VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Současné trendy vývoje uživatelského rozhraní Java aplikací

Actual Trends UI Development for Java Application

2018 Bc. Tomáš Přívozník

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Tomáš Přívozník**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Současné trendy vývoje uživatelského rozhraní Java aplikací
Actual Trends UI Development for Java Application

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je prozkoumat současný stav tvorby uživatelského rozhraní pro Java platformu, při zvážení přístupu tlustý i tenký klient, a nelezení přístupu, který patří mezi v současnosti nejpoužívanější. Dalším cílem je pak vytvořit nástroj, který zjednoduší a centralizuje vytváření uživatelského rozhraní dle nalezeného přístupu.

Práce bude zejména obsahovat:
1. Popis použité metody pro výzkum – například průzkum literatury, dotazníky, analýza dostupných zdrojových kódů OSS (Open Source Software) či jejich kombinace. Součásti popisu bude také zdůvodnění použití.
2. Popis provedeného výzkumu dané oblasti. Pokud ke sběru dat v rámci výzkumu budou použity původní nástroje, které budou vybranou část automatizovat, tak budou v práci obsaženy spolu s jejich popisem.
3. Vyhodnocení výsledků získaných v rámci výzkumu.
4. Model návrhu nástroje pro tvorbu uživatelského rozhraní a implementaci vybraných částí s ohledem na časovou náročnost vlastní implementace.
5. Skripty pro automatizované testování či popis testovacích scénářů.

Závěrečné poznámky:
Všechny zdrojové kódy či skripty budou uloženy v repozitáři aplikace, která je dostupná na adrese http://git.cs.vsb.cz. Vytvořené aplikace bude možno sestavit pomocí systému maven či gradle.

Seznam doporučené odborné literatury:

[1] WEAVER, James, Weiqi GAO, Stephen CHIN, Dean IVERSON a Johan VOS, 2014. Pro JavaFX 8: A Definitive Guide to Building Desktop, Mobile, and Embedded Java Clients. 1st ed. edition. Berkeley, CA: Apress. ISBN 978-1-4302-6574-0.
[2] TIDWELL, Jenifer, 2011. Designing Interfaces: Patterns for Effective Interaction Design. 2 edition. Beijing: O'Reilly Media. ISBN 978-1-4493-7970-4.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jan Kožusznik, Ph.D.**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 27. dubna 2018 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 27. dubna 2018 .....

Rád bych poděkoval panu Ing. Janu Kožusznikovi, Ph.D. za profesionální vedení této práce a zpětnou vazbu k vypracovanému tématu.

Abstrakt

Tato diplomová práce se zabývá porovnáním současných trendů vývoje UI pro Java aplikace.

Cílem teoretické části práce je nalézt přístupy vývoje UI, které patří v současnosti k nejpoužívanějším. Praktická část práce navazuje na výsledky teoretické části. Cílem praktické části práce je vytvoření nástroje ulehčující vytvoření projektu s vybraným přístupem pro tvorbu UI.

Úvodní část je věnována metodám výzkumu. Během výzkumu byly použity metody Google trendy, prohledání literatury, Stack Overflow a ZeroTurnaround statistiky a analýza veřejného repositáře GitHub. Po vyhodnocení výsledků výzkumu byly vybrány čtyři přístupy pro tvorbu UI. Všechny vybrané přístupy mají popsanou funkcionalitu.

Závěr práce popisuje nástroj vytvořený v praktické části práce. Konkrétně jsou popsány implementační rozhodnutí a jednotlivé části nástroje.

Klíčová slova: Java, JavaScript, trendy, Google, Stack Overflow, ZeroTurnaround, GitHub, výzkum, Angular, React, JSF, JSP, Struts

Abstract

This diploma thesis deals with the comparison of current trends UI development for Java applications.

The aim of the theoretical part is to find the approaches of UI developments, which currently belongs the most used. The practical part of the thesis builds on the results of the theoretical parts. The aim of the practical part of the thesis is to create a tool facilitating the creation of a project with a selected approach for UI creation.

The introductory part is devoted to research methods. Methods used during the research were Google Trends, Literature research, Stack Overflow, and ZeroTurnaround statistics and analysis of the public repository GitHub. Four approaches for UI creation have been selected after evaluating the research results. All selected approaches have their functionality described.

The conclusion of the thesis describes the tool created in the practical part of the thesis. In particular, implementing decisions and individual parts of the tool are described.

Key Words: Java, JavaScript, trends, Google, Stack Overflow, ZeroTurnaround, GitHub, research, Angular, React, JSF, JSP, Struts

Obsah

Seznam použitých zkratek a symbolů

AJAX	–	Asynchronous JavaScript And XML
API	–	Application Programming Interface
CDI	–	Contexts and Dependency Injection
CRUD	–	Create Read Update Delete
CSS	–	Cascading Style Sheets
DOM	–	Document Object Model
EE	–	Enterprise Edition
EL	–	Expression Language
GWT	–	Google Web Toolkit
HTML	–	Hypertext Markup Language
IDE	–	Integrated Development Environment
IT	–	Information technology
JRE	–	Java Runtime Environment
JS	–	JavaScript
JSF	–	JavaServer Faces
JSNI	–	JavaScript Native Interface
JSON	–	JavaScript Object Notation
JSP	–	JavaServer Pages
JSR	–	Java Specification Requests
JSTL	–	JavaServer Pages Standard Tag Library
MVVM	–	Model-View-ViewModel
MVC	–	Model–View–Controller
OGNL	–	Object-Graph Navigation Language
OOP	–	Object-Oriented Programming
POJO	–	Plain Old Java Object
REST	–	REpresentational State Transfer
RIA	–	Rich Internet Application
RPC	–	Remote Procedure Call
SPA	–	Single-Page Applications
SQL	–	Structured Query Language
UI	–	User Interface
URL	–	Uniform Resource Locator
XML	–	Extensible Markup Language

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Současný vývoj Java aplikací nabízí celou řadu přístupů pro tvorbu UI. Obecně lze přístupy rozdělit do dvou kategorií podle typu aplikace. V jednom případě aplikace reprezentuje tenkého klienta a v druhém případě tlustého klienta.

Cílem práce je prozkoumat dnešní možnosti ve tvorbě UI pro Java aplikace. Vzhledem k velkému množství stylů tvorby UI se práce věnuje pouze některým stylům. Jednotlivé styly jsou analyzovány a detailněji popsány. Druhá část práce se snaží nalézt a vytvořit přístup pro zjednodušení vytvoření vybraného stylu tvorby UI.

Úvod práce se zabývá rozhodnutím jestli budou sledovány přístupy tvorby UI pro tenkého nebo tlustého klienta. Poté byl proveden výzkum zjišťující jaké jsou aktuálně používané přístupy pro tvorbu UI. Pro dosažení co největší přesnosti výzkumu, bylo nutné najít rozličné zdroje poskytující validní informace. Zdroje poskytovaly informace přímo nebo nepřímo. Přímým zdrojem informací je myšlena už vytvořená statistika v kontextu Java aplikací. Nepřímý zdroj informací představují surová data, která musí být použita k vytvoření statistik.

Po analýze výsledků výzkumu byly vybrány nejpoužívanější přístupy a ty byly detailněji popsány. Jednotlivé přístupy jsou popsány pro lepší pochopení funkcionality, popis dále odhaluje jak se jednotlivé přístupy od sebe liší.

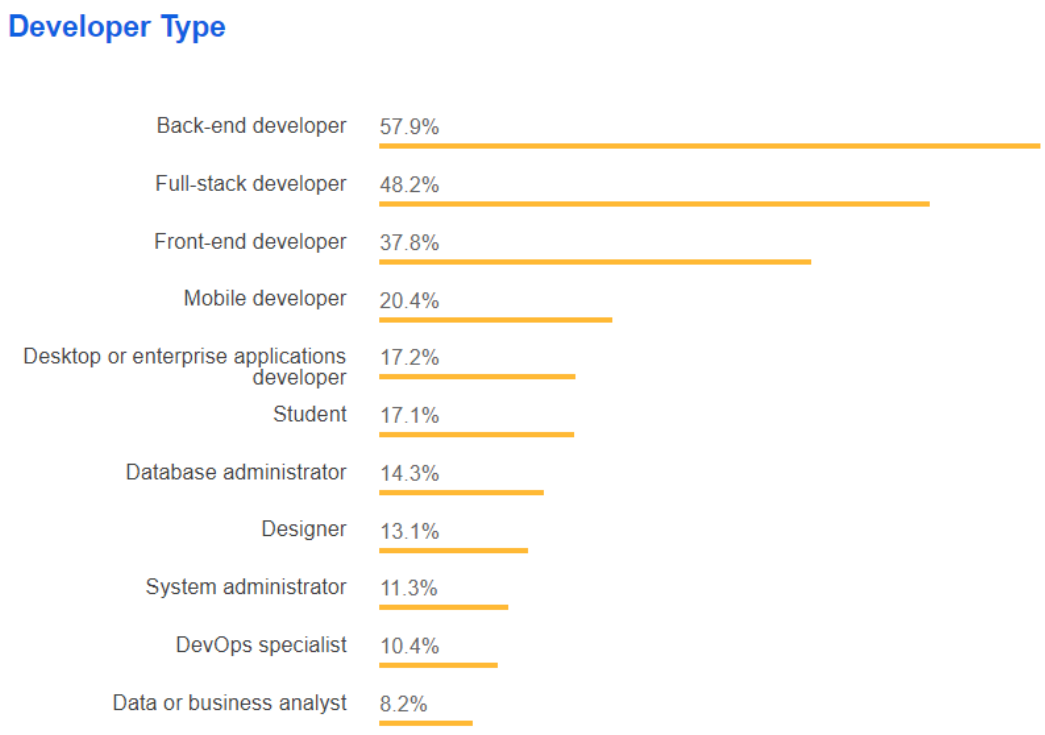
Konec práce je věnován popisu nástroje zjednodušující tvorbu UI vybraného přístupu. Součástí popisu nástroje jsou implementačním rozhodnutím a model návrhu vytvořeného přístupu.

2 Popis výzkumu

Prvotní představa výzkumu spoléhala na veřejný repositář *GitHub* jako hlavní na zdroj informací. Získané výsledky by byly ověřeny vůči jiným zdrojům pro dosažení co největší přesnosti výzkumu.

Skutečný výzkum se mírně lišil od autorovy představy. Bylo použito více metod výzkumu než bylo zamýšleno.

Prvotní představa je důležitá pro nastínění prvotního směru vývoje, u této práce tomu nebylo jinak. Součástí představy bylo nalezení prvotního přístupu jak k výzkumu přistupovat. To bylo rozhodnutí, zda budou zkoumány trendy v oblasti tenký nebo tlustý klient. Ovšem v dnešní době, kdy k vývoji nejen podnikových aplikací se používá velké množství rámců se rozdíl mezi tenkým a tlustým klientem ztrácí. Tato skutečnost vyplývá ze situace, kdy klient nabaluje stále více funkcí a kódu čím se zvyšují nároky na výpočetní výkon klienta. Tato práce je založena na autorově rozhodnutí, tudíž jsou sledovány trendy podnikových aplikací. Rozhodnutí bylo učiněno na základně průzkumu popularity jednotlivých přístupů. Aktuální statistika na adrese `https://insights.stackoverflow.com/survey/2018/` ukazuje přibližné zastoupení vývojářů jednotlivých komunit. Ve statistice jasně vítězí komunita vývojářů podnikových aplikací. Na obrázku je ukázána pouze nejpodstatnější část statistiky. Přibližně 92 000 vývojářů odpovědělo na tuto otázku.



Obrázek 1: Statistika zastoupení vývojářů v jednotlivých komunitách [15]

2.1 Metody výzkumu

Níže jsou popsány tři hlavní zdroje použité během výzkumu. Všechny tyto zdroje byly využity a zkombinovány pro dosažení co nejpřesnějšího výsledku při porovnávání aktuálních trendů.

2.1.1 Veřejný repositář

Nejprve se jevila jako nejlepší a nejspolehlivější metoda výzkumu prozkoumání veřejného repositáře zdrojových kódů konkrétně (*GitHub*). Jeden z hlavních principů repositáře je uložení projektu na jednom centralizovaném místě. Z toho pohledu veřejný repositář obsahuje velké množství projektů. Pokud vezmeme projekty, které mají vytvořené UI, každý z těchto projektů tedy představuje nějaký přístup pro tvorbu UI. Aktuální trendy by bylo možné získat z pouhého počtu přístupů jednotlivých projektů.

Veřejný repositář dále nabízí celou řadu prostředků jak zjistit popularitu určitého programovacího jazyka. U každého repositáře lze vidět jeho popularitu prostřednictvím počtu hvězd ukazující, kolika lidem se projekt líbí a počet větvení označující kolik lidí si projekt stáhlo a dále upravovalo. Díky těmto kritériím lze použít pokročilé vyhledávání pro zobrazení projektů vytvořených v daném jazyce s danou popularitou. Bohužel vyhledávač vracel nepřesné výsledky při hledání projektů z více různých programovacích jazyků. Proto bylo nutné zkombinovat všechny dostupné metody a výzkum provést nad výsledky z několika zdrojů. Obtíže s pokročilým vyhledáváním v repositáři nevyřazují tuto metodu ze seznamu validních zdrojů. Obecná oblíbenost jednotlivých programovacích jazyků je stále relevantní.

2.1.2 Literatura

Prozkoumání literatury vedlo napříč knihami publikovanými od roku 2002 až po rok 2018. Tato literatura se zaobírala tématy, jako jsou moderní architektura webových aplikací, návrh rozhraní, vývoji Java EE a další. Veškerá literatura se obecně zabývala architekturou klient-server u podnikových aplikací a jak je vyvíjeno u takových to aplikací uživatelské rozhraní. Z analýzy literatury lze zjistit jaké technologie a techniky byly v době publikování aktuální, a co je považováno za standard.

2.1.3 Statistiky

Validní informace pro tento výzkum poskytují statistiky vyhledávání od společnosti *Google*, statistky popularity webových rámců pro Javu od společnosti *ZeroTurnaround* a nebo průzkum využívání webu Stack Overflow za aktuální rok. Všechny tyto statistiky poskytují aktuální výsledky.

2.2 Počátky výzkumu

Odborné články mohou na úvod poskytnout základní informace, jaké jsou trendy pro vývoj UI podnikových aplikací. V takovémto nepříliš specifikovaném rozsahu, výsledky vyhledávání jednoznačně ovládly JavaScript rámce použité na klientské části aplikace. Podnikové aplikace s daným rámcem na klientské straně mohou být velice populární, ovšem není zaručeno, že serverová část aplikace je napsána v jazyce Java. Pakliže chceme mít výsledek výzkumu relevantní pro tuto práci, je nutné ověřit tyto výsledky v odborné literatuře. Pro dosažení co největší přesnosti tohoto výzkumu, byly vyhledány rámce čistě pro Java aplikace. Tento výsledek už obsahoval relevantní informace o aktuálních trendech. Rozsah zkoumaných rámců byl vymezen na tři nejpoužívanější Java rámce a jeden používaný JavaScript rámec.

V prvotních krocích zkoumání dominovaly rámce Angular, React, Meteor a Vue.js. Jak bylo zmíněno dříve, tyto výsledky musely být ověřeny s jinými zdroji. Literatura zabývající se touto problematikou potvrdila vývoj podnikových aplikací, vytvořených propojením technologie Angular a jazyka Java. Takovéto aplikace byly předmětem i mnoha dalších odborných článků. Další, ovšem méně časté bylo propojení rámce React a Java EE aplikace.

