Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra informačních technologií

Studijní program: Aplikovaná informatika

Obor: Informační systémy a technologie

Framework RichFaces

DIPLOMOVÁ PRÁCE

Student : Barbora Kořistková

Vedoucí : Ing. Jarmila Pavlíčková

Oponent : Ing. Jana Válková

2014

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracovala samostatně a že jsem uvedla všechny použité prameny a literaturu, ze které jsem čerpala.

V Praze dne 7. května 2014 . . . . . .

Barbora Kořistková

Poděkování

Na tomto místě bych chtěla poděkovat Ing. Jarmile Pavlíčkové za skvělé vedení a neocenitelné rady během tvorby diplomové práce i celého studia.

Abstrakt

Tato diplomová práce se zabývá důkladnou analýzou a ohodnocením komponentového frameworku JBoss Richfaces s ohledem na jeho konkrétní silné a slabé stránky. Framework je zkoumán jak z teoretického, tak z praktického hlediska. Mimo to je podroben komparativní analýze, která vztahuje jeho vlastnosti k ostatním produktům podobného zaměření, které se na trhu vyskytují. Kromě teoretického pozadí problému a detailního objasnění problematiky zahrnuje práce také praktickou část, která se skládá z plnohodnotné referenční aplikace s fragmenty komentovaného kódu a důkladného vysvětlení praktického využití frameworku.

Klíčová slova

Komponentový framework, vývoj webových aplikací, JavaServer Faces

Abstract

The thesis is concerned with an in-depth analysis and evaluation of the JBoss RichFaces component framework with regards to its particular strengths and weaknesses. The framework is examined both from the theoretical and practical point of view and also subjected to comparative analysis with other similar products on the market. Apart from the theoretical background and profound subject explanation, the practical part of the thesis offers a full-fledged referential application with fragments of commented code and thorough exploration of the framework's professional usage.

Keywords

Component framework, web application development, JavaServer Faces

Obsah

1 Úvod

Tato práce se bude zabývat zevrubnou analýzou webového frameworku RichFaces vydávaného americkou společností Red Hat Inc. Výše zmíněný framework představuje open source řešení pro snadné a rychlé vytváření prezentační vrstvy webových aplikací psaných v jazyce Java. Svým zaměřením je určen zejména pro podnikové využití.

Analýza frameworku bude zahrnovat identifikaci a ohodnocení jeho relativních silných a slabých stránek vzhledem k vybraným konkurenčním produktům na trhu, a prozkoumání technického řešení frameworku a technologií, na nichž je postaven. V neposlední řadě se tato práce bude také věnovat formám užití frameworku, práci s jeho komponentami a omezeními, popř. výhodami, které jeho zapojení do vývoje aplikace přináší. Veškerá analytická činnost se bude soustředit na využití nástroje v korporátním prostředí.

Téma z oblasti komplexního vývoje podnikových aplikací jsem zvolila proto, že mi vzhledem k mému odbornému zaměření a profesní historii je velmi blízké a představuje pro mě možnost podrobněji analyzovat funkcionalitu a použití jednoho z předních nástrojů v této oblasti.

Zároveň se jedná o nástroj, se kterým jsem se ve svém profesním životě krátce setkala, tudíž mohu čerpat i z osobních zkušeností. Mimo to disponuji i detailní znalostí velmi podobného konkurenčního nástroje PrimeFaces od turecké společnosti PrimeTek, díky čemuž mohu lépe srovnávat výhody a nevýhody plynoucí z užití jednotlivých variant.

Tato práce sleduje několik dílčích cílů, jejichž naplnění bylo přizpůsobeno i členění celého textu. Tyto cíle jsou následující:

1. Vymezení podstatných odborných pojmů pro lepší orientaci a porozumění čtenáře
2. Analýza konkurenčního prostředí na trhu webových frameworků
3. Teoretická analýza struktury a funkcionality frameworku
4. Analýza akademických prací ze stejné problémové oblasti, popř. na příbuzná témata
5. Praktické zhodnocení užití frameworku včetně identifikace omezení a nedostatků

Prvnímu cíli, tedy vymezení důležitých termínů z problémové oblasti, se práce věnuje hned ve své první obsahové kapitole. Tato kapitola definuje klíčové pojmy pro daný tematický okruh, vysvětluje jejich význam, resp. význam, v jakém budou používány v následném textu, což zajišťuje bezpečné dosažení tohoto cíle.

Třetí kapitola se souhrnně věnuje dosažení druhého a třetího cíle, tj. analýze konkurenčních produktů a jejich vzájemnému srovnání s frameworkem RichFaces a dále analýze struktury a funkcionality frameworku samotného. Nástroje vybrané ke srovnávání jsou

podrobeny vícekriteriálnímu srovnávání a výsledné bodové ohodnocení je uvedeno do kontextu a okomentováno. Tento proces slouží k naplnění druhého cíle.

Pro dosažení třetího cíle zkoumá relevantní část třetí kapitoly technologickou strukturu frameworku, tj. na jakých technologiích je založen, jaká jsou jeho omezení a jaké má vyhlídky na budoucí rozvoj. Dále je v tomto kontextu zkoumáno typické využití a charakteristické rysy frameworku.

Rešerší akademických prací na příbuzná témata či témata ze stejné problémové oblasti se věnuje čtvrtá kapitola. V tomto oddílu je provedena analýza prací z českých i zahraničních univerzit, které se týkají webových frameworků a jejich vzájemného srovnávání. Texty jednotlivých prací byly získány z odpovídajících úložišť závěrečných prací mateřských institucí. Práce z domácích univerzit jsou uváděný jako první, světové akademické texty jsou analyzovány jako druhé v pořadí.

Poslední obsahová kapitola demonstruje na vývoji referenční aplikace možnosti využití nástroje a jeho komponent, možné komplikace a omezení plynoucí z jeho zapojení do projektů. Popis a zhodnocení jednotlivých části funkcionality je doplněn o názorné ukázky kódu, ať už se jedná o konfiguraci, zdrojový kód v jazyce Java nebo samotné použití komponent frameworku.

Tato práce předpokládá jistou odbornou erudici na straně čtenáře, tj. je určena pro publikum z řad odborníků v oboru informačních technologií, popř. o zájemce o tuto oblast. Text práce vyžaduje alespoň obecnou znalost programování a algoritmizace přinejmenším na úrovni porozumění zdrojovému kódu. Čtenář by proto měl být obeznámen s programovacím jazykem Java nebo jiným jazykem s obdobnou syntaxí.

Vzhledem k zaměření práce na oblast webových aplikací je také předpokládána jistá orientace v této oblasti, zejména co se způsobu zpracování požadavků a generování výsledných stránek týče.

Struktura tohoto akademického textu byla již naznačena v části pojednávající o cílech celé práce. Tato úvodní část se zabývá představením zkoumané problematiky, zaměření práce a shrnutím toho, čemu se text bude v dalších kapitolách věnovat. Druhá kapitola vymezuje významné odborné termíny, jichž bude v textu hojně užíváno, a tedy je potřeba stanovit vyznění, v němž budou v tomto kontextu chápány.

Třetí kapitola podrobně zkoumá strukturu, historii i budoucnost frameworku RichFaces. Věnuje se také analýze jeho konkurentů na trhu a zhodnocení silných a slabých stránek jednotlivých nástrojů. Čtvrtá kapitola uzavírá teoretickou část práce a obsahuje zevrubnou rešerši akademických prací, které byly napsány na příbuzná témata, ve snaze inspirovat se přístupem jiných autorů a také jejich pojetí kriticky zhodnotit.

Pátá kapitola představuje praktickou část práce. Zaobírá se vývojem referenční aplikace, na němž ukazuje způsob práce s frameworkem, různé alternativní cesty pro dosažení téhož efektu, omezení zejména na straně funkcionality a konfigurace, a také komentuje překážky,

které se při vývoji aplikace vyskytly. Závěrečná část pak shrnuje obsah a účel práce, hodnotí dosažení cílů a framework jako takový.

V návaznosti na existenci vícero cílů je třeba i výstupy práce zařadit do různých kategorií. Mezi očekávanými produkty práce pak lze identifikovat následující dílčí celky:

1. Vícekriteriální analýza konkurenčního prostředí na trhu webových frameworků s kvantifikovatelnými výsledky
2. Analýza akademických přístupů k porovnání webových frameworků v domácím i zahraničním univerzitním prostředí
3. Referenční aplikace s komentovaným kódem

První dva výstupy jsou spíše akademické povahy a mohou sloužit jako základ pro další analýzu nástrojů pro vývoj prezentační vrstvy webových aplikací. Lze na ně navázat mnoha způsoby – např. lze obohatit základnu srovnávaných produktů, rozšířit, zúžit či modifikovat množinu kritérií, popř. provést přehodnocení vah, které jsou jim přiřazeny, s ohledem na konkrétní zaměření následné analýzy. Dále lze zahrnout větší množství akademických prací ze širšího okruhu.

Tyto výstupy jsou určeny pro zájemce o výzkum v oblasti trhu s produkty výše zmíněného typu, popř. odborníkům z oblasti informačních technologií, kteří se potřebují před zahájením projektu rozhodnout, které nástroje do své aplikace zapojí.

Referenční aplikace s komentovaným kódem představuje prakticky využitelný výstup, který je určen zejména softwarovým vývojářům z oblasti webových aplikací. Představuje konzistentní ukázku možností a využití frameworku s konkrétními příklady nastavení i použití jednotlivých komponent. Komentář pak poskytuje cenné informace o možnostech a omezeních nástroje, o formách spolupráce a v některých případech až symbióze s jinými nástroji a o problémech, se kterými se vývojář může setkat.

Na závěr této úvodní části je vhodné podotknout, že text práce je psán s ohledem na čtivost při zachování vysoké úrovně odbornosti. Vzhledem k technické povaze tématu se v některých případech nelze vyhnout zapojení anglických výrazů, které se v praxi běžně používají. Anglickou formu termínu text upřednostňuje pouze tehdy, pokud neexistuje dostatečně kompaktní a přesný český ekvivalent.

2 Vymezení základních pojmů

Tato část bude věnována definici některých pojmů, které mají stěžejní význam pro tuto práci a bez jejichž pochopení není možné následnému textu porozumět. Mnohé z těchto pojmů lze chápat více způsoby v různých kontextech, proto je důležité vymezit ten význam, v němž budou používány v mé práci, aby se zamezilo nežádoucím dezinterpretacím zde předkládaného textu.

2.1 Framework

Pod pojmem framework se rozumí mnoho věcí, v případě mé práce vždy půjde konkrétně o softwarový framework. Softwarový framework je modelem určité problémové oblasti nebo jejího důležitého aspektu. Touto oblastí může být technická doména (např. řízení transakcí) nebo byznys doména (např. bankovnictví). Framework poskytuje znovupoužitelný design a implementaci této problémové oblasti (Riehle, 2000). Kód frameworku je obvykle nemodifikovatelný, uživatelské změny jsou možné pouze za pomoci rozšiřování funkcionality ve vhodných místech.

Využívání nejrůznějších frameworků výrazně zefektivňuje a zrychluje vývoj všech typů aplikací, nejen těch webových. Výhodou jejich užití je například skutečnost, že pro každou řešenou oblast lze vybrat nejlepší dostupný nástroj a není nutné se zdržovat vlastním vývojem, když již existuje obdobný a pravděpodobně lépe optimalizovaný nástroj.

Frameworky se samozřejmě dají využít pouze pro opakované problémy, které se objevují v mnohých projektech. Jedná se např. o práci s databází, navigaci ve webových aplikacích či export dat do souborů typu XLS(X)[1] a PDF. Pro unikátní funkcionalitu daného projektu pochopitelně žádný framework existovat nemůže. Využití frameworků tak umožňuje uživateli soustředit se na své jedinečné problémy a typové oblasti řešit pomocí kódu třetí strany.

Při zapojení většího množství frameworků však může nastat problém s jejich integrací, ať už s vlastním jádrem projektu nebo mezi sebou navzájem. Zejména u mladších nástrojů dochází také k problémům se zpětnou kompatibilitou, kdy uživatel v zájmu aktuálních technologií přejde na vyšší verzi frameworku a náhle zjistí, že související kód už nefunguje a aplikace se chová zcela jinak, než byl zvyklý. U zralejších produktů k podobným problémům dochází zřídka, výjimkou je např. framework Tapestry, který se úmyslně zcela vzdal zpětné kompatibility mezi verzemi 4 a 5.

[1] Formáty souborů využívané rozšířeným tabulkovým editorem MS Excel.

Webový framework je pak framework určený pro vývoj webových aplikací (např. RichFaces).

Definicí webové aplikace se zaobírá následující oddíl.

2.2 Webová aplikace

Když v roce 1991 vyšla první verze protokolu HTTP[2], jeho tvůrci předpokládali, že bude sloužit pouze k přenosu jednoduchých HTML[3] stránek bez potřeby náročnějšího formátování. V následujících dvou dekádách ale došlo k obrovskému rozvoji komunikace s využitím internetu a jedním z objevivších se fenoménů se staly právě webové aplikace.

Webová aplikace je v zásadě jakákoliv aplikace, ke které mohou uživatelé přistupovat po internetu prostřednictvím svého prohlížeče. Může se jednat o komplexní podnikové aplikace psané v JEE[4] nebo o jednoduché stránky psané v kombinaci skriptovacích jazyků (často PHP [5]a JavaScript).

V tomto kontextu se také často používá výraz "tenký klient" jako protiváha ke "tlustému klientu" klasicky instalovaných aplikací. Tenký klient v sobě obsahuje pouze prezentační logiku a potřebná data dostává z jiného zdroje (tzv. aplikační server), kdežto tlustý klient zahrnuje všechnu potřebnou funkcionalitu včetně práce s daty a aplikační logiky. V případě webových aplikací je tímto tenkým klientem právě internetový prohlížeč.

Prakticky každé dnes fungující webové stránky se kromě těch nejjednodušších dají nazývat webovou aplikací. Tyto aplikace obvykle slouží k elektronické podpoře prodeje (elektronické obchody), ke komunitní komunikaci (sociální sítě), pro přístup k Vašim penězům (internetové bankovnictví) a myriádě dalších účelů.

2.3 Webový šablonovací systém

Webový šablonovací systém je nepostradatelnou součástí většiny podnikových aplikací. Jedná se o prostředek, který umožňuje do předpřipravených šablon webových stránek doplnit patřičná data (ať už aplikační či z databáze) a předat výsledek do uživatelova prohlí-

---

[2] Hyper-Text Transfer Protocol.

[3] Hyper-Text Markup Language.

[4] Java Enterprise Edition.

[5] PHP: Hypertext Preprocessor.

žeče v dobře zobrazitelné podobě. Šablony webových stránek se skládají z jednotlivých komponent daného šablonovacího jazyka, např.:

Př. 2.3.1: *Jednoduchá komponenta, šablonovací systém Facelets*

```
<h:outputText value="#{sampleBean.getGreeting()}"/>
```

Výsledkem zpracování této komponenty pak bude následující fragment kódu[6]:

Př. 2.3.2: *Výsledek zpracování šablony*

```
<p>Hello World!</p>
```

Výhodou využití šablonovacího systému je kvalitní oddělení prezentační, datové a aplikační logiky v souladu s principem MVC[7]. Šablony dále představují obecné znovupoužitelné prvky, které lze do sebe prakticky libovolně skládat pro dosažení žádaného výsledku, což opět snižuje množství nezbytného úsilí při vývoji webových aplikací a umožňuje rozdělení práce mezi vývojáře a návrháře grafické podoby budoucích stránek. Mezi populární šablonovací systémy patří např. Velocity, Facelets nebo Django.

2.4 Komponenta

Tento pojem se v softwarovém kontextu často skloňuje v nejrůznějších významech a užitích. Pro účely této práce se komponentou rozumí ohraničený znovupoužitelný prvek užívaný při vývoji softwaru.

V tomto textu se slovo “komponenta” vyskytuje v několika spojeních

- Komponenta frameworku – v tomto případě se jedná o stavební kámen daného frameworku, tj. zásadní technologii či koncept, na nichž je framework postaven.
- Komponenta programu – z pohledu programu důležitý objekt (třída).
- Komponenta šablonovacího jazyka – uzavřený prvek šablony webové stránky.

[6] Kód podkladové třídy je úmyslně vynechán pro svou triviální povahu a irelevantnost vzhledem k tématu.

[7] Model-View-Controller.

3 Obecný popis frameworku

Framework JBoss RichFaces (dále jen „framework“) se v podnikové praxi stal neocenitelným pomocníkem pro tvorbu webových aplikací a zejména jejich uživatelského rozhraní. Stejně jako v případě jiných frameworků[8] se jeho komponenty zakládají na prvcích standartu JSF[9], což umožňuje podporu uplatnění zásad oddělení vzhledu aplikace od jejího chování. Vtělením tohoto principu je koncept MVC . Tento framework se řadí do kategorie vhodné pro tvorbu aplikací typu RIA[10] a obsahuje zabudovanou podporu technologie AJAX[11].

Za primární vývoj a správu frameworku zodpovídá společnost Red Hat, Inc. (divize JBoss), jedná se však o open-source produkt. Framework byl kompletně napsán v jazyce Java a je určen pouze pro projekty psané ve stejném jazyce. Často se využívá v kombinaci s oblíbeným frameworkem Spring[12], který se stará o vlastní funkcionalitu aplikace a doplňuje chybějící funkce, jako např. ověřování identity uživatele a jeho práv.[13]

Jako velmi efektivní se také ukazuje spojení RichFaces se Spring WebFlow, což je rozšíření základního Springu pro účely implementace tzv. “toků” aplikací, tj. definování sousledností stránek a podmínek přechodů mezi nimi v čisté a přehledné podobě. Takovým tokem může být např. proces objednávky zboží. V původní verzi standardu JSF nebylo možné toky definovat, nicméně nová verze standardu již tuto funkcionalitu obsahuje.

RichFaces 4 v aktuální verzi 4.3.5 nabízí celkem 83 komponent, z toho 16 s nativní podporou AJAXu. Tyto komponenty pokryjí většinu potřebné funkcionality, nicméně v nabídce zcela chybí např. generování grafů a jiných vizuálních reprezentací dat.

Společnost JBoss také dodává nástroje pro efektivní práci s frameworkem (a jinými produkty této společnosti). Jedná se zejména o rozšíření do vývojového prostředí Eclipse dodávané pod názvem JBoss Tools, které umožňuje vizuální návrh a editaci vlastních komponent přímo uvnitř vývojového prostředí, což výrazně usnadňuje tvorbu nových prvků.

Kolem frameworku se vytvořila čilá vývojářská komunita (ve smyslu vývojářů, kteří užívají RichFaces ve svých projektech), která aktivně spolupracuje na řešení obvyklých i neobvyklých problémů při užívání frameworku a fungují také částečně jako rozhodovací

[8] PrimeFaces, IceFaces aj.
[9] JavaServer Faces.
[10] Rich Internet Application.
[11] Asynchronous Java and XML.
[12] Spring Framework od SpringSource. Open-source nástroj pro podnikové aplikace v Javě. Viz http://www.springsource.org/spring-framework
[13] Angl. authentication and authorization.

těleso, kdy s nimi autoři frameworku konzultují nejpalčivější problémy, které by se měly v budoucích verzích řešit, a nejžádanější nové prvky, které by měli do budoucna zařadit.

3.1 Komponenty frameworku

Aktuální verze frameworku byla vydána 27. ledna 2014. Jedná se o verzi 4.3.5, která je založena na JSF verze 2.0 a vyšší. Od verze 4.0 framework opouští technologii JSP[14] a nahrazuje ji systémem Facelets.

3.1.1 JavaServer Pages

Technologie JavaServer Pages v současné verzi 2.2 je součástí platformy JEE. Tato serverová technologie je určena pro snadnou tvorbu webových aplikací a je založena na zpracování předložených šablon do validního HTML výstupu, který obdrží prohlížeč uživatele.

Šablony mohou existovat v mnoha podobách a ačkoliv JSP zahrnuje i vlastní knihovnu komponent, mnohem častěji se pro návrh webových stránek používají komponenty standardu JSF. V rámci šablon se kromě definice podoby stránky lze také odkazovat na programový kód, tj. obsah stránek se dynamicky generuje a přizpůsobuje aktuálním datům.

