SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

SPRING: DIFFERENCE BETWEEN SPRING MVC AND SPRING BOOT

SEMINÁRNA PRÁCA

2022 Rendek, Richterová, Sokol, Ulrichová, Včelková

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

SPRING: DIFFERENCE BETWEEN SPRING MVC AND SPRING BOOT

SEMINÁRNA PRÁCA

Študijný program:	Aplikovaná informatika
Predmet:	I-ASOS – Architektúra softvérových systémov
Prednášajúci:	RNDr. Igor Kossaczký, CSc.
Cvičiaci:	Ing. Stanislav Marochok

Bratislava 2022 Rendek, Richterová, Sokol, Ulrichová, Včelková

Obsah

Zoznam obrázkov a tabuliek

Zoznam skratiek

AOP	Aspect-oriented programming
API	Application Programming Interface
DI	Dependency Injection
EE	Enterprise Edition
EJB	Enterprise Java Beans
IoC	Inversion of Control
JDBC	Java database connectivity
JDK	Java Development Kit
JSF	Java Server Face
JSP	JavaServer Pages
JTA	Java Transaction API
MVC	Model-View-Controller
ORM	Object-Relational Mapping
OS	Operation System
POJO	Plain Old Java Objects
UI	User Interface

Úvod

V súčastnosti, v tejto uponáhľanej dobe, sa čoraz viac stretávame s tým, že ľudia hľadajú cestu, ako si veci zjednodušiť. Spring framework umožňuje vývojárom venovať sa samotnej tvorbe aplikácie, či biznis logike, vďaka svojej konfigurácii a tomu, že sa stará o infraštruktúru. V našej semestrálne práci si povieme, čo to Spring framework je, a taktiež si povieme o Spring Boot a Springom MVC. Ku jednotlivým frameworkom si rozoberieme aké výhody a nevýhody ich použitie prináša a bližšie popíšeme rozdiely medzi Spring MVC a Spring Boot.

1 Spring

Framework je súbor preddefinovaného kódu, do ktorého môžu vývojári pridávať svoje riešenia problému, v podobe kódu. Existuje mnoho populárnych Java frameworkov vrátane Java Server Faces (JSF), Maven, Hibernate, Struts a Spring.

Framework Spring je open-source framework, ktorý poskytuje infraštruktúrnu podporu pre vývoj aplikácií v jazyku Java. Patrí medzi najpopulárnejšie frameworky Java Enterprise Edition (Java EE). Spring pomáha vývojárom vytvárať vysoko výkonné aplikácie s použitím obyčajných objektov Java - Plain Old Java objects (POJO).

Autorom frameworku Spring je Rod Johnson, ktorý ho prvýkrát vydal pod licenciou Apache 2.0 v júni 2003 a umiestnil ho na SourceForge. Milióny vývojárov na celom svete používajú Spring framework na vytváranie vysoko výkonného, ľahko testovateľného a opakované použiteľného kódu. [1, 2]

Spring je nenáročný, pokiaľ ide o veľkosť a transparentnosť. Základná verzia Spring frameworku má okolo 2 MB.

1.1 Prečo práve Spring?

Programy v jazyku Java sú zložité a obsahujú veľa náročných komponentov. Náročné znamená, že vzhľad a vlastnosti komponentov závisia od základného operačného systému (OS).

Spring sa považuje za bezpečný, lacný a flexibilný framework. Zvyšuje efektivitu programovania a skracuje celkový čas vývoja aplikácie, pretože je odľahčený, efektívne využíva systémové zdroje a má veľkú podporu.

Spring odstraňuje zdĺhavú konfiguráciu a stará sa o infraštruktúru, takže sa vývojári môžu sústrediť na písanie biznis logiky a na tvorbu aplikácie.

1.2 Ako funguje Spring

Webová aplikácia, ktorú vieme vytvárať prostredníctvom frameworku Spring (vrstvená architektúra) bežne obsahuje tri vrstvy:

- **Prezentačná vrstva/užívateľské rozhranie (UI)** - Ide o najvzdialenejšiu vrstvu, ktorá sa stará o prezentáciu obsahu a interakciu s používateľom.
- **Vrstva biznis logiky** - Centrálna vrstva, ktorá sa zaoberá logikou programu.
- **Vrstva prístupu k údajom** - Hlboká vrstva, ktorá sa zaoberá získavaním údajov zo zdrojov.

Aby aplikácia fungovala, vrstvy musia navzájom od seba závisieť. Inými slovami, prezentačná vrstva komunikuje s vrstvou biznis logiky, ktorá komunikuje s vrstvou prístupu k údajom. Voľné prepojenie je ideálne, pretože voľne prepojené komponenty sú nezávislé, čo znamená, že zmeny v jednom z nich neovplyvnia fungovanie ostatných.
Základom logiky Spring je dependency injection. Dependency injection je programovací vzor, ktorý umožňuje vývojárom vytvárať viac oddelených architektúr. Spring vďaka nim rozumie rôznym anotáciám jazyka Java, ktoré vývojár umiestňuje na vrcholy tried. Spring vie, že vývojár chce vytvoriť inštanciu triedy a že by ju mal Spring spravovať. Spring tiež rozumie závislostiam a zabezpečuje, aby všetky vytvorené inštancie mali správne vyplnené závislosti.

Aby Spring Framework inštancioval objekty a vyplnil závislosti, programátor jednoducho povie Springu, ktoré objekty má spravovať a aké sú závislosti pre každú triedu. Vývojár tak urobí pomocou anotácií, ako napr:

- **@component** - Dáva Springu vedieť, ktoré triedy má spravovať (vytvárať). Označí beans (objekty) ako spravované komponenty, čo znamená, že Spring bude tieto triedy autodetekovať.
- **@autowired** - Informuje Spring, ako má postupovať pri inštanciácii triedy (začne hľadať danú závislosť medzi komponentmi/triedami, aby našiel zhodu). To ušetrí čas od spájania kódu a umožní to, aby Spring našiel, čo je potrebné kde injektovať.