Neustále upravování a přidávání metod výzkumu vedlo k upravení základních parametrů vyhledávání. Nové vyhledávání se zaobíralo trendy mezi čistě Java webovými rámci. Takto směřovaný výzkum odhalil již vypracované statistiky na dané téma. Uvedené zdroje poskytly nové možnosti pro získání aktuálních trendů při vývoji UI.

3 Porovnání výsledků metod výzkumu

V této kapitole jsou přiblíženy jednotlivé metody výzkumu, a jaké výsledky tyto metody poskytly.

3.1 Literatura

Knihy byly vybírány na základě prvotního rozhodnutí zaměření se na podnikové aplikace. Z toho důvodu je většina knih zaměřených aplikace Java EE nebo Spring. Obecně veškeré knihy zabývající tvorbou UI u podnikových aplikací jsou považovány za vhodný zdroj informací.

Po prozkoumání literatury byl vytvořený přehled ukazující jaké technologie se objevily v jednotlivých knihách. Technologie Angular a JSP se v literatuře objevovaly nejvíce. JSP se objevovalo v literatuře, která byla publikována okolo roku 2012. Knihy zabývající se moderní architekturou podnikových aplikací obsahovaly ukázky propojení Java EE s rámcem Angular. Níže je tabulka ukazující výsledky prozkoumání pouze relevantní literatury. Symbol *X* označuje, která literatura se zabývala danou technologií.

Tabulka 1: Prohledání literatury.

Technologie	1	2	3	4	5	6	7	8	9	10
Angular	-	X	X	-	X	-	-	X	-	X
React	-	-	-	-	-	-	-	-	-	X
JSF	-	-	-	X	X	X	X	-	-	-
JSP	X	X	-	X	-	X	X	-	-	-
JSTL	-	-	-	-	-	X	-	-	-	-
Thymeleaf	-	X	-	-	-	-	-	-	X	-
JQuery UI	-	-	X	-	-	-	-	-	-	-
Sencha Touch	-	X	-	-	-	-	-	-	-	-
ExtJs	-	X	-	-	-	-	-	-	-	-

Prozkoumaná literatura:

1. HTML5 Enterprise Application Development (2013) [26]
2. Enterprise Application Development with Ext JS and Spring (2013) [27]
3. Enterprise Web Development (2014) [28]
4. Web 2.0 and Social Networking for the Enterprise (2009) [29]
5. Digital Java EE 7 Web Application Development (2015) [30]
6. Java EE 7 Developer Handbook (2013) [31]
7. Java Web services (2002) [32]

8. Java EE 8 and Angular: A practical guide to building modern single-page applications with Angular and Java EE (2018) [33]

9. Learning Spring Boot 2.0 - Second Edition: Simplify the development of lightning fast applications based on microservices and reactive programming (2017) [34]

10. Front-End Reactive Architectures: Explore the Future of the Front-End using Reactive JavaScript Frameworks and Libraries (2018) [35]

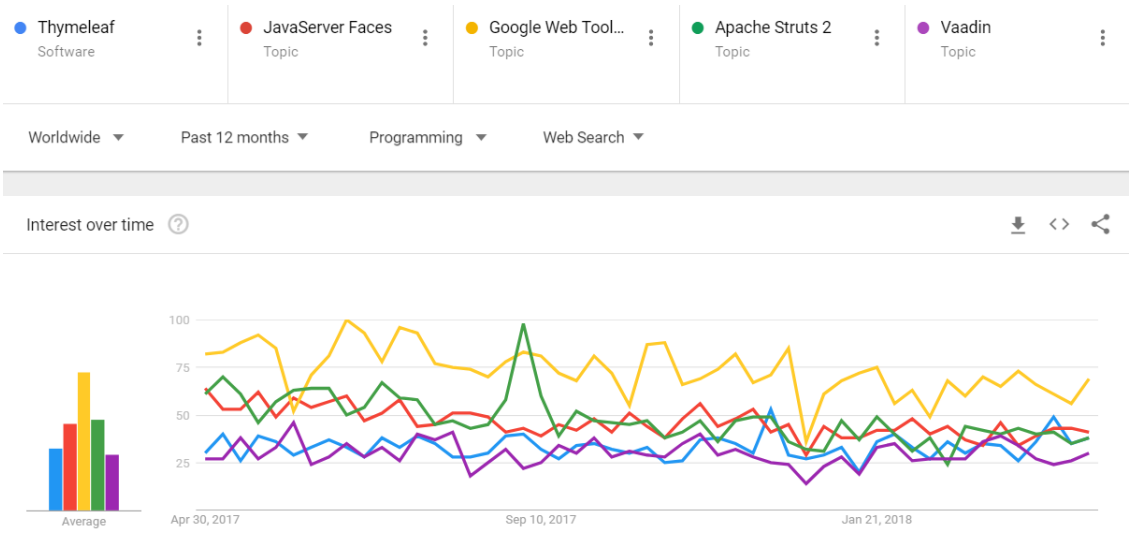
3.2 Google trends

Společnost Google nabízí informace o vyhledávání pro všechny uživatele webové služby `https://trends.google.com/trends/`. Úvodní stránka informuje o aktuálně nejvyhledávanějších tématech. Uživatel má možnost specifikovat, pro která témata má být statistika zobrazena. Parametry jako období nebo kategorie upravují výsledná data.

Vyhledávač Google jakožto jeden z nejpoužívanějších vyhledávačů může být nápomocen při určování aktuálních trendů pro tvorbu UI. V kritériích vyhledávání lze nastavit období za poslední rok a vymezit výsledek na nejaktuálnější data. Pro ještě detailnější přesnost je možné vybrat kategorii k jaké se má hledané téma vztahovat.

Statistika porovnává popularitu vyhledávání pro Java webové rámce. Tyto technologie byly vybrány na základě prohledání literatury a jiných statistik. Technologie JSP byla z porovnání vynechána kvůli velkému procentu zkreslení, i když byla vybrána kategorie *Programování*.

GWT byl za uvedené období nejvyhledávanějším rámcem v porovnání s ostatními rámci. Můžeme si všimnout, že druhé místo v oblíbenosti vyhledávání obsadil rámec Apache Struts 2, který nepatří mezi nové rámce. To nemění nic na tom, že může být stále používaný. Ostatní srovnávané rámce mají podle této statistiky vcelku podobnou popularitu.

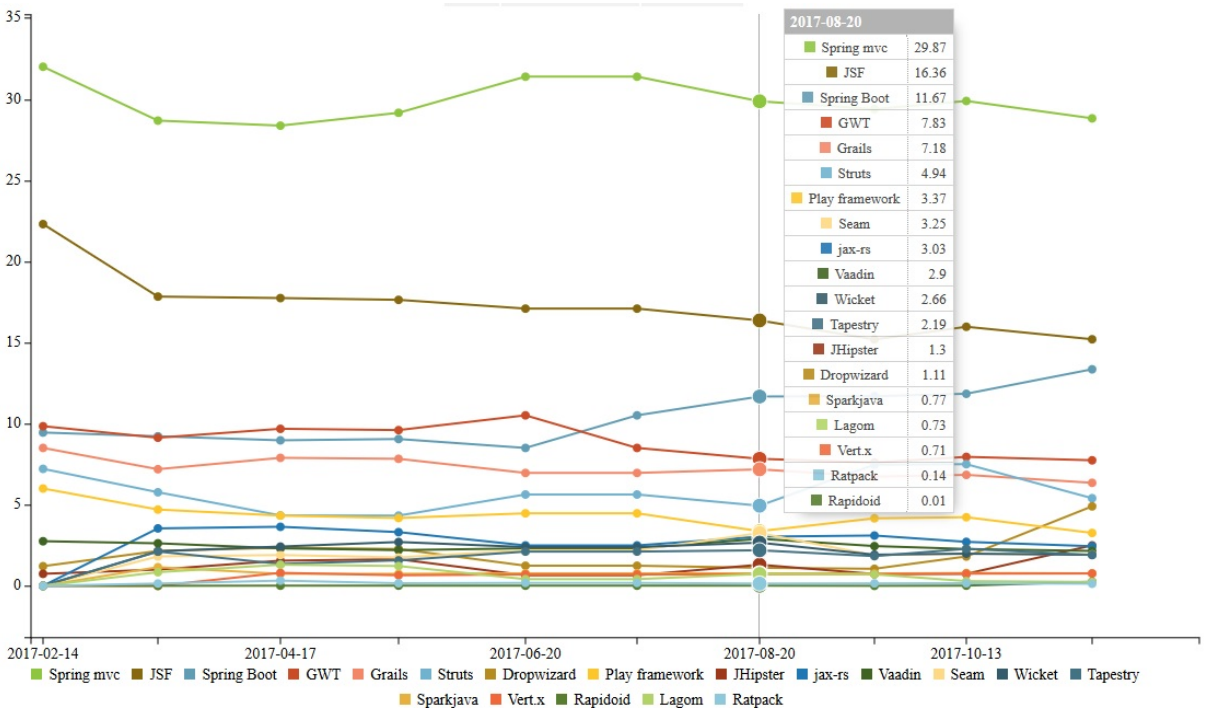


Obrázek 2: Trend vyhledávání rámců v roce 2018 [24]

3.3 ZeroTurnaround statistiky

Statistiku vytváří profesionální společnost pohybující se na poli jazyka Java více než 10 let. Porovnání Java webových rámců kdy každému rámci je přiřazen index označující jeho popularitu, je dostupné na adrese `https://zeroturnaround.com/webframeworksindex/`. Výsledky jsou periodicky aktualizovány. Relevantnosti popularity rámců se snaží dosáhnout kombinací několika různých zdrojů informací. Jsou zkombinovány zdroje jako GitHub, Google, Stack Overflow a také LinkedIn. Sociální síť LinkedIn nepřímo ukazuje popularitu různých technologií. Počet pracovních pozic pro danou technologii je také relevantní měřítko popularity.

Statistika zahrnuje veškeré rámce, které jsou aktuálně nejpoužívanější a jsou založeny na Javě. Pro tento výzkum jsou relevantní rámce, které se nějakým způsobem uplatňují při tvorbě vzhledu podnikové aplikace. To nás přivádí na rámec JSF, který se umístil jako první z rámců vyhovující těmto kritériím. Pod ním se umístil GWT. I tato statistika nasvědčuje poměrně velkému zájmu rámce Apache Struts 2, který se umístil pod GWT.



Obrázek 3: Statista popularity webových rámců pro jazyk Java [23]

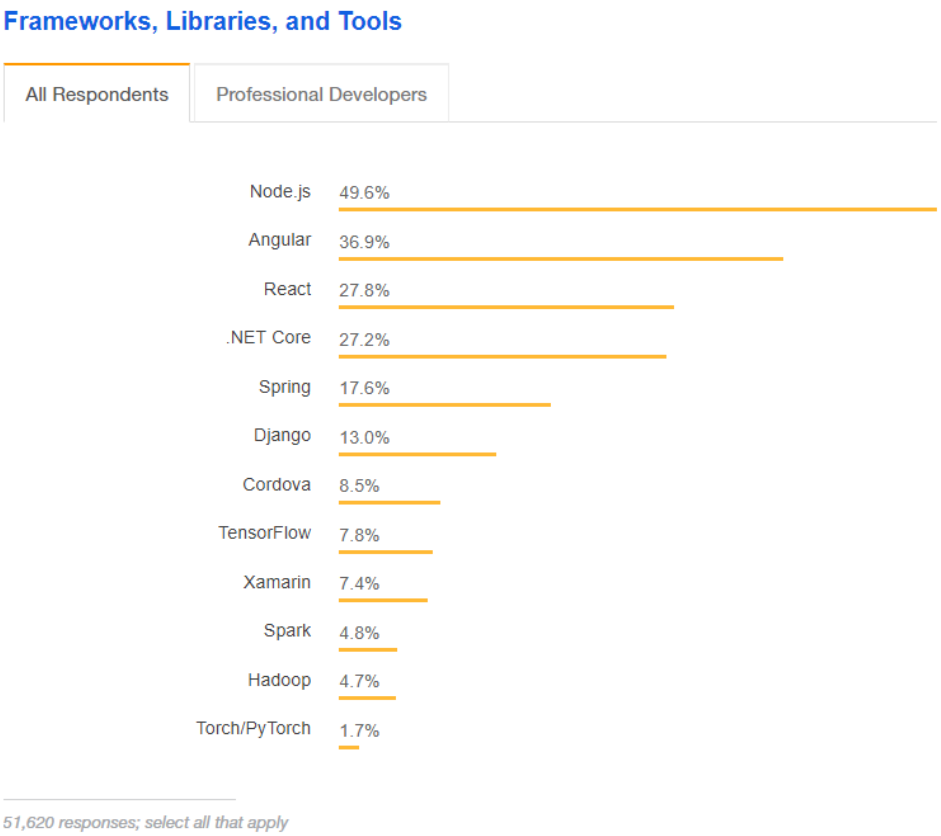
3.4 Stack Overflow statistiky

Stránka Stack Overflow se stala útočištěm pro mnoho vývojářů. Díky tomu se na jednom místě koncentruje skoro nepřeberné množství informací okolo programování. Tyto informace poskytují zázemí pro další různé statistiky. Například zaměstnanci z řad Stack Overflow vytvářejí další oficiální přehledy, zabývající se srovnáním oblíbenosti štítků jednotlivých technologií nebo analýzou používaných technologií.

3.4.1 Průzkum

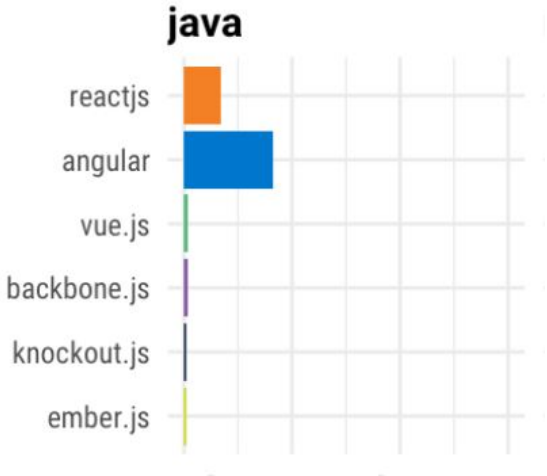
Jednou za rok je vydán obsáhlý průzkum zkoumající aktuální aspekty vývoje softwaru. Výsledky jsou k dispozici na `https://insights.stackoverflow.com/survey/2018`. Tento průzkum je vytvořen na základě informací poskytnutých komunitou uživatelů. Průzkum zahrnuje srovnání používaných technologií, rozmezí platu jednotlivých pracovních pozic v IT, zkušenosti vývojářů, geografické zobrazení místa pobytu vývojáře a mnoho dalších témat. Z toho průzkumu bylo pro tuto práci relevantní procentuální srovnání nejpoužívanějších technologií.

Určitá sekce průzkumu se zabývá používanými technologiemi, konkrétně jaké jsou aktuálně nejpoužívanější. Mezi prvními se opět umístily JS rámce. Statistika napovídá většímu použití rámce Anguar. Popularity byly určeny na základě odpovědí více než padesáti tisíc vývojářů.



