží jednotlivé objekty.

Objekt *Container* je základní objekt, který může být použitý – je to nejobecnější objekt, který nemá konkrétní zaměření a slouží k tvorbě rámců v aplikaci. V přenesení opět na naši webovou stránku, bych tuto komponentu jen z dálky přirovnal k *div* elementu v HTML. Sám *Container* si nenese žádnou vizuální prezentaci, stejně jako *div*, pokud mu tedy nenastavíme pozadí a podobně jako u *div* elementu. Tomuto blokovému prvku jde nastavit všechny možné vlastnosti, od pozice, velikosti a samozřejmě třeba taky layoutu, tedy způsobu skládání dalších objektů – a to buď vertikálně za sebou, horizontálně vedle sebe, či absolutní layout, kde hraje roli nastavení pozice, a objekty se mohou překrývat. Navíc objekt *Container* přináší do naší aplikace ještě jednu, řekl bych nejdůležitější vlastnost a to tu, že každý *Container* se skládá opět ze scén, tj. obsahuje objekt *Scenes* stejně jako náš program na začátku – tady vidíme, proč je *Container* nejzákladnější objekt – celý program na začátku je *Container*. Využití této funkce je třeba u menu, kdy si vytvoříme container menu, kam rozmístíme navigační tlačítka. Pod container menu vytvoříme další container pro obsah, kde do každé scény dáme nějaký obsah. Tlačítkům v menu jen nastavíme události a akce přepínání scén v obsahové části a máme aplikace, kde není třeba vykreslovat menu vícekrát a mění se jen obsahová část.

Objekt *Graphic* slouží na vykreslování grafických podkladů všech datových typů (respektive všech datových typů podporovaných technologií Flash a Adobe Air). Objekt se vytvoří buď, vykreslením grafického containeru na scénu, nebo pouhým přetažením objektu z knihovny (panel library si popíšeme později). Objekt *Graphic,* nemá žádné scény jako objekt *Container*, pouze se stará o zobrazení grafických dat. To se děje tak, že objekt pošle požadavek přes *Communication Layer* do knihovny *Loaders Library*, kterou jsme si popsali v *Core Layer*. Knihovna po načtení souboru posílá zpět už jen obrazová data, která jsou vykreslena.

Objekt *Video* je velice podobný objektu *Graphic* s jediným rozdílem, že se stará o vykreslení videa. Vše probíhá úplně stejně. Jediný podstatný rozdíl, který je dobré zmínit, je že video umí přehrávat formát *.flv a rovněž může přehrávat obraz z webkamery. Další rozdíl oproti *Graphic* objektu je ten, že video nemůže být přibaleno do výsledného *.swf souboru jako *Graphic* objekty, nýbrž je do aplikace streamován.

Dále si popíšeme panely, které modul *Editor* na vrstvě *Plugin Layer* obsahuje:

- Component panel
- Container panel
- Event panel
- Library panel
- Properties panel
- Scene panel
- Tools panel

Panel *Component*, který obsahuje předpřipravené komponenty, které jsou složeny ze základních objektů – tedy prvků *Container*, *Graphic* či *Video*. Jako příklad bych zde uvedl nejjednodušší komponentu a tou je tlačítko. Tlačítko obsahuje dvě scény, kde na každé scéně je objekt *Graphic*. To nám představuje stavy tlačítka, kdy je tlačítko normálně zobrazeno a druhý stav je při najetí myši na tlačítko, tedy nějaký hover efekt. Tato komponenta má už i nadefinované události – právě třeba po najetí a odjetí myši, se mění scéna s grafickými podklady.

Panel *Container* umožňuje v aplikaci vytvářet, upravovat, mazat a obecně spravovat všechny objekty na jedné scéně. V seznamu objektů je jméno containerů a typ (*Container*, *Graphic*, *Video*). Tento panel funguje i jako správa vrstev, které známe z grafických editorů (Photoshop, Gimp,...). Jen zde fungují opačně, tedy vrstva nahoře je vykreslována dole. Je to z toho důvodu, aby nové objekty se řadily v seznamu dolů, ale samozřejmě nový objekt musí být vykreslen nahoře.

Event panel obsahuje rozhraní pro přidávání, upravování a mazání události označeného objektu na scéně. Uživatel po označení objektu může přidat událost ze seznamu – tento seznam obsahuje pouze události, které jdou přidat na daný objekt, tedy vždy projde filtrem, který se postará o relevantní výsledky. Dále po vybrání události se musí vybrat cíl, tedy jaký objekt, například po kliknutí myši, budeme ovlivňovat. Další věc je akce, kterou aplikujeme na námi vybraný cíl – to může být třeba změna scény, či přehrání videa. Opět akce jsou závislé na typu cíle, uživateli se tedy nestane, že by dostal irelevantní data. Poslední krok je nastavit hodnotu, která se přenese s akcí – u změny scény je to název scény, který se má změnit.

Library panel slouží ke správě veškerých podkladů – tedy grafiky programu, videí, zvuků a dalšího. Pokud uživatel chce importovat soubory do aplikace, panel pošle požadavek přes *Communication Layer* do knihovny *Air Library*, která uživateli nabídne dialogové okno pro výběr (i hromadný výběr) souborů, pošle cesty k souborům, které se uloží v knihovně.

Properties panel obsahuje všechna možná nastavení, od nastavení programu, tedy rozlišení scény, barvy pozadí až po nastavení jednotlivých věcí, jako jsou objekty, které byly pospány výše. Dle označení konkrétního objektu, či stavu aplikace, se v panelu *Properties* zobrazí dané nastavené vlastností – například při vybrání *Container* objektu se uživateli ukáží vlastnosti tohoto objektu, tedy pozice, velikost atd.