Dôležité pojmy

- **Automatické zapájanie** - (z angl. Autowiring) proces, ktorým Spring identifikuje závislosti, zhody a zapĺňa ich.
- **Bean** - Spring bean je objekt, ktorý je inštanciovaný, vytvorený a spravovaný IoC kontajnerom. Beans sú základom aplikácie.
- **Dependency injection** - Návrhový vzor programovania, vďaka ktorému je kód voľne previazaný, čo znamená, že akákoľvek zmena v aplikácii jedného, neovplyvní druhý.
- **Inverzia kontroly** - (z angl. Inversion of Control (IoC)) - Odňatie kontroly triede a jej odovzdanie Spring Frameworku.

- **Inverzný kontajner riadenia** - (z angl. Inversion of control container) je jadro Spring Frameworku, kde sa objekty vytvárajú, spájajú, konfigurujú a spravujú počas celého životného cyklu.

Pred príchodom Enterprise Java Beans (EJB) museli vývojári v jazyku Java používať JavaBeans na vytváranie webových aplikácií. Hoci JavaBeans poskytovala pomoc pri vývoji komponentov používateľského rozhrania (UI), nebola schopná poskytovať služby, ako je správa transakcií a bezpečnosť, ktoré boli potrebné na vývoj robustných a bezpečných korporátnych aplikácií. Príchod EJB sa považoval za riešenie tohto problému. Vývoj korporátnych aplikácií s EJB však nebol jednoduchý, pretože vývojár musel vykonávať rôzne úlohy, ako napríklad vytvárať rozhrania Home, Remote a implementovať metódy spätného volania životného cyklu, čo viedlo k zložitosti poskytovania kódu pre EJB. Kvôli tejto komplikácii začali vývojári hľadať jednoduchší spôsob vývoja korporátnych aplikácií. Framework Spring (ktorý je všeobecne známy ako Spring) sa objavil ako riešenie všetkých týchto komplikácií. Spring využíva rôzne nové techniky, ako napríklad aspektovo orientované programovanie (AOP), Plain Old Java objekt (POJO) a Dependency injection (DI) na vývoj korporátnych aplikácií, čím odstraňuje zložitosti spojené s ich vývojom pomocou EJB. Tento framework sa zameriava najmä na poskytovanie rôznych spôsobov, ktoré pomáhajú spravovať biznis objekty. V porovnaní s klasickými frameworkami Java a rozhraniami API (Application Programming Interfaces), ako sú napríklad Java database connectivity(JDBC), JavaServer Pages(JSP) a Java Servlet, výrazne uľahčil vývoj webových aplikácií.
Spring framework možno považovať za kolekciu čiastkových frameworkov, nazývaných aj vrstvy, ako napríklad Spring AOP. Spring Object-Relational Mapping (Spring ORM). Spring Web Flow a Spring Web MVC. Je to odľahčený aplikačný framework, používaný na vývoj enterprise aplikácií. Pri vytváraní webovej aplikácie môžete použiť ktorýkoľvek z týchto modulov samostatne. Moduly sa môžu aj zoskupiť, aby poskytovali lepšie funkcie vo webovej aplikácii. Framework Spring je voľne viazaný vďaka funkcii Dependency Injection. [1, 3, 4]

1.3 Výhody a nevýhody Springu

Následne si uvedieme výhody a nevýhody Spring Frameworku, ktoré treba zvážiť pred jeho použití.

Výhody:

1. **Použitie POJO**

Spring Framework používa Plain Old Java Object (POJO), pričom jeho použitie na vývoj aplikácie spočíva v tom, že nepotrebujete korporátny kontajner, ako je aplikačný server. Pomáha zbaviť sa konvenčných Enterprise Java Beans (EJB) tým, že vám umožňuje používať robustný servletový kontajner, ako je Tomcat. To robí zo Spring Frameworku odľahčený framework.

2. **Flexibilita pri konfigurácii Spring**
   Spring podporuje používanie konfigurácie XML, ako aj anotácií založených na jazyku Java na konfiguráciu Spring Beans. Preto poskytuje flexibilitu pri používaní ktorejkoľvek z nich pri vývoji korporátnej aplikácie.

3. **Nie je potrebný server**
   Ako viete, framework Spring poskytuje odľahčený kontajner. Ten je možné aktivovať bez použitia webového alebo aplikačného servera.

4. **Použitie Spring AOP**
   Modul Spring AOP prináša vývojárovi niekoľko výhod. Umožňuje vývojárovi mať inú kompilačnú jednotku alebo samostatný loader tried. Spolu s tým využíva IoC na dependency injection, čo umožňuje normálne konfigurovať aspekty.

5. **Nie je potrebné znovu vynaliezať**
   Jednou z hlavných výhod frameworku Spring Framework pri vývoji korporátnych aplikácií je to, že môžete využívať výhody zo Springu. Spring využíva technológie, ako sú JDK časovače, ORM rámce, Java EE atď. Takže vývojári sa nemusia učiť všetky tieto technológie alebo frameworky, aby mohli vyvíjať aplikácie.

6. **Využívanie modularity**
   Framework Spring poskytuje vývojárom modularitu. Pomáha im vybrať si, ktoré balíky alebo triedy môžu použiť alebo ignorovať. Vďaka množstvu tried a balíkov je pre vývojárov prínosom, že môžu bez problémov identifikovať a vyberať balíky alebo triedy.

7. **Jednoduchá testovateľnosť**
   Jedna z funkcií Spring Dependency injection pomáha pri zlepšovaní testovateľnosti. Zjednodušuje vkladanie testovacích údajov pomocou JavaBean POJO.