Obrázek 4: Popularita rámců v roce 2018 [16]

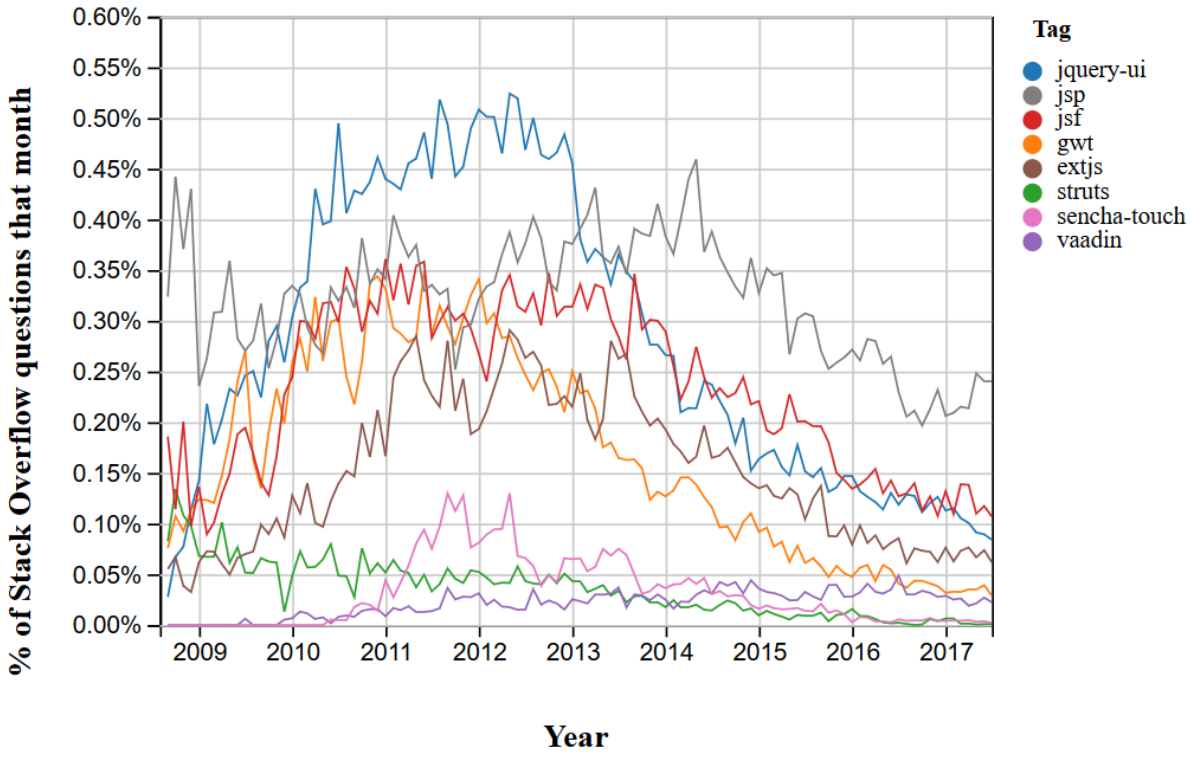
Další průzkum z roku 2018 se zabývá čistě JavaScript rámci. V průzkumu jsou porovnány štítky, podobně jako tomu je v této práci. Další část průzkumu ovšem poskytuje nové informace o popularitě propojení JavaScript rámců s jazykem Java. Zdroj informací je k nalezení zde `https://stackoverflow.blog/2018/01/11/brutal-lifecycle-javascript-frameworks/`.



Obrázek 5: Kolaborace JS rámců a jazyka Java [17]

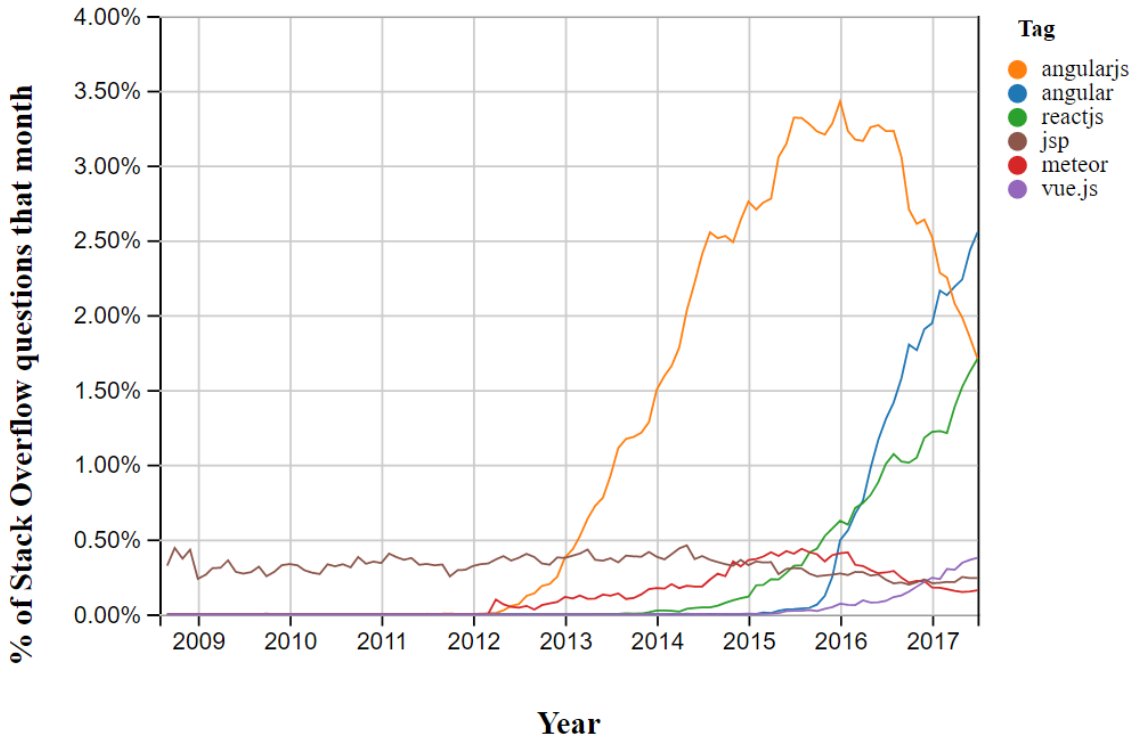
3.4.2 Srovnání štítků

Porovnání štítků zobrazuje vývoj popularity štítku v závislosti na čase. Tyto štítky jsou myšleny pouze v kontextu webu Stack Overflow. Nejpopulárnějším štítkem se jednoznačně stal štítek *jsp*, v závěsu za ním se nachází štítek *jsf*.



Obrázek 6: Porovnání oblíbenosti štítků z roku 2018 [18]

Na obrázku níže je porovnání oblíbenosti štítků pro JS rámce. Porovnání štítků je rozděleno na dva grafy z důvodu velkého rozmezí popularity štítků. Štítky JS rámců zkreslují průběh popularity méně populárních štítků, jak můžeme vidět u štítku *jsp* na obrázku níže. Štítek pro rámec *jsp* byl vybrán jako zástupce technologií z prvního porovnání, neboť se umístil na prvním místě oblíbenosti.



Obrázek 7: Druhé porovnání oblíbenosti štítků z roku 2018 [19]

3.5 GitHub

Kromě toho, že je možné zjistit popularitu repositáře, dalším zdrojem informací jsou témata. Témata se mohou vztahovat k programovacím jazykům, obecným pojmům nebo konkrétním produktům.

3.5.1 Téma

Při vyhledání tématu *Angular* je možné vidět aktivní repositáře zabývající se tímto tématem. Toto téma se vztahuje k 10 514 repositářům. Tento počet není úplně přesný, pakliže vezmeme v úvahu jednotlivé verze produktu. Každá verze má vlastní téma, popřípadě repositář.

Téma *React* má na sebe navázáno přibližně tři krát více repositářů. Přesný počet v době výzkumu byl 34 450.

3.5.2 Repositář

Každý repositář má svůj počet hvězd, ten označuje počet vývojářů, kteří si oblíbili daný repositář. Při prozkoumání repositáře pro samotný produkt Angular, je možné vidět, že produkt získal popularitu 35 231 hvězd.

Obdobně jako počet repositářů i oblíbenost produktu React je větší. Tento projekt má 93 839 hvězd.

Rámec GWT si oblíbilo 994 vývojářů. Těsně za ním je rámec Struts 2 s 827 vývojáři, kteří si ho oblíbili.

3.6 Vyhodnocení

Literatura se z velké části zabývala vývojem podnikových aplikací. Nejčastější technologie byly Angular a JSP. Obě technologie se objevily v pěti z deseti zkoumaných knih. Těsně za nimi se umístil rámec JSF, který se vyskytl ve čtyřech knihách z deseti.

ZeroTurnaround statistiky poskytly solidní výsledky v oblasti Java webových rámců. Mezi nejpoužívanějšími rámci se umístilo JSF, GWT a Struts. Všechny tyto rámce byly porovnány v kontextu popularity vyhledávaní přes vyhledávač Google. Žádný z porovnávaných rámců nebyl značně vyhledávanější než jiný.

Průzkumy populární stránky pro vývojáře ukázaly absolutní dominanci JS rámců nad výše uvedenými technologiemi. V těchto statistikách byl veden boj hlavně mezi rámcem Angular a React. Vítězem v ohledu rozšířenosti se stal rámec Angular. Při porovnávání štítků byly porovnány všechny výše uvedené technologie. Stejně jako v průzkumu i tady byly nejpopulárnější JS rámce. Při vynechání JS rámců, štítky JSP a JSF byly nejpopulárnější. V grafu lze pozorovat, že popularita rámce Apache Struts 2 se blíží k nule. To je ovšem v rozporu s popularitou vyhledávání na Google vyhledávači. To ukazuje, že Struts se již neřadí k používaným rámců při vývoji nových aplikací. To ovšem neznamená, že není použit u velkého množství již vyvinutých aplikací, které jsou aktivní.

Veřejný repositář GitHub obsahuje repositáře pro projekty Angular a React. Rámec React byl v této komunitě mnohem populárnější. Větší počet vývojářů označilo repositář jako oblíbený.

3.6.1 Vybrané rámce

Na základě výzkumu byly vybraný rámce Angular, GWT, Struts a JSF. Každý z rámců bude popsán ve své vlastní kapitole. Tyto kapitoly si nekladou za cíl detailně popsat veškerou funkcionalitu jednotlivých rámců, nýbrž mají pouze uvést základní principy funkcionality.

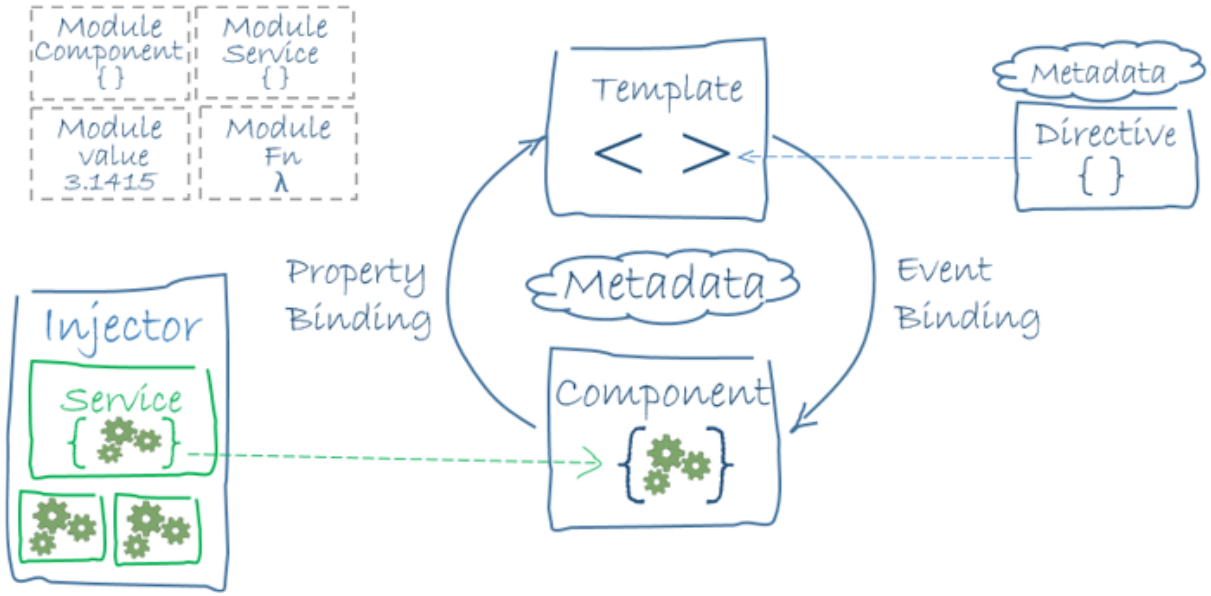
4 Angular

Angular je open-source webový aplikační rámec, který vývojářům nabízí spoustu možností prostřednictvím stabilní kódové základny, živého společenství a bohatého ekosystému [1]. Je podporován a licencován společností Google. Pro vyvíjení Angular aplikací je potřeba mít nainstalovaný Node.js a výchozí balíčkový manažer (npm). Angular je napsán v jazyce TypeScript. Angular prošel vývojem několika verzí od AngularJS až po aktuální Angular 5. V nejbližší době vyjde nová verze Angular 6. K lepšímu pochopení tohoto rámce budou vysvětleny pojmy jako SPA a jiné.

4.1 Funkčnost

Základní stavební prvky aplikace Angle jsou *NgModuly*, které poskytují kontext pro kompilaci komponent. NgModuly sbírají související kód do funkčních sad, Angular aplikace je definována sadou NgModulů. Aplikace má vždy alespoň kořenový modul, který umožňuje bootstrapping, obvykle má mnoho dalších modulů [2].

Pojem bootstrapping obvykle označuje samo-startovací proces neboli posloupnost kroků. Tento proces pro svůj běh nepotřebuje vstupy od uživatele.



Obrázek 8: Model propojení základních komponent [2]

Komponenta s šablonou tvoří pohled, který představuje část HTML stránky. Ten to princip souvisí s pojmem SPA, který je vysvětlen níže. Angular poskytuje směrovací službu pro definování navigace mezi jednotlivými pohledy.

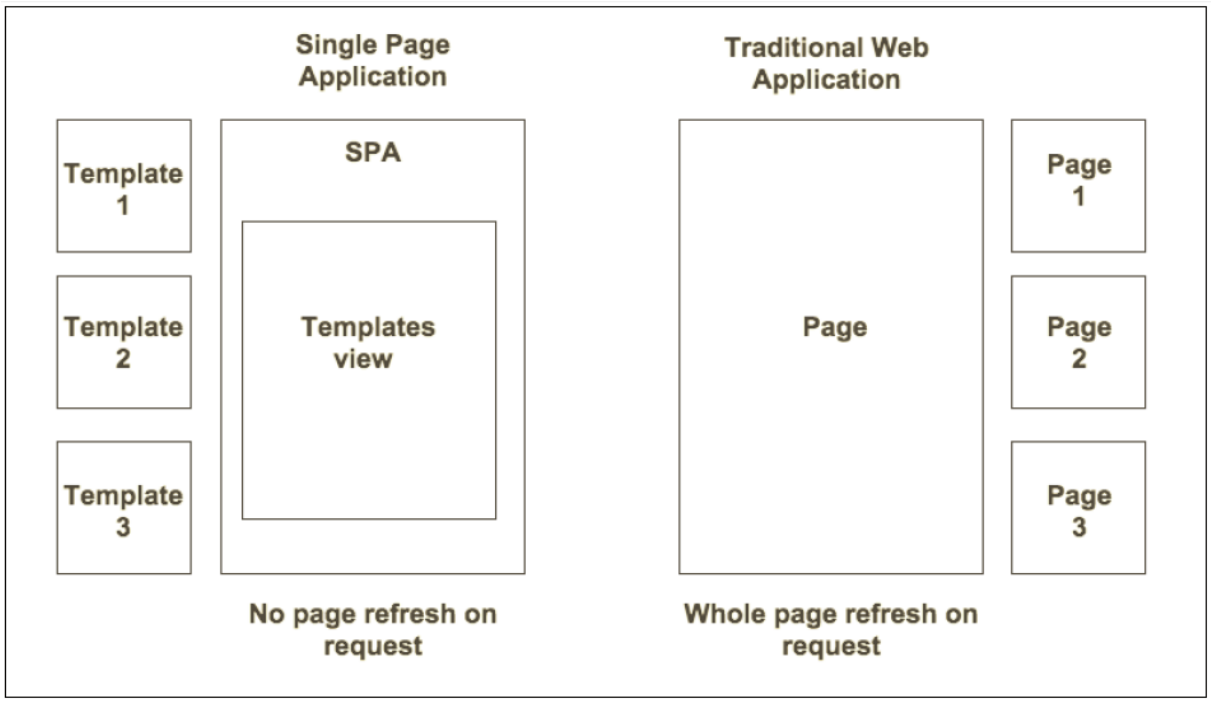
Třída se stává komponentou po přidání dekorátoru, ten přidává metadata třídě. Šablona používá události komponenty a komponenta poskytuje vlastnosti šabloně.