Scene panel pracuje obdobně jako *Container* panel, s tím rozdílem, že místo objektů se stará o scény. Tedy dovoluje vytvářet, editovat a mazat scény. Je zde několik omezení – vždy musí zůstat alespoň jedna scéna, název scény musí být na jedné úrovni jedinečný a název nesmí být prázdný řetězec. Dále, ve *Scéně* panelu, se kromě názvu scény, dá nastavit, která scéna má být startovací, tedy která ze scén se zobrazí první.

Tools panel obsahuje pouze přepínání režimu modulu *Editor*. Jedná se o režim SELECT (režim výběru objektů), CREATE (režim vytváření objektů) a režim RESIZE (režim úpravy velikosti objektů). Kromě změny módu také obsahuje další nastavení, jako třeba nastavení kreslících nástrojů, kdy jde přepnout tvorbu nových objektů na existující typy (*Container*, *Graphic* a *Video*).

5.2.3.1 Preview

Knihovna *Preview* není v současné aplikaci implementována, přesto popíšu připravené rozhraní, na které by se implementace této knihovny nasadila. Z návrhu víme, že tato vrstva slouží k náhledu aplikace, aby si uživatel mohl aplikaci v rychlosti prohlédnout a nemusel ji exportovat do výsledného formátu a spustí se mu ve virtuálním prostředí aplikace. Tento náhled nemá zabrat žádný čas navíc, to je díky využití datových struktur, které jsou obsaženy v knihovně *Editor*. Knihovna *Preview* pošle požadavek právě na knihovnu *Editor* přes komunikační vrstvu *Communication Layer*, přes kterou posléze dostane odpověď ve formě dat – jediné, o co se musí tato knihovna postarat, je vykreslení scény.

5.2.3.2 Export

Knihovna *Export* slouží k vyexportování aplikace do požadovaného formátu. V současné době je implementován jediný formát, který jsme si zmínili v návrhu aplikace – tedy Flashový formát a výstupem je Flashový soubor spolu s webou stránkou, kam je zasazen.

Export se dělí na dvě základní části:

- Export do meziformátu
- Export do cílového formátu

Export do meziformátu

Exportem do meziformátu je myšlen export do xml sturktury, který se provede v první fázi exportu. Tento export probíhá na požadavek z *Communication Layer* a provádí se v knihovně *Editor*. XML meziformát by se dal rozdělit na dvě části a to nastavení celého dokumentu (velikost, barva pozadí,...) a část druhou, to je samotná struktura tvořeného formátu, mezi kterou patří rozdělení scén, rozmístění objektů a jejich parametrů, včetně přidaných událostí.

Export do cílového formátu

Tento export spočívá v překladu meziformátu do konečného formátu. To se děje již v knihovně *Export*. Tato knihovna zatím nemá implementované uživatelské rozhraní a celý export probíhá na pozadí. V budoucnu je plánováno přidat možnosti úprav exportu (výběr formátů a dalšího nastavení) a taky infomrací o právě probíhající konverzi(o tom více později). Knihovna *Export* po obdržení odpovědi na její požadavek, který je v aplikačním meziformátu, zkopíruje všechny potřebné soubory do projektových složek, vytvoří konfigurační soubory aplikace a nachystá vše potřebné pro kompilátor. Aktuálně, jak bylo výše popsáno, je podporován export do Flashového formátu *.swf a tedy jsou veškeré soubory generovány v jazyce Action Script 3.0. Následně je volán Flashový kompilátor, který musí mít uživatel nainstalován na svém počítači a musí mít rovněž nastavenou cestu k tomuto kompilátoru v nastavení své aplikace. Po úspěšném zkompilování projektu je vyvolán dotaz na uložení exportovaných souborů do vybrané lokace uživatele.

5.2.4 Application Layer

Vrstva *Application Layer* je nevyšší vrstvou zapouzdření programu. Jak jsme si uvedli v návrhu – tato vrstva je nástavbou *Engine* Layer. Implementačně je vše vyřešeno pomocí importování knihovny *Engine* do aplikace.

Vrstva *Application Layer* neobsahuje žádné uživatelské rozhraní, pouze obsahuje místo, kam je vložen kompletní program, to nám umožňuje velice snadno měnit cílový formát aplikace. V aktuálním návrhu máme možnost vložit aplikaci do dvou podporovaných formátů, které vybraná technologie Flash podporuje.

A to do formátu:

- Adobe Air
- Adobe Flash

Základní rozdíl je tedy v tom, že Adobe Air nám vytváří desktopovou aplikaci, která funguje jako kterákoliv jiná desktopová aplikce a přináší nám možné funkční rozšíření technologie Adobe Air. V případě Adobe Flash, by naše aplikace fungovala jako webová a byla by spouštěna ve Flash

Player pluginu webového prohlížeče. V rámci této práce je odevzdána Adobe Air varianta vrstvy *Application Layer* a to z důvodu potřeby volání Flashového kompilátoru, který je nainstalován na uživatelově počítači. Do budoucna je však aplikace výborně připravena, aby mohla existovat jako webová aplikace (více později, v části Závěr).