8. **Konzistentnosť správy transakcií**
   Spring Framework dokáže jednoducho škálovať lokálne aj globálne transakcie pomocou JTA. Je to vďaka konzistentnému rozhraniu správy transakcií.

9. **Dobre navrhnutý webový framework**
   Spring má dobre navrhnutý MVC framework, ktorý poskytuje skvelé alternatívne riešenie k staršiemu webovému frameworku.

10. **Inverzné riadenie a API**
    Spring Framework poskytuje kontrolu inverzie a API na prevod výnimiek vyhadzovaných JDBC, Hibernate na nekontrolované a konzistentné.

Nevýhody:

1. **Zložitosť frameworku Spring**
   Hlavnou kritikou, ktorú tento framework zaznamenal je jeho zložitosť a chýbajúce jasné zameranie, pretože má viac ako 2400 tried so 49 ďalšími nástrojmi, ktoré vývojárom komplikujú jeho používanie.

2. **Vysoká krivka učenia**
   Ak ste novým vývojárom v tejto oblasti, bolo by pomerne ťažké naučiť sa Spring Framework. Je to spôsobené novými programovými metódami a podrobným rozpracovaním každej z nich, ktoré si vyžaduje vývoj aplikácie.

3. **Tony paralelných mechanizmov**
   Tento bod je tiež uvedený v časti o výhodách, ale môže to byť nevýhodné vzhľadom na to, že to môže vývojárom poskytnúť veľa možností, a to spôsobuje zmätok. Zo strany vývojára je potrebné veľa pochopenia, aby vedel, ktoré metódy alebo triedy môžu byť užitočné pri konkrétnej príležitosti.

4. **Veľa XML v jazyku Spring**
   Ak ste niekedy pracovali s frameworkom Spring zistíte, že vývoj aplikácie Spring si vyžaduje veľa kódu XML.

5. **Nedostatok usmernení**
   V dokumentácii Spring nie sú pre vývojárov k dispozícii jasné usmernenia k niekoľkým témam, ako sú napríklad cross-site scripting útoky alebo útoky cross-site request forgery. Takisto má niekoľko bezpečnostných dier.

2 Spring MVC

Spring MVC je framework používajúci jazyk Java určený na vytváranie webových aplikácií. Slúži ako nadstavba na Spring, ktorá vytvorí komunikáciu pomocou HTTP protokolu. Framework používa návrhový vzor Model-View-Controller (MVC). Model slúži na zapuzdrenie dát aplikácie vo všeobecnosti pozostávajúcich z POJO. View vykresľuje údaje modelu a tiež generuje HTML výstup, ktorý dokáže interpretovať prehliadač klienta. Controller spracuje požiadavky používateľov a odovzdá ich do zobrazenia na vykreslenie. HTTP požiadavky a odpovede zabezpečuje DispatcherServlet. Jeho funkciou je odosielať prichádzajúce požiadavky, HTTP requesty, správnym obslužným programom špecifikovaným pomocou anotácií ako napríklad @Controller, @RestController či iných anotácií označených znakom @. [5, 6]

2.1 Výhody a nevýhody [7]

Výhody

1. **Jednoduché a rýchle nasadenie:** Na nasadenie na lightweight servlet sa používa kontajner, ktorý zabezpečuje jednoduchosť a rýchlosť.
2. **Rýchly a paralelný vývoj:** Vývoj je rýchly kvôli efektívnemu spôsobu prepájania prvkov prostredníctvom MVC, a taktiež paralelný v zmysle možnosti práce, na rôznych funkcionalitách bez toho, že by jedna priamo ovplyvnila druhú.
3. **Jednoduchšia aktualizácia aplikácie:** Aktualizácia je jednoduchšia pretože stačí iba zmeniť zdrojové súbory na serveri alebo inej platforme.
4. **Aplikácia je viac úrovňová:** Čo umožňuje rýchlejšie ladenie aplikácie.
5. **Škálovateľnosť:** Spring MVC zabezpečuje vysokú mieru škálovateľnosti.
6. **Oddeľovanie rámcov:** Aplikácia oddeľuje rámce pre jednoduchšie vykonávanie.

Nevýhody

1. **Vhodný len pre väčšie aplikácie:** Nie je vhodný pre malé aplikácie. Pretože negatívne ovplyvňuje výkon a dizajn aplikácie.
2. **Náročnosť:** Je náročný na pochopenie a tvorbu aplikácie, zvlášť pre začiatočníka.
3. **Zastaranosť:** Objemný so starším kódom, čo pri malých aplikáciach zbytočne zvyšuje náročnosť aplikácie.

2.2 Použitie

Tento framework používa funkcionálne programovanie čo znamená množstvo premenlivých objektov, anotácií, konfiguračných súborov XML a ďalších prostriedkov pre tvorbu aplikácie, čo zjednodušuje ako programovanie, tak aj nasadenie a zmenu. Tým sa stáva vhodným pre použitie na rozsiahle projekty, čo má ale za následok, že z počiatku čistý kód sa stane veľmi neprehľadným. [8]

Spring MVC Framework funguje nasledovne:

1. Všetky prichádzajúce požiadavky sú zachytené DispatcherServletom, ktorý funguje ako front controller.
2. DispatcherServlet potom získa záznam mapovania obsluhy zo súboru XML a odovzdá požiadavku kontroleru.
3. Objekt ModelAndView je vrátený kontrolerom.
4. DispatcherServlet skontroluje záznam resolveru view v súbore XML a zavolá príslušný view komponent.

Závislosť pre použitie tameplateovacieho jazyka timelief:

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

3 Spring Boot

Spring Boot je open source framework založený na jazyku Java, ktorý sa používa na vytvorenie mikroslužieb.