JSP a JSF byly v minulosti často používány společně, nicméně ve skutečnosti obě technologie mohou existovat zcela odděleně. JSP mají vlastní komponenty pro definici stránky a JSF mohou využít jinou zpracující technologii. Pro některé nedostatky JSP byl vývojářskou komunitou systematicky vyvíjen tlak na úplné opuštění této technologie v rámci standardu JSF, k čemuž ve verzi 2.0 a pozdějších skutečně došlo. (Brown et al., 2005)

3.1.2 JavaServer Faces

Technologie JavaServer Faces v aktuální verzi 2.2 patří stejně jako JSP mezi standardní součástí platformy JEE a tedy je součástí všech dedikovaných JEE serverů. Zkratka JSF označuje zároveň standardní specifikaci toho, jak má vypadat rozhraní moderního frameworku pro tvorbu uživatelského rozhraní na bázi komponentového vývoje, a implementaci této specifikace, tedy komponentový framework pracující na straně serveru. Pro zpracování komponent se dlouhou dobu používala technologie JSP, která byla od verze 2.0 nahrazena systémem Facelets. (Brown et al., 2005)

Na této specifikaci je vystavěno mnoho různých webových frameworků – jmenujme např. IceFaces, PrimeFaces, MyFaces a samozřejmě RichFaces. Stejně jako JSP i tuto specifika-

[14] JavaServer Pages.

ci spravuje společnost Oracle. JSF lze použít i bez systému Facelets, např. ve spolupráci s technologií XUL[15].

Při samostatném využití frameworku roli modelu (tj. zdroje dat) hrají tzv. managed beans[16], tedy komponenty, které jsou definovány v konfiguračních souborech (obvykle `faces-config.xml`) a podávají data šablonám. Obvyklejší je spojení s podnikovým frameworkem typu Spring, kde jsou komponenty JSF modelu nahrazeny třídami daného frameworku.

3.1.3 Facelets

Systém Facelets (aktuální verze 2.0) byl vytvořen pro potřeby podpory JSF, neboť původní technologie již nedokázala pokrýt veškeré potřeby tohoto standardu. Jedná se o lehký framework vyvinutý opět společností Oracle, taktéž se stal součástí platformy Java EE spolu s JSF 2.0.

Stejně jako v případě předchozí technologie JSP i Facelets představuje zejména zpracující technologii pro tvorbu uživatelského rozhraní webových aplikací s využitím komponentově orientovaného přístupu. Zároveň se pod stejným názvem skrývá i vlastní šablonovací jazyk pro tvorbu stránek za pomoci JSF. (Oracle, 2013)

Facelets umožňují využít současně základní JSF prvky, XHTML[17] značky a také uživatelsky definované komponenty. Jednotlivé druhy šablonovacích elementů pak rozlišuje za pomoci deklarace jmenných prostorů v záhlaví šablony. Tato vlastnost poskytuje tvůrci šablony větší flexibilitu a volnost při použití vhodných technologií. Zajímavou vlastností je také možnost deklarovat značky v klasickém XHTML a přidat jim příznak, který systému řekne, že se jedná o Facelets komponentu. Tento zdánlivě příliš zdlouhavý přístup má svou výhodu: umožňuje editovat šablony ve WYSIWYG[18] editorech. Tato výhoda pro normálního vývojáře příliš neznamená, může být ale nedocenitelná pro webové návrháře.

---

[15] XML User Interface Language.

[16] Doslova „spravované komponenty.“ Jedná se o datové struktury, které jsou přístupné odkudkoliv z aplikace a jsou deklarovány v konfiguračních souborech. Z nedostatku českých ekvivalentů používám původní anglický výraz, který je v praxi obecně užíván a chápán.

[17] eXtensible HyperText Markup Language.

[18] What You See Is What You Get.

3.1.4 AJAX

Technologie AJAX (Asynchronous JavaScript And XML) umožňuje vývoj efektivnějších webových stránek díky tomu, že požadavky na server se řeší na pozadí a uživatel nemusí čekat na jejich dokončení. Nejvýraznějším efektem pro koncového uživatele je skutečnost, že se překreslují pouze části stránky, nikoliv stránka jako celek, jak tomu bylo dříve. AJAX tedy umožňuje dynamičtější interakci a úspornější využití serverového času. Technologie byla standardizována v roce 2006. (Deitel, 2009)

Mezi uživatelsky neviditelné výhody tohoto přístupu patří snížení serverové zátěže díky požadavkům na menší objem překreslovaných dat, což při stále vzrůstajících nárocích na internetovou síť rozhodně není na škodu.

AJAX je nativní součástí RichFaces od verze 3.

3.1.5 jQuery

Framework jQuery je založen na jazyku JavaScript a představuje nejpopulárnější nástroj ve své kategorii (DevRates, 2013). Jedná se o open-source nadstavbu nad standardním JavaScriptem, která umožňuje efektivnější a jednodušší práci s tímto skriptovacím jazykem. Mj. nabízí mnoho pokročilých funkcí zaměřených zejména na často se opakující operace (jako např. vybrání prvku na stránce dle identifikátoru) a složitější grafické efekty, což výrazně urychluje kódování i ladění a přispívá k čistotě kódu.

Framework jQuery je kompatibilní se všemi moderními prohlížeči, což výrazně usnadňuje přenositelnost aplikace a zvyšuje možnost oslovení různého publika. Tento jednoduchý a populární vyvíjí jQuery Foundation, jejímž cílem je mj. prosazování webových standardů a meziprohlížečové kompatibility.

Kromě samotného javascriptového jádra jQuery nabízí nadstavbu v podobě předstylovaných[19] grafických komponent, nástrojů pro mobilní použití a framework Qunit pro psaní javascriptových testů. (jQuery Foundation, 2013)

RichFaces využívají jQuery ve verzi 1.6.4 (aktuální verze 1.10) pro odesílání ajaxových požadavků a grafické efekty na stránce. Kromě jQuery se do verze 3.x používal také javascriptový framework Prototype. (JBoss, 2013)

[19] Slovo „styl“ se ve slově „předstylované“ komponenty používá ve smyslu užití kaskádových stylů CSS. Předstylované komponenty jsou takové komponenty, které mají předdefinovaný vzhled, obvykle pro pohodlí uživatele frameworku.

3.2 Charakteristické rysy

3.2.1 Model-View-Controller

RichFaces (stejně jako ostatní JSF frameworky) podporuje implementaci a využití prinicpu MVC (Model-View-Controller). Jedná se koncept rozdělení aplikace do tří částí, kde jedna se stará o výslednou podobu stránky, druhá k tomu poskytuje data a třetí obstarává navigaci a obsluhu událostí (Veit and Herrmann, 2003).Tento princip napomáhá udržovatelnosti kódu a vytvoření kvalitní architektury, mimo to pak také umožňuje dělbu práce mezi návrhářem stránek, programátorem a architektem. Kromě RichFaces a příbuzných frameworků je na principu MVC postaven např. i populární Spring.

(Veit and Herrmann, 2003) identifikoval také některé slabé stránky tohoto konceptu. View a Controller jsou spolu úzce svázané a vyvíjejí se pro konkrétní Model. Proto tyto komponenty nejsou znovupoužitelné pro jiný model a je nutné vytvářet je znovu. Lze tomu předejít vysokou měrou abstrakce a vytvořením striktních pravidel, která Model musí splňovat, nicméně takový přístup značně omezuje flexibilitu návrhu a použití ideálních technik. Existence slabých stránek nicméně nic nezmění na tom, že prakticky všechny moderní webové frameworky tento koncept využívají.

3.2.2 Rapid Application Development

Rapid Application Development (RAD) představuje alternativu ke tradičnímu způsobu vývoje aplikací, tj. životnímu cyklu specifikace – návrh – implementace - užívání. Tento klasický životní cyklus vychází z předpokladu, že po specifikaci požadavků je možné je implementovat bez nutnosti větších změn. Praxe však ukazuje, že výsledkem takovéhoto přístupu je často aplikace, která vyhovovala požadavkům zákazníka na počátku životního cyklu, ale v době uvedení do provozu je prakticky bezcenná. (Howard, 2002)

(Howard, 2002) dále rozlišuje dva typy rychlého vývoje aplikací a upozorňuje na kritický rozdíl v přístupu mezi oběma druhy. Jedná se o RPD (Rapid Program Development) a RSD (Rapid System Development).

Na počátku projektu typu RPD existuje specifikace programu a projekt cílí na implementaci v nejkratším možném čase. Práce tohoto typu se zadává zkušeným programátorům a nepředpokládá se komunikace s budoucími uživateli. Takový typ vývoje ovšem neodpovídá zavedeným standardům současné doby, kdy se klade maximální důraz na zahrnutí uživatele do projektu. V případě programů s jasným zadáním (obvykle se jedná o program běžící na pozadí) však nepřítomnost uživatele není na škodu.

RSD naopak předpokládá intenzivní využití času a vědomostí koncových uživatelů pro maximálně efektivní spolupráci a co nejkvalitnější výsledek, čímž zapadá mezi agilní me-

todiky a principy. Interaktivní a průběžné zpracování a přizpůsobování specifikace ušetří zbytečnou práci vývojářům a náklady na tuto práci zadavateli projektu a výsledný produkt má mnohem větší šanci na úspěšné spuštění a následný život. Mezi často využívané techniky patří např. prototypování.

Při využití RSD se klade velký důraz na přidanou hodnotu, kterou vyvíjený systém přinese k reálným podnikovým procesům, a také na dobu trvání dodávky. Zejména ve velmi dynamických oblastech[20] může každý den zdržení znamenat obrovskou finanční ztrátu nebo postih regulatorního orgánu[21].

RAD samozřejmě přináší i některé problémy, často bývá obviňován z produkce nekvalitního a neudržovatelného kódu. V praxi se ukázalo, že při využití RAD se tyto techniky aplikují spíše ve vývojových než modelovacích nástrojích a modelování a plánování v důležitosti spíše ustupuje. Klesá také důraz na analýzu, což může v dlouhém období vyústit v kritické problémy a další nepoužitelnost projektu. Zahrnutí těchto technik také v projektových manažerech vzbuzuje nerealistická očekávání stran míry zrychlení vývoje. (Agarwal et al., 2000)

Všechna výše zmíněná rizika však lze omezit a minimalizovat jejich dopad na projekt. Celkově tento přístup převrací klasická zavedená paradigmata v softwarovém vývoji a přesouvá důraz na přínos programů pro hlavní činnost podniku, na spolupráci s uživateli, pochopení jejich potřeb a přizpůsobení se jejich rytmu.

3.2.3 Rich Internet Application

Aplikace typu Rich Internet Application (RIA) umožňují přenést komfort klasických instalovaných počítačových aplikací do internetového prohlížeče. Takové aplikace nabízejí snadné ovládání, příjemné uživatelské rozhraní, snížení komunikace mezi klientem a serverem a zvýšení interaktivity aplikace.

Vzhledem ke vzrůstající intenzitě internetového podnikání roste i důležitost komfortu ovládání a užívání aplikací, které toto umožňují. Proto se do popředí zájmu dostávají frameworky a jiné nástroje, které vývoj takových aplikací usnadňují. (Bozzon et al., 2006a)

(Bozzon et al., 2006a) dále rozlišuje celkem čtyři typy přístupů k tvorbě RIA:

1. skriptovací – logika je implementována s pomocí skriptovacích jazyků[22]

[20] Např. investiční bankovnictví.

[21] Např. při nedodržení termínu pro zahrnutí nové legislativy, aktuálně např. Dodd Frank Act v bankovnictví.

[22] Např. JavaScript

2. modulový[23] - logiku obstarávají doplňky (rozšíření) prohlížeče[24]
3. prohlížečový – logika je nativně podporována prohlížečem přes definované rozhraní v jazyce XML[25]
4. desktopový – aplikace běží mimo prohlížeč[26].

Základní sada charakteristik, které jsou typické pro RIA přístup k tvorbě webových aplikací, by se dala shrnout do následujících bodů[27]:

- redukce komunikace se serverem na absolutní minimum,
- aplikační data mohou být ukládána na klientu i na serveru,
- zpracování dat (ukládání, mazání) lze provádět na klientu i na serveru,
- aplikace funguje i bez přístupu k internetu (zejména pro desktopový přístup).

Aplikace typu RIA v současné době zažívají obrovský rozmach a nahrazují některé zavedené služby[28] (Anderson, 2007). Někteří autoři varují před bezpečnostními nedostatky, které pocházejí už z vývojových platforem RIA aplikací (Flash, .NET, Java) a zdůrazňují nutnost neustálé aktualizace verzí jednotlivých aplikací (Seltzer, 2010). Shodnou se však na tom, že přínosy těchto aplikací jsou obrovské a trend nezastavitelný. (Kay, 2009)

3.3 Funkcionalita

Jak bylo zmíněno výše, framework RichFaces nabízí celkem 83 komponent pro skládání jednotlivých stránek, z toho 16 komponent s podporou technologie AJAX. Pokrývá zejména základní funkcionalitu pro sběr a prezentaci dat pro účely vývoje podnikových aplikací.

---

[23] Originální výraz zní *plugin-based*, tedy doslova "pluginový přístup," nicméně k tomuto slovu neexistuje vhodný český ekvivalent. Náhradní slovo "modul" vyjadřuje podstatu přístupu a přitom je dostatečně srozumitelné.

[24] Např. Flash.

[25] Např. vykreslovací jádro prohlížeče Mozilla Firefox Gecko.

[26] Např. s využitím technologie Java Web Start.

[27] V následujícím výčtu se slova klient a server používají ve smyslu klient-server architektury. Klientem v tomto kontextu je obvykle webový prohlížeč.

[28] Např. klasické televizní vysílání je pomalu nahrazování internetovými televizemi a streamovaným vysíláním.

Přesnou definici komponent s vizuální prezentací lze získat na stránkách výrobce: http://showcase.richfaces.org/.

V této části se soustředím na obecnou kategorizaci a popis jednotlivých částí funkcionality, kterou framework nabízí spíše než na detailní popis konkrétních komponent. Cílem této podkapitoly je na vhodné úrovni podrobnosti uvést schopnosti a omezení zkoumaného frameworku a předložit je ke zvážení.

3.3.1 Datový vstup

Sběr dat patří bezesporu k základním funkcím většiny podnikových aplikací, tudíž je nezbytné umožnit jejich efektivní zpracování a validaci. Zároveň v rámci zachování výkonnosti a omezení síťového vytížení by tato validace ideálně měla probíhat přímo na klientské straně, tj. v prohlížeči, bez nutnosti kontaktovat server.

Kromě klasických vstupních políček a políček pro zadání hesla nabízí framework i následující komponenty:

- kalendář s možností automatické definice formátu dle aktuální lokalizace,
- automatické našeptávání textu na základě údajů z databáze nebo jiného zdroje
- WYSIVYG editor s možností úpravy jeho chování,
- komponenty pro stanovení rozpětí číselných hodnot (např. cena od 500 do 5000),
- rozbalovací menu s mnoha možnostmi nastavení a úpravy,
- nahrávání souborů.

Doprovodnou funkcionalitou komponent datového vstupu je v souladu s výše uvedenými požadavky možnost ajaxové validace ještě před odesláním na server. Validace umožňuje zkontrolovat jen vstup na klientovi nebo kontrolu celého objektu na pozadí a vypisovat chybová hlášení a varování v zásadě kdekoliv na stránce.

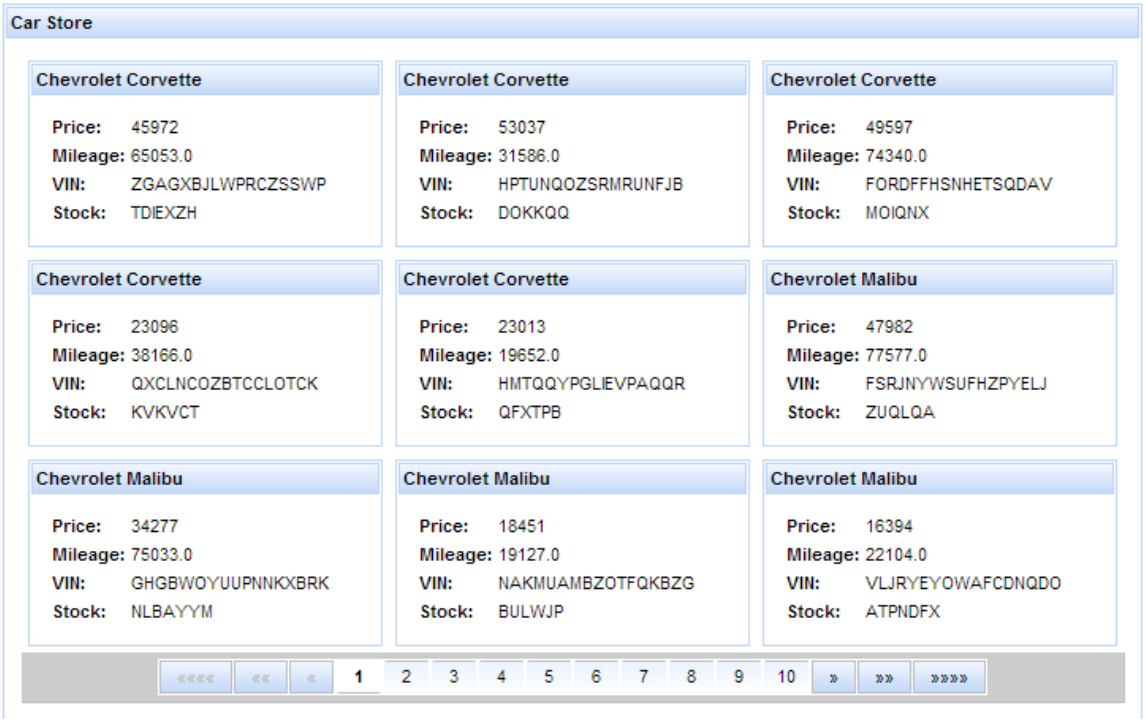
3.3.2 Datový výstup

V prezentaci dat je framework RichFaces ve srovnání s konkurencí (zejména s PrimeFaces) poněkud omezený, nenabízí v zásadě žádné náročnější grafické zpracování databázových údajů, jako jsou např. grafy a diagramy. Kvůli tomu nelze tento framework uplatnit

například při tvorbě populárních projektových kokpitů[29]. RichFaces se omezuje na následující komponenty:

- Klasické tabulky s možností stránkování, automatického filtrování a řazení. Skutečně databázové stránkování je nutno naprogramovat ručně na pozadí s využitím odpovídajících rozhraní frameworku.[30]

- Datovou prezentaci v podobě "samolepek" s informacemi o konkrétním datovém objektu (viz Obr. 1).



Obr. 1: *Datová prezentace v podobě "samolepek".*
Zdroj: (JBoss Foundation, 2012a)

- Podpora dynamického vytváření menu a kontextových nabídek. Překvapivě nenabízí podporu populární drobečkové navigace.

- Organizace dat do stromové struktury

Tyto nedostatky lze samozřejmě řešit, ovšem toto řešení bude s největší pravděpodobností mimo RichFaces. Do projektu lze v souběhu s tímto frameworkem zapojit např. Prime-

[29] Angl. project dashboard. Jedná se o grafickou prezentaci fundamentálních charakteristik podniku pro účely vedení firmy. Manažer má tak přehled o všem, co se ve firmě děje, na agregátní úrovni a může operativně reagovat na vzniklé problémy.

Faces nebo GWT bez větších problémů s kompatibilitou, jediným úskalím při integraci RichFaces-PrimeFaces mohou nastat v rozdílných verzích JSF.

3.3.3 AJAX

RichFaces nativně podporuje technologii AJAX v celkem 16 komponentách (tj. 19,27%). S pomocí těchto komponent lze přidat podporu AJAXu v zásadě do jakékoliv jiné komponenty prostým zanořením ajaxové komponenty do té "obyčejné." Tyto komponenty umožňují:

- Dynamicky nastavovat atributy v objektech na serveru z klientské části
- Aktualizace stránky systémem Push (server aktualizuje klienta) i Pull (klient si o aktualizace požádá).
- Definovat javascriptovou funkci v podobě komponenty na stránce
- Přiřadit jakékoliv komponentě posluchače[31], který zareaguje na definované události
- Reagovat na jakoukoliv javascriptovou událost a vyvolávat tyto události
- Redukovat množství odeslaných a zpracovávaných ajaxových požadavků s pomocí fronty
- Vytvořit modální panel[32] po dobu trvání nějaké operace
- Vypisovat události, které probíhají na klientské straně, pro lepší ladění aplikace.