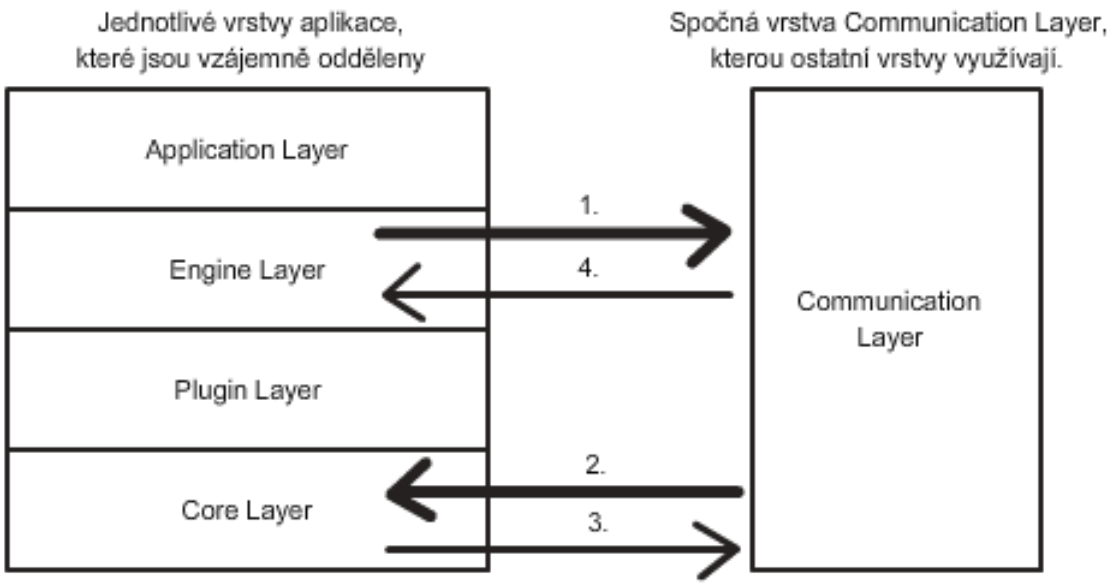
Jediné, co je potřeba udělat v našem projektu v programu Adobe Flash Builder je importovat do Adobe Air nebo Adobe Flash aplikace vrstvu *Engine Layer*, tedy jedinou knihovnu, která tato vrstva obsahuje – *Engine Library* a umístit hlavní komponentu programu *Editor_comp* do vybrané aplikace (Adobe Air nebo Adobe Flash).

5.2.5 Communication Layer

Vrstva *Communication Layer* obsahuje jedinou knihovnu *Gateway Library*. Jako jediná tato knihovna je obsažena v každé knihovně v aplikaci, aby kterákoliv část programu mohla komunikovat přes toto rozhraní napříč programem.

Veškerá komunikace je zabezpečena skrz připravené metody a tak jednotlivé knihovny nemají přímý přístup do dalších knihoven či objektů, hlavně co se týče jiných vrstev. Pokud potřebujeme komunikovat mezi dvěma knihovnami, musíme přidat do vrstvy *Communivation Layer* metodu, která bude celou komunikaci zprostředkovávat.

V této sekci si blíže představíme celou komunikaci. V návrhu byl kladen důraz na schopnost téměř jakoukoliv knihovnu v programu oddělit či přidat a to za co nejmenšího počtu vynaložených úprav aplikace. Proto veškerá komunikace je vedena přes události a ne přímým voláním metod jednotlivých objektů v knihovnách. V programovacím jazyce Action Script jsou události zachytávány tzv. Listenery, tedy nějakými naslouchači událostí. Každá událost, musí mít svůj cíl, kde je vyvolána přes takzvaný Event Dispatcher. Jako cíl, je ve většině případů komunikace určeno samotné jádro v *Core Layer* – tedy knihovna *Core Library*. V některých případech funguje cílení přímo do konkrétního modulu či panelu, ale to jen za přepdpokladu, že knihovna, která vznáší dotaz je ve stejné vrstvě jako cílová knihovna a i přesto musí být komunikace vedena přes tuto vrstvu, nikoliv napřímo. Jediný rozdíl je v prohloubení bezpečnosti některých komunikačních cest a omezení komunikace z jiných vrstev, než kam patří cílový objekt – tedy jeho knihovna. Tento systém je pro nás velice důležitý právě v příkladě odebrání některé z knihoven, protože pokud nějakou knihovnu odebereme, nestane se nic jiného než to, že na daný požadavek vrstvy *Communication Layer* nikdo neodpoví a program nespadne, či nevyvolá žádnou vyjímku, jen příjdeme o funkce, které odebraná knihovna podporovala a to je i požadovaný záměr.



Obrázek 5.3: Ukázka komunikace – Vrstva *Engine Layer* vznáší požadavek přes komunikační rozhraní, které ho předává cílové vrstvě, tedy *Core Layer*, tak po provedení požadavku vrací odpověď opět přes komunikační rozhraní.

6 Testování

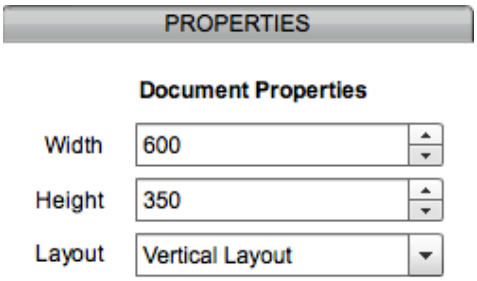
Při návrhu apikace byl požadavek na prezentování programu na příkladu tvorby Flashového video přehrávače. Proto testování proběhlo na tomto demonstračním příkladu, který si uvedeme krok za krokem.

Cíle testování:

- Aplikace bude obsahovat vložené video
- Video bude možné ovládat prvky Play / Pause / Stop
- Po kliknutí na video se aplikace Spustí / Pozastaví
- Aplikace bude zobrazovat informace o načtení videa a přehrávání
- V aplikaci bude možné skákat na požadované místo ve videu

6.1 Nastavení aplikace

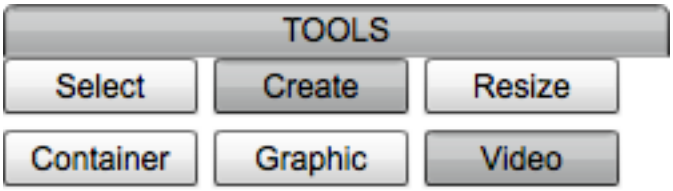
Jako první je dobré si ujasnit, jak velká naše aplikace bude. Tyto parametry si můžeme nastavit v panelu *Properties*, konkrétně v nastavení dokumentu. To se nám zobrazí jak na začátku jako defaultní okno s vlastnostmi, nebo pokud nebude označen žádný objekt na scéně (to docílíme kliknutím do hlavního modulu mimo nějaký objekt). Nastavíme si naše hodnoty na 600px šířku a 350px výšku. Dále je pro jednoduchost skladby aplikace vhodné, zvolit si nějaký layout aplikace, pro naše potřeby si zvolíme *Vertical Layout*.