Spring Boot poskytuje vývojárom v jazyku Java dobrú platformu na vývoj samostatnej a produkčnej springovej aplikácie, ktorú stačí spustiť. Môžete začať s minimálnymi konfiguráciami bez toho, aby ste potrebovali celú konfiguráciu Spring. [9]

3.1 Aký je rozdiel medzi Spring a Spring Boot?

Pochopenie rozdielov medzi Springom a Spring Bootom je kľúčové pre rozhodnutie, či tento framework použiť, alebo nie. Spring Boot je postavený nad Spring frameworkom a dodáva sa s mnohými závislosťami, ktoré možno zapojiť do springovej aplikácie. Niektoré príklady sú Spring Kafka, Spring LDAP, Spring Web Services a Spring Security. Vývojári však musia každú "stavebnú tehlu"nakonfigurovať sami pomocou množstva konfiguračných súborov XML alebo anotácií. Spring framework sa zameriava na poskytovanie flexibility prostredníctvom funkcie dependency injection. Pomáha rýchlo injektovať požadované závislosti, ale aj vyvíjať aplikáciu voľne viazaným spôsobom.

Spring Boot je odľahčený framework. Pomáha pri voľnom spájaní závislostí a testovateľnosti. Jeho modulárna architektúra umožňuje vybrať si časti, ktoré potrebujete a izolovať ich. Má podporu pre konfiguráciu XML aj anotácií. Poskytuje abstrakciu na softvér ORM na vývoj logiky perzistencie ORM.Je kompatibilný s mnohými službami middleware. Podporuje framework JDBC, ktorý zvyšuje produktivitu a znižuje počet chýb. Na druhej strane Spring Boot sa zameriava na skrátenie dĺžky kódu a poskytuje jednoduchý spôsob spúšťania aplikácie Spring. [10]

Spring Boot je v podstate rozšírenie frameworku Spring, ktoré odstraňuje šablónovité konfigurácie potrebné na nastavenie aplikácie Spring. Preberá názorový pohľad na platformu Spring, čo otvára cestu k rýchlejšiemu a efektívnejšiemu vývojovému ekosystému. [11]

Funkcie v Spring Boot:

- Štartovacie závislosti na zjednodušenie zostavovania a konfigurácie aplikácie,
- Vstavaný server na zabránenie zložitosti pri nasadzovaní aplikácií,

- Metriky, kontrola stavu a externá konfigurácia,
- Automatická konfigurácia pre funkcie Spring - vždy, keď je to možné.

Spring Boot sa skladá z 2 slov Spring a Boot prvá časť slova spring predstavuje spring framework (framework, ktorý umožňuje písať korporátne java aplikácie), teraz druhá časť boot, ktorá je bootstrap, takže spojenie oboch slov. Spring Boot znamená niečo, čo vám umožňuje bootstrapovať spring aplikáciu od nuly.

Kľúčovým rozdielom alebo kľúčovou vlastnosťou Spring je dependency injection a v prípade Spring Boot je to autokonfigurácia. [12]

Čo je lepšie naučiť sa Spring alebo Spring Boot? Spring Boot obsahuje všetky funkcie štandardného frameworku Spring a zároveň výrazne uľahčuje vývoj aplikácií. V porovnaní so Springom môžete aplikáciu spustiť za podstatne kratší čas, pretože všetky atribúty Spring Bootu sú automaticky nakonfigurované. Odpoveďou je teda Spring Boot.

Prečo je Spring Boot najlepší? Spring Boot znamená niečo, čo vám umožní zaviesť springovú aplikáciu od základov. Framework Spring Boot sa používa najmä na vývoj rozhraní REST API.

Nahrádza Spring Boot Spring? Spring Boot nemá byť náhradou za Spring; skôr má urýchliť a uľahčiť prácu s ním. V dôsledku toho väčšina úprav potrebných na migráciu aplikácie súvisí s konfiguráciou. Naše na mieru vytvorené kontroléry a ďalšie komponenty zostanú z veľkej časti nezmenené.

Koľko trvá naučiť sa Spring? Odpoveď na túto otázku sa líši od človeka k človeku, pretože každý má svoju vlastnú rýchlosť, ale v priemere sa dá Spring naučiť približne za 4 týždne (mesiac).

Kedy by ste nemali používať Spring boot? Spring Boot nie je vhodný pre rozsiahle projekty, pretože vytvára veľa nepoužívaných závislostí, čo vedie k veľkým nasadzovacím súborom. [12]

3.2 Výhody a nevýhody

Spring Boot je navrhnutý tak, aby uľahčil používanie frameworku Spring. Spring poskytuje voľne viazané aplikácie, čo je samo o sebe skvelá vlastnosť. Keď však ide o niekoľko voľne previazaných blokov, sledovanie všetkých sa stáva únavnou a nevďačnou úlohou. Práve tu prichádza Spring Boot, ktorý vám pomôže udržať veci jednoduché vďaka nasledujúcim funkciám:

Výhody:

1. Je veľmi jednoduché integrovať aplikáciu Spring Boot s jej ekosystémom Spring, ako

	Spring	**Spring Boot**
Kde sa používa?	Spring framework je java EE framework, ktorý sa používa na vytváranie aplikácií.	Framework Spring Boot sa používa najmä na vývoj rozhraní REST API.
Kľúčová funkcia	Hlavnou alebo najdôležitejšou funkciou frameworku Spring je dependency injection.	Hlavnou alebo primárnou funkciou Spring Bootu je autokonfigurácia.
Prečo sa používa	Jeho cieľom je uľahčiť vývoj Java EE a umožniť vývojárom vyššiu produktivitu.	Spring Boot poskytuje frameworku Spring funkciu RAD (Rapid Application Development) na rýchlejší vývoj aplikácií.
Typ vývoja aplikácie	Framework Spring pomáha vytvoriť voľne previazanú aplikáciu.	Spring Boot pomáha vytvoriť samostatnú aplikáciu.
Závislosť od serverov	V Springu musíme na testovanie projektu explicitne nastaviť servery.	Spring Boot ponúka vstavané alebo zabudované servery, ako napríklad Tomcat a jetty.
Deskriptor nasadenia	Na spustenie Spring aplikácie je potrebný deskriptor nasadenia.	V aplikácii Spring Boot nie je deskriptor nasadenia potrebný.
Podpora databázy v pamäti	Spring neposkytuje podporu pre in-memory databázu.	Spring Boot poskytuje podporu pre in-memory databázu, napríklad H2.
Boilerplate kód	Spring framework vyžaduje príliš veľa riadkov kódu (boilerplate kód) aj pre minimálne úlohy.	Vyhnete sa boilerplate kódu, čo skracuje čas a zvyšuje produktivitu.
Konfigurácie	Musíte vytvoriť konfigurácie manuálne.	Existujú predvolené konfigurácie, ktoré umožňujú rýchlejšie zavádzanie.
Závislosti	Na vytvorenie webovej aplikácie je potrebných niekoľko závislostí.	Na druhej strane Spring Boot dokáže aplikáciu spustiť len s jednou závislosťou. Počas zostavovania sa vyžaduje niekoľko ďalších závislostí, ktoré sa štandardne pridávajú do finálneho archívu.
HTTP autentifikácia	Základné overenie HTTP slúži na povolenie bezpečnostných potvrdení. Označuje, že na povolenie zabezpečenia je potrebné povoliť niekoľko závislostí a konfigurácií. Spring vyžaduje na nastavenie zabezpečenia v aplikácii štandardné závislosti spring-security-web a spring-security-config. Ďalej musíme pridať triedu, ktorá rozširuje WebSecurityConfigurerAdapter a využíva anotáciu @EnableWebSecurity.	Spring Boot vyžaduje na svoju činnosť aj tieto závislosti, ale stačí definovať závislosť spring-boot-starter-security, pretože tá automaticky pridá všetky príslušné závislosti do classpath.
Testovanie	Testovanie v Spring je v porovnaní so Spring Boot náročné kvôli veľkému množstvu zdrojového kódu.	Testovanie v Spring Boot je jednoduchšie kvôli menšiemu množstvu zdrojového kódu.
XML konfigurácia	Vyžaduje sa konfigurácia XML.	Konfigurácia XML nie je potrebná.
Nástroje CLI	Spring neposkytuje žiadny CLI nástroj na vývoj a testovanie aplikácií.	Spring Boot poskytuje nástroj CLI na vývoj a testovanie aplikácií.
Pluginy	Spring neposkytuje žiadny plugin pre maven, Gradle atď.	Spring Boot poskytuje pluginy na kompiláciu pre Maven a Gradle.

sú Spring JDBC, Spring ORM, Spring Data, Spring Security atď.

2. Riadi sa prístupom "Opinionated Defaults Configuration". Pomáha vyhnúť sa všetkej manuálnej práci pri písaní boilerplate kódu, anotáciách a zložitých konfiguráciách XML.

3. Poskytuje množstvo pluginov na veľmi jednoduchú prácu s embedded a in-memory databázami.

4. Dodáva sa s integrovanými HTTP servermi, ako sú Jetty a Tomcat, na vývoj a testovanie webových aplikácií.

5. Poskytuje nástroj CLI (Command Line Interface) na vývoj a testovanie Spring Boot (Java alebo Groovy) aplikácií z príkazového riadku veľmi jednoducho a rýchlo.

6. Rýchly a jednoduchý vývoj aplikácií založených na technológii Spring.

7. Zjednodušuje závislosť a dodáva sa s vloženým Servlet kontajnerom.

8. Zjednodušená správa závislostí bez konfliktov verzií prostredníctvom štartovacích POM.

9. Môžeme rýchlo nastaviť a spustiť samostatné, webové aplikácie a mikroslužby za veľmi krátky čas.

10. Spring Boot poskytuje koncové body HTTP na prístup k interným údajom aplikácie, ako sú podrobné metriky, vnútorná práca aplikácie, stav atď.

Vďaka funkciám, ako je automatická konfigurácia, vám Spring Boot ušetrí starosti s programovaním a zbytočnou konfiguráciou. Framework Spring vám neponúka len funkcie ako dependency injection alebo spracovanie transakcií, ale slúži aj ako základ pre ďalšie frameworky Spring. Najlepším príkladom je Spring Boot. Spring Boot používa ako základ Spring Framework. Zjednodušuje závislosti Spring a spúšťa aplikácie priamo z príkazového riadka. Nevyžaduje ani externý aplikačný kontajner. Spring Boot pomáha externe ovládať a prispôsobovať komponenty aplikácie.

Spring Boot má veľa cenných funkcií, ktoré oslovujú vývojárov a používateľov (hoci používatelia málokedy vedia, prečo uprednostňujú digitálne produkty vytvorené pomocou Spring). Napriek tomu existujú aj nevýhody výberu Spring Boot.

Medzi najčastejšie sťažnosti alebo nevýhody vývojárov patria napríklad:

- Najväčšou výzvou, s ktorou sa mnohí vývojári stretávajú pri používaní Spring Boot, je nedostatok kontroly nad systémom. "Opinionated Defaults Configuration" – Opinionated stratégia alebo autokonfigurácia, ktorá je výhodou, však inštaluje, respektíve vytvára, mnoho ďalších závislostí, o ktorých predpokladá, že ich systém bude potrebovať. Inštaláciou všetkých týchto dodatočných závislostí, ktoré sa však vôbec nemusia všetky využívať, sa stane, že veľkosť súboru na nasadenie bude v aplikáciách Spring Boot veľmi veľká.

- Ďalším problémom, s ktorým sa vývojári stretávajú je to, že môže byť pomerne zložité a časovo náročne konvertovať či aktualizovať staršie systémy alebo existujúce Spring projekty na aplikáciu s použitím Spring Boot. Na riešenie tejto nevýhody je odporúčane použiť nástroj, ako je Spring Boot CLI (Command Line Interface), ktorý nám vie pomôže konvertovať starší kód.