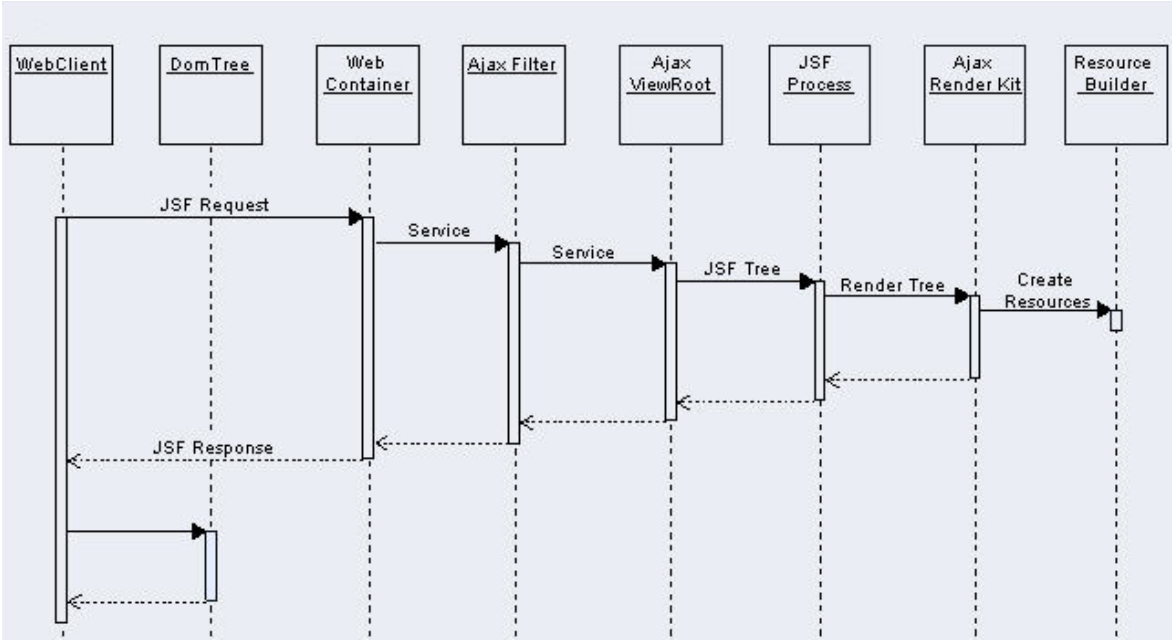
Základním prvkem při nakládání s ajaxovou funkcionalitou je ajaxový filtr zabudovaný ve frameworku, který slouží k úpravě generovaného kódu v souladu s požadavky prohlížeče. Při "obyčejném" požadavku odesílá celý komponentový strom, kdežto při ajaxovém požadavku odesílá pouze změněnou oblast.

Definice skutečnosti, že požadavek má využít AJAX se provádí přidáním některé z výše zmíněných komponent k tomu určených. Aktualizaci jednotlivých dotčených oblastí na stránce provádí javascriptové jádro frameworku. Rozdíl mezi zpracováním klasického a ajaxového požadavku lze demonstrovat na následujících obrázcích.

---

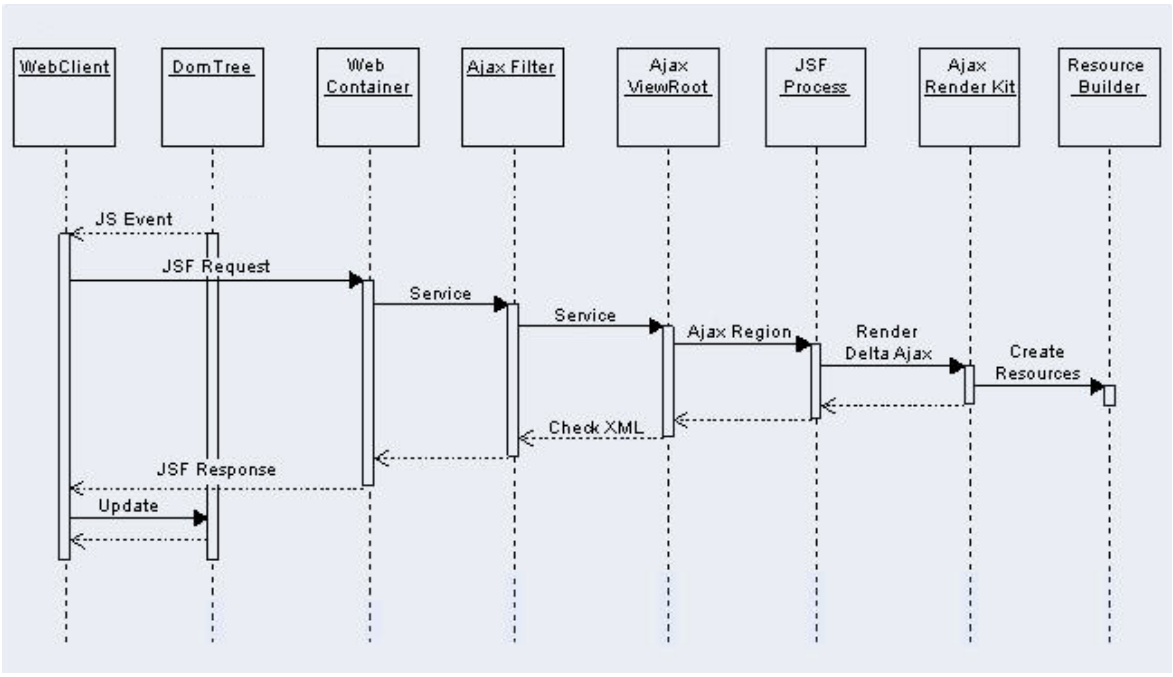
[31] Angl. listener.

[32] Modální panel zabrání uživateli nakládat s podkladovou stránkou po dobu jeho existence.



Obr. 2: *Zpracování klasického požadavku*

Zdroj: (JBoss, 2013)



Obr. 3: *Zpracování ajaxového požadavku*

Zdroj: (JBoss, 2013)

Z těchto diagramů je evidentní rozdíl při zpracování obou typů požadavků. Při ajaxovém požadavku se přehodnocuje a překresluje pouze změněná oblast, kdežto při klasickém požadavku dochází ke znovunačtení a znovuvykreslení celé stránky se všemi závislostmi a databázovými dotazy. Je tedy zřejmé, že využívání ajaxových požadavků je výrazně efektivnější a rychlejší.

3.3.4 Ostatní funkcionalita

RichFaces poskytuje komponenty pro přímé zapojení JavaScriptového frameworku jQuery do aplikace, nicméně vzhledem k tomu, že RichFaces sám o sobě obsahuje svou vlastní verzi tohoto frameworku, dochází na stránkách ke konfliktům a neočekávanému chování.

Pro stránku je dále možné definovat klávesové zkratky a navěsit na ně odpovídající funkce. RichFaces také poskytuje podporu pro drag&drop, tedy přetahování objektů po stránce.

3.3.5 Známá omezení

Ve vývojářské dokumentaci samotní autoři frameworku zmiňují některá funkční omezení pro fungování RichFaces kromě těch, která uvádím výše. Framework je zejména omezen v oblasti

- mazání a přidávání nových objektů – na stránce vždy musí být zástupný symbol[33], do něhož se objekt dle potřeby přidá nebo se odtud smaže,
- stylování objektů, které nejsou při načtení stránky přítomny – v souvislosti s předchozím problémem RichFaces neumožňuje aplikovat na takto přidané objekty kaskádové styly nebo JavaScriptové funkce.

Verze 4.x je navíc omezená oproti nižší verzi v rozsahu nabízených komponent. V nejnovější verzi chybí např. komponenta pro podporu Gogole Maps, grafická komponenta `paint2D` (umožňovala tvorbu nadpisů podobných WordArtu), podpora Google Earth, část drag&drop funkcionality a také komponenta `insert`, která umožňovala vkládání částí kódu do textu stránky[34] s možností zvýraznit syntaxi dle konkrétního jazyka. Poslední dvě komponenty by měly být v dalších verzích frameworku doplněny.

3.3.6 Zhodnocení funkcionality

Framework bezesporu poskytuje veškerou funkcionalitu potřebnou pro standardní webové aplikace podnikového typu, založené zejména na formulářích, tabulkách a zobrazování textu. Pro dynamické webové stránky s marketingovým potenciálem je ovšem poněkud nevhodný, protože neposkytuje v zásadě žádné možnosti kreativnějšího grafického zobrazení sesbíraných dat. Stejně tak komponenty tohoto frameworku neobsahují v zásadě žádné

---

[33] Angl. placeholder.

[34] Tj. vědomé zobrazování kódu, užívané např. na vývojářských stránkách. Takto vkládaný kód je obvykle nutné ošetřit tak, aby ho vykreslovací jádro prohlížeče ignorovalo a kód neprovádělo.

animace, tudíž výsledný efekt je poněkud fádní. Tyto nedostatky lze vyřešit zapojením jiného frameworku, např. PrimeFaces nebo GWT.

Za výrazný funkční nedostatek lze považovat také chybějící podporu drobečkové navigace, která je při tvorbě komplikovaných webových aplikacích neocenitelná a bohatě využívaná. Narozdíl od předchozí verze chybí také propojení na mapové služby Google Maps, jejichž využití skýtá mnohé možnosti.

Konkurenční PrimeFaces také nabízí mj. takové speciality, jako myšlenkové mapy, zabudovanou fotogalerii či možnost pořizovat fotky z webkamery, které ovšem v korporátním prostředí nenacházejí denního využití, rozhodně však motivují vývojáře, aby tento framework používali i ve svých soukromých projektech či v projektech zaměřených spíše vůči veřejnosti.

Obecně lze tedy dojít k závěru, že funkcionalita poskytovaná frameworkem RichFaces je spíše základního typu, nicméně pro klasickou intranetovou či podnikovou aplikaci bude naprosto dostačující.

Zdá se, že komunita stojící za vývojem RichFaces si svá omezení a nedostatky uvědomuje a pokouší se je řešit tak, aby framework neztrácel uživatele ve prospěch mladších a progresivnějších PrimeFaces. Výsledkem jejich vědomé snahy o pokrok je tak např. implementace mobilní verze některých komponent, migrace chybějících komponent ze starší verze do nové a vylepšování stávajících komponent.(Leathem, 2011)

3.3.7 Budoucí rozvoj

Tato subkapitola bude věnována perspektivám budoucího rozvoje frameworku v mnoha různých ohledech. Dostatečný vývoj každého softwarového nástroje představuje nesmírně důležitý faktor pro jeho použitelnost, komunitní oblíbenost a konkurenceschopnost vůči ostatním nástrojům. Zejména softwarové firmy kladou důraz na aktuálnost svých nástrojů a zapracování nejnovějších trendů. Nástroj, který se nevyvíjí, je předem odsouzen k zániku.

Kolem RichFaces se kromě vývojářů samotných pohybuje i široká komunita uživatelů, kteří přidávají vlastní prvky, poskytují podněty k dalšímu pokroku a pracují na integraci s jinými nástroji. Rozvoj RichFaces je dále výrazně ovlivněn vývojem podkladových nástrojů, jako je např. framework jQuery aj.

Významným faktorem pro budoucí směřování frameworku je vydání nové verze standardu JSF (konkrétně se jedná o verzi 2.2), který přináší některé nové funkce a možnosti. Dle (Tijms, 2012) se zejména jedná o:

- Podporu HTML 5 – zejména možnost definovat vlastní atributy HTML prvků[35]

[35] Tzv. pass-through attributes

- Definování toků aplikací (tj. souslednosti obrazovek, viz výše) – v tomto ohledu JSF zcela otevřeně přiznává inspiraci frameworkem Spring WebFlow

- Vylepšení v oblasti technologie AJAX – zejména se jedná o lepší řazení a zacházení s požadavky (tj. vylepšení v oblasti fronty) a asynchronní nahrávání souborů

- Bezstavové pohledy[36] - JSF si za normálních okolností ukládají informace o aktuálním stavu vykreslovaných komponent, nyní je možné toto chování zcela vypnout

- Dynamickou změnu vzhledu – za běhu aplikace pro každého uživatele zvlášť (plánováno do budoucích verzí standardu JSF, ve verzích 3.x využívajících starší variantu standardu je možné měnit vzhledy pouze před spuštěním aplikace a pro všechny uživatele naráz)

Všechny tyto změny přinášejí výrazně širší možnosti pro vývojáře a tvůrci RichFaces je budou muset ve svých komponentách reflektovat. Zejména v oblasti změny vzhledu si RichFaces zakládá na kvalitě, tudíž lze předpokládat kvalitní komponenty pro dynamickou změnu motivů použitých v aplikaci. Zavedení definice toků na úrovni JSF výrazně usnadní vývojářskou práci a odstraní nutnost využívat externí framework, tudíž bude možno přistoupit k používání RichFaces v čisté podobě bez nutnosti přizpůsobení navigačnímu nástroji. Vlastní atributy u HTML prvků skýtají široké možnosti využití pro definování různého chování a vykreslování. Úplná integrace s JSF 2.2 je naplánována do verze 5.

Tvůrci RichFaces také poměrně překvapivě uvedli, že se v budoucnu hodlají zaměřit kromě rozvoje své standardní sady komponent i na komponenty javascriptové, a to zejména v podobě modulárních rozšíření pro framework jQuery. Tyto nové prvky by měly být znovupoužitelné a nezávislé na jádru frameworku, tj. bude možné je použít i ve spolupráci s jiným webovým frameworkem. Jedná se rozhodně o zajímavou iniciativu, která mimo jiné slibuje zkrácení vývoje nových komponent a lepší podporu principů RAD. Zároveň tím tvůrci RichFaces chtějí částečně poděkovat open source projektu jQuery za jeho příspěvek k fungování frameworku. (Leathem, 2012a)

Kromě proprietárního vývoje nových komponent lze také očekávat rozšíření nástrojů pro tvorbu vlastních přizpůsobených komponent s využitím připraveného CDK[37]. Šablona nové komponenty je ve verzích 3.x definována v XML podobě, do této šablony se pak vkládá chování pomocí Java kódu na pozadí. Tato forma přidávání nových prvků doznala už ve verzích 4.x značných změn a v budoucnu by se měla stát mnohem jednodušší a přístupnější v souladu se snahou o podporu principů RAD.

Na vývoj standardních komponent samozřejmě nebude zapomenuto. V současné době se pracuje na dokončení převodu chybějících komponent z verze 3.x do verze 4.x, na vyladění

---

[36] View. Ve smyslu Model-View-Controller.

[37] Component Development Kit. Nástroj pro tvorbu komponent.

chyb a nedostatků a optimalizaci chování již transformovaných komponent. Cílem této činnosti je doladit veškeré nedostatky plynoucí z radikální změny, kterou přechod na verzi 4 rozhodně byl, a dokončit celý komponentový balík.

Zároveň s pracemi na vylepšení verzí 4.x probíhá vývoj verze 5 (první alfa verze už byla vydána 6. února 2014, finální datum vydání není zatím známo). O nových funkcích toho zatím není příliš známo, vývojáři slibují zejména:

- Snazší změnu vzhledu – RichFaces budou umožňovat zapojení externích vzhledů definovaných v jazyce LESS[38], tj. uživatelé budou moci využívat předdefinované vzhledy volně ke stažení
- Integraci s nově vydaným standardem JSF 2.2
- Nové komponenty (ačkoliv zatím není přesně stanoveno jaké)
- Snadnou migraci z verze 4 na verzi 5

Poslední zmiňovaný krok představuje velkou snahu o uklidnění komunity a znovupřilákání některých uživatelů, kteří se kvůli vleklým problému s kompatibilitou verzí přiklonili ke konkurenčním produktům.

RichFaces také klade stále větší důraz na mobilní komponenty a kompatibilitu s mobilním zobrazováním. Výhodou tohoto přístupu je, že práce s mobilními komponentami se na straně vývojáře v zásadě neliší od standardního přístupu, jsou to tytéž postupy a tytéž principy, obohacené o snahu minimalizovat počet požadavků na server a stahovaných souborů (obvykle přidružené stylopisy[39] a javascripty). V tomto ohledu lze do budoucna očekávat vyšší míru standardizace, minimalizace velikosti zdrojových souborů nutných ke stažení ze serveru a optimalizaci výkonnosti.

Z výše uvedeného jasně vyplývá, že framework rozhodně není statický a neustále se vyvíjí. Tento vývoj je obvykle založen na vizi vývojářského týmu, kteří ho nicméně konzultují s uživatelskou komunitou tak, aby došlo k co možná nejefektivnější prioritizaci jednotlivých úkolů.

Zohledňování přání uživatelské veřejnosti lze frameworku jen přičíst k dobru. Nevýhodou tohoto přístupu ovšem je nemožnost definovat nějaký dlouhodobý trend vývoje, neboť plánování probíhá v zásadě z verze na verzi v závislosti na aktuálně nejpalčivějších problémech uživatelské báze. RichFaces budou tak i do budoucna výrazně ovlivňovány náladou v komunitě a změnami podkladových nástrojů.

---

[38] Dynamický stylovací jazyk, vytvořený jako nadstavba nad standardním CSS.

[39] Definice stylů v jazyce CSS.

3.4 Konkurenční frameworky

Jak již bylo nastíněno, standardní specifikace JSF posloužila jako základ mnoha dalších frameworků, které tak poskytují podobné, nikoliv ale stejné možnosti jako RichFaces. V korporátní praxi se samozřejmě využívají i jiné technologie, např. framework Apache Struts nebo Google Web Toolkit (GWT). V této části se text bude věnovat charakteristice jednotlivých frameworků a jejich vzájemnému porovnání a pochopitelně také srovnání s frameworkem RichFaces.

Z frameworků založených na JSF jsem k srovnání vybrala PrimeFaces proto, že se jedná o nejoblíbenější nástroj[40] pro vývoj RIA aplikací v Javě (DevRates, 2013). Pro zajímavost, RichFaces ve stejném srovnání získávají páté místo. Konkureční IceFaces vynechávám zejména proto, že jejich kód je na PrimeFaces založen a tedy je to poměrně zbytečné (Optimus Prime, 2012).

Frameworky Struts a Tapestry zde zastupují nástroje pro tvorbu webových rozhraní mimo rodinu JSF, aby srovnání bylo dostatečně vypovídající a úplné.

3.4.1 Popis frameworků

PrimeFaces

Framework PrimeFaces (aktuální verze 4.0) vyvíjí softwarová společnost Prime Technology, která ho zároveň využívá v projektech pro své klienty. Produkt má zabudovanou podporu technologie AJAX a nabízí širokou škálu komponent. Ve svých webových aplikacích tento framework využívají i světové známé firmy jako např. UniCredit Bank, Deloitte, Ford či Boeing. Komponenty dále disponují možností být upraveny pro mobilní zobrazení. (Prime Technologies, 2012)

V současné době se jedná o nejpopulárnější framework pro tvorbu podnikových webových aplikací v Javě (DevRates, 2013) a narozdíl od konkurence nabízí také placenou verzi, která v nejvyšší kategorii umožňuje klientům zadat dodavateli vývoj nových komponent, jako by se jednalo o interní framework.

[40] Jedná se skutečně o hodnocení popularity využití daného nástroje mezi vývojáři profesionálního i amatérského charakteru, nikoliv o přímé hodnocení kvality nástroje na základě předem stanovených kriterií. Pořadí v tomto žebříčku odráží zejména snadnost práce a integrace daného nástroje, což výrazně přispívá k vývojářské spokojenosti.

Tapestry

Apache Tapestry je open source framework pro vytváření dynamických a robustních aplikací v Javě vyvíjený organizací Apache Software Foundation již od roku 2000. Tento framework se skládá z knihovny komponent a šablonovacího systému na bázi HTML. Bez větší námahy lze rozdělit proces tvorby šablon a samotný vývoj, což tento nástroj činí velice populárním v korporátním vývoji. (Apache Software Foundation, 2013a)

Tapestry patří v popularitě u vývojářů hned za PrimeFaces (DevRates, 2013), nicméně tato popularita doznala významného poklesu, když při přechodu z verze 4 na verzi 5 byla úmyslně zcela vynechána jakákoliv zpětná kompatibilita a nebyl poskytnut ani návod pro migraci stávajících aplikací na novou verzi frameworku. Nyní se tato rána v oblíbenosti frameworku pomalu zaceluje, neboť od verze 5 je zpětná kompatibilita opět přislíbena.