Obrázek 6.1: Nastavení aplikace

6.2 Video Container

Ústředním prvkem naší aplikace je Video, proto si musíme vytvořit *Video Container*, do kterého posléze naimportujeme požadované video.



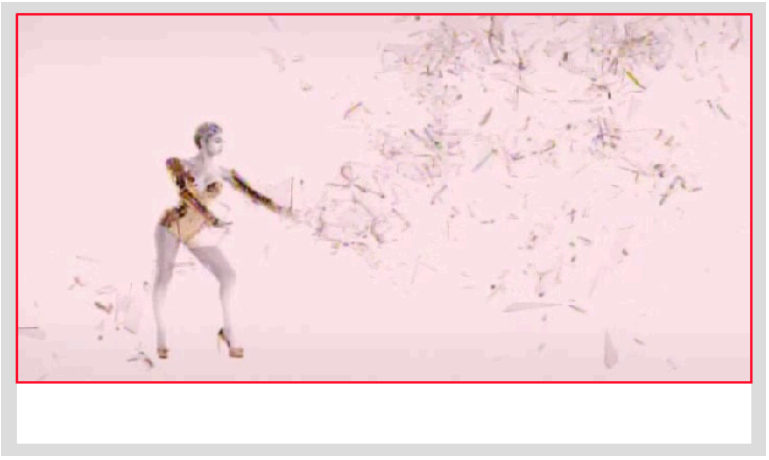
Obrázek 6.2: Vytváření *Video Containeru*

Začneme tím, že si v panelu *Tools* nastavíme možnost tvorby objektů (je defaultně zapnutá při spuštění aplikace) a přepneme typ containeru na Video. Metodou Drag & Drop v editoru nakreslíme náš objekt – všimněte si, že náš objekt se začíná tvořit od souřadnic [0, 0]. To je kvůli nastavenému layoutu aplikace.

Po vytvoření objektu, si ho můžeme v panelu *Properties* ještě upravit – nastavíme mu šířku na 600px a výšku 300px. Dále si náš container přejmenujeme, to uděláme v panelu *Container*. Objekt si vybereme (pokud není) v seznamu a tlačítkem Edit, se dostaneme do možnosti upravit název našeho objektu – pojmenujeme ho video_box.

Samotný container by nám nebyl k ničemu, to nejdůležitější, tedy samotný obsah si přidáme nyní. V panelu *Library* zvolíme možnost importovat soubor a vybereme požadované video ve formátu *.flv (na přiloženém CD je složka Resources, kde je video přibaleno). Po vybrání videa, se nám objeví položka s videem v naší knihovně (import videa může malou chvíli trvat, dané video se totiž kopíruje z vybrané složky do projektové složky aplikace). Po úspěšném importu videa do aplikace nám stačí video metodou Drag & Drop přetáhnout nad náš container, do kterého se při najetí myší nad něj vloží jako náhled a po puštění objektu se video umístí natrvalo (Při tažení videa se nám přehrává v průběhu tažení náhled videa a při přetažení nad editor se nám náhled zvětší na skutečnou velikost videa. Pokud video v tuto chvíli upustíme, vytvoří automaticky *Video Container* o skutečných rozměrech videa).

Po dokončení tohoto kroku máme video na scéně, a jak můžete vidět, přehrává se v containeru živý náhled videa.



Obrázek 6.3: Hotový *Video Container*

6.3 Ovládací prvky

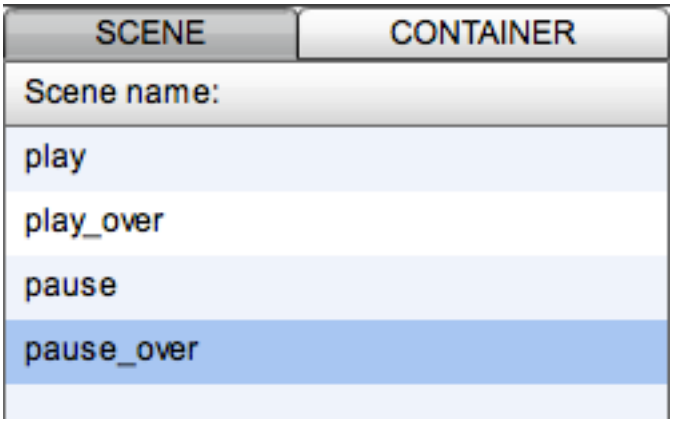
Naší aplikaci budeme muset nějak ovládat, vytvoříme si ovládací komponentu – panel, který bude obsahovat tlačítko Play / Pause, tlačítko Stop a dále Progress Bar, který bude ukazovat načtení videa, přehrávání videa a taky nám bude umožňovat přeskakovat na libovolnou část videa.

Jako první si vytvoříme objekt *Container*, který bude sloužit jako obal celé komponenty. Stejně jako u videa si přepneme panel *Tools* na požadovaný objekt a nakreslíme náš ovládací panel, který ještě v *Properties* upravíme na šířku 600px a výšku 30px. I v tomto případě se nám bude hodit nastavit layout daného objektu – vybereme možnost *Horizontal* Layout. Nyní si panel přejmenujeme stejným způsobem, jak tomu bylo u videa a nazveme ho control_panel. Nakonec si tento container otevřeme. K otevření můžeme použít několik způsobů, buď si v panelu *Tools* přepneme režim na Select, a dvojklikem na náš objekt jej otevřeme, nebo stačí v panelu *Container* rovněž dvakrát kliknout na položku našeho objektu.

Dalším krokem bude vytvoření tlačítka Play / Pause. Aby bylo hezké, budeme chtít, aby když je video zastavené, zobrazoval se znak Play, po najetí na tlačítko se zobrazí over effect tlačítka Play. Pokud video pustíme, musí se tlačítko chovat stejně, ale místo znaku Play zde bude znak Pause. Tedy cílem je vytvořit takové *Toggle* tlačítko.