- Spring Boot je nevhodný pre rozsiahle projekty. Framework bol vytvorený ako agilný a odľahčený, zameraný hlavne na API a mikroslužby, preto by sa nemal používať na vytváranie veľkých monolitických aplikácii, v čom sa zhodnú aj viacerí vývojári. [13]

Niektoré ďalšie nevýhody sú:

- Ak ste nikdy predtým nepracovali so Springom a chcete sa naučiť o proxy, dependency injection a AOP programovaní, neodporúča sa začať so Spring Bootom, pretože nepokrýva väčšinu týchto detailov.

- Modifikácie nie sú v Spring Boote jednoduchou úlohou. Ak nemáte silné znalosti o systémoch a histórii Springu, nemôžete modifikovať alebo troubleshootovať problémy so Spring Boot.

- Ak nepoznáte iné projekty ekosystému Spring, ako napríklad Spring Security, Spring AMQP, Spring Integration atď), pri ich používaní so Spring Boot vám budú chýbať mnohé koncepty, ktoré by ste pochopili, keby ste ich začali používať nezávisle.

V záver ako zhrnutie môžeme skonštatovať, že Spring Boot bol navrhnutý tak, aby pomohol vývojárom skrátiť čas potrebný na vývoj a zlepšiť výkon tým, že poskytuje predvolené nastavenie a integračné testy. Ak chcete rýchlo začať vývoj Java aplikácie, môžete jednoducho pracovať s predvolenými hodnotami a vynechať XML konfiguráciu.

Spring Boot je jednoducho rozšírením samotného Springu, aby bol vývoj, testovanie a nasadenie pohodlnejšie.

3.3 Použitie

V tejto podkapitole si zhrnieme dôležité body v prípade použitia Spring Bootu.

Spring Boot automaticky konfiguruje aplikáciu na základe závislostí, ktoré ste pridali do projektu pomocou anotácie @EnableAutoConfiguration. Napríklad, ak je databáza MySQL na vašej classpath, ale nenakonfigurovali ste žiadne databázové pripojenie, potom Spring Boot automaticky nakonfiguruje databázu v pamäti.

Vstupným bodom spring boot aplikácie je trieda obsahujúca anotáciu @SpringBootApplication a metódu main. Spring Boot automaticky skenuje všetky komponenty zahrnuté v projekte pomocou anotácie @ComponentScan.

Štartér systému Spring Boot: Správa závislostí je pre veľké projekty náročná úloha. Spring Boot rieši tento problém tým, že poskytuje sadu závislostí pre pohodlie vývojárov.

Ak napríklad chcete používať Spring a JPA na prístup k databáze, stačí, ak do projektu zahrniete závislosť spring-boot-starter-data-jpa.

Všetky Spring Boot startery sa riadia rovnakým vzorom pomenovania spring-boot-starter- *, kde * označuje, že ide o typ aplikácie. [9]

Príklady:

Webová závislosť Spring Boot Starter sa používa na zápis Rest endpointov:

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

Závislosť Spring Boot Starter Security sa používa pre Spring Security:

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Prečo Spring Boot?

Spring Boot si môžete vybrať kvôli funkciám a výhodám, ktoré ponúka:

- Poskytuje flexibilný spôsob konfigurácie Java Beans, konfigurácií XML a databázových transakcií.

- Poskytuje výkonné dávkové spracovanie a spravuje koncové body REST.
- V Spring Boot je všetko automaticky nakonfigurované; nie je potrebná žiadna manuálna konfigurácia.
- Ponúka Spring aplikáciu založenú na anotáciách.
- Uľahčuje správu závislostí.
- Obsahuje vstavaný kontajner servletov.

4 Porovnanie rozdielov

So Spring Boot sa pracuje oveľa pohodlnejšie ako so Spring MVC, ale výber môže závisieť aj od typu aplikácie. Oba tieto frameworky majú svoje výhody, napríklad Spring MVC je dobrou voľbou na vývoj modulárnych webových aplikácií a vynecháva prekrývajúcu sa úlohu spracovania požiadaviek HTTP. Na druhej strane Spring Boot pomáha šetriť čas a znižuje námahu tým, že znižuje počet konfigurácií. Jednou spoločnou vecou, ktorá môže spôsobiť zmätok, je používanie anotácií na vývoj aplikácie v oboch typoch technológií. Ďalšie porovnanie je prehľadne zhrnuté v nasledovnej tabuľke. [6, 14]

Spring MVC	**Spring Boot**
Spring MVC je framework založený na MVC, ktorý sa používa na vytváranie webových aplikácií.	Spring Boot je framework na tvorbu REST API. Používa sa na vytváranie samostatných webových springových aplikácií.
Tento framework vyžaduje veľa konfigurovania, napríklad konfigurácie DispatcherServlet a konfigurácie View Resolveru.	Spring Boot spracúva konfiguráciu automaticky pomocou funkcie automatickej konfigurácie.
Rýchlosť vývoja sa znižuje, pretože je potrebné porozumieť závislostiam doplnkov.	Kratší čas vývoja, produktivita sa zvyšuje.
Každú závislosť je potrebné špecifikovať samostatne, aby sa funkcie spustili.	Spring Booth má koncept štartérov, ktorý po pridaní do classpath poskytne všetky závislosti potrebné na vývoj webovej aplikácie.
Spring MVC vyžaduje veľa manuálnych konfigurácií pre JAR balenie.	Spring Boot umožňuje vstavanému serveru spustiť funkciu samostatne.
Pre framework Spring MVC sa vyžaduje deskriptor nasadenia.	Pre framework Spring Boot sa nevyžaduje deskriptor nasadenia.
Štyri komponenty Spring MVC sú Model, View, Controller a Front Controller (trieda DispatcherServlet).	Štyri vrstvy Spring Bootu sú prezentačná vrstva, biznis vrstva, perzistenčná vrstva a databázová vrstva.
Spring MVC je navrhnutý len na vývoj dynamických webových stránok a webových služieb RESTful.	Spring Boot umožňuje vytvárať aj rôzne iné druhy aplikácií.
Neposkytuje výkonné batch spracovanie.	Poskytuje výkonné batch spracovanie.
Každá dependency je v Spring MVC špecifikovaná samostatne.	V Spring Boot sú všetky dependencie zabalené do jedného celku.