Struts

Apache Struts patří mezi další z projektů Apache Software Foundation v aktuální verzi 2.3. Jedná se o jeden z nejpopulárnějších nástrojů pro tvorbu webových aplikací vůbec, za což vděčí mj. použití technologie JSP a implementace principů MVC. Kromě šablonovacího systému zahrnuje také knihovnů komponent a podporuje technologie AJAX, SOAP[41] a REST[42].

Struts prošly již dlouhou historií (první verze byla vydána roku 2000) a přestože jejich popularita konstantně klesá, zachovávají si značnou část uživatelské základny i přes kritiku z pozice zastaralosti a porušování principů MVC. (Apache Software Foundation, 2013b)

3.4.2 Stanovení srovnávacích kritérií

Proces výběru vhodného webového frameworku se vždy musí řídit určitými kritérii, která dobře vystihnou kritické potřeby subjektu, jenž nástroj vybírá. Mezi obvyklá kritéria patří zejména programovací jazyk, v němž je daný framework napsán, možnost integrace s jinými frameworky a kompatibilita výsledné aplikace se zvolenými webovými prohlížeči. Náplň těchto obecných kategorií však bude výrazně individuální[43] a výrazně závislá na vývojovém prostředí. Toto srovnání je zaměřeno na webové frameworky napsané v jazyce Java.

---

[41] Simple Object Access Protocol.

[42] Representational State Transfer.

[43] Např. vývojáři v .NET se často omezují na kompatibilitu s prohlížečem Microsoft Internet Explorer, zatímco Java projekty se obvykle snaží o všestrannou kompatibilitu.

Dle (Laakso and Niemi, 2008) jsou pro volbu správného nástroje důležitá tato obecněji definovaná kritéria:

- zralost produktu,
- zpětná kompatibilita,
- aktivita komunity,
- typ licence,
- bezpečnost,
- podpora technologie AJAX,
- škálovatelnost,
- úroveň dokumentace,
- existence ladících nástrojů a
- zda je produkt open-source.

Jejich výzkum se orientoval na srovnání frameworků Spring, RichFaces (coby zástupce JSF rodiny), Apache Tapestry a Echo2, nicméně výše zmíněná kritéria lze bezesporu použít i na zde zvolené frameworky.

Důležitost výše uvedených kritérií pro použitelnost daného nástroje ve vývoji je neoddiskutovatelná. Pod pojmem *zralost produktu* si lze představit kombinaci délky setrvání produktu na trhu, kvalitu jeho rozvoje a údržby a jeho stabilitu. Ta je do značné míry podpořena *zpětnou kompatibilitou*, což je jednoznačně žádoucí rys u každého nástroje, neboť usnadňuje přechod na novější verze (čímž se mj. obvykle snižují bezpečnostní rizika).

Kritéria *typ licence* a *open source* povaha projektu do jisté míry představují nákladový a právní pohled na nástroj, tj. zda je nutné za něj platit a za jakých podmínek ho lze použít. Náklady na zavedení nového nástroje do projektu či projektů také výrazně klesají tehdy, pokud je *úroveň* dostupné *dokumentace* dostatečně kvalitní, její rozsah je dostačující (tj. pokrývá všechny relevantní oblasti užití nástroje) a je přiměřeně aktuální. To vyžaduje zejména nesmírnou péči ze strany tvůrců dotyčného softwarového nástroje.

Na kvalitě dokumentace se také velkou měrou podílí uživatelská *komunita*, která obvykle upozorňuje na chyby a nedostatky jak v dokumentaci, tak v projektu samotném. Různá tematická fóra, kde uživatelé-vývojáři řeší své projektové problémy, jsou neocenitelným zdrojem dodatečné dokumentace a příkladů. Rychlost adaptace na nový nástroj a efektivita práce s ním také závisí na dostupnosti *ladicích nástrojů*, tedy nástrojů pro krokování aplikace a odhalování chyb. Bez podobných služeb si moderní vývoj nelze příliš představit.

Kritéria *škálovatelnost* a *podpora technologie AJAX* do značné míry souvisí s neustále se zvyšujícím tlakem na výkonnost a rychlost webových aplikací. Uživatelé v současné době požadují, aby webové aplikace reagovaly v zásadě stejně rychle jako obyčejné textové stránky, aby byly interaktivní a vizuálně atraktivní. Při obrovském počtu uživatelů s přístupem k internetu je tak nutné myslet na rozložení této zátěže mezi více produkčních serverů a na omezení objemu přenášených dat, což výše zmíněná kritéria postihují.

V neposlední řadě k významným nárokům, které zvolený nástroj musí uspokojit, patří požadavky jeho *bezpečnosti*. Webové aplikace musí dnes a denně odolávat mnohým hrozbám v podobě útoků brutální silou, zneužívání uživatelských údajů, neoprávněné manipulace s daty, spamu a jiným útokům, tudíž bezpečnost aplikace představuje prioritní charakteristiku. Kvalitní zabezpečení aplikace minimalizuje riziko její nedostupnosti (a potenciální obchodní ztráty s tím spojené) a také integritu a zajištěnost uživatelských dat.

Úprava sady srovnávacích kritérií

Rozhodla jsem se vynechat několik kritérií tak, aby struktura celého hodnocení více odpovídala mým záměrům. Původní kolekce kritérií byla uzpůsobena na míru konkrétnímu projektu, což mírně omezuje jejich obecné využití. Vynechána byla kritéria *typ licence* a *open-source*, neboť všechny zkoumané produkty jsou open-source a dle licence také volně k užití. Zahrnutí těchto požadavků tedy lze tedy považovat za nadbytečné a zbytečně matoucí.

Výčet kritérií je naopak doplněn o požadavek *snadné změny vzhledu*. Může se to zdát jako triviální záležitost, ale z vlastní zkušenosti mohu s jistotou tvrdit, že jednotný vzhled aplikací laděný do firemních barev výrazně snižuje obtížnost zacházení s aplikací pro koncové uživatele a je také důležitý pro korporátní identitu.

Dalším přidaným kritériem z mé strany pak je *obtížnost konfigurace*, tedy jak složité a časově náročné je zapojit produkt do projektu. Díky tomu navíc počet kritérií zůstane stejný a není nutno výrazně přepracovávat váhy, které jsou jednotlivým kritériím přiřazeny.

K pozměnění váhy jednotlivých kritérií došlo s ohledem na zaměření této práce na korporátní využití. Váha byla zvýšena u kritéria *zpětná kompatibilita*, protože pro podniky je nesmírně důležité, aby použité technologie byly stabilní a podporované ze strany dodavatele[44]. Častá obměna použitých nástrojů, které jsou často použity u většího množství projektů, přináší nemalé dodatečné náklady a vnáší destabilizující prvek do celého systému.

Naopak bylo provedení snížení kritéria *obtížnost konfigurace*. Přestože je tento aspekt bezesporu důležitý, lze jeho dopad zmírnit splněním kritérií *aktivita komunity* a *kvalita dokumentace*. Navíc v softwarově orientovaných firmám či podnicích, které své IT obsluhují

[44] Zejména pokud jste natolik rigidní subjekt jako např. banky.

interně, obvykle bývá křivka učení poměrně strmá. Kompletní nastavení vah kritérií viz tabulka 1.

Tabulka 1: *Přehled kritérií*

Kritérium	**Váha**
Zralost produktu	0,15
Aktivita komunity	0,1
Zpětná kompatibilita	0,1
Bezpečnost	0,1
Technologie AJAX	0,15
Škálovatelnost	0,15
Kvalita dokumentace	0,05
Existence ladících nástrojů	0,1
Snadná změna vzhledu	0,05
Obtížnost konfigurace	0,05

3.4.3 Postup vyhodnocení frameworků

Při vyhodnocování kvalit jednotlivých frameworků jsem vycházela zejména z informací obsažených na příslušných webových stránkách a v referenční dokumentaci. K praktickému provedení ohodnocení jsem využila navrženého postupu vicekriteriální analýzy dle (Laakso and Niemi, 2008), kdy každému kritériu je přiřazena určitá váha. Součet těchto vah je 1. Každé kritérium může nabývat hodnot od -1 do 1 dle míry uspokojení požadavku hodnotitele.

Tabulka 2: *Systém hodnocení (Zdroj:* (Laakso and Niemi, 2008)*)*

Skóre	Interpretace
1	Framework plně vyhovuje kritériím.
0,5	Framework částečně vyhovuje kritériím.
0	Vyhovění kritériím je neznámé nebo nelze říci, že by framework kritériím vyhovoval či šel proti nim.
-0,5	Charakteristiky frameworku jsou částečně v rozporu s kritérii.
-1	Charakteristiky frameworku jsou zcela v rozporu s kritérii.

U frameworků RichFaces a Tapestry, které jsou analyzovány v původním výzkumu, jsem jejich hodnocení přezkoumala a až na detaily převzala. Originální článek vyšel v roce 2008, kdy RichFaces existovaly ve verzi 2.0 bez nativní podpory AJAXu. To se od té doby změnilo (podpora AJAXu byla zaváděna od verze 3), a proto se změnilo i hodnocení v této kategorii.

Stejně tak bylo nezbytné změnit hodnocení zpětné kompatibility tohoto frameworku, neboť nová verze již není zpětně kompatibilní s JSP, které byly standardem pro předchozí vydání. Kompatibilita je zachována pouze částečně proto, že i v dobách RichFaces 2.0 existovaly implementace využívající šablonovací systém Facelets a nikoliv JSP.

RichFaces se zlepšily i v kategorii *existence ladících nástrojů*, nejnovější verze přidala nový prvek (obdobný se již dříve nacházel u konkurenčních PrimeFaces), který umožňuje výpis akcí na straně klienta (tedy prohlížeče) a výrazně usnadňuje hledání chyb při užívání tohoto nástroje.

Všechny tyto skutečnosti potvrzují, že se RichFaces neustále vyvíjejí a jejich tvůrci ve spolupráci s komunitou usilovně pracují na tom, aby přinejmenším dohnali konkurenci a přizpůsobili se aktuálním technologickým trendům.

3.4.4 Výsledky srovnání

Z níže uvedeného srovnání nejlépe vychází framework PrimeFaces od turecké společnosti Prime Technologies, který se těší i značné oblibě mezi vývojáři. Mezi jeho nesporné přednosti patří zejména kvalitní ladící nástroje a technologická aktuálnost. Nad RichFaces zvítězil v kategorii *snadná změna vzhledu*.

Framework RichFaces za svým konkurentem tedy zaostává v bodovém hodnocení o 5%, nicméně srovnáme-li jeho skóre s předchozím hodnocením, došlo k výraznému zlepšení[45]. Nativní podpora AJAXu a existence ladících nástrojů výrazně přispívají ke kvalitě a použitelnosti frameworku.

Apache Struts na třetí příčce zaostává zejména v otázkách bezpečnosti a kvality dokumentace, což ve spojení s obtížností konfigurace může znamenat velké zdržení v projektu.

Apache Tapestry skončil na posledním místě a za svými konkurenty výrazně zaostává. Není bez zajímavosti, že jako jediný získal i záporné body, protože neposkytuje žádné bezpečnostní prvky a není vůbec zpětně kompatibilní (Tapestry 5 je v zásadě zcela nový framework, který s Tapestry 4 nemá prakticky nic společného). Na druhou stranu je také jediný, který získal pozitivní ohodnocení v kategorii *škálovatelnost*.

Srovnání frameworků potvrzuje, že v současné době trend směřuje spíše k lehkým frameworkům s kvalitní podporou ze strany komunity i dodavatele (v podobě dokumentace). Bezpečnost obvykle zajišťuje jiný nástroj, např. Spring. Na prvních dvou příčkách se umístily frameworky založené na JSF 2.0, což z nich ve součtu se snadnou integrací se Springem dělá pro korporátní vývoj volbu č.1.

[45] Celkové hodnocení v roce 2008 bylo 0,6. (Laakso and Niemi, 2008)

Tabulka 3: *Srovnání frameworků dle zadaných kritérií.*

	Váha	**RichFaces**		**PrimeFaces**		**Tapestry**		**Struts**	
		Hodnota	Skóre	Hodnota	Skóre	Hodnota	Skóre	Hodnota	Skóre
Zralost produktu	0,15	1	0,15	1	0,15	1	0,15	1	0,15
Aktivita komunity	0,1	1	0,1	1	0,1	0,5	0,5	1	0,1
Zpětná kompatibilita	0,1	0,5	0,05	0,5	0,05	-1	-0,5	0,5	0,05
Bezpečnost	0,1	0,5	0,05	0,5	0,05	-0,5	-0,05	0	0
Technologie AJAX	0,15	1	0,15	1	0,15	0,5	0,075	0,5	0,075
Škálovatelnost	0,15	0	0	0	0	1	0,15	0	0
Kvalita dokumentace	0,05	1	0,05	1	0,05	0,5	0,025	1	0,05
Ladící nástroje	0,1	1	0,1	1	0,1	1	0,1	1	0,1
Snadná změna vzhledu	0,05	0,5	0,025	1	0,05	1	0,05	0,5	0,025
Obtížnost konfigurace	0,05	1	0,05	1	0,05	0,5	0,025	0,5	0,025
Celkem	1,0		**0,725**		**0,75**		**0,525**		**0,575**

Zdroj: (Laakso and Niemi, 2008)*, autor.*

3.5 Související frameworky

Tato kapitola je věnována některým frameworkům, které se v kombinaci s RichFaces často užívají nebo mají potenciál doplnit chybějící části funkcionality. Mnohé z nich jsou v tomto textu již dříve zmíněny.

Samozřejmě je možné vytvořit webovou aplikaci s využitím pouze RichFaces, nicméně v korporátním vývoji je obvykle výhodnější použít robustnější framework, do něhož se RichFaces začlení jako nástroj pro tvorbu prezentační vrstvy.

3.5.1 Spring

Spring je modulární framework určený pro aplikace psané v Java Standard i Enterprise edici. V zásadě je určen k vytvoření infrastruktury potřebné jako základ aplikace, čímž vývojářům uvolňuje ruce ke skutečně kreativní a přínosné práci. Architektura frameworku je postavena na uznávaných návrhových vzorech a osvědčených postupech[46] v oboru.

Tento framework se skládá z přibližně 20 modulů, přičemž různé moduly se zabývají různými problémovými oblastmi, např. bezpečností, přístupem k datům, webovými službami či mobilními principy. Framework také poskytuje prostředky pro tvorbu extenzivních integračních testů a lze ho využít ve spojení s v zásadě jakýmkoliv serverem.

Typickým rysem pro Spring je využití principu MVC a tzv. Dependency Injection. Dependency Injection ve stručnosti znamená, že framework se sám postará o inicializaci objektů, které aplikace potřebuje (např. služeb, továrních objektů a různých pomocných tříd), aniž by se o to uživatel musel snažit. (Johnson et al., 2013) Obvykle stačí dotyčný objekt opatřit patřičnou anotací a Spring se o ostatní postará za vás. Tento princip mj. umožňuje zapojit do aplikace různé implementace téže komponenty v závislosti na konfiguraci. Této funkcionality lze s výhodou využít např. pro testování, kdy se užité databáze a služby zamění za lokální implementaci pro testovací účely.

Další charakteristickou vlastní Springu je přístup “convention over configuration”, tj. snaha o minimalizaci nutné konfigurace. Pokud vývojář dodržuje při pojmenovávání objektů a jiné obdobné činnosti stanovené konvence frameworku, výrazně se tím snižuje nutnost konfigurace. Např. pokud se atribut objektu jmenuje stejně jako odpovídající sloupec v databázi, není nutná žádná konfigurace k tomu, aby se hodnota z databáze správně přiřadila. Pouze pokud se jména liší, je nutné uvést název sloupečku, kterému atribut odpovídá.

[46] Angl. best practices.

Jádro frameworku (tzv. Core Container) se skládá z následujících komponent

- Core and Beans
- Context
- Expression Language

Komponenta Core and Beans představuje základ celého frameworku, které se stará o Dependency Injection a mj. odstraňuje nutnost starat se vytváření singleton objektů.

Modul Context shromažďuje objekty vytvořené pomocí jádra podobně jako JNDI[47], tj. umožňuje k nim přístup z kteréhokoliv místa aplikace. Context dále umožňuje internacionalizaci a přístup ke vzdáleným komponentám.

Modul Expression Language (tj. výrazový jazyk) poskytuje prostředky pro manipulaci s objekty za běhu aplikace. Spring Expression Language byl vytvořen pro účely vývojářské komunity a pro použití v rámci všech modulů. (Johnson et al., 2013)

Tvůrci Springu se snažili o jeho maximální integrovatelnost s jinými frameworky, mezi nejoblíbenější kombinace patří framework Hibernate a právě RichFaces. Ve spolupráci s RichFaces Spring poskytuje nutnou podkladovou infrastrukturu pro webovou vrstvu.

3.5.2 Spring WebFlow

Již několikrát zmiňovaný Spring WebFlow představuje jeden ze dvaceti modulů mateřského frameworku Spring. Je mu zde věnováno více prostoru proto, že oproti ostatním modulům má pro aplikace s RichFaces výrazně větší význam.

Jak název napovídá, WebFlow umožňuje vytváření tzv. flows neboli toků aplikací. Tok v zásadě představuje pevnou definovanou souslednost stránek, která má stanovený začátek a konec, jako např. proces objednávky v e-shopu. Při takovéto souslednosti je nutno udržovat určité množství společných dat uložených v objektech, jako např. údaje o zákazníkovi nebo obsah nákupního košíku. K tomuto účelu byl vytvořen právě Spring WebFlow.

Jednotlivé kroky v takovém toku WebFlow nazývá “stavy” a každý stav obvykle vyústí v zobrazení nové stránky. Modul umožňuje definovat operace, které se mají provést při vstupu do stavu, tj. ještě před vykreslením stránky (např. kontrola práv uživatele, pokud nesplňuje požadavky, je přesměrován na jinou stránku). Dále lze stanovit možnou navigaci na jiné stránky v závislosti na vstupu od uživatele. Lze také definovat povinné vstupní parametry pro stránku a vytvářet objekty dostupné v rozsahu tohoto toku. (Donald et al., 2013)

---

[47] Java Naming and Directory Interface.

WebFlow se v souvislosti s RichFaces užívá pro tvorbu souslednosti stránek a efektivní navigaci skrze aplikaci, která udržuje společné objekty dostupné na pozadí a odstraňuje nutnost neustále si je odněkud vytahovat.

3.5.3 RichFaces Bootstrap

RichFaces Bootstrap je vedlejší projekt vývojářů RichFaces, který se soustředí na integraci prvků populárního šablonovacího frameworku Twitter Bootstrap. Jak bylo zmíněno v části o budoucím rozvoji frameworku, úsilí vývojářů směřuje mj. k dynamické změně vzhledu, kde by právě sady vzhledů definovaných jako Bootstrap šablony mohly výrazně přispět k jejich popularitě a rozšíření.

Tento projekt představuje novou třetí sadu komponent (jako doplněk ke standardním RichFaces komponentám a ajaxovým komponentám). Tyto komponenty zaprvé mohou využívat předdefinovaných stylů (např. pro vytvoření oblých prvků), zadruhé přidávají možnost připojit k uživatelem zadané hodnotě předem definovaný text (např. “@”). Bootstrap dále doplňuje RichFaces o některé komponenty, které v něm citelně chybí, např. drobečkovou navigaci a prvky pro detailní a přesné rozvržení stránky mezi různé sekce.

Komponenty jsou také automaticky stylovány v případě různých akcí na stránce, např. pokud pole neprojde validací, automaticky je zobrazeno červeně atp. Součástí projektu je také rozsáhlá sada ikonek, které lze použít prostým odvoláním se na jejich název. (JBoss Foundation, 2012b)

Bootstrap představuje potenciálně velmi přínosnou iniciativu, která umožní vytváření znovupoužitelných vzhledů přenosných i do jiných systémů, které sice nevyužívají RichFaces, ale využívají Twitter Bootstrap. Zároveň umožňují uživatelům RichFaces profitovat z bohaté kolekce vzhledů, která pro Twitter Bootstrap byla vytvořena. V budoucnu by také mělo být možné měnit vzhled aplikace dynamicky za běhu. Bootstrap komponenty by se měly stát plnohodnotou součástí RichFaces v připravované verzi 5. (Leathem, 2012b)

4 Rešerše prací na příbuzná témata

Tématika webových frameworků, jejich kvalit a vzájemného porovnání patří k často zpracovávaným okruhům bakalářských i diplomových prací nejen u nás, ale i ve světě. Bezesporu je tomu tak i proto, že se jedná o předmět denní praxe mnohých autorů a stále aktuální problém. Tato část obsahuje rešerši prací na příbuzná témata, která se mého předmětu zájmu dotýkají.