Proto opět využijeme základní objekt *Container*, který si nakreslíme o velikosti 56px x 30px. Tento container si přejmenujeme na toggle_button a otevřeme si jej.

Nyní si vyzkoušíme práci se scénami, v panelu *Scene* obdobně jako u containeru si nejprve přejmenujeme scénu, která se jmenuje Default na scénu play a dále přidáme 3 další scény: play_over, pause, pause_over.



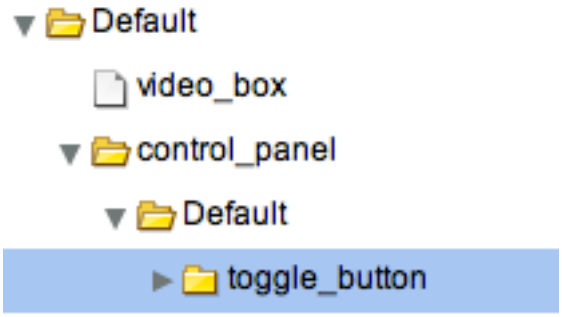
Obrázek 6.4: Rozvržení scén v objektu toggle_button

Dalším krokem je importovat do knihovny 4 obrázky, které budeme potřebovat: play.png, play_over.png, pause.png a pause_over.png (všechny jsou ve složce Resources na CD). Nyní

postupně umístíme do jednotlivých scén požadované obrázky. Scénu si přepneme dvojklikem na její název a poté daný obrázek přetáhneme na pozici [0, 0]. Nyní, pokud je vše hotovo se přepneme do scény play, která by měla vypadat takto:

Obrázek 6.5: Scéna play objektu toggle_button

Nyní si objekt označíme, pokud nemáme a v panelu *Event* přiřadíme potřebné události. Začneme tlačítkem Add, které nám zobrazí strom naší aplikace. V tomto prvním kroku vybíráme cíl, který se po nějaké události bude měnit, respektive se kterým budeme pracovat. My si vybereme náš toggle_button objekt (ten se nachází v Default -> control_panel -> Default -> toggle_button).



Obrázek 6.6: Vybrání cíle toggle_button

V dalším kroku určíme událost, při které se bude vyvolávat nějaká akce. U tlačítka Play nám stačí jediná, a to aby po najetí kurzoru myši nad tento obrázek play.png se přešlo v našem cíli (objekt toggle_button) na scénu play_over. Proto jako událost vybereme MouseEvents.over a v dalším kroku, kde nastavujeme akci, která se po události vykoná, tedy vybereme NavigateAction.changeScene. To je událost, která v objektu cíle (toggle_button) přepne scénu na hodnotu, kterou nastavíme v posledním kroku. Ta bude play_over.

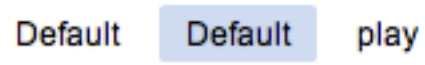
EVENT	
Event	Action
over	changeScene

Obrázek 6.7: Stav panelu *Event* po dokončení přidávání události

Obdobně u dalších obrázků nastavíme události a to takové:

- u obrázku play_over.png nastavíme:
    - cíl video_box, událost MouseEvents.click, akce VideoAction.play a hodnotu nenastavujeme.
    - cíl toggle_button, událost MouseEvents.click, akce NavigateAction.changeScene a hodnotu pause_over
    - cíl toggle_button, událost MouseEvents.out, akce NavigateAction.changeScene a hodnotu pause
- U obrázku pause.png
    - cíl toggle_button, událost MouseEvents.over, akce NavigateAction.changeScene a hodnotu pause_over
- U obrázku pause_over.png
    - cíl video_box, událost MouseEvents.click, akce VideoAction.pause a hodnotu nenastavujeme
    - cíl toggle_button, událost MouseEvents.click, akce NavigateAction.changeScene a hodnotu play_over
    - cíl toggle_button, událost MouseEvents.out, akce NavigateAction.changeScene a hodnotu play

Nyní se musíme přesunout do objektu control_panel, to uděláme v navigaci, která je součástí editoru nahoře a přepneme se na scénu Default, druhou zprava, což je scéna právě našeho containeru control_panel.



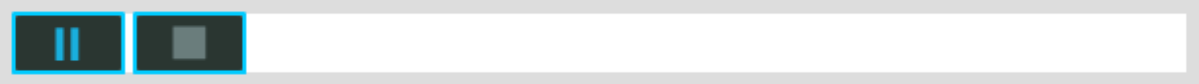
Obrázek 6.8: Navigace v modulu *Editor*

Zde si vytvoříme ještě tlačítko stop a poté náš progress bar. Začneme tlačítkem stop, opět si vytvoříme objekt *Container* o stejné velikosti jako je toggle_button, který se díky layoutu zarovnal automaticky vedle tlačítka toggle_button. Nový objekt si přejmenujeme na stop_button a otevřeme si ho. Nyní si v něm vytvoříme dvě scény: stop a stop_over, kde scénu stop vytvoříme přejmenováním Defaultní scény. Do knihovny si jako v případu toggle_button objektu naimportujeme obrázky stop.png a stop_over.png, které umístíme do požadovaných sekcí. Poslední krok k úspěšchu, je nastavit události:

- U obrázku stop.png
    - cíl stop_button, událost MouseEvents.over, akce NavigateAction.changeScene a hodnotu stop_over

- U obrázku stop_over.png
    - cíl video_box, událost MouseEvents.click, akce VideoAction.stop a hodnotu nenastavujeme
    - cíl stop_button, událost MouseEvents.out, akce NavigateAction.changeScene a hodnotu stop

Pokud se nyní přepneme do scény stop a následně v navigaci přejdeme do objektu control_panel. Vše by mělo vypadat takto:



Obrázek 6.9: Objekt control_panel

Posledním prvkem je náš progress bar, vytvoříme si opět objekt *Container*, pojmenujeme ho progress_bar a nastavíme mu velikost 280px x 10px. Následně si jej otevřeme a do knihovny naimportujeme soubory: loading1.png, loading2.png a loading3.png . V dalším kroku je přetáhneme do scény v pořadí loading3.png, loading1.png a loading2.png. Pozor na nastavení layoutu v objektu progress_bar, musíme mít absolutní layout, aby se dané obrázky překryly. Nakonec ještě přejmenujeme objekty, loading3.png nazveme background, loading1.png buffer a loading2.png nazveme timer.