4.1.1 SPA

Tento druh aplikace uvádí jiný způsob chováni aplikace. Princip je zachycen na obrázku níže. Jeden z důvodů proč vyvíjet tento druh aplikace je samotný princip SPA kdy HTML stránka je poslána klientovi pouze jednou. U klasických aplikací, které mají více stránek je HTML stránka sestavena na straně serveru a je odeslána klientovi. V případě SPA je HTML stránka sestavena u klienta a server posílá pouze data. Nemusí se aktualizovat celá stránka, ale jenom některé její části. Uživateli se zdá, že aplikace běží rychleji, protože server neposílá celé HTML stránky [3].

Jak funguje přesměrování na jinou stránku, když aplikace má pouze jednu HTML stránku? Namísto konverze dat na HTML na serveru a následné odeslání po síti, v SPA je nyní přesunut tento proces konverze ze serveru na klienta [4].

Tento typ aplikace má i několik nevýhod. V počátcích kdy vyhledávače nebyly připraveny na SPA aplikace, bylo skoro nemožné vyhledat takovéto aplikace. Vyhledávače měly problémy se správným indexováním SPA aplikací. Další překážku představovalo použití tlačítka zpět v prohlížeči, to mohlo způsobit ztrátu dat. Ani historie webového prohlížeče nefungovala úplně spolehlivě.



Obrázek 9: Rozdíl mezi tradiční a SPA aplikací [3]

4.2 Vlastnosti rámce

Níže jsou uvedeny klíčové vlastnosti rámce a aspekty funkcionality.

4.2.1 TypeScript

TypeScript je volně šiřitelný jazyk, za jehož vytvořením stojí *Microsoft*. Jeho cílem je posunout vývoj v jazyce JavaScript na vyšší úroveň. TypeScript je staticky napsaný kompilovaný jazyk, který generuje kód JavaScript, který lze použít pro různé platformy [14]. Vývojáři pohybující se v OOP jazycích uvítají určité vlastnosti, které má TypeScript navíc oproti jeho předchůdci.

Novinkou jsou třídy, moduly nebo například generiky ulehčující vývoj velkých aplikací. Další užitečnou vlastností je myšlenka veřejných a privátních členů. Pokud se objekt pokusí přistoupit na privátního člena jiného objektu, kompilátor takovýto kód označí za chybný. Takovýto princip napomáhá k zapouzdření určité funkcionality.

4.2.2 Angular CLI

Jedna z prerekvizit pro vývoj Angular aplikací je npm. Skrze tento manažer se instaluje Angular CLI, který slouží k založení projektu, vygenerování komponent aplikace, ovládání životního cyklu aplikace, testování nebo zabalení aplikace k nasazení. Toto rozhraní pro příkazovou řádku výrazně ulehčuje vývojářům práci s aplikací. V souboru package.json je seznam nainstalovaných balíčků. Npm balíčky lze stáhnout přes npm klienta nebo yarn klienta. Po založení projektu, soubor package.json obsahuje závislosti důležité pro běh aplikace.

4.2.3 Komponenty

První verze AngularJS fungovala na principu MVVM. Kdežto od Angular 2 a výše jsou základními prvky aplikace komponenty. Každá aplikace má kořenový element, který má pod komponenty a tak to jde dále. Z toho vyplývá, že komponenty tvoří stromovou strukturu.

Komponenty jsou v podstatě třídy komunikující s HTML souborem komponenty, který je zobrazen v prohlížeči. Komponenty používají služby, které poskytují funkcionalitu nepřímo spojenou s pohledem. Třída komponenty je označena dekorátorem *@Component*.

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-heroes',
  templateUrl: './heroes.component.html',
  styleUrls: ['./heroes.component.css']
})
export class HeroesComponent implements OnInit {
  heroes: Hero[];

  constructor() { }

  ngOnInit() {
```

```
    this.getHeroes();
  }

  getHeroes(): void {
   // get data
  }
}
```

Výpis 1: Ukázka Angular komponenty

4.2.4 Směrnice

Směrnice jsou v podstatě to samé co komponenty. Komponenty rozšiřují směrnice, důvod proč mají vlastní dekorátor je jejich významnost v celém rámci. Směrnice jsou třídy s dekorátorem *@Directive*. Když rámec vykresluje stránku, transformuje DOM podle instrukcí daných směrnic. Existují dva druhy směrnic: strukturální a atributové směrnice.

4.2.4.1 Strukturální směrnice Strukturální směrnice utvářejí rozvržení HTML stránky. Upravují strukturu DOM, obvykle přidáním, odstraněním nebo manipulací s prvky. Strukturální směrnice se vkládají přímo do HTML prvků stejným způsobem jako jiné směrnice.

Ukázka níže ukazuje podmínkovou směrnici, která očekává řídící hodnotu typu boolean. Nebo například jiná směrnice **ngFor* je iterativní a vytvoří seznam prvků na základě předaného pole. Angular má i další vestavěné směrnice.

```
<div *ngIf="hero" class="name">{{hero.name}}</div>
```

Výpis 2: Ukázka podmínkové směrnice s hvězdičkou

Lze si všimnout speciálního znaku * použitého ngif nebo ngfor. Tento znak je takzvaným syntaktickým cukrem. Jedná se o vytvořené ulehčení práce pro programátory. V tomto případě je ukázka výše přepsána na ukázku níže.

```
<ng-template [ngIf]="hero">
  <div class="name">{{hero.name}}</div>
</ng-template>
```

Výpis 3: Ukázka podmínkové směrnice bez hvězdičky

4.2.4.2 Atributové směrnice Atributové směrnice mění vzhled nebo chování prvku, komponenty nebo jiné směrnice. Směrnice *ngModel* slouží k obousměrnému vázání dat.

```
<input [(ngModel)]="hero.name">
```

Výpis 4: Ukázka atributové směrnice

4.2.5 Pipes

Trubky umožňují deklarovat transformace hodnot zobrazení v HTML šabloně. Třída s dekorátorem @Pipe definuje funkci, která transformuje vstupní hodnoty na výstupní hodnoty pro zobrazení v pohledu.

Angular poskytuje několik již nadefinovaných transformací, ale je možné vytvořit si vlastní transformaci. Syntaxe je stejná jako pro klasickou datovou vazbu mezi modelem a pohledem, ale navíc se přidá operátor a název trubky. Transformace může přijímat parametry ovlivňující její chování a tyto transformace lze libovolně řetězit.

```
<!-- shortTime format: output '9:43 AM' -->
<p>The time is {{today | date:'shortTime'}}</p>
```

Výpis 5: Ukázka transformace dat

4.2.6 Routing

Směrovače umožňují navigaci mezi pohledy. Aplikace má jedinou instanci služby pro směrování. Směrovač mapuje cesty vypadající jako URL adresy na komponenty. Jakmile se změní URL adresa prohlížeče, směrovač hledá odpovídající cestu, ze které může určit jaká komponenta má být zobrazena.

Pokud směrovač zjistí, že aktuální stav aplikace vyžaduje určitou funkcionalitu, která není načtena, směrovač může modul zodpovědný za tuto funkcionalitu dočíst mechanismem lazy load.

Vygenerovaná aplikace nemá nakonfigurované žádné směrovače. Konfigurace musí být explicitně vytvořena. Níže se nachází ukázka konfigurace směrovače. Atribut path označuje adresu pro komponentu. Speciální znak ** slouží pro výchozí zobrazení jakékoliv nespecifikované adresy. Pořadí tras hraje důležitou roli, směrovač totiž při porovnávání adres používá strategii první shoda vítězí. Proto je nejlepší dát specifické adresy na začátek a ty obecné na konec.

```
const appRoutes: Routes = [
  { path: 'crisis-center', component: CrisisListComponent },
  { path: 'hero/:id',      component: HeroDetailComponent },
  { path: '**', component: PageNotFoundComponent }
];

@NgModule({
  imports: [
    RouterModule.forRoot(appRoutes)
    // other imports here
  ],
  ...
})
```

```
export class AppModule { }
```

Výpis 6: Ukázka konfigurace směrovačů

4.2.7 Služby

Služby jsou implementovány jako klasické třídy s dekorátory označující jejich typ a poskytující informace jak je má aplikace používat. Služba se označuje dekorátorem @Injectable a tím umožní vkládat službu do komponent jako závislost. Služba typicky obsahuje data nebo funkcionalitu, která je sdílena mezi komponentami. HttpClient je mechanismus pro komunikaci se serverem přes HTTP protokol.

Samotné komponenty by neměly načítat nebo ukládat data. Jejich účel je prezentace data a delegování přístupu k službě, která data poskytne nebo provede jejich zpracování.

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs/Observable';
import { User } from './user';

@Injectable()
export class UserService {

  private userUrl = 'http://localhost:8080/users';

  constructor(private http: HttpClient) { }

  /** GET users from the server */
  getUsers (): Observable<User[]> {
    return this.http.get<User[]>(this.userUrl);
  }
}
```

Výpis 7: Ukázka Angular služby

4.2.8 Testování

Stejně jako AngularJS i pozdější verze jsou vyvinuty s velkým důrazem na testovatelnost. Nově založený projekt je možné hned otestovat. K testování používáme Angular CLI. Pro ulehčení testování jsou použity dva rámce: Karma a Jasmine. Jakmile je testování spuštěno

4.3 Výhody

Jedna z největších výhod toho rámce je Angular CLI, tento nástroj výrazně ulehčuje vývojářům práci. Založený projekt je bez dalšího zásahu vývojáře připraven ke spuštění. Jakékoliv změny v projektu jsou okamžitě aplikovány. Další důležitou vlastností je snadná testovatelnost. Celý rámec byl vytvořen tak aby šlo vše otestovat. Obousměrná vazba dat zajišťuje automatickou aktualizaci dat v modelu v závislosti na pohledu a naopak. Práce s POJO objekty přináší dobrou integraci s jinými technologiemi. Angular aplikace není závislá pouze na jedené technologii jako rámce JSF nebo Apache Struts 2.

5 JavaServer Faces

JavaServer Faces je rámec pro vývoj UI webových aplikací v Javě. Podobně jako většina jiných rámců, i JSF dbá na separaci prezentační a aplikační logiky. Tato technologie byla představena jako Java Specification Request (JSR) 127 v roce 2001 a finální verze specifikace (JSF 1.0) byla vydána v roce 2004 [5]. Jedním z klíčových cílů JSF bylo vyhnout se spoléhání na jednu konkrétní zobrazovací technologii. Takže JSF poskytuje modulově založené rozhraní umožňující vývojářům integraci s různými zobrazovacími technologiemi.

5.1 Funkčnost

Po kliknutí na tlačítko v JSF aplikaci, je odeslán požadavek z webového prohlížeče na server. JSF je zodpovědné za překládání tohoto požadavku do události, která může být zpracována logikou na serveru. Je dále zodpovědný za to, že každá grafická komponenta, která je definována na serveru, je správně zobrazena v prohlížeči.

JSF ve své architektuře využívá MVC styl, tím je schováno mnoho procesů běžících na pozadí. Vývojáři se tak mohou soustředit na vytváření komponent a jejich interakce.

Java webové aplikace komunikují přes HTTP protokol prostřednictvím Servlet API a typicky používají nějakou zobrazovací technologii, například JSP. Takovéto technologie slouží k definování uživatelských rozhraní, která jsou složena z komponenty, které komunikují s Java kódem. Architektura komponent využívá technologii JavaBean pro vystavení vlastností a událostí.

5.2 Vlastnosti rámce

Níže jsou uvedeny klíčové vlastnosti rámce a aspekty funkcionality.

5.2.1 Servlet

Java Servlet API poskytuje objektově orientovaný pohled, HTTP požadavky a odpovědi jsou zaobaleny jako objekty. Poskytuje vstupní a výstupní proudy pro přečtení uživatelovy odpovědi a zapsání dynamického obsahu. Požadavek je zpracováván pomocí servletu jinými slovy objektu, který zpracovává určitou sadu HTTP požadavků. Servlety běží uvnitř kontejneru, který je v podstatě Java aplikace, která provádí veškerou práci spojenou s během servletů.

Webové aplikace řeší bezstavovost pomocí takzvaných Session, tím se uživatel jeví pořad jako aktivní i když ve skutečnosti aktivní není.

5.2.2 Konfigurace s faces-config.xml

Stejně jako většina rámců má konfigurační soubor, JSF má faces-config.xml. Tento XML soubor umožňuje definovat pravidla pro navigaci, inicializaci JavaBeans, zaregistrování vlastních JSF komponent, validátory a další aspekty aplikace JSF.

```
<managed-bean>
    <managed-bean-name>sampleBean</managed-bean-name>
    <managed-bean-class>org.gwtbook.SampleBean</managed-bean-class>
    <managed-bean-scope>request</managed-bean-scope>
</managed-bean>
```

Výpis 8: Ukázka ManagedBean konfigurace

5.2.3 Konfigurace s web.xml

Všechny Java EE aplikace mají konfigurační soubor web.xml. U většiny moderních aplikací je tato konfigurace přesunuta do JavaBean. Rámec JSF vyžaduje specifikovat FacesServlet, což je obvykle hlavní servlet aplikace. Na tento servlet musí být mapovány požadavky.

5.2.4 Expression language

Java Expression Language (EL) je kompaktní a výkonný mechanismus, který umožňuje dynamickou komunikaci v aplikacích JSP a JSF (včetně vývoje rámce založené na JSF, jako jsou PrimeFaces, ICEfaces a RichFaces) [6].

EL v prezentační vrstvě slouží ke komunikaci s aplikační logikou. V podstatě se jedná o obousměrnou komunikaci, proto můžeme poskytnout uživateli data z aplikační logiky, ale zároveň je možné odeslat uživatelská data ke zpracování.

Primárně EL uvidíme v JSF stránce, ale lze jej použít například v faces-config.xml. Dále jsi uvedeme několik vybraných funkčností, které EL nabízí ke komunikaci mezi webovou stránkou a aplikací.

Obecně řečeno, EL může být použito k naplnění požadavků HTTP s uživatelskými daty, extrahovat a vystavovat data z HTTP odpovědí, aktualizovat HTML DOM a mnohem více.