V mnohých ohledech jejich zkušenosti a postřehy posloužily jako inspirace a byly zapracovány do tohoto textu, v některých aspektech se mé názory odlišují a pokud jsem s nimi v rozporu, vysvětluji i důvod, proč tomu tak je. Práce ostatních autorů mi byly cenným zdrojem nápadů a občas i podnětem k přehodnocení svého pohledu na věc.

Rešerši jsem podrobila zejména práce domácích univerzit obdobného zaměření, tj. domácí VŠE, Českého vysokého učení technického v Praze a Vysokého učení technického v Brně. V oblasti zahraničních univerzit jsem se zaměřila na univerzity obdobného zaměření nebo alespoň s informatickou fakultou, aby bylo dosaženo patřičné relevance prací. Zdrojem textů byla ve všech případech databáze závěrečných prací dané univerzity.

(Vagner, 2009) i (Koščejev, 2009) se ve svých pracech soustředí na porovnání webových frameworků pro jazyk Java. Jejich přístup se v mnohých ohledech shoduje s mým, nicméně oba se zaměřují spíše na technickou stránku věci, tzn. instalaci, kvalitu chybových hlášek a podkladové návrhové vzory.

Kritéria porovnání jednotlivých produktů se v případě (Koščejev, 2009) ve velké míře shoduje s mnou navrhovanými kritérii. Jedná se zejména o kritéria kvality dokumentace, velikosti a aktivity uživatelské komunity a snadnosti nasazení a užívání. Kromě toho přidává ještě kritéria zaměřená na srovnání funkcionality a architektury jednotlivých produktů, která byly ze zde předkládaného srovnání vynechány, protože neodpovídají mému přístupu k hodnocení v této práci.

Vzhledem k tomu, že práce pana Koščejeva je přímo zaměřená na srovnání frameworků, jde samozřejmě v hodnocení a popisu produktů do mnohem větší hloubky a detailů. Autor v závěru textu podotýká, že nedospěl k jednoznačnému vítězi tohoto srovnání a nechává tedy výběr vhodného kandidáta na zkušenostech a preferencích konkrétního uživatele.

(Vagner, 2009) jde v technických detailech do ještě větší hloubky a zkoumá mechanismus validace formulářových dat, podporu ve vývojových prostředí, komponentovou skladbu frameworku a diskovou velikost souborů, které framework tvoří. Ve shodě s mým přístupem pak zkoumá též kvalitu dokumentace a podpory jednotlivých nástrojů (tj. uživatelskou komunitu a existenci ladících nástrojů).

(Válek, 2008) se zaměřuje na koncept rychlého vývoje aplikací (RAD, viz výše). Za vhodné nástroje pro tento typ vývoje však považuje pouze frameworky založené na skriptova-

cích jazycích (PHP, Ruby aj.), což z takové metodologie zcela vylučuje frameworky založené na jazyce Java, což je zcela v rozporu např. s oficiální referenční dokumentací k RichFaces. Naopak vyzdvihuje objektově orientované programování. Za nepříliš vhodné považuji striktní trvání na jediném návrhovém vzoru[48], což nepříjemně omezuje vývojové možnosti. Autor navíc ani nevysvětluje důvod, proč právě tento návrhový vzor představuje tak fundamentální prvek vhodných nástrojů.

(Pösinger, 2007) se ve své práci detailně zabývá strukturou kódu v závislosti na použitém frameworku, množstvím nutných konfiguračních souborů a složitostí konfigurace. Krátce se také zabývá rychlostí učení se novému nástroji a složitostí jeho užívání. Za zajímavý prvek považuji také analýzu poptávky po specialistech na daný framework, kterému se věnuje i (Koščejev, 2009).

Naproti tomu (Changpil, 2012) zvolil zcela odlišný přístup a zkoumá vhodnost daného frameworku pro konkrétní projekt na základě analýzy nutně vynaloženého času pro zvládnutí práce s tímto nástrojem, požadované odbornosti projektového týmu a celkových nákladů. Taktéž čerpá z výzkumu (Laakso and Niemi, 2008), který oceňuje z hlediska analytického přístupu, na druhou stranu ho však kritizuje proto, že nezohledňuje nákladový přístup. Pohled této práce se tedy blíží spíše přístupu projektového manažera.

Přístup tohoto autora předpokládá existenci funkční specifikace na počátku životního cyklu projektu, což je skutečnost, která v oblasti agilního programování nenastává příliš často a naopak částečně popírá základní agilní principy. Z těchto důvodů lze jeho závěry pro mé účely převzít jen s podrobnou revizí.

(Podolka, 2007) je autorem jediné práce, která spojuje kritéria funkčního charakteru s kritérii nákladové analýzy a uplatnitelnosti znalosti frameworku při hledání zaměstnání. Řeší také cenovou dostupnost jednotlivých nástrojů, což vzhledem k bezplatné povaze většiny vedoucích frameworků představuje poněkud nadbytečnou kategorii.

(Changpil, 2012) dále také očekává, že proces výběru webového frameworku patří ke standardním součástem životního cyklu projektu a opakuje se před započetím každého projektu. V korporátní praxi naopak spíše existuje jeden standardně užívaný nástroj (nebo malý počet nástrojů), který se nasazuje ve všech projektech.

Za zajímavý aspekt lze považovat autorův předpoklad, že jakékoliv požadavky lze naplnit s použitím jakéhokoliv frameworku, jediným limitujícím předpokladem je čas a schopnosti projektového týmu. V mnoha případech by to znamenalo nutnost doprogramování některých částí vlastními silami, nicméně i to může být efektivní, pokud ve výsledku takový projekt bude levnější, než využití jiného nástroje.

(van der Tol, 2012) se ve své práci zaměřuje přímo na framework JavaServer Faces a jeho význam pro moderní vývoj webových aplikací. Vysvětluje stávající omezení internetu,

[48] Konkrétně se jedná o Front Controller.

která vycházejí z historicky odlišeného určení celé této sítě, a popisuje vliv, jaký tato omezení mají na vývoj aplikací typu RIA. Dále se věnuje popisu fungování frameworku v souladu s principem MVC. Šablona stránky (view) je vytvořena v šablonovacím jazyce Facelets a uložena v separátním souboru. Se stránkou je na pozadí spojena třída (tzv. backing bean, představuje controller), která stránce poskytuje data, jež získává z entit (model), obvykle uložených v databázi.

Za hlavní přednosti JSF považuje integraci s AJAXem a možnost částečného překreslování stránek, snadnou integraci s prakticky jakýmkoliv projektem a dodržování principu MVC. Cílem jeho práce bylo dokázat, že standard JSF je kompatibilní se skupinou technologií HTML 5, což se mu také podařilo.

Závěrem lze říci, že problematikou webových frameworků a vzájemného srovnání jejich kvalit se zaobírá značné množství prací a jejich závěry a metody se v mnohých ohledech shodují. Není nicméně bez zajímavosti, že pohledy, ze kterých k problematice přistupují, se výrazně odlišují. Technologicko-vývojářsky orientované práce se zaměřují ve shodě s mým přístupem na kvalitu dokumentace jednotlivých nástrojů, která se projevuje v rychlosti učení se novému nástroji, na aktivitu uživatelské komunity a také na jejich architektonické kvality. Projektověji zaměřené práce pak zdůrazňují roli nákladů na pořízení a osvojení si nového nástroje a na koordinaci týmové práce, zejména mezi různými typy úkolů (návrh, vývoj, grafické práce).

5 Referenční aplikace

Součástí této práce je také vývoj referenční aplikace, která bude využívat nabízené funkcionality frameworku RichFaces a demonstrovat jejich užití, výhody a nedostatky a alternativní řešení. V následujícím textu se mj. zaměřuji na popis možností definice a změny vzhledu, validace uživatelského vstupu a způsoby spolupráce s jinými frameworky.

Referenční aplikace je nasazena na http://diplomka.balhar.net.

Přihlašovací údaje:

Uživatelské jméno: admin@gmail.com

Heslo: admin1234

V rámci aplikace lze také samozřejmě vytvořit nový uživatelský účet s vlastními přihlašovacími údaji. Vzhledem k omezeným možnostem sdíleného hostingu je možné, že aplikace bude o něco pomalejší, než bývá zvykem, popř. bude dočasně nedostupná.

5.1 Popis aplikace

Funkcionalita aplikace se zakládá na reálných požadavcích převzatých ze skutečného projektu, nicméně jejich zpracování je pojato tak, aby bylo možné v maximální možné míře předvést možnosti frameworku RichFaces.

Referenční aplikace je určena pro účely webové prezentace zábavné společenské hry na bázi improvizovaného divadla. Účast ve hře předpokládá, že se potenciální účastník zaregistruje, vybere si postavu, za kterou chce hrát, a zvolí pro svou postavu příběhové linky, kterých se chce účastnit. Aplikace tedy musí poskytovat formulář pro registraci s patřičným zabezpečením vkládaných dat, funkcionalitu pro volbu postavy a příběhových linek. Volba dramatických linií je omezena typem postavy, kterou si hráč zvolil.

Údaje o volných postavách a příběhových liniích a již přihlášených hráčích se volně zobrazují bez nutnosti přihlášení. Informace o registrovaných hráčích a možných postavách jsou na stránkách k dispozici v různém grafickém provedení (tedy s využitím různých komponent frameworku).

Hráči mají dále možnost spravovat svůj účet, tj. aplikace má zabezpečenou část, kam se lze dostat pouze po přihlášení. Správa účtu zahrnuje změnu hesla, nahrání profilového obrázku a změnu osobních údajů. Uživatelé dále mohou měnit vzhled i jazyk aplikace za běhu.

V zabezpečené části aplikace mají registrovaní uživatelé také možnost vytvářet vlastní příběhové linie. Každá příběhová linie může zahrnovat více rolí, tj. zúčastnit se jí může více hráčů a jejich účinkování je předem omezeno právě definicí role. Pokud příběh např.

vypráví o vraždě, jeden z hráčů bude hrát roli vraha, druhý pak roli možného svědka se vztahem k oběti. Vzájemné jednání těchto postav je pak do značné míry určeno vztahem vyplývajícím z příběhu.

Hráči, resp. jejich postavy se mohou účastnit více příběhových linií, nicméně jejich počet je omezen tzv. příběhovými body. Každá postava při svém vytvoření dostane přidělený výchozí počet příběhových bodů, které může za účast v příbězích utratit. Jednotlivé role se v bodovém ohodnocení liší podle důležitosti pro hru a dramatického dopadu.

5.2 Prostředí

Aplikace je vyvíjena jako čistě RichFaces/JSF aplikace bez pomoci všeobecných korporátních frameworků jako např. Spring, aby bylo možné plně vyhodnotit silné a slabé stránky nástroje při použití frameworku v základní verzi.

Ze zřejmých důvodů je aplikace postavena na nejnovější verzi RichFaces, tj. 4.3.5. Final, která využívá JSF 2. Ke správě závislostí[49] využívá nástroj správy softwaru Maven 3 a běží na webovém serveru Apache Tomcat verze 8. RichFaces ve verzích 4.x vzhledem ke své závislosti na standardu JSF 2 může být používáno pouze s webovými servery, které jej podporují, tj. Apacha Tomcat verze 7 a vyšší. S jistými opatřeními lze použít i nižší verze, nicméně takové prostředí představuje značná omezení.

Do vývoje byl s výhodou zapojen také databázový framework Hibernate (v4.2.2.Final). Kromě toho, že zajišťuje perzistenci vkládaných dat, což je typický prvek webových aplikací, poskytuje také některé další funkce prezentační vrstvě. Framework RichFaces totiž umožňuje využít Hibernate anotace pro účely validace uživatelsky definovaných dat[50] (více viz podkapitola o validaci). Databáze, s níž Hibernate v případě této referenční aplikace komunikuje, je postavena na open source řešení MySQL 5.6.

Součástí aplikace je také nástroj pro manipulaci kolekcí LambdaJ[51] vyvíjená společností Google a jedná se o nástroj zapojený spíše na bázi osobní preference. Umožňuje mj. filtro-

---

[49] Angl. dependency. Výraz se používá pro označení softwaru třetích stran, na jejichž přítomnosti je fungování aplikace závislé. Závislosti (kromě jiného) jsou konfigurovány v souboru pom.xml, který představuje ústřední bod fungování nástroje Maven. Více viz http://maven.apache.org/

[50] Resp. tuto funkcionalitu poskytuje subprojekt Hibernate Validator, který je možné do libovolné aplikace zapojit nezávisle na hlavní části frameworku. (Více viz http://hibernate.org/validator/)

[51] https://code.google.com/p/lambdaj/wiki/LambdajFeatures

vat kolekce na základě předem definovaných kritérií, extrahovat vnořené objekty a provádět na nich další agregované funkce. Takové operace obvykle vyžadují opakované procházení kolekcí (a podkolekcí) a pečlivou, problematickou manipulaci s obsaženými objekty, což je po výpočetní stránce značně náročné. Nástroj LambdaJ tuto situaci značně usnadňuje.

Př. 5.2.1: *Tradiční kód*

```
List<GameCharacter> charList = ArrayList<GameCharacter>();
for (GameCharacter char : characters) {
    if (char.getType().equal(type) {
          charList.add(char);
    }
}
```

Př. 5.2.2: *Kód s využitím LambdaJ*

```
charList = filter(having(on(GameCharacter.class).getType(), equalTo(type)), cha-
racters);
```

Mimo mnou zapojené frameworky a nástroje existují také závislosti samotného frameworku RichFaces, které musí být v projektu přítomny, aby jakákoliv aplikace na něm založená mohla fungovat. Konkrétně se jedná o následující komponenty:

- Google Guava (víceúčelová knihovna, zaobírá se např. zpracováním textu a vyrovnávací pamětí)
- CSS Parser (zpracovává dynamicky definované CSS[52] styly)
- SAC (taktéž využívaný pro zpracování CSS)

Verze závislostí frameworku jsou výhodně spravovány s užitím Mavenu a rodičovské konfigurace (soubor `pom.xml`), tudíž není nutné se o ně starat a je zaručeno, že vždy budou nejnovější (při zachování kompatibility s verzi RichFaces). Zvednutí verze RichFaces je tak otázkou přepsání jediné hodnoty v konfiguraci.

5.3 Úvodní nastavení a učící křivka

Vzhledem k mým minimálním zkušenostem s kompletní konfigurací a rozběhnutím J2EE[53] aplikace včetně nezbytné infrastruktury, ať už s použitím korporátních frameworků či bez

[52] Cascading StyleSheet.

[53] Java Enterprise Edition.

nich, se vývoj referenční aplikace pro účely této práce stal ideální příležitostí k prozkoumání učící křivky frameworku.

Rozběhnutí základní verze aplikace (tzv. Hello World example[54]) trvalo přibližně čtyři hodiny[55]. Tato aktivita zahrnovala následující kroky:

- Instalace a konfigurace prostředí pro užití nástroje Maven
- Vygenerování skeletonu aplikace za pomoci odpovídajícího Maven archetypu
- Vyřešení problémů se závislostmi projektu
- Nastavení aplikačního serveru Tomcat a připojení k MySQL

Referenční dokumentace k RichFaces se ukázala jako užitečný zdroj informací, zejména pak jako vynikající startovní bod. Na druhou stranu postrádá některé důležité informace, které se v ní buď vůbec nenacházejí, jsou rozeseté po různých sekcích či nejsou dostatečně vysvětleny (např. chybí ukázky nastavení, dokumentace místy obsahuje pouze slovní popis). Mezi chybějící informace patří např. skutečnost, že od verze 4.x je nutné používat vyšší verze aplikačních/webových serverů, že běh frameworku závisí na přítomnosti sekundárních komponent od třetích stran a že optimalizace a komprimace souborů je ve výchozím nastavení zapnutá.

Z pochopitelných důvodů také chybí informace ke kompatibilitě mezi RichFaces a ostatními frameworky, které referenční aplikace používá. Tvůrci frameworku pochopitelně nemohli předvídat všechny možné kombinace vývojového prostředí. Za poměrně podstatné však považuji informace týkající se závislostí projektu, tj. např. pokud je v rámci aplikace použit jiný framework s validační funkcionalitou, je nutné zredukovat explicitně definované závislosti frameworku RichFaces, aby nedocházelo ke konfliktům mezi různými verzemi a duplicitními třídami[56].

---

[54] Oblíbený termín užívaný pro rozběhnutí základní verze aplikace, která na obrazovku obvykle vypisuje obligátní „Hello World!“ Hello-world aplikace se používají pro ilustraci příkladů pro začátečníky.

[55] Z tohoto času cíleně vynechávám dobu strávenou řešením zablokovaného root účtu pro přístup k MySQL, neboť se nejedná o typický problém při nastavování J2EE aplikace, ale důsledek předchozího užívání této služby.

[56] V tomto případě se konkrétně jedná o knihovnu javax.persistence, která patří jak mezi závislosti frameworku RichFaces, tak databázového nástroje Hibernate.

5.4 Tvorba stránky, šablonování

Každá stránka, která bude tvořit součást budoucí aplikace, musí mít svůj vlastní XHTML soubor, v němž bude obsažena její definice. Vytváření stránky probíhá velice podobně, jako při psaní standardního HTML kódu – používají se značky (tagy) a jejich atributy.

Stránka může být tvořena dvěma hlavními druhy prvků[57] – standardními RichFaces komponentami a ajaxovými komponentami. Doplňkovou roli hrají komponenty standardu JSF. Pro správné fungování stránky[58] musí její záhlaví kromě deklarace typu dokumentu obsahovat i tzv. jmenné prostory, které zpracovávajícímu procesu říkají, odkud se daná komponenta bere a jak ji zpracovat. V rámci RichFaces se obvykle lze setkat s pěti jmennými prostory, které zároveň odpovídají pěti základním typům komponent, které se při tvorbě stránky používají:

- JSF verze klasických HTML komponent (jmenný prostor http://java.sun.com/jsf/html)
- Jádro standardu JSF (jmenný prostor http://java.sun.com/jsf/core)
- JSF prvky pro definování rozložení stránky (jmenný prostor http://java.sun.com/jsf/facelets)
- Ajaxové komponenty RichFaces (jmenný prostor http://richfaces.org/a4j)
- Jádro frameworku RichFaces (jmenný prostor http://richfaces.org/rich)

Nejjednodušší definice stránky může vypadat zhruba následujícím způsobem[59]:

Př. 5.4.1: *Základní stránka*

```
<html xmlns:rich="http://richfaces.org/rich"
      xmlns:h="http://java.sun.com/jsf/html">
<h:head/>
<h:body>
    <rich:panel>
          <h:outputText value="Hello World!"/>
    </rich:panel>
</body>
</html>
```

[57] V rámci nově vyvíjené verze 5.0 (stále ve verzi alfa) budou obě komponentové sady sloučeny do jedné kolekce.

[58] Deklarace jmenného prostoru kromě zabezpečení fungování stránky také u většiny vývojářských nástrojů umožňuje využít jejich potenciálu a velice užitečného našeptávání názvu prvků i atributů a jejich možných hodnot.

[59] Pro účely jednoduchosti vynechávám deklaraci typu dokumentu.