V navigaci přejdeme do objektu control_panel a na objekt progress_bar přidáme událost:

- cíl video_box, událost MouseEvents.click, akce VideActions.seekRatio a typ hodnoty nastavíme na xRatio

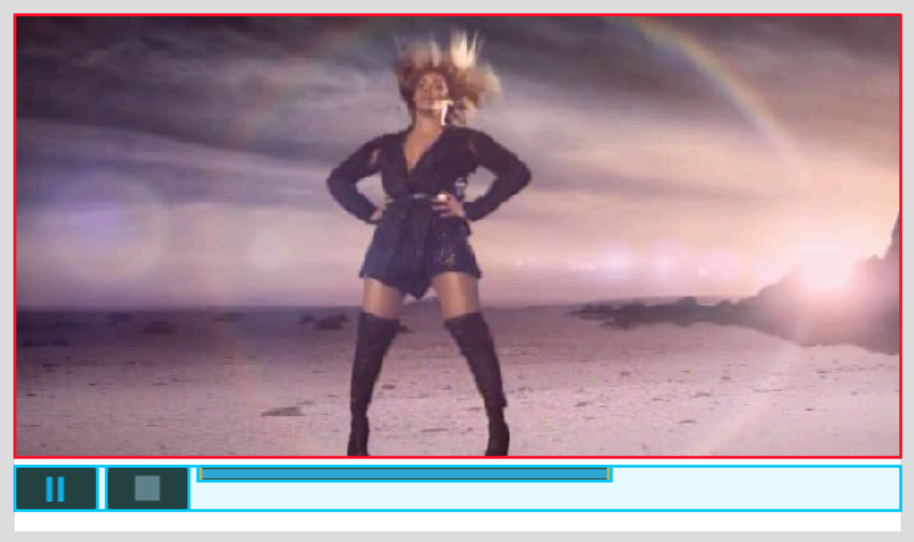
Tím docílíme přeskakování videa po kliknutí na komponentu progress_bar.

Na závěr ještě přidáme nějaké události na objekt video_box, proto budeme muset přejít v navigaci na scénu Default, nejvíce vlevo, která představuje základní plochu celé aplikace, ve které se video_box nachází. Zde si označíme objekt video_box a v panelu *Event* mu přidáme pár událostí:

- cíl toggle_button, událost VideoEvents.play, akce NavigateAction.changeScene a hodnotu pause
- cíl toggle_button, událost VideoEvents.pause, akce NavigateAction.changeScene a hodnotu play
- cíl video_box, událost MouseEvents.click, akce VideoAction.toggle a hodnotu nenastavujeme
- cíl timer, událost VideoEvents.timeChange, akce PropertieAction.scaleX a typ hodnoty timeRatio

- cíl buffer, událost VideoEvents.bufferChange, akce PropertieAction.scaleX a typ hodnoty bufferRatio

Nyní je naše aplikace hotova (samozřejmě si s rozvržením prvků můžeme dále vyhrát, ale jako rychlé demo tohle uplně stačí) a můžeme ji vyexportovat (hlavní horizontální menu – Project -> Export), ve výsledku bude vypadat hlavní scéna takto:



Obrázek 6.10: Finální rozvržení scény

7 Závěr

Na závěr bych chtěl vyjádřit uspokojení, které mi tento projekt přinesl. Již delší dobu jsem se chtěl pustit do takového projektu, a proto jsem si ho vybral za téma v bakalářské práci. Přesvědčil jsem se, že můj odhad na začátku byl správný a daná aplikace může fungovat. Bylo tu určité riziko, se kterým jsem do projektu šel a na úplném konci mohu zhodnotit práci za velmi uspokojivou. Samozřejmě kvůli časovému tlaku není zcela dle mých představ, ale věřím, že jednou bude, protože hodlám v tomto projektu velice aktivně pokračovat.

Ještě jednou shrnu, co bylo cílem práce. Za úkol bylo udělat aplikaci na generování uživatelských rozhraní, včetně přiřazování funkcí – daný program byl prezentován na ukázkovém příkladu Flashového video přehrávače. Musím konstatovat, že jsem daného cíle se dosáhlo, a že se cesta editoru, který nebude potřebovat psaní programovacího kódu pro určité typy aplikací našel.

Je zde i na místě uvést plány do budoucna. Jistě si chci stanovit velké cíle, aby program byl úspěšný. Nejdříve chci implementovat části, které byly navrženy, ale nebyly implementovány – jedná se o knihovnu *Preview*, kterou považuji za klíčovou. Dále chci doplnit podporu všech vlastností, pro Flashovou platformu a poté začít připravovat další platformy (webové služby, exporty na mobilní zařízení, ...). Rád bych také celou aplikaci uvedl do online měřítka a celou záležitost exportování aplikace prováděl online na serveru – tedy využitím Cloud Computingu. Jistě v budoucnu nastane i potřeba opustit technologii Adobe Air, na které dosavadní aplikace stojí, a to kvůli spotřebě výkonu. Je to taky jeden z cílů vzdálenější budoucnosti.