5.2.4.1 EL okamžité a odložené vyhodnocení Okamžité vyhodnocení vrátí výsledek okamžitě po vykreslení stránky. Tento druh výrazů slouží pouze pro čtení. Zapisují se tímto způsobem:

```
$ {}
```

Odložené vyhodnocení může vrátit výsledek v různých fázích životního cyklu stránky. JSF může vyhodnotit výrazu v různých fázích životního cyklu. Tyto výrazy mohou být vlastnosti ManagedBean nebo metody a zapisují se takto:

```
# {}
```

5.2.4.2 Bean vlastnosti Na hodnotové výrazy je možné narážet velice často ve vztahu k objektům, jejich vlastnostem a atributům. Takové výrazy se používají k vyhodnocení výsledků nebo k nastavení vlastnosti Bean za běhu aplikace. Vlastnost nebo atribut objektu nemusí být jen primitivní datový typ, lze používat i kolekce nebo výčtový datový typ Enum. EL poskytuje implicitní objekty, přes které je možné získat informace vztahující se k aktuálnímu požadavku a prostředí.

Jak bývá u Bean zvykem všechny vlastnosti jsou privátní a přístup k nim je zprostředkován pouze pomocí veřejných metod. V ukázce webové stránky ovšem přistupujeme přes tečkovou notaci rovnou k vlastnosti od ModuleBean. Na pozadí jsou volány již zmíněné veřejné metody get a set.

```
@Named
@SessionScoped
public class ModuleBean implements Serializable {
    /**
     * Name of the module
     */
    private String moduleName;

    String getModuleName() {
        return moduleName;
    }

    void setModuleName(String moduleName) {
        this.moduleName = moduleName;
    }
}
```

Výpis 9: Ukázka ManagedBean

Ukázka webové stránky. Pro přístup k vlastnosti se používá tečková nebo závorková notace.

```
<h:inputText id="moduleName" value="#{moduleBean.moduleName}" />
<h:inputText id="secondModuleName" value="#{moduleBean['moduleName']}" />
```

Výpis 10: Ukázka HTML stránky

5.2.4.3 Kolekce a výrazy metod Ke kolekcím (polím, setům aj.) přistupujeme stejně jako ke klasické vlastnosti. Proto v ukázce rovnou použijeme volání metody pro vrácení kolekce. Pro vyhodnocení metody musíme použít vyhodnocení #. Metodě můžeme předat parametry. V atributech určených pro volání metod lze volat JavaScript funkce.

```
    ...
   /**
    * List of all connected modules
    */
   private List<Module> listOfModules;

   /**
    * Get all modules connected to the application
    *
    * @return List of modules
    */
   public List<Module> getConnectedModules() {
      listOfModules = moduleService.listAllRegisteredModules();
      return listOfModules;
   }
    ...
```

Výpis 11: Ukázka kolekce

HTML ukázka. Pro zobrazení kolekce je použita komponenta z knihovny PrimeFaces. Atribut value specifikuje jaká metoda má být použita (metoda musí být veřejná). Atribut var je kurzor pro získanou kolekci.

```
<p:dataGrid id="modulesPanelGrid" columns="1" layout="grid"
  value="#{moduleBean.getConnectedModules()}" var="module">
  <f:facet name="header">
    <p:panelGrid columns="2" layout="grid" styleClass="facetHeader
       pGridNoBorder">
     <h:outputText value="Dostupne moduly" styleClass="headerLeft facetText" />
      <p:commandButton value="pridat modul" styleClass="headerRight"
       update="addNewModuleForm" oncomplete="PF('addNewModuleDlg').show()" />
   </p:panelGrid>
  </f:facet>
  <p:panelGrid columns="1" layout="grid" styleClass="pGridNoBorder">
    <p:commandLink value="#{module.name}"
      action="#moduleBean.setSelectedModule(module.id)}" update="modulesForm" /
          >
  </p:panelGrid>
</p:dataGrid>
```

Výpis 12: Ukázka komponent PrimeFaces

5.2.4.4 Podmíněné výrazy Podmíněný výraz je zprostředkován pomocí ternárního operátoru. Můžeme jej využít například při rozhodování jaká CSS třída bude aplikována na daný element. CSS soubor

```
.red { color:#cc0000; }
.blue { color: #0000cc; }
```

Výpis 13: Ukázka CSS souboru

```
<h:outputText styleClass="#{playersBean.play == 'Left' ? 'red': 'blue'}"
  value="#{playersBean.play}"/>
```

Výpis 14: Ukázka HTML stránky s použitím ternárního operátoru

5.2.5 Managed Beany

Jsou znovu použitelné komponenty, u kterých se o vytváření instancí stará rámec. Použitím Managed Bean dosáhneme oddělení prezentace od aplikační logiky. JSF stránky se odkazují na vlastnosti Managed Bean a aplikační logika je uložena v implementaci třídy.

Třída je označena dvěma anotacemi. Anotace @ManagedBean je součást JSF rámce. Ovšem od verze 2.3 je tato anotace označena za zastaralou a neměla by se používat. Druhá anotace označuje dobu existence. Všechny ostatní anotace v balíku javax.faces.bean jsou také označeny jako zastaralé. Na místo zastaralých anotací bychom měli použít CDI anotace poskytnuté Java EE standardem.

Výpis několika existenčních anotací:

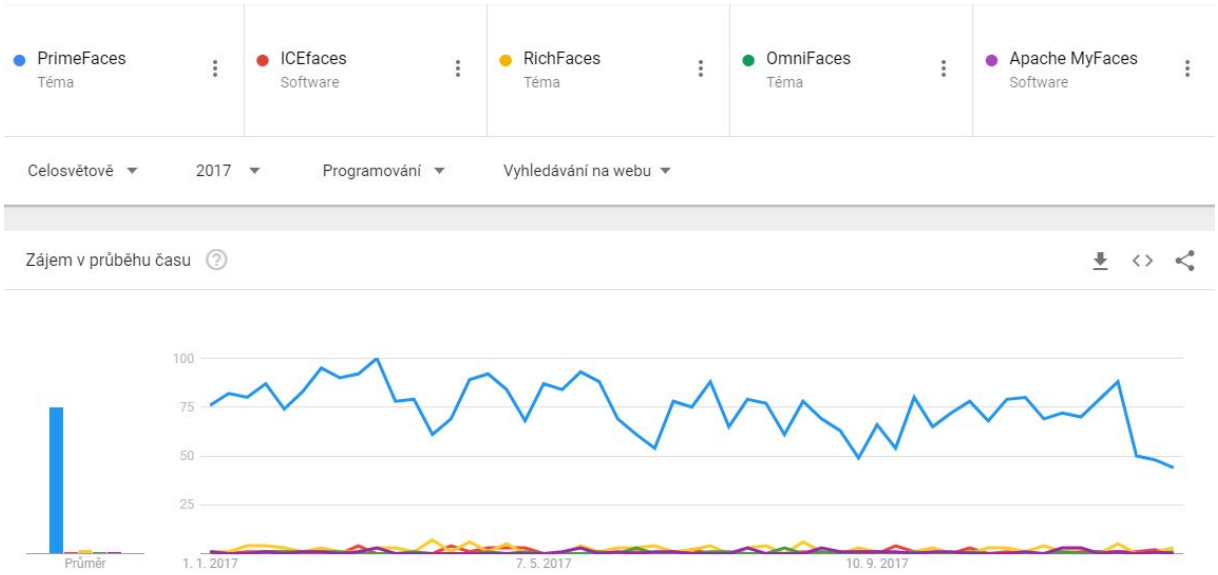
- @RequestScoped = Instance existuje tak dlouho, dokud probíhá HTTP žádost-odpověď. Vytvoří se na základě požadavku protokolu HTTP a zničí se, když je hotova odpověď HTTP.
- @ViewScoped = Instance existuje, dokud uživatel pracuje s JSF pohledem. Instance existuje, dokud uživatel pracuje s JSF pohledem. Vytvoří se na základě požadavku protokolu HTTP a bude zničen, jakmile se uživatel přepne do jiného pohledu.
- @SessionScoped = Instance existuje tak dlouho, jak dlouho trvá relace HTTP. Vytvoří se při prvním HTTP požadavku zahrnujícím tuto třídu v relaci a zničí se při neplatnosti relace.
- @ApplicationScoped = Instance existuje tak dlouho, dokud běží webová aplikace. Vytvoří se při prvním HTTP požadavku zahrnujícím tuto třídu a zničí se při ukončení webové aplikace.
- @NoneScoped = Instance existuje dobu vyhodnocení EL výrazu. Vytvoří se na základě vyhodnocení EL a zničí se po vyhodnocení EL.

JSF UI komponenty podporují atribut s názvem binding. Ten slouží k provázání mezi komponentou a atributem v třídě. Přes tuto vazbu lze ovládat celou komponentu, navíc takto jsou odkryty metody, které na webové stránce nejsou přístupny.

Vzhledem k tomu, že JSF při každé žádosti vytváří novou instanci komponenty, musí Manage Beana existovat po dobu požadavku, jinak může být komponenta sdílena mezi různými pohledy [6].

5.2.6 Statistika používaných knihoven

JSF umožňuje sestavit stránku za použití komponent z knihoven třetích stran. Samozřejmě technologie JSP a JSTL nejsou žádnou výjimkou a mohou být použity v JSF aplikaci. Knihovna PrimeFaces byla vybrána jako nejoblíbenější na základě oblíbenosti vyhledávání a používanosti technologie v literatuře.



Obrázek 10: Porovnání knihoven pro JSF [21]

5.2.7 HTML front end

Webová stránka se skládá z komponent, které jsou poskytnuty skrze URL adresu.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.
    org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns=http://www.w3.org/1999/xhtml
  xmlns:h=http://java.sun.com/jsf/html
  xmlns:ui=http://java.sun.com/jsf/facelets
  xmlns:p="http://primefaces.org/ui">
```

```
<ui:composition template="/templates/layout.xhtml">
  <ui:define name="content">
    <h:form id="adminForm" class="hr-form">
     <h1>Administrator page</h1>
      <p:panelGrid styleClass="users" columns="2" layout="grid"
        columnClasses="w70,w30">
      <h:panelGroup class="user-actions">
        <p:commandButton styleClass="negative" icon="fa fa-trash-o"
            action="#{adminBean.deleteHR(admin)}" update="adminForm" />
      </h:panelGroup>
     </p:panelGrid>
      ...
```

Výpis 15: Ukázka HTML stránky a použití komponent

5.3 Výhody

Vývoj aplikace může být rozdělen do více týmů, kdy jeden tým vyvíjí prezentační vrstvu a jiný tým aplikační logiku. To je výhodné, zejména pokud je ve vývojovém procesu tým zabývající se pouze prezentační vrstvou. Jednou z největších výhod je, jak již bylo zmíněno skrytí komunikace na úrovni HTTP protokolu mezi klientem a serverem. Vývojář se nemusí starat o posílání požadavků, ale soustředí se pouze na hlavní aspekty vývoje. To je prezentace dat a vytvoření Managed Bean. Díky toho je aplikace vyvinuta v poměrně krátkém čase.

6 Google Web Toolkit

V roce 2006 Google vydal Google Web Toolkit (GWT), sadu vývojových nástrojů, programovacích utilit a grafických komponent, které umožňují vytvořit RIA jinak než dříve. Specifickou vlastností toho rámce je, že kód na straně prohlížeče není napsán v JavaScriptu, ale je napsán v Javě [7]. Při vývoji byl kladen velký důraz na možnost integrace umožňující využití již hotového JavaScript kódu nebo s již existujícími službami na straně serveru. Jádrem GWT je kompilátor Java-do-JavaScript, který vytváří kód schopný běžet v prohlížeči Internet Explorer, Firefox, Mozilla, Safari a Opera.

6.1 Funkčnost

Vytvořená aplikace se řadí mezi RIA. Kompilátor Java-do-JavaScript je nejdůležitější části rámce. Jak napovídá název, vývojář napíše klasický Java kód, který je kompilátorem převeden na JavaScript. Aby bylo možné v Java kódu používat knihovny jako *lang*, *math* a další musely být tyto knihovny emulovány. Za tuto funkcionalitu zodpovídá část rámce zvaná *JRE Emulation Library.*

6.1.1 RIA

Je typ webové aplikace někdy taky zvané jako Chytrý klient. Tato aplikace stojí na rozmezí tenkého a tlustého klienta a bere si pozitivní vlastnosti obou přístupů. Aplikaci není nutné instalovat, je automaticky aktualizována a vypadá jako aplikace pro stolní počítač. Provedení jednotlivých funkcí aplikace je rozděleno na částečné zpracování u klienta a částečné zpracování na serveru.

6.2 Vlastnosti rámce

Níže jsou uvedeny klíčové vlastnosti rámce a aspekty funkcionality.

6.2.1 Java-do-JavaScript kompilátor

Kompilátor je zodpovědný za převod Java kódu na JavaScript, z určitého pohledu funguje stejně, jako Java kompilátor, vytváří bytecode z Java kódu.

Začíná se třídou, která je označena jako entry-point, následně se zpracují závislosti potřebné ke kompilaci Java kódu. Kompilátor GWT se liší od standardního kompilátoru jazyka Java, tím že do modulu nezakomponuje vše, ale ty věci, které jsou potřeba. Když použijeme knihovnu obsahující velké množství funkcí, kompilátor vloží pouze ty funkce, které jsou využity ve třídě entry-point. Samozřejmě to samé platí i pro třídy. Kompilátor má tři režimy stylů, které určují, jak vypadá výsledný JavaScript kód.

Výchozí styl je obfuskace, což způsobí, že kód je zkomprimován a lze těžce dešifrovat. Hlavní důvod pro použití toho stylu je zmenšení velikostí výsledných JS souborů. Za neprvotně plánovanou výhodu lze považovat špatná čitelnost kódu, znesnadňující zkopírování kódu nebo využití slabin v kódu. Níže se nachází ukázku kódu, tento úryvek je výstupem kompilátoru GWT s aplikováním daného stylu.

```
funtion b () {return this.c + '@' + this.d ();}
```

Výpis 16: Ukázka obfuskovaného JS kódu

Další styl se nazývá hezký a produkuje čitelný JavaScript. Ukázka kódu toho stylu je odvozena ze stejného zdroje jako předchozí ukázka. Nyní lze vidět, že funkce je klasický toString().

```
function _toString () {
    return this._typeName + '@' + this._hashCode ();
}
```

Výpis 17: Ukázka hezky vypsaného JS kódu

Poslední styl je detailní, který vytváří kód podobný hezkému a navíc přidává jméno třídy jako součást názvu. Takto lze lépe pochopit kód a zjistit zdroj Java kódu, ze kterého byl JavaScript vytvořen. Z ukázky je tedy patrné, že funkce toString() byla vygenerována pro kořenovou třídu Object z balíku lang.