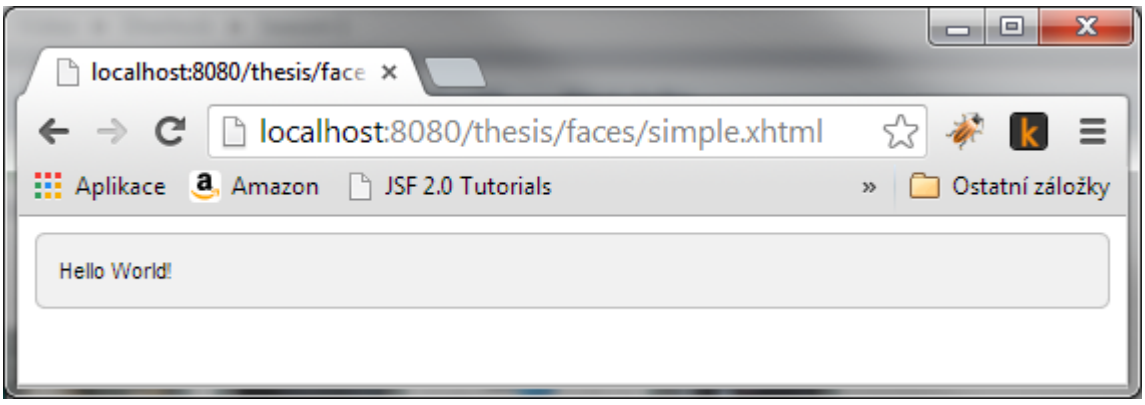
Tato definice stránky s užitím JSF a RichFaces značek projde zpracováním jádra frameworku a na základě výše uvedeného zápisu bude vygenerován validní HTML dokument, kde všechny elementy budou opatřeny náležitými stylovými třídami. Vygenerován bude také relevantní CSS soubor a v případě potřeby také JavaScript pro ovládání komplexnějších prvků.


```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <link type="text/css" rel="stylesheet"
          href="/thesis/faces/rfRes/skinning.ecss?db=eAFTvJp8DwAEqwI4"/>
    <script type="text/javascript"
            src="/thesis/faces/javax.faces.resource/jsf.js?ln=javax.faces&amp;stage=Development">
    </script>
    <script type="text/javascript"
            src="/thesis/faces/javax.faces.resource/jquery.js">
    </script>
    <script type="text/javascript"
            src="/thesis/faces/javax.faces.resource/richfaces.js">
    </script>
    <script type="text/javascript"
            src="/thesis/faces/javax.faces.resource/richfaces-base-component.js">
    </script>
    <link type="text/css" rel="stylesheet"
          href="/thesis/faces/rfRes/panel.ecss?db=eAFTvJp8DwAEqwI4&amp;ln=org.richfaces"/>
</head>
<body>
<div class="rf-p " id="j_idt5">
    <div class="rf-p-b " id="j_idt5_body">Hello World!</div>
</div>
</body>
```

Obr. 4: *Vygenerovaný zdrojový HTML kód*

Zdroj: autor

Výsledkem zpracování tohoto krátkého zápisu je následující stránka (kulaté rohy jsou důsledkem aplikace vlastního skinu, viz dále):



Obr. 5: *Výsledná webová stránka*

Zdroj: autor

Tato stránka obsahuje pouze statické prvky a v zásadě není důvod, aby byla generována s pomocí pokročilého šablonovacího frameworku. Předpřipravené šablony stránek se použí-

vají proto, aby bylo možné do statického neměnného rámce, který představuje kostra webové stránky, dynamicky vkládat výstupy programu, resp. aplikace, která běží na pozadí.

RichFaces se drží jmenných konvencí standardu JSF[60] a na atributy, resp. metody Javovských tříd se odkazuje s použitím konstrukce `#{nazevTridy.nazevAtributu}`, resp. `#{nazevTridy.nazevMetody()}`.

V případě, že se webová vrstva aplikace pokouší pracovat s atributy, není nutno specifikovat, zda se nám jedná o čtecí či přiřazovací operaci, framework vyhodnotí a použije správnou metodu dle kontextu[61]. Tuto skutečnost lze ilustrovat na jednoduchém příkladu vstupního pole ve formuláři, což je jeden z nejběžnějších prvků webových aplikací.

Př. 5.4.2: *Vstupní pole formuláře*

```
<h:form id="regForm">
    <h:outputText value="#{msg['form.label.name']}:"/>
    <h:inputText id="name" value="#{regBean.dummyUser.name}">
    // zbytek formuláře
</h:form>
```

Při vykreslování formuláře framework najde odpovídající Java třídu, na ní pak odpovídající objekt (v tomto případě instanci třídy `User`)[62] a zavolá čtecí metodu pro získání hodnoty atributu `name`. Naopak v případě odeslání formuláře po proběhnutí obdobného procesu zavolá metodu přiřazovací a aktualizuje hodnotu atributu novými daty.

Kromě těchto činností framework (resp. nižší vrstva v podobě standardu JSF) chrání webovou prezentaci aplikace před výjimkami typu Null Pointer, tj. před situací, kdy by hodnota objektu `dummyUser` byla tzv. `null`, tedy objekt by nebyl inicializovaný. Za normálních okolností by se při pokusu číst atributy neinicializovaného objektu objevila výjimka, která by se zobrazila uživateli a přerušila by chod aplikace. Při užití RichFaces se tyto výjimky prakticky nevyskytují, protože pokud je objekt, na němž voláme metody, prázdný, framework jednoduše potlačí volání těchto metod. Směrem k uživateli se pole zobrazí jako prázdné, což je jednoznačně uživatelsky přívětivější[63].

---

[60] V dřívějších verzích při užití standardu JSP byla níže zmíněná konstrukce značně podobná. Na atributy a metody se odkazovalo s užitím konstruktu `${nazevTridy.nazevAtributu}`.

[61] Častěji se používají anglické termíny getter (čtecí metoda) a setter (přiřazovací metoda).

[62] Pro získání objektu se pochopitelně také volá čtecí metoda na původní třídě.

[63] Problémy nastávají až při odeslání formuláře, kdy už nelze skutečnost, že neexistuje žádný objekt, kterému bychom mohli hodnoty přiřadit, řešit obecnou cestou.

Třídy, na jejichž metody a atributy framework „vidí“ (tzv. *managed beans*), musí splňovat několik podmínek. Zaprvé se musí jednat o objekty typu POJO[64], tj. musí mít bezparametrický konstruktor a všechny atributy musí mít čtecí i přiřazovací metody. Kromě toho musí být specifickým způsobem označeny tak, aby je jádro frameworku vidělo a zahrnulo do kontextu aplikace. Za těchto podmínek se lze na takové třídy odkazovat ze šablon stránek.

Označení třídy lze provést dvěma základními způsoby. Buď může být třída explicitně uvedena v konfiguračním souboru frameworku `faces-context.xml` nebo může být označení součástí samotného kódu třídy v podobě anotace. Stejně jako v případě dříve zmiňovaného frameworku Hibernate, i v tomto případě preferuji anotace, které jsou méně rozvláčné.

Př. 5.4.3: *Označení třídy s užitím konfiguračního souboru*

```
...
<managed-bean>
    <managed-bean-name>regBean</managed-bean-name>
    <managed-bean-class>org.docs.richfaces.security.RegBean</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
...
```

Př. 5.4.4: *Označení třídy s užitím anotace*

```
@ManagedBean
@RequestScoped
public class RegBean implements Serializable {
    // kód třídy
}
```

Obě varianty obsahují přesně stejné informace. Rozsah platnosti třídy (udán v tomto případě anotací `@RequestScoped`, resp. atributem `managed-bean-scope`) určuje omezení doby, po níž jsou data v třídě obsažená platná. Existuje několik běžně používaných alternativ:

1. Request Scope – data jsou platná po dobu jednoho HTTP požadavku,
2. View Scope – data jsou platná, dokud uživatel zůstane na jedné stránce,
3. Session Scope – data jsou platná, dokud platí i session[65] uživatele,
4. Application Scope – data jsou platná po celou dobu běhu aplikace.

---

[64] Plain Old Java Object. Třída s bezparametrickým konstruktorem a všemi čtecími a přiřazovacími metodami. (Fowler, 2003)

[65] Bez českého ekvivalentu. V zásadě se jedná o relaci/konverzaci mezi uživatelem a aplikací, která zahrnuje více HTTP požadavků.

5.4.1 Šablonování

Jednotlivé stránky v rámci stejné webové aplikace mají tradičně jedno společné – vypadají zhruba stejně. Obvykle obsahují nějaké záhlaví a zápatí, menu a hlavní obsah. Z vývojářova hlediska se tak vlastně jedná o duplicitní kód, který by bylo vhodné extrahovat někam stranou, aby mohl být znovupoužit a aby se nacházel pouze na jednom místě. Výhodou takového přístupu je rychlost úprav, dodržení konzistence (tj. změny se promítnou naráz do všech stránek a stejným způsobem) a možnost flexibilně tento kód zaměňovat za jiný (tj. přepínat mezi různými rozloženími).

Procesu rozdělování stránky na více podstránek a jejich vzájemného skládání se říká šablonování. Ve zkratce to znamená, že se vytvoří šablona stránky, která zahrnuje společné prvky (tj. zmiňované záhlaví, zápatí, menu a hlavní obsah), kterou budou používat všechny stránky v dotyčné aplikaci. Jednotlivé stránky se pak pouze odkazují na rodičovskou šablonu a samy definují pouze ty elementy, které jsou pro ně specifické.

V tomto ohledu se RichFaces spoléhají na podporu ze strany standardu JSF, tj. k definici a skládání šablon a jejich součástí využívají jádro standardu. JSF umožňuje také implementovat dědičnost mezi šablonami, tj. extrahovat vzájemně disjunktní podmnožiny sdílených rysů do provázaných šablon a použít vždy tu, která odpovídá aktuální stránce.

Jako příklad lze uvést např. rozhraní aplikace pro běžné uživatele a pro administrátory. Mnoho prvků mají společných, ale některé jsou navíc/chybí. Šablony pro obě rozhraní by v takovém případě dědily elementy od mateřské šablony a zároveň by doplňovaly další prvky v závislosti na typu uživatele. Stránky určené koncovým uživatelům by pak užívaly jinou šablonu než stránky administrátorské.

Referenční aplikace využívá jednoduchou šablonu[66], která definuje záhlaví, zápatí, menu a prostor pro hlavní obsah. Záhlaví a zápatí jsou vždy stejné, šablona tedy obsahuje přesný odkaz na soubor, který obsahuje definici těchto oblastí.

Naproti tomu název stránky a její obsah se může měnit, tudíž jsou v šabloně umístěny jen zástupné symboly[67], u nichž se očekává předefinování v další úrovni šablony nebo ve stránce samotné. Významné prvky šablon a stránek jsou v následujícím ukázkovém kódu vyznačeny tučně, přičemž propojení šablon a stránek je označeno zeleně, zástupné symboly červeně a definice vymezené oblasti pak modře.

---

[66] Ve skutečnosti jsou to dvě vnořené šablony.

[67] Anglicky placeholders. Jedná se o prvky, které vymezují místo na stránce, kam přijde později dodaný obsah.

Př. 5.4.5: Šablona stránky – nejvyšší úroveň (soubor `template.xhtml`)

```
<html>
<h:head>
    <ui:insert name="title">Default title</ui:insert>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <h:outputStylesheet library="css" name="skin.css"/>
    ...
</h:head>

<h:body>
    <f:view locale="#{langBean.locale}">
          <ui:include src="header.xhtml"/>                    //záhlaví
          <ui:insert name="body">Default content</ui:insert>
          <ui:include src="footer.xhtml"/>                    //zápatí
    </f:view>
</h:body>
</html>
```

Př. 5.4.6: *Šablona stránky – nižší úroveň (soubor* `pageTemplate.xhtml`*)*

```
<html>
<ui:composition template="/templates/template.xhtml">
    <ui:define name="body">
        <div id="infoDiv">
            <ui:include src="../includes/sideMenu.xhtml"/>
        </div>
        <div id="contentDiv">
            <ui:insert name="content"/>
        </div>
    </ui:define>
</ui:composition>
</html>
```

Př. 5.4.7: *Stránka využívající šablonu nižší úrovně*

```
<html>
<body>
    <ui:composition template="/templates/pageTemplate.xhtml">
    <ui:define name="content">
        <rich:panel>
            test content
        </rich:panel>
    </ui:define>
    </ui:composition>
</body>
</html>
```

5.5 Vzhled aplikace - definice a přizpůsobení

Framework RichFaces nabízí mnoho způsobů, jak rychle získat rozumně vypadající aplikaci a definovaný vzhled snadno a pružně přizpůsobovat, což je v souladu s principy RAD. V oblasti definice a přizpůsobení vzhledu vlastních komponent RichFaces nabízí tři možnosti:

1. Využití předpřipraveného vzhledu dodávaného s frameworkem
2. Vytvoření vlastního vzhledu
3. Rozšíření a modifikace existujícího vzhledu

Mimo přizpůsobení vzhledu nativních komponent RichFaces pochopitelně umožňuje zapojit i další prvky v podobě stylopisu. Tyto doplňující možnosti se obvykle užívají k definici rozložení stránky a pozicování jednotlivých elementů.

5.5.1 Předpřipravené vzhledy

Nejjednodušším způsobem, jak využít grafické možnosti frameworku, je zapojení jednoho ze sedmi předdefinovaných stylů. Tyto připravené vzhledy (obvykle se používá anglický termín "skiny") definují všechny potřebné CSS třídy, podle nichž jsou nativní komponenty frameworku vykresleny. Předdefinované skiny jsou součástí distribuce frameworku a jsou okamžitě k dispozici. Autor aplikace, která tovární nastavení vzhledu používá, nemusí napsat jedinou řádku CSS kódu a může se soustředit na podstatnější problémy, jako je např. funkcionalita.

Zapojení předdefinovaného vzhledu je velmi snadné, stačí přidat odpovídající parametr do konfiguračního souboru webové aplikace *web.xml*. Tento soubor představuje jádro webové aplikace a obsahuje všechna podstatná nastavení pro její běh, jako např. filtrování příchozích požadavků, umístění úvodní stránky a definici formátu adresy webové stránky, která má projít přes zpracování skrze jádro frameworku RichFaces.

Př. 5.5.1: *Nastavení statického vzhledu*

```
<context-param>
    <param-name>org.richfaces.skin</param-name>
    <param-value>ruby</param-value>
</context-param>
```

Stejnou konstrukci lze použít také k pružné změně vzhledu za běhu aplikace s malou obměnou. Vykreslení celé aplikace tak lze změnit prostým stiskem tlačítka, přičemž ovlivněny jsou z pochopitelných důvodů pouze vlastní komponenty frameworku a příbuzné komponenty z výše zmíněných jmenných prostorů.

Př. 5.5.2: *Nastavení dynamického vzhledu – web.xml*

```
<context-param>
     <param-name>org.richfaces.skin</param-name>
     <param-value>#{skinBean.skin}</param-value>
</context-param>
```

Př. 5.5.3: *Nastavení dynamického vzhledu – relevantní kód třídy SkinBean*

```
    @ManagedProperty(value="ruby")
    private String skin;

    public void changeSkin() {
        if (skin.equals("custom")) {
            setSkin("ruby");
        } else {
            setSkin("custom");
        }
    }
```

Užití továrního nastavení je pochopitelně poměrně generický, ač snadný přístup, který většině tvůrců moderních webových aplikací nebude vyhovovat, neboť jejich záměrem je obvykle své aplikace vzhledově odlišit od ostatních.

5.5.2 Vlastní vzhled, přizpůsobení, dědičnost

Pokud má vzhled aplikace zásadní význam nebo dodávané skiny nesplňují požadavky (např. kvůli potřebě dodržet firemní pravidla pro grafické rozhraní aplikací), framework umožňuje vytvářet vlastní definice toho, jak má aplikace vypadat.

Každá nativní komponenta je spojena se značným množstvím CSS tříd, které stanovují vzhled jednotlivých částí dotyčného prvku. Vytvoření nového vzhledu takzvaně na zelené louce tudíž představuje zdlouhavý a náročný proces, kdy hrozí nebezpečí opomenutí přepsání některých vlastností a následná grafická nekonzistence.

V mnoha případech tak může být jednodušší použít jako základ vzhledu nové aplikace některý z předdefinovaných skinů, které framework nabízí, a změnit pouze ty části, které neodpovídají představám tvůrce. RichFaces tedy nabízejí jistou omezenou dědičnost při zpracování grafické stránky aplikace.

Pro vytvoření takového skinu je potřeba pouze dodržet jmenné konvence[68] frameworku při vytváření souboru s novým vzhledem, umístit jej na classpath[69] aplikace a odkázat se na rodičovský vzhled.

---

[68] Název skinu musí mít následující formu: *názevSkinu.skin.properties*.

Př. 5.5.5: *Definice rozšiřujícího skinu*

```
baseSkin=ruby                  // definice rodičovského skinu

panelBorderRadius=5px          // stanovení vlastních hodnot CSS stylů
```

S implementací vlastních skinů je spojen problém technického rázu. RichFaces ve výchozím nastavení provádí kompresi a slučování všech konfiguračních souborů pro dosažení maximální efektivity. Zahrnutí uživatelsky definovaných skinů do tohoto procesu je otázkou netriviálního nastavení[70] s užitím Mavenu, který umožňuje vygenerování odpovídajících výsledků při zahrnutí uživatelských souborů. Ve většině případů, pokud výkonnost a efektivita nejsou stěžejní, stačí výše zmíněnou optimalizaci vypnout následujícím způsobem:

Př. 5.5.6: *Vypnutí optimalizace souborů*

```
<context-param>
       <param-name>org.richfaces.resourceOptimization.enabled</param-name>
       <param-value>false</param-value>
</context-param>
```

Třetí a poslední standardní možností v oblasti definice vzhledu je vytvoření vlastního vzhledu zcela od začátku. Soubor, který vzhled obsahuje, musí dodržovat stejné podmínky a jmenné konvence jako v případě rozšíření již existujícího skinu a jeho struktura je obdobná jako v příkladu 5.5.5 (viz výše). Parametr `baseSkin` z pochopitelných důvodů chybí.

V tuto chvíli se zdá vhodné vysvětlit, jakým způsobem jádro frameworku s uživatelsky definovanými skiny vlastně pracuje[71]. Při spuštění webové aplikace jsou parametry v příslušném souboru se skinem zpracovány a na základě těchto údajů framework generuje CSS soubor, který obsahuje definice stylových tříd jednotlivých komponent. Tento soubor je pak připojen k vygenerované webové stránce. Do jedné stylové třídy bývá mapováno více logicky souvisejících parametrů původního skinu. Zároveň jeden parametr může ovlivňovat více stylových tříd.

---

[69] Prakticky nepřeložitelný odborný termín, specifický pro vývoj v jazyce Java. Jedná se v zásadě o seznam adresářů a souborů, které virtuální stroj Javy „vidí“, tj. je schopen s nimi za běhu pracovat. Classpath bývá obvykle nastavována jako parametr v rámci proměnných prostředí (při užití OS Windows).

[70] Viz https://community.jboss.org/people/michpetrov/blog/2012/11/27/resource-optimization-with-maven-resources-plugin

[71] Podobným procesem prochází i předpřipravené skiny.

Vytvoření původního vzhledu zcela od začátku představuje mnoho hodin pečlivé práce, při které je třeba dbát toho, aby autor definoval hodnoty všech parametrů a aby jednotlivé parametry byly mezi sebou konzistentní. Počet parametrů (a výsledných CSS tříd) může být skutečně vysoký. Pro představu uvádím část parametrů a tříd pro komponentu `rich:dataTable`[72]. Celkový počet tříd a parametrů je ve skutečnosti ještě vyšší – 18 tříd využívá 9 různých parametrů pro definování jednoduché tabulky.

Tabulka 4: *Přehled parametrů a tříd pro komponentu* `rich:dataTable`

Výsledná CSS třída	**Parametry původnío vzhledu**	**Ovlivněné prvky**
.rf-dt	tableBackgroundCoolr tableBorderWidth tableBorderColor	Celá tabulka
.rf-dt-c	tableBackgroundColor tableBorderWidth tableBorderColor generalTextColor generalFamilyFont generalSizeFont	Buňka tabulky
.rf-dt-hdr-c	tableHeaderBackgroundColor tablerBorderWidth tableBorderColor tableHeaderTextColor generalFamilyFont generalSizeFont	Buňka záhlaví tabulky
.rf-dt-shdr-c	tableHeaderBackgroundColor tablerBorderWidth tableBorderColor tableHeaderTextColor generalFamilyFont generalSizeFont	Buňka pod-záhlaví tabulky

Zdroj: (JBoss, 2013)

[72] Komponenta je vykreslována jako klasická tabulka, nabízí však některé nadstandardní funkce, které se při vývoji aplikací bez komponentových frameworků obvykle řeší kombinací CSS a JavaScriptu. V rámci komponent RichFaces je tato komponenta relativně komplexnější.

5.5.3 Další možnosti, zapojení externího CSS

Výše uvedené možnosti skinování[73] se vztahují na nativní komponenty frameworku a jejich součásti. Generované webové stránky ovšem kromě těchto komponent používají i jiné prvky, jako např. klasické HTML nebo komponenty, které nabízí samotný standard JSF. Pro účely stylování těchto prvků je možné zapojit do vykreslení stránky také externí stylopisy.