Literatura

[1] Craig J,, Cooper M., Pappas L., Schwerdtfeger R., Seeman L.: Accesible Rich Internet Applications, W3C publication [online]. c2008-2011, aktualizováno 2011-01-18 [cit. 2011-05-14]. Dostupné na URL:<http://www.w3.org/TR/wai-aria/complete>

[2] Webová stránka: Rich Internet Application [online]. Aktualizováno 2011-04-30 [cit. 2011-05-14].
Dostupné na URL:<http://en.wikipedia.org/wiki/Rich_Internet_application>

[3] Apple Inc.: Remote Scripting with IFRAME [online]. c2011 [cit. 2011-05-14]. Dostupné na URL:<http://developer.apple.com/internet/webcontent/iframe.html>

[4] Keyes J.: X Internet: the executable and extendable Internet. 2007. ISBN 0849304180

[5] Bauer G.: The future of the Web: Rich Clients, Roch Browsers, Rich Portals [online]. Vancover: 2003. [cit. 2011-05-14]. Dostupné na URL:<http://luxor-xul.sourceforge.net/talk/vanx-mar-2003/slides.html>

[6] Webová stránka: Adobe Flash [online]. Adobe Systems Inc. [cit. 2011-05-14]. Dostupné na URL:<http://www.adobe.com/products/flash.html>

[7] Webová stránka: Adobe Flash Player [online]. Adobe Systems Inc. [cit. 2011-05-14]. Dostupné na URL:<http://www.adobe.com/products/flashplayer/>

[8] Zakas N., McPeak J., Fawcett J.: Professional Ajax. Zoner Press, 2006. ISBN 978-0-471-77778-6

[9] Webová stránka: Microsoft Silverlight [online]. Microsoft . [cit. 2011-05-14]. Dostupné na URL:<http://www.microsoft.com/cze/web/silverlight/>

[10] Webová stránka: .Net Framework Overview [online]. Microsoft. [cit. 2011-05-14]. Dostupné na URL:<http://www.microsoft.com/net/overview.aspx>

[11] Webová stránka: Java [online]. Oracle. [cit. 2011-05-14]. Dostupné na URL:<http://www.oracle.com/technetwork/java/javase/overview/index.html>

[12] Webová stránka: Shiva3D Editor [online]. Stonetrip, c2003-2011. [cit. 2011-05-14]. Dostupné na URL:<http://www.stonetrip.com/what-is-shiva-3d.html>

[13] Webová stránka: Unity3D [online]. Unity, [cit. 2011-05-14]. Dostupné na URL:<http://unity3d.com/>

[14] Webová stránka: Molehill [online]. Adobe Systems Inc. [cit. 2011-05-14]. Dostupné na URL:<http://labs.adobe.com/technologies/flashplatformruntimes/incubator/features/molehill.html>

[15] Hickson I.: HTML 5, W3C publication [online]. c2011, aktualizováno 2011-04-05 [cit. 2011-05-14]. Dostupné na URL:<http://www.w3.org/TR/html5/>

[16] Webová stránka: Adobe Flash Builder [online]. Adobe Systems Inc. [cit. 2011-05-14]. Dostupné na URL:<http://www.adobe.com/products/flash-builder.html>

[17] Webová stránka: Adobe Dreamweaver [online]. Adobe Systems Inc. [cit. 2011-05-14]. Dostupné na URL:<http://www.adobe.com/products/dreamweaver.html?promoid=DJDSZ>

[18] Webová stránka: Adobe Flash Catalyst [online]. Adobe Systems Inc. [cit. 2011-05-14]. Dostupné na URL:<http://www.adobe.com/products/flashcatalyst.html?promoid=GYSOG>

[19] Technická specifikace: SWF File Format Specification [online]. Adobe Systems Inc., 2008 [cit. 2011-05-14]. Dostupné na URL: <http://www.adobe.com/content/dam/Adobe/en/devnet/swf/pdf/swf_file_format_spec_v10.pdf>

[20] Weobvá stránka: Action Script 3.0 overview [online]. Adobe Systems Inc. [cit. 2011-05-14]. Dostupné na URL: <http://www.adobe.com/devnet/actionscript/articles/actionscript3_overview.html>

[21] Yergeau F., Maler E., Sperberg-McQueen C., Paoli J., Bray T.: Extensible Markup Language (XML) 1.0, W3C publication [online]. c2008, aktualizováno 2008-21-26 [cit. 2011-05-14]. Dostupné na URL:<http://www.w3.org/TR/xml/>

[22] Gassner G.: Flash Builder 4 and Flex 4 Bible [online]. Wiley Publisher, Inc. 2010. ISBN978-0-470-48895-9