5 Softvér

Na ukážku sme si pripravili jednoduchý softvér, ktorý nesie názov Insurance system. Úlohou tohto softvéru je ukladať údaje o ľuďoch, ktorý si uzavreli nejaké poistenie a taktiež aj ukladanie rôznych typov poistných zmlúv. Systém umožňuje ukladať nové dáta, modifikovať dáta, mazať dáta a zobrazovať všetky potrebné dáta.

Softvér sme vyhotovili v dvoch verziách. Pri jednej verzií sme vytvorili softvér s použitím Spring MVC. V druhej verzií sme pripravili ten istý softvér s použitím Spring Boot, ktorý poskytuje REST API pre všetky funkcionality.

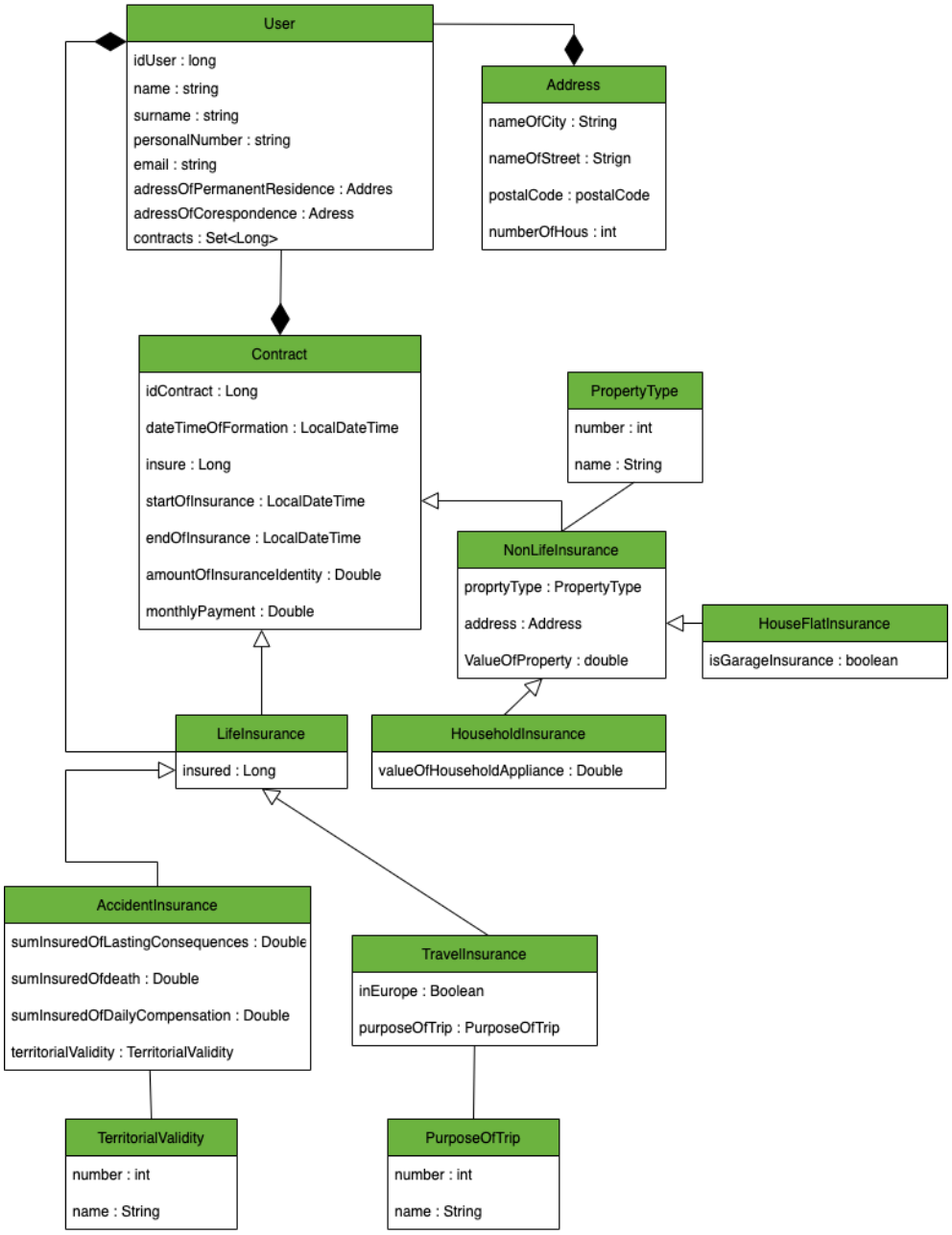
5.1 Architektúra softvéru

V oboch verziách nášho softvéru používame architektúru MVC.