```
function java_lang_Object_toString __ () {
    return this.java_lang_Object_typeName + '@' + this.hashCode __ ();
}
```

Výpis 18: Ukázka detailně vypsaného JS kódu

Poslední dva styly jsou využívány spíše při vývoji softwaru. Důvod je jasný, chyby zobrazené v prohlížeči je snadné vysledovat ke zdrojovému Java kódu. V produkčním prostředí je vhodnější použít obfuskační styl, pro snížení velikosti JS souborů a pro zamaskování kódu.

Další zajímavou vlastností kompilátoru je samostatný výstup kompilace kódu. Ten samý zdrojový kód je zkompilován do několika JS souborů. Protože každý prohlížeč poskytuje vlastní JavaScript API, vytvořený kód není jednoduše přenositelný na jiný prohlížeč. Proto GWT kompilátor vytváří vlastní JS soubor pro různé prohlížeče, podporované prohlížeče jsou Internet Explorer, Firefox, Mozilla, Opera, Safari and Google Chrome. Bootstrap skript sám použije správný soubor, když se aplikace načítá.

6.2.2 JSNI

Jedním z hlavních principů GWT je využití jazyka Java pro vytvoření klientské části aplikace. V některých situacích není vhodné použít Javu nebo to dokonce není možné a potřebujeme volat JavaScript. JavaScript Native Interface byl vytvořen přesně pro tyto situace, kdy v Java

kódu je možné spustit JavaScript. GWT kompilátor je schopen připojit nativní JavaScript k vygenerovanému JavaScript kódu.

```
public native int addTwoNumbers (int x, int y)
/*-{
    var result = x + y;
    return result;
}-*/;
```

Výpis 19: Ukázka použití JSNI

Java používá klíčové slovo native k označení metod, jejichž implementace je napsána v jiném programovacím jazyce. Takovéto metody nemohou mít implementaci, protože jsou již naimplementovány například v nějaké knihovně. V případě GWT jsou nativní metody používány odlišně. Pod nativní metodu se do komentáře napíše JavaScriptový kód, který poskytuje implementaci metody.

6.2.3 JRE Emulation Library

GWT kompilátor potřebuje přístup ke zdrojovému kódu, aby jej mohl převést do JavaScriptu. Knihovna emulace JRE poskytuje programátorům sadu některých balíčku Java, které mohou použít. Nelze ovšem využít všechny baličky například balíček java.util neobsahuje některé třídy nebo metody, balíček java.io je velice omezen, dále java.sql balíček není kompletní. Důvod tohoto omezení je jednoduchý, JavaScript kód vytvořený pomocí GWT je spuštěn v prostředí prohlížeče a ten nemá přístup k žádným místním souborům nebo tiskárnám [8].

6.2.4 RPC

Remote Procedure Calls spojují klientskou a serverovou část aplikace skrze zasílání požadavků. Například je potřeba načíst data, RPC zaobalí požadavek a po získání požadovaných informací je překreslena určitá část stránky. Je více způsobů jak poslat požadavek. Jeden je s využitím třídy RequestBuilder, další způsob je GWT-RPC.

6.2.4.1 RequestBuilder

Třída RequestBuilder umožňuje vytvořit požadavek pro odeslání na server a poskytuje přístup k výsledkům odeslaným ze serveru.

```
String url = "/service/search";
RequestBuilder rb = new RequestBuilder(RequestBuilder.GET, url);
try {
    Request request = rb.sendRequest("term=GWT+in+Action", new RequestCallback
        () {
        public void onResponseReceived (Request req, Response res) {
            // process here
```

```
        }

        public void onError (Request req, Throwable exception) {
            // handle error here
        }
    });
}
catch (RequestException e) {
    // handle exception here
}
```

Výpis 20: Ukázka použití RequestBuilder

6.2.4.2 GWT RPC Je mechanismus pro předávání Java objektů na server a ze serveru zpátky přes standardní HTTP protokol. Rámec jde využít pro transparentní volání Java servletů a nechá GWT ať se postará o detaily na nižší úrovni, jako je serializace objektů [9].

```
public interface StockPriceServiceAsync {
  void getPrices(String[] symbols, AsyncCallback<StockPrice[]> callback);
}
```

Výpis 21: Ukázka RPC s asynchronním voláním metody

Kontrakt StockPriceServiceAsync musím mít implementaci pro zajištění funkcionality. V ukázce se implementací toho kontraktu nebudeme zabývat, neboť to není věc nutná k pochopení principu RPC.

```
private void refreshWatchList() {
    // Initialize the service proxy.
    ArrayList<String> stocks = new ArrayList<String>();
    StockPriceServiceAsync stockPriceSvc = GWT.create(StockPriceService.class);

    // Set up the callback object.
    AsyncCallback<StockPrice[]> callback = new AsyncCallback<StockPrice[]>() {
        public void onFailure(Throwable caught) {
            // Do something with errors.
        }

        public void onSuccess(StockPrice[] result) {
            updateTable(result);
        }
    };
```

```
    // Make the call to the stock price service.
    stockPriceSvc.getPrices(stocks.toArray(new String[0]), callback);
}
```

Výpis 22: Ukázka RPC s asynchronním voláním metody

6.2.5 Historie prohlížeče

Jedena z velkých nevýhod aplikací, které mají jednu stránku, a její obsah je dynamicky měněn, je využívání tlačítka zpět. Stejný problém má také technologie AngularJS jak již bylo zmíněno. Prohlížeč změnu obsahu stránky nebere jako přepnutí stránky, proto daná technologie musí řešit tuto vlastnost, aby měl uživatel plnohodnotný zážitek z používání aplikace, jak je zvyklý u jiných aplikací.

Řešení tohoto problému vyžaduje přidání skrytého rámečku na stránku a použití několika skriptů, aby vše fungovalo.

```
<iframe src="javascript:''" id="__gwt_historyFrame" style="position:absolute;
    width:0;height:0;border:0"></iframe>
```

Výpis 23: Ukázka skrytého rámečku na stránce

Při každé změně obsahu stránky je vytvořen nový token označující jakoby odlišné stránky. Token je jednoduše řetězec, který může aplikace analyzovat, aby se vrátila do určitého stavu [10].

```
http://www.example.com/com.example.gwt.HistoryExample/HistoryExample.html#page1
```

Výpis 24: Ukázka URL tokenu

ValueChangeHandler je volán při kliknutí na tlačítko zpět. Obstarává získání tokenu pro navrácení na předchozí stránku a provede změnu obsahu stránky.

6.2.6 JUnit

GWT poskytuje podporu pro JUnit, který je četně využíván Java programátory. Třída obsahující testovací případy musí rozšiřovat třídu GWTTestCase. Následně je třeba implementovat metodu getModuleName() a tím věci spojené s přípravou pro testování končí a zbývá pouze vytvoření testovacích případů. GWT používá název modulu k nalezení konfiguračního souboru projektu. GWTTestCase umožňuje testovat RPC volání na server. GWT spouští vlastní verzi Tomcat serveru, zkompiluje zdrojový kód a testuje očekávané výsledky po zavolání logiky severu z klientské strany.

6.3 Výhody

Začneme pravděpodobně tou nejvýznamnější výhodou a tou je úplně jiný způsob psaní RIA. GWT umožňuje psát webové aplikace stejně jako při psaní Java aplikací pro stolní počítače. Tím že velká většina kódu je psaná v Javě není nutná dobrá znalost jazyka JavaScript. K vývoji můžeme používat populární technologie jako je JUnit. Aplikace je zkompilovaná do několika JS souborů podle toho jaký prohlížeč je aktuálně používán. Vývojář tak nemusí řešit nepříjemné problémy s nekompatibilní implementací funkcionality mezi prohlížeči. Hezkou vlastností GWT je RPC, které umožňuje předávat Java objekty mezi klientem a serverem. Serializace a deserializace objektů je zaobalena v RPC, takže je vytvoření GWT aplikace zase o něco jednodušší.

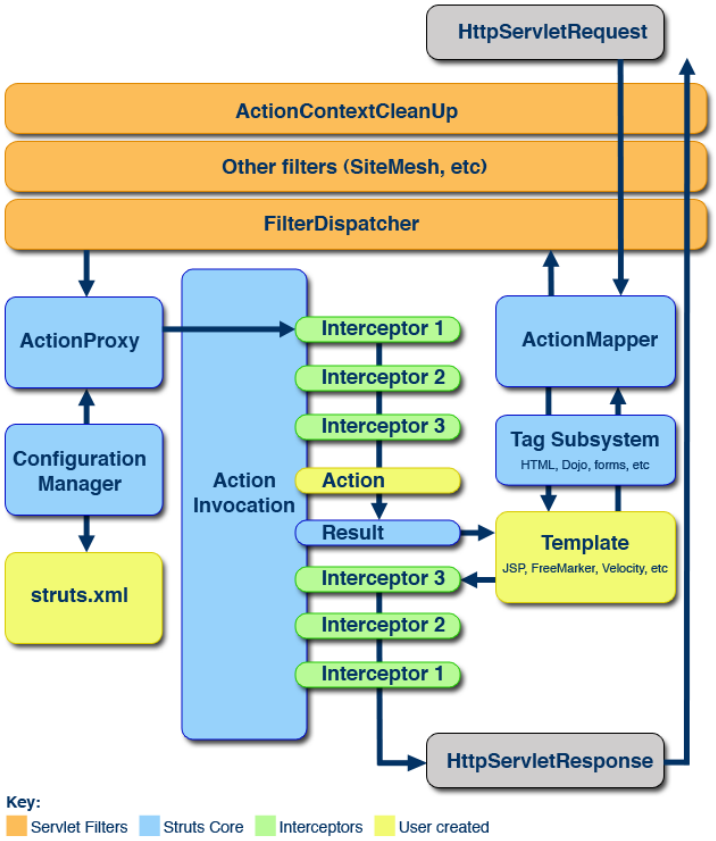
GWT také umožňuje komunikaci s jinými aplikacemi než Java prostřednictvím standardů JSON. Knihovny JSON jsou k dispozici pro většinu jazyků, takže integrace je relativně jednoduchý úkol.

7 Apache Struts 2

Apache Struts je bezplatný open source MVC rámec pro vytváření elegantních a moderních webových aplikací Java. Upřednostňuje konvence před konfiguraci, je rozšiřitelná pomocí architektury zásuvných modulů a dodává se se zásuvnými moduly, které podporují funkce REST, AJAX a JSON [11]. V podstatě se jedná o akčně orientovaný MVC rámec, akce jsou velkou a důležitou částí aplikací Struts 2. Apache Struts se může zdát jako zastaralá technologie a to v prvé řadě díky faktu, že první verze vydána v roce 2000. Apache Struts 2 byl vydán v roce 2006 a do roku 2017 byly vydávány další verze.

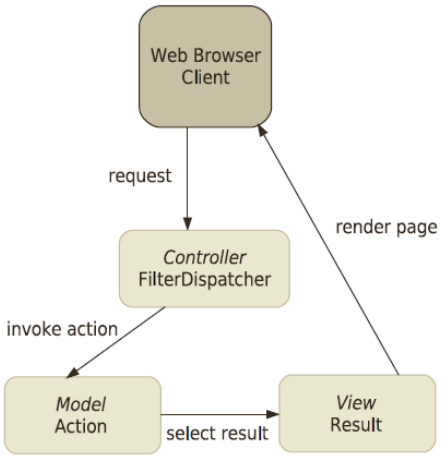
7.1 Funkčnost

Příchozí HTTP požadavek musí projít přes několik filtrů. FilterDispatcher používá ActionMapper k učení jaká akce má být provedena. Poté je řízení předáno ActionProxy. Akce musí projít přes všechny interceptory než muž být spuštěna. Stejnými interceptory musí akce projít i na cestě zpět k uživateli.



Obrázek 11: Abstraktní pohled na architekturu Struts 2 [22]

Jak již bylo zmíněno, v architektuře je použit MCV vzor. Tři hlavní pilíře tohoto vzoru jsou reprezentovány akce, výsledek a FilterDispatcher. Architektura je deklarovatelná dvěma způsoby. Je možné použít XML konfiguraci nebo konfiguraci pomocí Java anotací.



Obrázek 12: Struts 2 MVC [25]

7.2 Vlastnosti rámce

Níže jsou uvedeny klíčové vlastnosti rámce a aspekty funkcionality.

7.2.1 FilterDispatcher

FilterDispatcher plní roli kontroléru. Tento servlet filtr kontroluje každý příchozí požadavek, aby určil, která akce by měla požadavek zpracovat. Není nutné starat se o klasické věci kontroléru, stačí poskytnout mapování akcí na URL adresu požadavku. Obě metody konfigurace servletu jsou podporovány.

```
<filter>
    <filter-name>struts2</filter-name>
    <filter-class>
        org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
    </filter-class>
</filter>
<filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*.action</url-pattern>
</filter-mapping>
```

Výpis 25: Ukázka FilterDispatcher

7.2.2 Akce

Akce slouží ke dvěma účelům. Za prvé zapouzdřuje volání aplikační logiky. Za druhé slouží jako místo pro přesun dat. Jakmile akce získá řízení pro zpracování požadavku od kontroléru, připraví data a provede aplikační logiku. Až akce dokončí svou práci, předá výsledek komponentě pohled. Výsledek v kontextu akcí je pouze řetězec oznamující jak akce dopadla. Vybraný výsledek vykreslí odpověď. Registrování akce je ukázáno v kapitole konfigurace.

```
public class TextExamplesAction extends ActionSupport {
    private static Log log = LogFactory.getLog(TextExamplesAction.class);

    public String execute() {
        log.debug("Action executed");
        return SUCCESS;
    }
}
```

Výpis 26: Ukázka implementace akce

7.2.3 Výsledek

Konfigurace výsledků je to co mapuje návratovou hodnotu akce (jednoduchý řetězec) na stránku JSP, přesměrování, jinou akci a tak dále [12]. Běžně používané výsledky jsou “success” a “error”. Pokud akce rozšiřuje třídu ActionSupport má k dispozici několik základních řídících řetězců.

Předdefinované výsledky v ActionSupport:

- String SUCCESS = "success";
- String NONE = "none";
- String ERROR = "error";
- String INPUT = "input";
- String LOGIN = "login";

```
<result name="success" type="dispatcher">
    <param name="location">/ThankYou.jsp</param>
</result>
```

Výpis 27: Ukázka registrace výsledku

7.2.4 Interceptory

Jedna z klíčových vlastností rámce je zásobník interceptorů přes, které musí akce projít ještě předtím než je zpracována. Stejnými interceptory musí akce projít i na cestě zpět k uživateli. Není požadováno aby interceptor pracoval před i po provedení akce, stačí, aby fungoval v jednom směru. Interceptory zajišťují oddělení řízení tím způsobem, že nejprve je zpracován interceptor, poté se volá metoda execute() dané akce a nakonec je řízení opět předáno interceptoru. Celý proces zpracování požadavku může být ukončen už v prvním interceptoru, takže akce není vůbec provedena. Funkce systému jako je typová konverze, populace objektů, validace, nahrávání souborů a další jsou realizovaný pomoci interceptorů. Protože interceptory jsou koncipovány jako připojitelné moduly, každá akce může mít jinou sadu interceptorů. Interceptory jsou klasické třídy. Každý Interceptor má vlastní implementaci, na kterou se odkazuje atribut class.

```
<interceptors>
    <interceptor name="security" class="com.company.security.
        SecurityInterceptor"/>
    <interceptor-stack name="secureStack">
        <interceptor-ref name="security"/>
        <interceptor-ref name="defaultStack"/>
    </interceptor-stack>
</interceptors>
```

Výpis 28: Ukázka registrovaní interceptoru

Pro zaregistrování interceptoru slouží reference na poskytnuté jméno. Akce může mít takto několik interceptorů nebo může mít celou sadu.