Příloha A

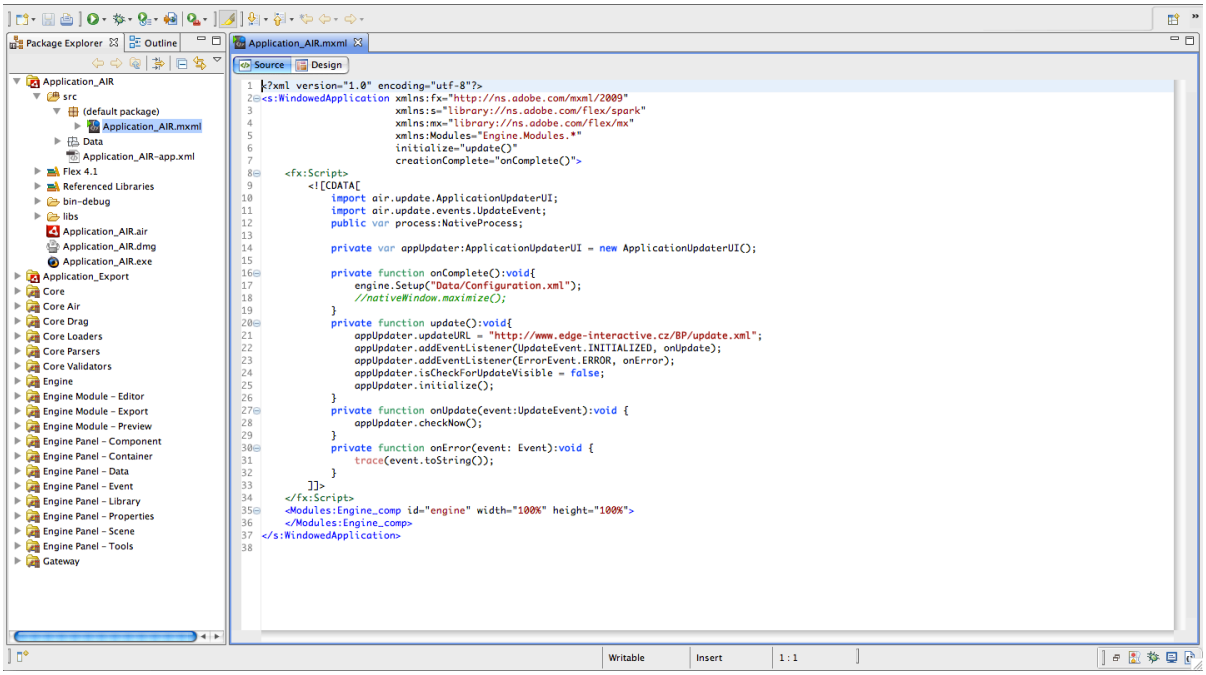
Obsah CD

CD obsahuje tyto adresáře a soubory:

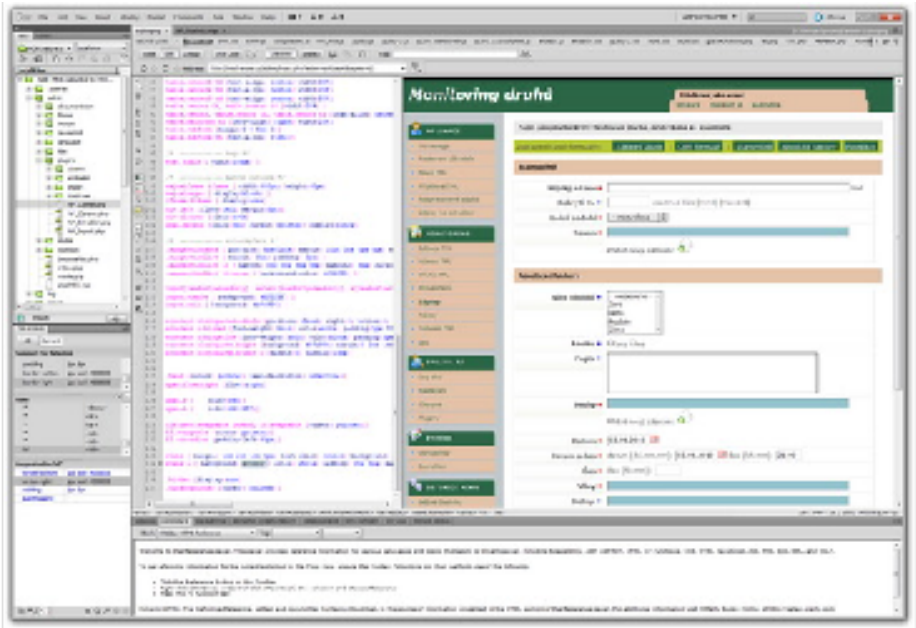
- BP.pdf – tato dokumentace
- Dokumentace.pdf – programová dokumentace a nápověda
- Resources – složka obsahující zdrojovou grafiku a video, které bylo použito při tvorbě testovací aplikace
- Export – složka obsahující vyexportovaný testovací program
- Sources – složka obsahující veškeré zdrojové kódy pro Adobe Flash Builder vývojové prostředí.
- RiaCreator.exe a RiaCreator.dmg – instalační soubory programu pro platformy Microsoft Windows a Mac Os X.

Příloha B

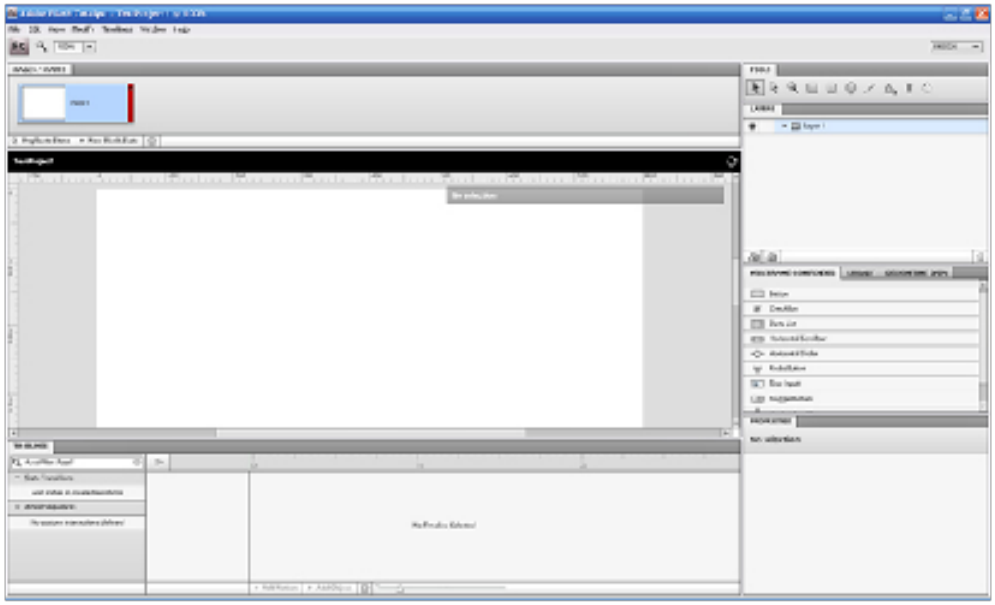
Ukázka konkurenčních vývojových prostředí



Obrázek B.1: Prostředí Adobe Flash Builder



Obrázek B.2: Prostředí Adobe Dreamweaver (zdroj: Internet)



Obrázek B.2: Prostředí Adobe Catalyst (zdroj: Internet)

Příloha C

Ukázka aplikace