Nelze však použít klasické vložení stylopisu jako při psaní "obyčejného" HTML, tj. značku `<link>`, protože díky zapojení RichFaces pro generování stránky by pak výsledná cesta ke stylopisu nebyla vygenerována správně. Stylopis lze připojit užitím JSF značky `<h:outputStylesheet>`. V takovém případě se cesta k souboru vytvoří správně. Podobná omezení platí i pro ostatní externí součásti stránky, tj. např. obrázky.

Př. 5.5.7: *Definice externího stylopisu*

```
<head>
    ...
    <h:outputStylesheet library="css" name="skin.css"/>
    ...
</head>
```

Výše uvedený příklad počítá s umístěním stylopisu na výchozí očekávané cestě projektu, konkrétně *umístění projektu/webapp/resources/css/skin.css*.

Vzhledem k tomu, že od verze 4.x už RichFaces neobsahují komponentu `rich:page` ani `rich:layout`, které sloužily k rozvrhování stránky, používají se externí stylopisy zejména pro definování kontejnerů, které budou obsahovat jednotlivé sekce stránky, a pro jejich pozicování.

5.6 Validace uživatelského vstupu

Uživatelská data představují pro webové aplikace důležitý a zároveň problematický prvek. Bez nich většina aplikací postrádá smysl, nicméně značné potíže nastávají při jejich zadávání.

Pomineme-li úmyslné pokusy o poškození aplikace vkládáním škodlivého kódu do vstupních polí v naději, že autor aplikace na takovou možnost zapomněl, nelze brát na lehkou váhu skutečnost, že uživatelé mají tendenci chybovat. Už zadávání obyčejného telefonního čísla může představovat problém – kolik číslic aplikace očekává? Mám zadat mezistátní předvolbu? Se symbolem plus nebo se dvěma nulami na začátku?

[73] Proces definice a přizpůsobení vzhledu (alias skinu).

K tomuto účelu je nutné při zpracování uživatelského vstupu provádět jeho validaci, tj. zkontrolovat, zda jsou data v takové podobě, v jaké je aplikace očekává, ať už se jedná o typ, délku či složení vstupních hodnot.

Co se škodlivého kódu týče, veškerý uživatelský vstup je automaticky filtrován a speciální znaky převedeny do neškodné podoby. RichFaces dále nabízí několik možností validace ostatních hledisek. Konkrétně se jedná o tyto typy validace:

1. Klientská validace dle standardu JSR-303[74],
2. Klientská validace dle standardu JSF
3. Validace na serveru

Klientská validace je prováděna v prohlížeči uživatele, tedy bez zátěže pro server. Její výhodou je zejména rychlost odezvy vůči uživateli, šetření se zdroji serveru a možnost vyhodnotit zadávaná data ještě předtím, než budou na server odeslána. Uživatel tak má možnost opravit své zadané informace prakticky okamžitě. Chybové hlášky při nevalidním vstupu obvykle obsahují i informace o tom, jak by měla zadávaná data vypadat.

Při validaci dle standardu JSR-303 je nutné do aplikace kromě RichFaces samotných zapojit framework, který poskytuje funkcionalitu pro její provedení. V rámci aplikace vyvíjené pro tuto práci se používá Hibernate Validator, který zároveň plní funkci referenční implementace tohoto standardu.

Pro aktivování této validace stačí vnořit element `rich:validator` do jakéhokoliv prvku, který umožňuje uživatelský vstup. Chybová hláška se pak objeví okamžitě poté, co dotyčný prvek ztratí kurzor. Kritéria pro validaci jsou specifikována za použití anotací výše zmíněného frameworku Hibernate Validator, což mimo jiné zajišťuje konzistenci mezi požadavky kladenými na uživatelský vstup v rámci webové aplikace i databáze.

Anotace poskytují široké možnosti pro specifikování potřebných kritérií. Lze definovat minimální a maximální délku textu, limity pro přípustné hodnoty čísel, regulární výraz a nenulovost odkazu na jiné objekty (tj. povinnost vztahu).

Př. 5.6.1: *Specifikace kritérií pro validaci dle standardu JSR-303*

```
public class User {
    @Size(min = 8, max = 500)
    private String password;
    ...
}
```

[74] Dokument týkající se validace tříd typu POJO. Autorem je expertní skupina, jejíž součástí jsou např. zástupci Sun Microsystems, Oracle a RedHat. (Bean Validation Expert Group, 2009)

Př. 5.6.2: *Použití validace v rámci definice stránky*

```
<h:inputSecret id="pass1" value="#{regBean.dummyUser.password}">
       <rich:validator/>
</h:inputSecret>
<rich:message for="pass1"/>
```

Jistou nevýhodou této validace je, že má čistě informativní povahu a nijak nezabraňuje uživateli chybná data odeslat, tj. nelze validaci navázat na stisknutí formulářového tlačítka. Jediný způsob, kterým se mi podařilo v rámci referenční aplikace navázat klientskou validaci na odeslání formuláře, je skrze chybové hlášky. Pokud nějaké v kontextu aplikace jsou, tlačítko bude neaktivní.

Př. 5.6.3: *Vyhodnocování přítomnosti chybových hlášek*

```
public boolean hasErrors() {
    FacesContext context = FacesContext.getCurrentInstance();
    return context.getMaximumSeverity()!= null;
}
```

S použitím standardu JSF lze dosáhnout stejného výsledku s užitím jiných prostředků. Místo elementu `rich:validator` se do vyhodnocovaných prvků vkládají elementy z rodiny `f:validate`, kdy je pro každý typ validace potřeba vnořit nový prvek. Aby bylo dosaženo stejného efektu jako v předchozím případě, je nutno zapojit dva různé validátory.

Př. 5.6.4: *Validace dle standardu JSF*[75]

```
<h:inputSecret id="pass1" value="#{regBean.dummyUser.password}">
    <f:validateLength minimum="8" maximum="500"/>
    <f:validateRequired/>
</h:inputSecret>
```

V obou výše zmíněných případech se jednalo o validaci na straně prohlížeče, což neumožňuje pokročilejší kontrolu, jako např. porovnání hodnot dvou různých vstupních polí. K tomuto účelu framework poskytuje komponentu `rich:graphValidator`, která byla navržena právě pro potřeby složitější validace na serveru.

Pole, která mají být součástí validace, je potřeba touto komponentou obalit. Veškeré relevantní metody, které se před odesláním formuláře mají spustit, se opatří anotací `@AssertTrue` nebo jinou anotací ze stejné rodiny. Jejich název musí dodržovat jmennou konvenci `isNázevMetody` a může jich být v zásadě nekonečně mnoho. Vykonávány jsou v abecedním pořadí.

---

[75] Relativně vysoká maximální délka hesla je způsobena nikoliv tím, že bych očekávala extrémně velká hesla, ale protože se heslo ukládá v zašifrované podobě, která je výrazně delší.

Př. 5.6.5: *Validační metody na serveru*

```
@AssertTrue(message = "#{msg['error.notLoggedIn']}")
public boolean isLoggedIn(){
   return authBean.getCurrentUser() != null;
}

@AssertTrue(message = "#{msg['error.passwordMismatch']}")
public boolean isPasswordsMatch() {
    return password.equals(passwordAgain);
}
```

5.7 Internacionalizace a lokalizace

Většina webových aplikací v současné době aspiruje na mezinárodní využití, a proto existuje ve více jazykových mutacích, tj. obsah aplikace je nabízen ve více variantách pro konkrétní prostředí.

V tomto kontextu se běžně užívají termíny internacionalizace a lokalizace, kde internacionalizace představuje přípravu aplikace na fungování ve více jazycích a lokalizace pak samotné přizpůsobení obsahu konkrétním jazykově-kulturním okolnostem. Lokalizace se neomezuje na prosté přeložení statických textů do jiných jazyků. Přizpůsobit je nutno také konvence pro zobrazení dat[76], rozložení kalendáře, symbolu měny aj.

RichFaces pro účely přizpůsobení textu využívá standardní přístup v podobě tzv. *property files*, tj. textových souborů, které jsou strukturovány na bázi klíč – hodnota, kde existuje zvláštní soubor pro každou mutaci. Klíče jsou pochopitelně shodné, hodnoty se pro jednotlivé mutace liší. Aplikace samotná se odkazuje pouze na klíč, přičemž konkrétní verze textu se volí za běhu na základě zvoleného jazyka.

Zobrazení dat a kalendářů aj. je řízeno automaticky samotným frameworkem. Pro využití této funkcionality je v zásadě potřeba pouze obalit mateřskou šablonu značkou `f:view` a předat jí jako hodnotu atribut odkaz na aktuálně používaný jazyk. Tento způsob uspořádání také umožňuje bez obtíží měnit lokalizaci za běhu aplikace.

Při využití validace na straně klienta dle standardu JSR-303 (viz výše) jsou chybové hlášky v lokalizované podobě poskytovány přímo frameworkem Hibernate Validator. V případě českého jazyka jsou nicméně poněkud nešikovně přeložené a obsahují nepřesné a zavádějící informace.

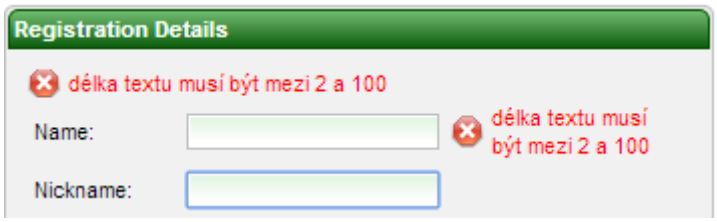
Framework naštěstí poskytuje možnost pro předefinování těchto hlášek tak, aby dávaly ten správný smysl a nemátly uživatele více, než je nutné. Za tímto účelem je nutné definovat

[76] Zde ve smyslu datum, tj. určení dne.

nový soubor s lokalizovanými hláškami pod názvem `ValidationMessages_cs.properties` v adresáři *umístění projektu/webapp/resources/*. Při zpracování lokalizace má tento soubor přednost před výchozími zprávami implementace standardu.

Obr. 6: *Výchozí chybová hláška.*

Zdroj: autor



Obr. 7: *Předefinovaná chybová hláška.*

Zdroj:autor

Referenční aplikace obsahuje českou a anglickou lokalizaci. Validační chybové hlášky jsou předefinovány pouze pro českou verzi, neboť v angličtině jsou po významové stránce v pořádku. Lokalizovány jsou všechny texty včetně programově přidávaných chybových hlášek.

5.8 Dokumentace, aktivita komunity

Kvalitní a úplná (nikoliv nutně obsažná) dokumentace představuje prvek se zásadním významem pro rozšíření a popularitu jakéhokoliv softwarového nástroje. Tato důležitost byla zmiňována již v úvodních kapitolách, konkrétně v části zabývající se porovnáváním jednotlivých frameworků.

RichFaces jako projekt open-source komunity je zdokumentován překvapivě dobře. Dokumentace je kvalitně zpracována a průběžně aktualizována, kromě vývojářských průvodců k jednotlivým verzím zahrnuje také kratší dokumenty vázané k jednotlivým subverzím. Tyto dokumenty obsahují změny, které se v dotyčné verzi udály, a tipy k migraci ze starší verze, je-li to třeba. Přes některé nedostatky a neúplnosti lze dokumentaci označit za výborně zpracovanou a snadno dostupnou.

Komunita okolo frameworku je neúnavně aktivní, jednotliví uživatelé hlásí chyby, podávají návrhy na zlepšení a vznášejí dotazy, popř. pomáhají ostatním s jejich problémy. Pouze

velmi kreativnímu vývojáři v silně atypickém projektu se může stát, že přijde s problémem, na který předtím nikdo nenarazil. Odezva komunity na problémy začátečníků i pokročilejších uživatelů je velmi rychlá a konstruktivní. V řešení problémů jsou aktivní i sami tvůrci frameworku, kteří se kromě vlastního dedikovaného fóra na oficiálních stránkách frameworku angažují i v pomoci na známých stránkách od vývojářů pro vývojáře StackOverflow[77].

Framework obsahuje také podporu pro vývojáře v podobě metadat o jednotlivých komponentách, což výrazným způsobem usnadňuje práci ve vývojovém prostředí. Díky těmto informacím totiž každý pokročilejší nástroj pro vývoj aplikací umí uživateli napovídat možné názvy značek, které chce použít, a použitelné atributy pro konkrétní značku.

Za mírný nedostatek je možné označit skutečnost, že metadata neobsahují i vymezení povolených hodnot. Tato funkcionalita by byla obzvláště vítána tam, kde validní hodnoty atributů závisí na kontextu a existuje jich omezené, předem definované množství.

Jako příklad lze uvést atribut `event` u značky `a4j:ajax`, kde se definují reakce aplikace na JavaScriptové události. Jiné události generuje komponenta tabulky (např. událost vybrání řádku), jiné události generuje zaškrtávací políčko, nicméně množina možných generovaných událostí je předem definovaná a poměrně malá. Napovídání možných hodnot atributů pro podobné případy by výrazně zrychlilo vývoj a odstranilo nutnost konzultovat referenční dokumentaci ke konkrétní komponentě.

5.9 Vlastní komponenty

Framework RichFaces disponuje také funkcionalitou k definici vlastních komponent. Jedná se o vlastně samostatný projekt vyvíjený z iniciativy tvůrců původního frameworku, tzv. Component Development Kit, dále jen CDK.

Autoři frameworku tvorbou tohoto projektu reagují na skutečnost, že softwaroví vývojáři v mnoha případech potřebují zkombinovat existující komponenty do jednoho komplexnějšího prvku a tuto kombinaci opakovaně využívat na různých místech aplikace. Typickým příkladem podobné komponenty je např. rozbalitelný kalendář jako rozšíření vstupního pole.

V takovém případě je pro vývojáře nesporně pohodlnější definovat mateřskou komponentu a její složení, než pokaždé opakovat stejnou kompozici na různých místech, což je navíc postup náchylný k chybám a konfliktům.

[77] http: //stackoverflow.com/

Projekt CDK se snaží umožnit vývojářům tvorbu nových komplexnějších komponent v čisté a snadno spravovatelné formě. Kromě definice HTML prvků, ze kterých se komponenta skládá, k nim umožňuje také připojit prvky a funkce, které nabízí JavaScript, zejména pak knihovna jQuery.

Tvorba vlastní komponenty probíhá jako separátní projekt ve vztahu k hlavní aplikaci. Iniciativa CDK předpokládá využití nástroje Maven, který z relevantních souborů vygeneruje patřičnou komponentu či komponenty. Výsledný archiv typu JAR pak hlavní aplikace uvede jako svoji závislost v rámci konfigurace a pro použití komponenty v rámci stránek je třeba definovat odpovídající jmenný prostor. Zapojení vlastních komponent do hlavního kódu je tedy poměrně velmi jednoduché a nenákladné.

Samotný proces tvorby komponenty doznal výrazných změn mezi verzemi 3 a 4. Došlo zejména k pozitivním posunům směrem k nižší složitosti vytváření komponent a zprůhlednění celého procesu.

K vytvoření komponenty je potřeba provést následující kroky:

- Vytvořit šablonu komponenty (XML soubor)
- Vytvořit abstraktní třídu jako reprezentaci této komponenty
- Vytvořit vykreslovací třídu jako rozšíření některé z existujících tříd tohoto typu

V zásadě nejsložitější částí celého procesu je tvorba XML definice cílové komponenty. Obě výše zmíněné třídy představují pouze jednoduché konstrukce, ze kterých bude skutečný operační kód komponenty automaticky vygenerován při kompilaci aplikace na základě konfigurace projektu a rodičovských tříd.

V rámci referenční aplikace byla vyvinuta komponenta, která kombinuje funkcionalitu vstupního pole a kalendářové komponenty frameworku jQuery dle (Leathem, 2014). Vzhledem k omezeným možnostem CDK komponenta nepodporuje validaci.

5.10 Zhodnocení frameworku

Framework RichFaces poskytuje komplexní sadu komponent pro potřeby tvorby podnikových webových aplikací. Ve srovnání s konkurenčními projekty, zejména s frameworkem PrimeFaces, se množina komponent jeví poněkud omezená. Chybí např. funkcionalita pro vytváření grafů a graficky zajímavějších prezentací dat. Zcela však postačuje pro vytváření běžných stránek a umožňuje využívat mnohé moderní prvky vývoje aplikací určených pro webový prohlížeč, jako např. částečné překreslování stránky, dynamické generování obsahu či propracovaná validace uživatelského vstupu.

Framework jednoznačně prokazuje vzestupnou tendenci, co se nabízené funkcionality a snadnosti jeho užívání týče. Ve srovnání se starší verzí 3.x se výrazně zlepšily možnosti využití komponent a jejich flexibilita. V tomto ohledu se framework rychle přibližuje vůdci na trhu, jakým jsou bezesporu úspěšnější PrimeFaces. Komponentová základna doznává výrazně pomalejších změn, oproti starším verzím byla přidána v zásadě pouze podpora aplikací určených pro mobilní zařízení.

Zvládnutí práce s frameworkem je i díky kvalitní dokumentaci a aktivitě uživatelsko-vývojářské komunity poměrně hladký proces s výraznou akcelerací. V rámci této činnosti vývojář získává velmi cennou obecnou průpravu pro využití frameworků, které jsou založeny na standardu JSF.

Zapojení RichFaces do projektu tak bude mnohem výhodnější pro tým, který pracoval s obdobným konkurenčním produktem (PrimeFaces, IceFaces) a naopak, vývojář pracující s RichFaces se bez větších obtíží zapojí do projektu, který některý z konkurenčních nástrojů využívá. Zkušenosti v této oblasti jsou tedy snadno přenositelné, což představuje značný přínos pro vývojáře, nicméně pro framework samotný to znamená obrovskou konkurenci ze strany ostatních podobných nástrojů a možné oslabení uživatelské základny.

Za silnou stránku frameworku lze považovat také skutečnost, že je schopen bezproblémově fungovat a poskytovat veškerou svou funkcionalitu bez ohledu na strukturu projektu. Pro uplatnění celého potenciálu frameworku tak není spoléhat na přítomnost nadřazeného nástroje v aplikaci, jako např. Spring. Podobná spolupráce ovšem není bez výhod, zejména co se týče funkcionality typu dependency injection a řízení databázové vrstvy.

Vzhledem k neutuchající aktivitě na straně autorů frameworku a stálé popularitě mezi vývojáři lze framework doporučit i pro projekty, které přikládají velkou důležitost zpětné kompatibilitě a kontinuálnímu pokroku. Komponentová základna je vhodná pro malé i velké projekty, které nicméně nejsou zaměřeny na přímé získávání zákazníků (např. e-shopy) vzhledem k omezenosti grafické prezentace dat. Pro ostatní aplikace se jedná o výbornou volbu s mnoha pozitivními dopady.

6 Závěr

Tato práce zamýšlela poskytnout celkem tři základní výstupy, které byly popsány a rozpracovány ve čtyřech obsahových kapitolách. Těmito výstupy byly následující dílčí části:

1. Vícekriteriální analýza konkurenčního prostředí na trhu webových frameworků s kvantifikovatelnými výsledky
2. Analýza akademických přístupů k porovnání webových frameworků v domácím i zahraničním univerzitním prostředí
3. Referenční aplikace s komentovaným kódem

Vicekriteriální analýza byla zpracována s využitím staršího výzkumu na podobné téma – viz (Laakso and Niemi, 2008). Tento výzkum poskytl zejména základ množiny kritérií a jejich vyhodnocování. Mezi přínosy práce v této oblasti patří zejména aktualizace bodových ohodnocení jednotlivých kritérií dle současných verzí srovnávaných nástrojů a úprava sady kritérií v souladu s mírně odlišným zaměřením této práce. Oproti původnímu výzkumu byla také upravena množina nástrojů, které byly ke srovnání použity s ohledem na zájmovou oblast práce.

S ohledem na výsledky zpracování analýzy lze bezpečně tvrdit, že všechny zkoumané nástroje s postupem času dosahují v jednotlivých kategoriích lepších hodnocení, tj. na trhu existuje nesporný vzestupný trend v kvalitě a přístupnosti. Zároveň není bez zajímavosti, že konečné bodové hodnocení srovnávaných produktů se nepohybuje v příliš širokém intervalu, přičemž na horních příčkách se umístily oba nástroje založené na standardu JSF.