```
<action name="VelocityCounter" class="org.apache.struts2.example.counter.
    SimpleCounter">
    <result name="success">...</result>
    <interceptor-ref name="security"/>
</action>
```

Výpis 29: Ukázka přidání výsledku a interceptoru k akci

7.2.5 OGNL

je výrazovým jazykem pro získávání a nastavování vlastností objektů jazyka Java a dalších doplňků, jako je zobrazení seznamu a výrazy lambda. Pro získání a nastavení hodnoty vlastnosti je používán stejný výraz [13]. OGNL umožňuje přístup k datům, která jsou na nějakém centrálním místě. V případě Struts 2 je tímto centrálním místem ValueStack obsahující data, na která se můžeme odkazovat, a která můžeme upravovat skrze EL.

7.2.6 ValueStack

Je kontejner obsahující veškerá data, která jsou potřebná v průběhu zpracování požadavku. Řídící tok spjatý s využitím ValueStack začíná přesunutím dat na toto centrální místo, data jsou upravována v průběhu vykonávání akce a nakonec jsou data přečtena výsledkem, který sestavuje výslednou stránku.

Nad kontejner k ValueStack je ActionContext, který je přístupný po celou dobu života akce. ActionContext uchovává mnohem více než jen ValueStack, obsahuje informace o požadavku, relaci a dalších. Každý požadavek má vlastní ActionContext, takže není třeba řešit synchronizovaný přístup několika vláken k jednomu kontejneru. I když je možné přímo přistupovat k ActionContext, není to doporučeno. Doporučený způsob přístupu k ActionContext je používaní nástrojů jako je OGNL.

7.2.7 Konfigurace

Jak již bylo zmíněno Struts 2 poskytuje dva typy konfigurací. Výběr stylu konfigurace je pouze o osobní preferenci vývojáře. Obě konfigurace mají stejný rozsah možností a vedou ke stejnému výsledku.

7.2.7.1 XML konfigurace Hlavním souborem u této konfigurace je struts.xml. I když je možné mít veškerou konfiguraci v tomto souboru, většina projektů bude mít několik konfiguračních souborů. V takovém případě je struts.xml považován za vstupní bod konfigurace, ve kterém jsou odkazy na ostatní konfigurační soubory. Takto je dosaženo lépe čitelné modulární konfigurace.

```
<action name="HelloWorld" class="manning.chapterOne.HelloWorld">
    <result name="SUCCESS">/chapterTwo/HelloWorld.jsp</result>
    <result name="ERROR">/chapterTwo/Error.jsp</result>
</action>
```

Výpis 30: Ukázka registrace akce

7.2.7.2 Java konfigurace Přímo do Java kódu jsou vloženy anotace specifikující metadata. Výhodou této konfigurace je možnost využití nástrojů pro čtení metadat, které vygenerují například přehled konfigurace aplikace. Struts 2 skenuje Java třídy a hledá anotace pro konfiguraci. V souboru web.xml je poskytnuto umístění takovýchto Java tříd. Aby bylo možné rozpoznat třídy akcí, musí implementovat rozhraní Action nebo název třídy musí končit slovem Action.

```
@Result(name="SUCCESS", value="/chapterTwo/HelloWorld.jsp" )
public class AnnotatedNameCollector extends ActionSupport {
    /* EMPTY */
}
```

Výpis 31: Ukázka registrace akce

Použití anotací automaticky přináší ulehčení práce, protože některé konfigurační prvky lze zjistit z třídy samotné. Každá Java třída musí mít jméno, anotace pro konfiguraci se vždy vážou k nějaké třídě. Rámec si sám zjistí tyto informace, a tudíž není potřebné explicitně uvádět tyto informace uvnitř anotace.

7.2.8 Ukázka HTML stránky

Jednoduchá HTML stránka s akcí *HelloWorld*.

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
<head>
    <title>Name Collector</title>
</head>
<body>
    <h4>Enter your name </h4>
    <s:form action="HelloWorld">
        <s:textfield name="name" label="Your name"/>
        <s:submit/>
    </s:form>
</body>
</html>
```

Výpis 32: Ukázka HTML stránky

7.3 Výhody

Struts 2 umožňuje dva způsoby konfigurace, vývojář si může zvolit takový styl konfigurace, jaký mu vyhovuje. Velkou částí rámce je konfigurace například akcí, interceptorů a výsledků, tyto konfigurace může provádět i člověk, který není s programováním zcela obeznámen. Používání EL je mírně odlišné oproti jiným rámcům. Speciální znaky pro označení výrazu nejsou potřebné (alespoň ve většině případů). Například když vytváříme stránku v JSF a chceme mít vstupní element propojený s objektem z ManagedBean musíme napsat speciální znaky označující výraz "#{admin.login}", pokud zapomeneme napsat speciální znak nebo jej napíšeme spatně, daný HTML element nebude provázán s objektem. Ve Struts 2 stačí pouze napsat "admin.login".

8 Praktická část

Cílem praktické části práce je vytvořit nástroj, který zjednodušuje vytvoření Angular projektů ve spojení se Spring Boot aplikací. Aplikace má vytvořené testy pro otestování vybrané funkcionality a otestování pokrytí.

Řešení je pouze prototyp, Spring Boot aplikace nemá do detailu vyřešené závislosti. Pro každou entitu v konfiguraci jsou vytvořeny třídy, které potřebují závislosti Web (web) a JPA (data-jpa). Bez těchto závislostí nepůjde projekt spustit. V případě, že projekt má přidanou závislost na JPA, musí mít navíc přidanou závislost na datový zdroj (například PostgreSQL). Datový zdroj musí být nakonfigurován pro správný běh aplikace.

Veškerý kód je uložen v Git repositáři. Konfigurátor (neboli podniková aplikace) je dostupný na `https://git.cs.vsb.cz/pri0097/web-configurator`. Na adrese `https://git.cs.vsb.cz/pri0097/project-generator` je nahraný nástroj pro generování projektů.

8.1 Návrh

Mezi první aktivity při vyváření návrhu neodmyslitelně patří zvážení možných přístupů a vytvoření vize. Po analýze jiných nástrojů generující projekty byly nalezeny dva přístupy, které splňovaly aspekty autorovy vize. Jedna z možností je vytvoření konzolové aplikace, kterou si uživatel stáhne a po spuštění se vygeneruje projekty. Veškerou konfiguraci pro vytvoření projektů by musel uživatel zadat přes příkazovou řádku. Druhá možnost je generování projektů na serverové straně. Uživatel by si přes webové rozhraní nakonfiguroval projekty, pak si jen stáhne archív s vytvořenými projekty dle zadané konfigurace.

8.2 Použitý přístup

Jako nejlepší přístup se nejprve jevil přístup podnikové aplikace. Uživatel by se nemusel o nic starat, pouze by vyplnil konfiguraci a zbytek práce by udělal server. Výhodou je, že uživatel nemusí mít nainstalované na svém počítači nástroje jako je Angular CLI, aby si mohl vygenerovat projekt. Ovšem po hlubším přezkoumání tohoto přístupu se objevily negativní vlastnosti vedoucí ke změně přístupu.

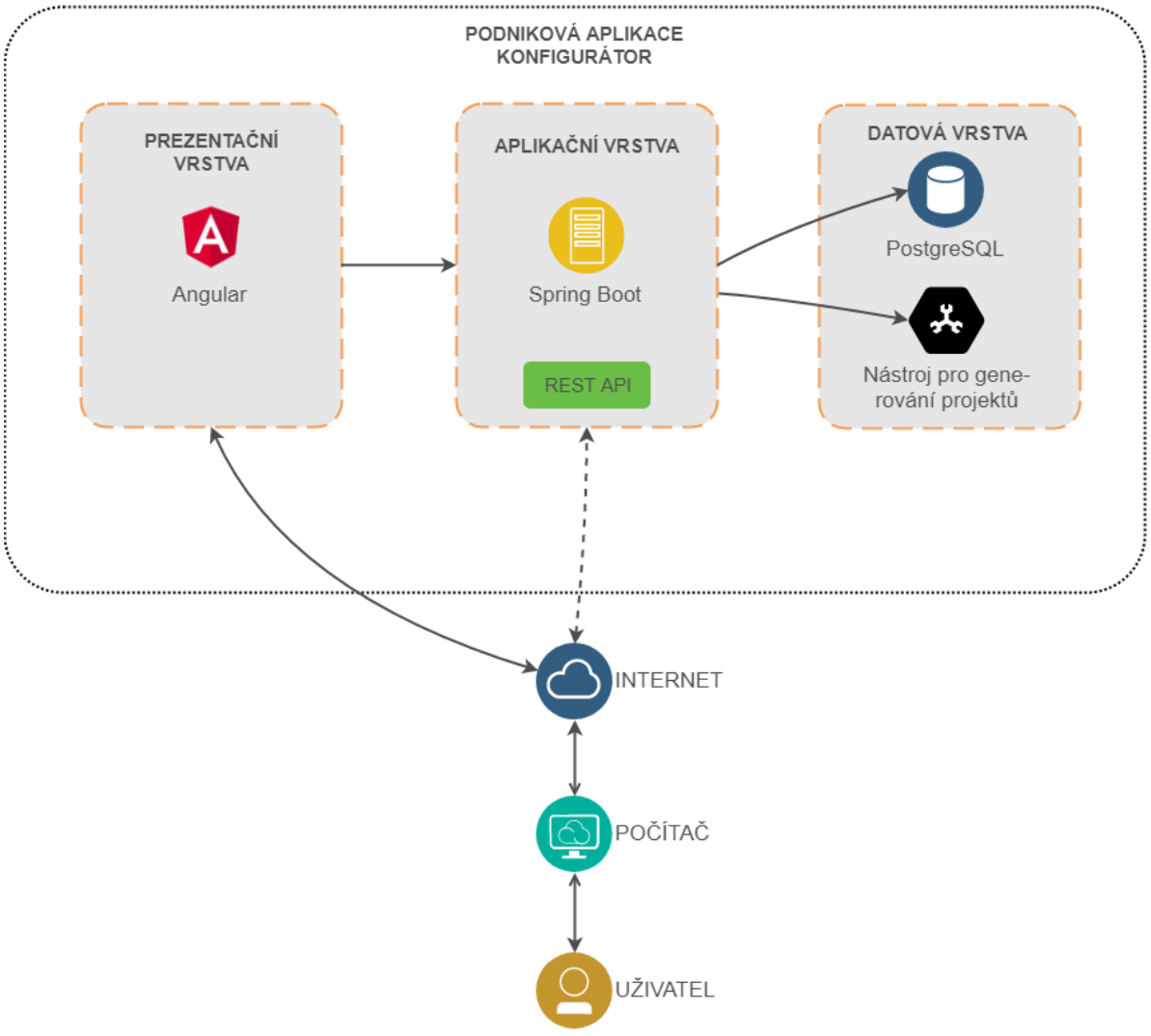
Velikost Angular projektu po vygenerování činí přibližně 350 MB. Stahování takového objemu dat je časově náročná věc. Když se k tomu připočte čas strávený generováním projektů a jejich zabalení do archívu, výsledný čas je příliš velký. Bylo by možné, aby do výsledného archivu byly zabaleny pouze zdrojové soubory Angular aplikace, ovšem aby se tyto závislosti stáhly klient by musel mít nainstalované Node.js a Angular CLI. Pokud uživatel již má tento software u sebe nainstalovaný, nabízí se možnost generování Angular aplikace přímo u klienta. To umožní lepší zpětnou vazbu pro uživatele a sníží se nároky na server.

Proto byl zvolen nový přístup kombinující již výše zmíněné přístupy. Uživatel jsi přes webové rozhraní nakonfiguruje projekty, které budou vytvořeny. Tato konfigurace je vložena do konzo-

lové aplikace, kterou si uživatel stáhne. Po spuštění konzolové aplikace se bez zásahu uživatele vygenerují projekty. Uživatel může v konzolové aplikaci sledovat výpis aktuálně prováděných kroků a jejich stav.

Další možnost implementace je zabalit konfigurační soubor mimo jar soubor. Nástroj pro generování projektů by si vždy zkusil načíst tento soubor, pokud by ho nenašel, dal by uživateli možnost použít výchozí konfigurační soubor zabalený v jar souboru. Výhoda toho přístupu je, že uživatel má snadnější přístup ke konfiguračnímu souboru a může si jej jednodušeji upravit.

Aktuální verze řešení je rozdělena na dva projekty. Jeden projekt je konzolová aplikace provádějící generování aplikací a druhý projekt je podniková aplikace. Klientskou část tvoří aplikace založená na rámci Angular 5 a serverovou část poskytuje Spring Boot aplikace.



Obrázek 13: Model návrhu nástroje

8.3 Popis řešení

Níže jsou popsány technické rozhodnutí tři hlavních komponent nástroje. Podniková aplikace poskytuje uživateli prostředky pro nakonfigurování a sestavení projektu. Konfiguraci projektu zajišťuje webový formulář na klientské části aplikace. Ten je poslán na serverovou část aplikace, která odešle uživateli sestavovací nástroj i s konfigurací.

8.3.1 Klientská část konfigurátoru

Aplikace založená na rámci Angular složí pro centralizaci konfigurace projektů Spring Boot a Angular. Předmětem konfigurace je například jméno aplikace, typ zabalení nebo seznam entitních tříd.

Aplikace má pouze jedinou HTML stránku, která je součástí komponenty *configuration-form*. Stránka obsahuje veškeré atributy spojené s konfigurací. Formulář ukládá uživateli omezení na vyplnění povinných polí nutných pro správný běh sestavení projektu. Kvůli velkému množství položek ve formuláři bylo nutné použít upravený CSS styl aplikace. (uživatel je zvyklý na určitý standard vzhledu)

Po potvrzení formuláře se odešle požadavek na server, který vrátí spustitelný soubor s uživatelovou konfigurací. Konfigurace je uložena v požadavku na server, který je poslán přes HTTP protokol.

8.3.2 Serverová část konfigurátoru

Aplikační logika se stará o zabalení uživatelovy konfigurace do nástroje pro generování projektů a odeslání archivu zpátky k uživateli. Serverovou část tvoří Spring Boot aplikace, která má vystavenou funkcionalitu přes REST API, takže serverová část aplikace není závislá na té klientské. Kontrolér aplikace reaguje pouze na HTTP požadavek typu POST. Tento typ požadavku je používán, když klient posílá nová data na server.

Aplikační kontrolér je mapován na */configurator*. Kdokoliv kdo pošle správně utvořený požadavek, obdrží upravený archiv, který má server uložený na pevném disku. Kontrolér v požadavku očekává konfiguraci ve formátu JSON, ta je vložena do nástroje pro generování projektů.

Uvnitř nástroje je nalezen soubor pro konfiguraci, následně je do něj zapsána konfigurace. Obsah souboru pro konfiguraci vždy obsahuje předchozí konfiguraci a ta je přepsána aktuální konfigurací. Původní archiv je takto znovu používán, protože se mění pouze konfigurace.

8.3.3 Nástroj pro generování projektů

Cílem nástroje je vygenerování podnikové aplikace kde klientskou část tvoří Angular a serverovou část tvoří Spring Boot. Výsledkem implementace je jar soubor obsahující funkcionalitu pro generování projektů a konfiguraci pro projekty. Archiv dále obsahuje všechny knihovny použité při implementaci aby nebyl start nástroje podmíněn přítomností potřebných knihoven v počítači

klienta. Prerekvizity pro spuštění nástroje jsou Java verze 8, Node.js a Angular CLI. Každá akce je provázena výpisem aby měl uživatel přehled co se děje.