Na výsledky výše uvedené analýzy lze velmi snadno navázat a rozšířit, popř. upravit její platnost. Nabízí se například zahrnutí většího počtu podobně zaměřených nástrojů, tematická úprava sady kritérií, návrh jemnější stupnice hodnocení či přehodnocení vah, které jsou jednotlivým kritériím přisuzovány.

Do rešerše akademických prací na příbuzná témata bylo zahrnuto celkem sedm prací, přičemž 5 pochází z českých univerzit a dvě jsou zahraničního původu. Texty se shodují na významu kvalitní dostupné dokumentace a minimálních požadavků na nastavení srovnávaných nástrojů.

Přístupy k dané problematice se v mnoha ohledech liší, přestože závěry se spíše shodují. Domácí práce se zaměřují spíše na technické kvality a architekturu zkoumaných produktů, zatímco ty zahraniční doplňují pohled o projektové řízení a nákladovou analýzu. V této části lze za hlavní přínos práce považovat kritické zhodnocení jednotlivých prací, porovnání jejich přístupů a závěrů a zdůraznění jejich předností. I zde se dá velmi snadno navázat zahrnutím většího množství prací i se vzdáleněji příbuznými tématy.

Hlavním původním přínosem této práce je však bezesporu referenční aplikace s komentářem, vytvořená v rámci praktické části textu. Aplikace slouží k názornému předvedení prá-

ce s frameworkem. Zaměřuje se zejména na typické formy užití nástroje k vytváření prezentační vrstvy webové aplikaci, jimiž jsou definice a modifikace vzhledu aplikace, validaci uživatelského vstupu a internacionalizaci.

Jednotlivé kroky při vývoji referenční aplikace jsou okomentovány s důrazem na identifikaci omezení a možností frameworku. Kromě formy jistého návodu k použití tak lze tuto část použít jako kritické a prakticky zaměřené zhodnocení frameworku vzhledem ke způsobu práce s ním i ve vztahu k ostatním konkurenčním produktům.

Závěrem lze tedy říci, že práce poskytuje komplexní zhodnocení frameworku z mnoha různých pohledů, a to po stránce teoretické i praktické. Porovnává framework s ostatními produkty podobného zaměření a analyzuje výsledky, hodnotí práci s frameworkem v praktickém nasazení a poskytuje doporučení pro vyšší efektivitu zapojení tohoto produktu do vývoje aplikace.

Závěry této práce pak lze rozšířit o další pole působnosti, která byla v tomto případě cíleně vynechána. Hovořit lze např. o vývoji aplikací pro mobilní zařízení, nasazení aplikací v jiné oblasti, než jsou podnikové systémy, popř. chování aplikací při stresovém zatížení.

Text práce mohou využít jak softwaroví vývojáři, kteří s frameworkem pracují či o tom uvažují, tak odborníci na úrovni softwarových architektů popř. projektových vedoucích při rozhodování o tom, který nástroj pro tvorbu prezentační vrstvy své webové aplikací zvolí.

Terminologický slovník

Termín	Význam (zdroj)
AJAX	Asynchronous Java and XML. Technologie pro částečné překreslování webových stránek. (Deitel, 2009)
Anotace	Způsob definice metadat v jazyce Java v podobě @Anotace. (Zarnett et al., 2010)
Archetyp	Šablona pro vygenerování skeletonu aplikace. (Apache Software Foundation, 2013c)
Classpath	Výchozí cesta aplikace. (autor)
CSS	Cascading Style Sheets. Jazyk pro vytváření stylopisů. (Deitel, 2009)
Dependency Injection	Mechanismus pro inicializaci aplikace včetně všech závislostí mezi jednotlivými komponentami. (Johnson et al., 2013)
HTML	HyperText Markup Language. Značkovací jazyk pro tvorbu webových stránek. (Deitel, 2009)
HTTP	Hypertext Transfer Protocol. Základní protokol internetové komunikace. (Deitel, 2009)
Internacionalizace	Proces návrhu a vývoje aplikace tak, aby zobrazované hodnoty byly odděleny od samotného kódu a bylo možné snadno přepnout mezi různými jazyky. (autor)
J2EE	Soubor nástrojů pro vývoj podnikových aplikací. (Oracle, 2013)
JavaScript	Skriptovací jazyk pro definici chování webové stránky v prohlížeči uživatele. (Deitel, 2009)
Jazyková mutace	Verze aplikace v konkrétním jazyce. Viz lokalizace. (autor)
JEE	Viz J2EE.
Jmenný prostor	Kategorie pro komponenty, v jejímž rámci musí být jméno komponenty unikátní. (Deitel, 2009)
JNDI	Java Naming and Directory Interface. Technologie pro katalogizaci komponent aplikace v distribuovaném prostředí. (Deitel, 2009)
JSF	JavaServer Faces. J2EE framework na bázi MVC pro tvorbu webových aplikací. (Deitel, 2009)
JSP	JavaServer Pages. J2EE framework na bázi MVC pro tvorbu webových aplikací, předchůdce JSF. (Deitel, 2009)
JSR-303	Standard pro validaci tříd typu POJO. (Bean Validation Expert Group, 2009)

Termín	Význam [zdroj]
Lokalizace	Proces přizpůsobení obsahu aplikace konkrétnímu jazykově-kulturnímu prostředí nebo verze aplikace v konkrétním jazyce. (Deitel, 2009)
Managed Bean	Třída spravovaná JSF frameworkem. (Deitel, 2009)
Metadata	Data o datech. Informace o struktuře a vlastnostech dat. (autor)
MVC	Model-View-Controller. Návrhový vzor používaný při vývoji webových aplikací. (Veit and Herrmann, 2003)
Open-source	Forma vývoje a distribuce softwaru. V závislosti na typu licence je možné open-source software používat zdarma i ke komerčním účelům. (Deitel, 2009)
POJO	Plain Old Java Object. Třída s bezparametrickým konstruktorem a přiřazovacími a čtecími metodami pro všechny datové atributy. (Fowler, 2003)
Pozicování	Proces rozložení jednotlivých částí stránky, obvykle s užitím jazyka CSS. (autor)
Property file	Textový soubor s příponou .properties na bázi klíč – hodnota, který slouží k uložení jednotlivých překladů textu aplikace. (autor)
RAD	Rapid Application Development. Metoda vývoje aplikací s důrazem na rychlost a snadnost celého procesu. (Howard, 2002)
RIA	Rich Internet Application. Webová aplikace s uživatelským prostředím a funkcionalitou, která poskytuje podobný komfort a možnosti, jako aplikace běžící mimo prohlížeč. (Bozzon et al., 2006b)
Rozsah platnosti	Angl. scope. Definice vymezení časového úseku, po který jsou data platná. (autor)
Session	Relace/konverzace mezi uživatelem a aplikací, která zahrnuje více HTTP požadavků. (Oracle, 2013)
Singleton	Třída, která v rámci aplikace má pouze jedinou přesně definovanou instanci. (Oracle, 2013)
Skeleton	Vygenerovaná kostra aplikace obsahující validní adresářovou strukturu a prázdné konfigurační soubory. (Oracle, 2013)
Skinování	Proces vytváření zaměnitelných vzhledů aplikace. (autor)
Stylopis	Definice vzhledu webových stránek s pomocí jazyka CSS. (autor)
Stylování	Proces definice grafické podoby stránky, obvykle s užitím jazyka CSS. (autor)
Validace	Kontrola faktické a strukturální správnosti uživatelského vstupu. (autor)
Termín	**Význam [zdroj]**

Validátor	Komponenta pro validaci. (autor)
WYSIWYG editor	Editor založený na vizuální prezentaci výsledného textu bez potřeby psát kód, ten je generován automaticky na pozadí. (JBoss Foundation, 2012a)
XHTML	eXtensible HyperText Markup Language. Značkovací jazyk pro tvorbu webových stránek. (Deitel, 2009)
XUL	Značkovací jazyk pro tvorbu webových rozhraní, vyvíjí Mozilla. (Quiroz et al., 2007)
Zástupný symbol	Angl. placeholder. Prvek označující místo, kam má být frameworkem doplněn obsah. (autor)
Závislost	Angl. dependency. Stav, kdy je jedna komponenta nutná pro fungování druhé. (Oracle, 2013)

Seznam literatury

Agarwal, R., Prasad, J., Tanniru, M., Lynch, J., 2000. Risks of rapid application development. Commun. ACM 43. doi:10.1145/352515.352516

Anderson, T., 2007. Why rich internet applications matter. IT Week 13.

Apache Software Foundation, 2013a. Tapestry - Introduction [WWW Document]. Tapestry 5. URL http://tapestry.apache.org/introduction.html (accessed 3.19.13).

Apache Software Foundation, 2013b. Apache Struts - Introduction [WWW Document]. Apache Struts. URL http://struts.apache.org/#Welcome (accessed 3.19.13).

Apache Software Foundation, 2013c. Apache Maven Project [WWW Document]. Introduction to Archetypes. URL (accessed 3.20.14).

Bean Validation Expert Group, 2009. JSR 303: Bean Validation.

Bozzon, A., Comai, S., Fraternali, P., Carughi, G.T., 2006a. Conceptual modeling and code generation for rich internet applications, in: Proceedings of the 6th International Conference on Web Engineering, ICWE '06. ACM, New York, NY, USA, pp. 353–360. doi:10.1145/1145581.1145649

Bozzon, A., Comai, S., Fraternali, P., Carughi, G.T., 2006b. Capturing RIA concepts in a web modeling language, in: Proceedings of the 15th International Conference on World Wide Web, WWW '06. ACM, New York, NY, USA, pp. 907–908. doi:10.1145/1135777.1135938

Brown, S., Dalton, S., Li, S., Jepp, D., Raible, M., Johnson, D., 2005. Pro JSP 2. Apress.

Changpil, L., 2012. An Evaluation Model for Application Development Frameworks for Web Applications. Ohio State University.

Deitel, P.J., 2009. Java for programmers, Deitel developer series. Prentice Hall, Upper Saddle River, N.J.

DevRates, 2013. DevRates [WWW Document]. DevRates. URL http://devrates.com/project/list?query=%5Bweb+framework%5D+%5Bjava%5D (accessed 3.2.13).

Donald, K., Vervaet, E., Grelle, J., 2013. Spring Web Flow Reference Guide.

Fowler, M., 2003. POJO [WWW Document]. URL http://www.martinfowler.com/bliki/POJO.html (accessed 3.15.14).

Howard, A., 2002. Rapid Application Development: rough and dirty or value-for-money engineering? Commun. ACM 45, 27–29. doi:10.1145/570907.570925

JBoss, 2013. Developer Guide [WWW Document]. Community Documentation. URL http://docs.jboss.org/richfaces/latest_4_3_X/Developer_Guide/en-US/html/ (accessed 6.4.13).

JBoss Foundation, 2012a. Richfaces Showcase [WWW Document]. RichFaces Showcase.

URL http://showcase.richfaces.org/richfaces/component-sample.jsf?demo=dataGrid&skin=blueSky (accessed 6.4.13).

JBoss Foundation, 2012b. RichFaces Bootstrap [WWW Document]. RichFaces Bootstrap. URL http://bootstrap-richfaces.rhcloud.com/ (accessed 6.20.13).

Johnson, R., Hoeller, J., Donald, K., 2013. Introduction to Spring Framework [WWW Document]. Spring Framework Reference Documentation. URL http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/overview.html (accessed 6.20.13).

jQuery Foundation, 2013. How jQuery works [WWW Document]. jQuery. URL http://learn.jquery.com/about-jquery/how-jquery-works/ (accessed 6.4.13).

Kay, R., 2009. Rich Internet Applications. Computerworld 43, 34.

Koščejev, A., 2009. Moderní Java frameworky pro tvorbu webových aplikací a jejích porovnání. Vysoká škola ekonomická.

Laakso, T., Niemi, J., 2008. An evaluation of AJAX-enabled java-based web application frameworks, in: Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia, MoMM '08. ACM, New York, NY, USA, pp. 431–437. doi:10.1145/1497185.1497278

Leathem, B., 2011. RichFaces 4.1.0.Final Release Announcement [WWW Document]. Experiences of a Developer. URL http://www.bleathem.ca/blog/2011/12/richfaces-410final-release-announcement.html (accessed 6.7.13).

Leathem, B., 2012a. Roadmap and Planning for RichFaces 4.3 and beyond [WWW Document]. RichFaces Development. URL https://community.jboss.org/thread/175973 (accessed 6.9.13).

Leathem, B., 2012b. RichFaces Bootstrap [WWW Document]. Experiences of a Developer. URL http://www.bleathem.ca/blog/2012/05/richfaces-bootstrap.html (accessed 6.20.13).

Leathem, B., 2014. Java EE - Experiences of a Developer [WWW Document]. URL http://www.bleathem.ca/blog/tags/CDK/ (accessed 3.22.14).

Optimus Prime, 2012. Final Words on PrimeFaces Fork [WWW Document]. Prime Faces Blog. URL http://blog.primefaces.org/?p=1750 (accessed 3.2.13).

Oracle, 2013. The Java EE 6 Tutorial.

Podolka, L., 2007. Porovnání web frameworků v Javě. Vysoká škola ekonomická.

Pösinger, M., 2007. Porovnání Java frameworků pro vytváření webových stránek. Vysoká škola ekonomická.

Prime Technologies, 2012. Why PrimeFaces? [WWW Document]. PrimeFaces. URL http://primefaces.org/whyprimefaces.html (accessed 3.3.13).

Quiroz, J.C., Louis, S.J., Dascalu, S.M., 2007. Interactive Evolution of XUL User Interfaces, in: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Compu-

tation, GECCO ’07. ACM, New York, NY, USA, pp. 2151–2158. doi:10.1145/1276958.1277373

Riehle, D., 2000. Framework Design - A Role Modelling Approach (Dizertační práce). Swiss Federal Institute of Technology, Zurich.

Seltzer, L., 2010. Rich Internet Applications: Richer platforms, Richer targets. eWeek 27, 23,26–27.

Tijms, A., 2012. What’s new in JSF 2.2? J-Development.

Vagner, V., 2009. Webové frameworky implementované v jazyce Java (Diplomová práce). České vysoké učení technické, Praha.

Válek, O., 2008. Webové aplikační frameworky pro rychlý vývoj aplikací. Vysoká škola ekonomická.

Van der Tol, M., 2012. JavaServer Faces (Diplomová práce). University of Amsterdam.

Veit, M., Herrmann, S., 2003. Model-view-controller and object teams: a perfect match of paradigms, in: Proceedings of the 2nd International Conference on Aspect-Oriented Software Development, AOSD ’03. ACM, New York, NY, USA, pp. 140–149. doi:10.1145/643603.643618

Zarnett, J., Tripunitara, M., Lam, P., 2010. Role-based Access Control (RBAC) in Java via Proxy Objects Using Annotations, in: Proceedings of the 15th ACM Symposium on Access Control Models and Technologies, SACMAT ’10. ACM, New York, NY, USA, pp. 79–88. doi:10.1145/1809842.1809858

Seznam obrázků a tabulek

Základní text

Seznam obrázků

Seznam tabulek

Seznam ukázek kódu

Přílohy

Seznam obrázků

Příloha A: Dokumentace k referenční aplikaci

Tato část práce slouží jako uživatelský manuál k práci s aplikací. Základní podoba aplikace je dostupná i bez přihlášení a uživatel má přístup do následujících sekcí

- Základní informace – sekce obsahuje stránky týkající se registrace, informací o hře a ostatních registrovaných hráčů
- O hře – sekce obsahuje stránky se seznamy aktuálně dostupných postav a příběhových linií
- Organizace – sekce obsahuje kontaktní informace na organizátora, v tomto případě se jedná o kontakt na autora této práce

Výchozí lokalizace aplikace závisí na nastavení prohlížeče uživatele, pokud není nastaven preferovaný jazyk, použije se angličtina. Do češtiny lze zobrazení snadno přepnout pomocí ikonky české vlajky v pravém horním rohu stránky pod záhlavím.

Po registraci a přihlášení je uživateli k dispozici sekce Můj účet. V této sekci uživatel může prohlížet a upravovat informace, které zadal při registraci, a vytvářet vlastní příběhové linie a role.

Obr. 8: *Hlavní stránka*

A.1 Registrace a přihlášení

Registrační formulář je k dispozici po kliknutí na položku menu "Základní informace" a pak na položku "Registrace". Registrační formulář po uživateli požaduje vyplnit následující údaje:

- Jméno (délka mezi 2 a 100 znaků)
- Přezdívku (délka mezi 3 a 25 znaků)
- Heslo (délka mezi 8 a 500 znaků)
- Datum narození
- Telefonní číslo (délka mezi 9 a 13 znaků)
- E-mail (kontroluje se správný formát e-mailu a musí se jednat o unikátní hodnotu)
- Ulice (minimální délka 3 znaky)
- Město (minimální délka 3 znaky)
- PSČ (předepsaná délka přesně 5 znaků)

Datum narození neprochází validací, protože se jedná o vlastní vytvořenou komponentu a CDK v této verzi ještě validaci nepodporuje.

Po úspěšné registraci bude uživatel přihlášen a přesměrován na úvodní stránku s detaily jeho účtu. Odhlásit se lze stisknutím tlačítka Odhlásit se v levé horní části stránky. Pokud je uživatel odhlášen, bude po stisknutím tlačítka Přihlásit se ve stejné části stránky přesměrován na přihlašovací formulář.

A.2 Postavy

Postavy jsou připravovány organizátorem hry, a proto je nelze vytvářet v rámci rozhraní aplikace. Na stránce Postavy v sekci O hře si přihlášený hráč může postavu vybrat, pokud ještě žádnou nemá. Vybírat lze pouze postavy, které jsou ještě volné, tj. žádný hráč si je ještě nezapsal. Každá postava má organizátorem přidělený počet bodů, který může utratit za účast v různých příběhových liniích.

Obr. 9: *Výběr postav*

A.3 Příběhové linie

Novou příběhovou linii může vytvořit pouze přihlášený uživatel a to na stránce Moje příběhy stisknutím tlačítka Přidat příběh. Zobrazí se modální panel, kde uživatel musí vyplnit jméno příběhu a jeho popis.

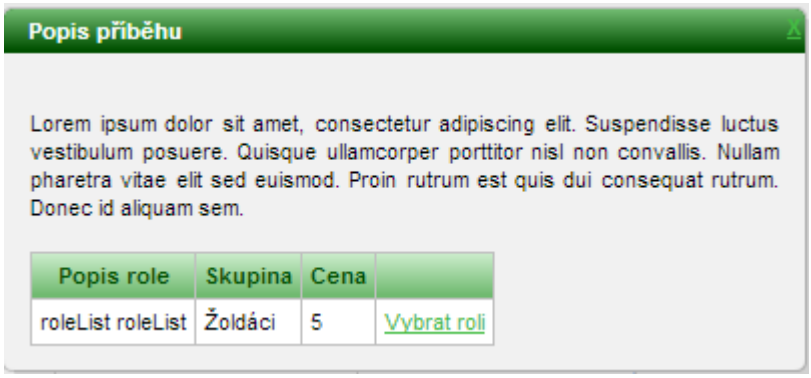
Obr. 10: *Příběhové linie*

Jakmile má uživatel vytvořený příběh, může do něj přidávat role stisknutím tlačítka Přidat roli. Zobrazí se modální panel, kde uživatel musí vyplnit skupinu, do které postava, která si tuto roli bude chtít zapsat, musí patřit, cenu role a popis role.

Obr. 11: *Tvorba příběhu*

Obr. 12: *Tvorba role*

Příběhy a role se zobrazují na stránkách Příběhové linii v sekci O hře. Nepřihlášený uživatel si kliknutím na odkaz Zobrazit detaily může prohlížet pouze popis příběhu. Přihlášený uživatel navíc uvidí také seznam rolí, které k příběhu patří, a může si některou roli pro svou postavu vybrat.



Obr. 13: *Výběr role*

Výběr postavy je povolen pouze tehdy, pokud jsou splněny všechny následující podmínky

- Uživatel je přihlášen a má již vybranou postavu
- Postava má dostatek příběhových bodů, aby zaplatila cenu role
- Postava nemá v příběhu vybranou jinou roli

Informace o dostupných příbězích a rolích se automaticky aktualizují.

A.4 Správa uživatelského účtu

Do sekce Můj účet mají přístup pouze registrovaní uživatelé. Stránka poskytuje přehled o informacích, které uživatel zadal během registrace, umožňuje mu změnit heslo a ostatní registrační údaje a nahrát profilovou fotku.

Obr. 14: *Administrace uživatelského účtu*