8.3.3.1 Implementační rozhodnutí Nástroj je implementován s ohledem na obecnost aby bylo možné v budoucnu přidat další projekty, které mohou být generovány. Toto zajišťuje implementovaný vytvářecí vzor Stavitel. Každý typ projektu má vlastní stavitelskou třídu. Uvnitř třídy jsou kroky vedoucí k sestavení projektu. Součásti vize nástroje byla možnost použití jednoho kroku nebo jinými slovy akce pro více projektů. Protože například zabalení projektů do archivu by muselo být provedeno pro všechny projekty a tento krok obecně představuje stejnou akci. Proto jsou jednotlivé akce implementovány jako návštěvníci. Tímto způsobem jsou akce obecné a každý projekt si v rámci akce udělá, co potřebuje. Jednotliví návštěvníci jsou zavoláni v konkrétní stavitelské třídě pro projekt. Samozřejmě v rámci návrhu byly zvažovány i jiné implementace ohledně sestavovacích akcí. Například každá akce by mohla být implementována jako dekorátor, kdy třída by byla akce, a výsledkem by byl hotový projekt. Další možnou implementací sestavení projektu je vzor řetěz odpovědnosti. Každá akce by byla odpovědná třída v řetězu sestavení. Každý projekt by měl vlastní řetěz sestavení.

Konfigurační soubor je sestaven stylem, aby bylo možné přidat nebo odebírat konfigurace. Na základě konfigurace je vybrán konkrétní stavitel projektu.

8.3.3.2 Stavitel pro Angular projekt Projekt je sestavován v několika krocích. Začíná se s načtením konfigurace, aby následující kroky měly potřebná data. Pro generování projektu je použito Angular CLI. Programově jsou spuštěny příkazy v příkazové řádce generující dílčí komponenty projektu nebo samotný projekt. Pro úspěšné provedení příkazů musí klient splňovat prerekvizity. Generování funguje na operačním systému Windows a systémech založených na jádře Linux.

8.3.3.3 Stavitel pro Spring Boot projekt Spring Boot stavitel stejně jako jeho kolega nejprve načte konfiguraci projektu jako je název, popis, verze jazyka Java, způsob zabalení aplikace a další.

Založení projektu ovšem používá velice odlišný přístup. Webová služba na adrese `https://start.spring.io/` poskytuje generování Spring Boot projektu. Obdobně jako autorova aplikace, i tato webová služba používá webové rozhraní pro nastavení konfigurace projektu. Zajímavosti je, že IDE IntelliJ IDEA využívá tuto službu, když vývojář zakládá Spring Boot projekt. Vývojář si nastaví jméno aplikace, závislosti a mnoho dalších věcí, které jsou odeslány v požadavku na server a ten vrátí archiv s nakonfigurovaným projektem. Stejný přístup využívá autorův Spring Boot stavitel. Požadavek nese veškeré parametry konfigurace v URL.

Nástroj generuje klasické entitní třídy, třídy pro služby, rozhraní pro přístup k databázi a kontrolér. Je v podstatě vygenerováno REST API pro jednoduché CRUD operace. Pro gene-

rování souborů je použit šablonovací nástroj FreeMarker. Jednotlivé soubory jsou šablony, do kterých je doplněn název entity. FreeMarker je přidán do aplikace jako knihovna.

Ještě předtím než jsou soubory vytvořeny, je vytvořena adresářová struktura odpovídající stylu Spring Boot apliakce. Na začátku cesty je název aplikace, pak vždy následuje *src/main/java* a cesta balíčku. Na konci cesty je název výsledné složky (neboli název balíčku). Všechny názvy specifické pro projekt jsou získány z konfigurace.

Jakmile je vytvořena adresářová struktura pro nový soubor a veškeré vlastnosti pro šablonu jsou nastaveny, vše je připraveno pro vytvoření samotného souboru. Tímto způsobem jsou vytvořeny další soubory jako je rozhraní a další.

9 Závěr

Jednotlivé metody výzkumu přinesly informace potřebné pro vytvoření náhledu jakým směrem se současné trendy vývoje UI pro Java aplikace ubírají. Většina metod analyzovala data za období od roku 2017 po rok 2018. Prohledaná literatura byla publikována ve větším časovém rozmezí.

Prozkoumaná literatura se ve většině případů zabývala vývojem podnikových aplikací. Knihy byly vydány v rozmezí od roku 2002 až po rok 2018. Starší knihy (okolo roku 2012) se spíše zabývaly standardem podnikových aplikací. U těchto knih byly nejčastěji používané technologie JSP a JSF. Novější knihy se více zaměřovaly na tvorbu UI za pomocí JS rámce. U takovýchto knih se častěji objevoval rámec Angular (pět knih z deseti prozkoumaných se zaměřily na tento rámec).

Společnost *Stack Exchange Inc.* vytváří každý rok statistky ohledně vývojářské komunity. Statistika za rok 2017 a 2018 ukazuje, že k nejpopulárnějším rámcům obecně patří rámec Angular a React. Na základě odpovědí této komunity se stal rámce Angular populárnějším oproti rámci React.

Společnost *ZeroTurnaround* neustále aktualizuje statistiku nejpopulárnějších Java webových rámců. Statistika uvádí JSF, GWT a Struts jako nejpopulárnější rámce pro tvorbu UI.

Veřejný repositář *GitHub* při porovnání rámců Angular a React ukázal větší oblíbenost rámce React. Zkoumání obecných informací o repositářů ukázalo, že rámec React má o 58 829 vývojářů více, kteří si oblíbili dané repositář. Rámec GWT si oblíbilo 994 vývojářů. Těsně za ním je rámec Struts 2 s 827 vývojáři, kteří si tento rámec oblíbili. Uvedené počty jsou aktuální ke dni 24. 4. 2018.

Po kombinaci výsledků z uvedených metod výzkumu byly vybrány rámce Angular, JSF, GWT a Apache Struts 2. Z vybraných technologií je Angular jako jediný JS rámec.

Pro vypracování praktické části byl vybrán rámec Angular. Výsledkem této části práce je nástroj zjednodušující tvorbu UI, který se skládá ze dvou projektů. Jedním je webový konfigurátor, jehož účelem je uživatelsky přívětivé vytvoření konfigurace projektů. Důležitější částí nástroje je druhý projekt generující projekty. Nástroj vygeneruje Angular aplikaci propojenou s aplikací Spring Boot.

Webový konfigurátor je podniková aplikace, která má oddělenou prezentační a aplikační vrstvu. Angular aplikace tvoří prezentační vrstvu, které je závislá na aplikační vrstvě tvořené Spring Boot aplikací. Nástroj pro generování projektů je konzolová aplikace.

Literatura

[1] RUEBBELKE, Lukas a Brian FORD. AngularJS in action. Shelter Island, NY: Manning, 2015. ISBN 9781617291333.

[2] Architecture overview. Angular [online]. Mountain View, California, United States: Google, c2010-2018 [cit. 2018-04-04]. Dostupné z: `https://angular.io/guide/architecture`

[3] MONTEIRO, Fernando. Learning Single-page Web Application Development. Birmingham B3 2PB, UK: Packt Publishing, 2014. ISBN 978-1-78355-209-2.

[4] Angular Single Page Applications (SPA): What are the Benefits?. Angular University [online]. Mountain View, California, United States: Google, c2018 [cit. 2018-03-28]. Dostupné z: `https://blog.angular-university.io/why-a-single-page-application-what-are-the-benefits-what-is-a-spa/`

[5] SHIP, Howard M. Lewis. Tapestry in action. Greenwich: Manning, c2004. ISBN 1-932394-11-7.

[6] LEONARD, Anghel. Mastering JavaServer Faces 2.2: Master the art of implementing user interfaces with JSF 2.2. Birmingham B3 2PB, UK: Packt Publishing, 2014. ISBN 978-1-78217-646-6.

[7] HANSON, Robert. a Adam. TACY. GWT in action: easy Ajax with the Google Web toolkit. Greenwich, CT: Manning, c2007. ISBN 1-933988-23-1.

[8] KEREKI, Federico. Essential GWT: building for the web with Google Web toolkit 2. Upper Saddle River, NJ: Addison-Wesley, c2011. Developer's library. ISBN 978-0-321-70514-3.

[9] Ajax Communication: Introduction. [GWT] [online]. Mountain View, California, United States: Google, 2006 [cit. 2018-02-06]. Dostupné z: `http://www.gwtproject.org/doc/latest/tutorial/clientserver.html`

[10] History. [GWT] [online]. Mountain View, California, United States: Google, 2006 [cit. 2018-02-07]. Dostupné z: `http://www.gwtproject.org/doc/latest/DevGuideCodingBasicsHistory.html`

[11] Apache Struts [online]. Forest Hill, Maryland, United States: Apache Software Foundation, c2000-2018 [cit. 2018-03-09]. Dostupné z: `https://struts.apache.org/`

[12] NEWTON, Dave. Apache Struts 2 Web Application Development: Design, develop, test, and deploy your web applications using the Struts 2 framework. Birmingham, B27 6PA, UK: Packt Publishing, 2009. ISBN 978-1-847193-39-1.

[13] OGNL - Apache Commons OGNL - Object Graph Navigation Library. Apache Commons – Apache Commons [online]. Forest Hill, Maryland, United States: Apache Software Foundation, c1997-2013 [cit. 2018-03-26]. Dostupné z: http://commons.apache.org/proper/commons-ognl/

[14] NANCE, Christopher. TypeScript Essentials: Develop large scale responsive web applications with TypeScript. Birmingham B3 2PB, UK: Packt Publishing, 2014. ISBN 978-1-78398-576-0.

[15] Developer Type. In: Stack Overflow Insights [online]. New York, USA: Stack Exchange, 2018 [cit. 2018-02-18]. Dostupné z: https://insights.stackoverflow.com/survey/2018#developer-profile-developer-type

[16] Frameworks, Libraries, and Tools. In: Stack Overflow Insights [online]. New York, USA: Stack Exchange, 2018 [cit. 2018-02-20]. Dostupné z: https://insights.stackoverflow.com/survey/2018#technology-frameworks-libraries-and-tools

[17] Frameworks used along with programming technologies. In: Stack Overflow Blog - A destination for all things related to development at Stack Overflow [online]. New York, USA: Stack Exchange, 2018 [cit. 2018-02-19]. Dostupné z: https://stackoverflow.blog/2018/01/11/brutal-lifecycle-javascript-frameworks/

[18] Stack Overflow Trends. In: Stack Overflow Insights [online]. New York, USA: Stack Exchange, c2018 [cit. 2018-02-25]. Dostupné z: https://insights.stackoverflow.com/trends?tags=jquery-ui%2Cjsp%2Cgwt%2Cjsf%2Cextjs%2Cstruts%2Csencha-touch%2Cvaadin

[19] Stack Overflow Trends. In: Stack Overflow Insights [online]. New York, USA: Stack Exchange, c2018 [cit. 2018-02-25]. Dostupné z: https://insights.stackoverflow.com/trends?tags=angularjs%2Cangular%2Creactjs%2Cvue.js%2Cjsp%2Cmeteor

[20] Overview2. In: Angular [online]. Mountain View, California, United States: Google, c2010-2018 [cit. 2018-02-30]. Dostupné z: https://angular.io/guide/architecture

[21] Trends. In: Trends [online]. Mountain View, California, United States: Google, 2018 [cit. 2018-03-06]. Dostupné z: https://trends.google.com/trends/explore?cat=31&q=%2Fm%2F011f50zg,%2Fm%2F03gswc9,%2Fm%2F047fy_g,%2Fm%2F0131kq0f,%2Fm%2F02622b8

[22] Big Picture. In: Apache Struts [online]. Forest Hill, Maryland, United States: Apache Software Foundation, c2000-2018 [cit. 2018-03-06]. Dostupné z: https://struts.apache.org/core-developers/big-picture

[23] Java Web Frameworks Index. In: JRebel & XRebel By ZeroTurnaround zeroturnaround.com [online]. Tartu, Estonia: ZeroTurnaround, c2007-2018 [cit. 2018-04-04]. Dostupné z: https://zeroturnaround.com/webframeworksindex/

[24] Trends. In: Trends [online]. Mountain View, California, United States: Google, 2018 [cit. 2018-04-03]. Dostupné z: https://trends.google.com/trends/explore?cat=31&q=%2Fm%2F0h96s3c,%2Fm%2F026mhl,%2Fm%2F0d6jq1,%2Fm%2F011807pt,%2Fm%2F0641ly1

[25] BROWN, Donald, Chad Michael. DAVIS a Scott. STANLICK. Struts 2 in action. Greenwich, CT.: Manning, c2008. ISBN 1-933988-07-x.

[26] SHAH, Nehal a Gabriel José BALDA ORTÍZ. HTML5 Enterprise Application Development: A step-by-step practical introduction to HTML5 through the building of a real-world application, including common development practices. Birmingham B3 2PB, UK: Packt Publishing, 2013. ISBN 978-1-84968-568-9.

[27] GIERER, Gerald. Enterprise Application Development with Ext JS and Spring: Develop and deploy a high-performance Java web application using Ext JS and Spring. Birmingham B3 2PB, UK: Packt Publishing, 2013. ISBN 978-1-78328-545-7.

[28] FAIN, Yakov, Victor RASPUTNIS, Anatole TARTAKOVSKY, Viktor GAMOV, Sharon WILKEY a Rebecca DEMAREST. Enterprise web development: building HTML5 applications : from desktop to mobile. Sebastopol, California: O'Reilly Media, 2014. ISBN 978-1-449-35681-1.

[29] BERNAL, Joey. Web 2.0 and social networking for the enterprise: guidelines and examples for implementation and management within your organization. Upper Saddle River, NJ: IBM Press, c2010. ISBN 978-0-13-700489-8.

[30] PILGRIM, Peter. Digital Java EE 7 Web Application Development: Develop Java enterprise applications to meet the emerging digital standards using Java EE 7. Birmingham B3 2PB, UK: Packt Publishing, 2015. ISBN 978-1-78217-664-0.

[31] PILGRIM, Peter. Java EE 7 Developer Handbook: Develop professional applications in Java EE 7 with this essential reference guide. Birmingham B3 2PB, UK: Packt Publishing, 2013. ISBN 9781849687942.

[32] CHAPPELL, David A. a Tyler. JEWELL. Java Web services. Sebastopol, CA: O'Reilly, c2002. ISBN 0-596-00269-6.

[33] PADMANABHAN, Prashant. Java EE 8 and Angular: A practical guide to building modern single-page applications with Angular and Java EE. Birmingham B3 2PB, UK: Packt Publishing, 2018. ISBN 978-1-78829-120-0.

[34] PADMANABHAN, Prashant. Learning Spring Boot 2.0: implify the development of lightning fast applications based on microservices and reactive programming. 2nd ed. Birmingham B3 2PB, UK: Packt Publishing, 2017. ISBN 978-1-78646-378-4.

[35] Front-end reactive architectures. New York, NY: Springer Science+Business Media, 2018. ISBN 978-1-4842-3179-1.

A Seznam příloh na CD

- `PRI0097.pdf` ..... Diplomová práce ve formátu PDF/A
- `web-configurator` ..... Konfigurátor pro generování projektů
  - `client` ..... Angular aplikace, která je klientská část konfigurátoru
    - `e2e`
    - `src`
    - `.angular-cli.json`
    - `.editorconfig`
    - `.gitignore`
    - `README.md`
    - `karma.conf.js`
    - `package-lock.json`
    - `package.json`
    - `protractor.conf.js`
    - `tsconfig.json`
    - `tslint.json`
  - `server` ..... Spring Boot aplikace, která je serverová část konfigurátoru
    - `.mvn`
    - `src`
    - `.gitignore`
    - `.mvnw`
    - `mvnw.cmd`
    - `pom.xml`
    - `README.md`
- `project-generator` ..... Nástroj pro generování projektů
  - `src`
  - `.gitignore`
  - `pom.xml`
  - `README.md`