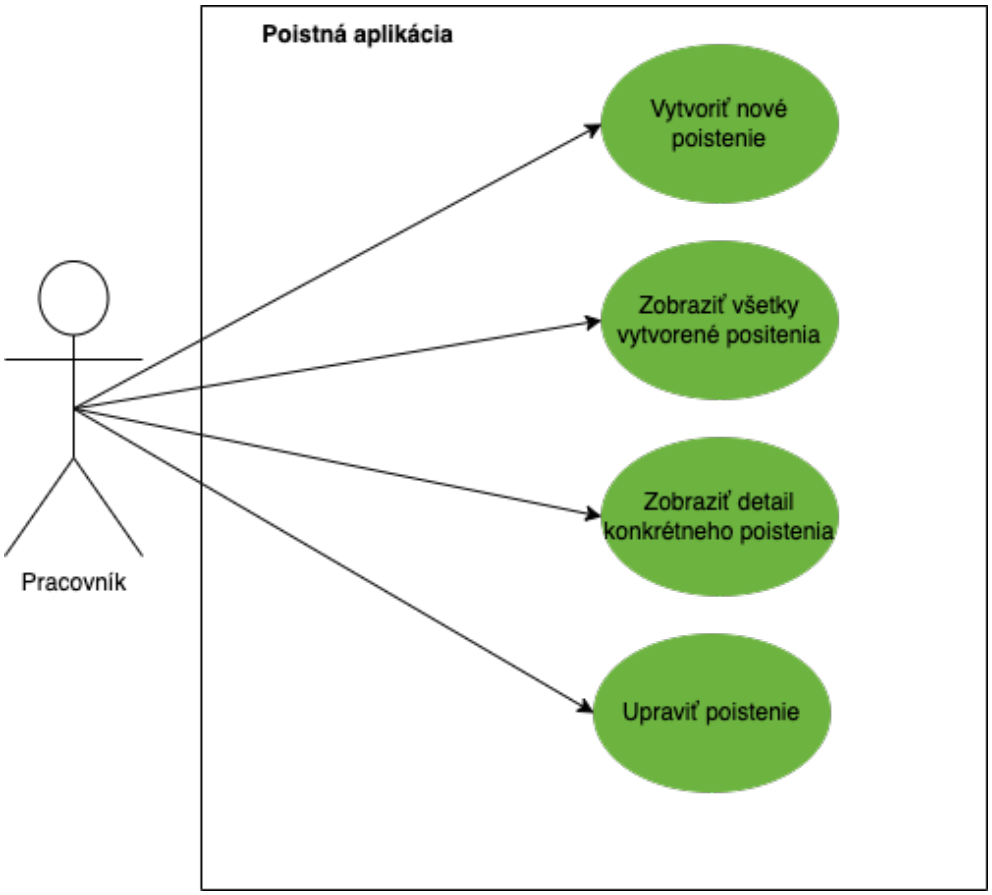
Vo vrstve model sa nachádzajú naše Doménové triedy, ktoré nesú dáta o používateľoch a zmluvách. V tejto vrstve sa nachádzajú aj servisné triedy, ktoré sa starajú o biznis logiku našej aplikácie. Vo vrstve controller sa nachádzajú triedy, ktoré sa starajú o komunikáciu medzi modelom a view vrstvou. V modifikácií aplikácie, kde používame Spring MVC máme aj view vrstvu v ktorej sa nachádzajú html súbory, ktoré slúžia na vykresľovanie dát a interakciu s užívateľom. V druhej modifikácií aplikácie, ktorá poskytuje REST API a je zameraná na Spring Boot, vrstva view chýba. Je to z dôvodu jednoduchosti softvéru.

Aj štruktúra usporiadania nášho kódu je prispôsobená tejto architektúre. V priečinkoch domain a service sa nachádzajú zdrojové kódy, ktoré tvoria model aplikácie. Priečinok web tvorí controller aplikácie.

Obidve verzie aplikácie sú rovnaké. Líšia sa iba v kontroleroch. Pri Spring MVC používame template language Thymeleaf. Na strane servera sa vygeneruje HTML súbor so všetkými potrebnými informáciami, ktorý sa následne užívateľovi načíta. Pri Spring Boot verzií kontroler posiela užívateľovi všetky potrebné dáta vo formáte json súboru a záleží už na klientskej aplikácií ako tieto dáta budú zobrazené užívateľovi.



Obr. 1: Class diagram popisujúci dátový model aplikácie



Obr. 2: Use case diagram

5.2 Použitie

- Ak vytvárate konzolovú aplikáciu, popíšte vstupné argumenty.
- Popíšte príklady používania vášho softvéru.
- Opíšte výstupy, ak váš softvér vytvára nejaké výstupy.
- Opíšte varovania a známe chyby, ak váš softvér obsahuje nejaké chyby.
- Opíšte niektoré zaujímavé prístupy použité na riešenie programátorských problémov, s ktorými ste sa stretli počas implementácie softvéru (ak je niečo zaujímavé).

Záver

Oba frameworky, Spring MVC ako aj Spring Boot majú svoje výhody ako aj nevýhody. Tu je dôležité si uvedomiť, že pravidlo pre výber jedného či druhého frameworku záleži na tom, čo má byť výstupnou aplikáciou. Ak by sa jednalo o webovú aplikáciu, je najvhodnejšou voľbou Spring MVC a to zvlášť v tom prípade, ak táto aplikácia vyžaduje modulárnosť. Na truhej strane Spring Boot nie je vhodný pre takýto druh aplikácie nakoľko jeho silnou stránkou je automatická konfigurácia, ktorá často pri webových aplikácia nestačí na dosiahnutie mieneného výsledku. Preto je Spring Boot vhodný práve na aplikácie, kde sa chceme vyhnúť nadmernému konfigurovaniu napríklad servletového servera či konfigurácii komunikácie z databázov a podobne. Ak by sme sa teda pozreli na tieto dva frameworky s pohľadu ich použitia Spring Boot je jednoduchší a použiteľnejší na širšiu škálu projektov. Oproti nemu Spring MVC je robustnejší no to hlavne kvôli jeho pripravenosti pre použitie na webové aplikácie a všetky problémy s tým súvisiace. Nedá sa tak povedať ktorý z nich je objektívne lepší všeobecne, no pre konkrétne použitie si vieme jasne vybrať správny nástroj.

Zoznam použitej literatúry

1. *Spring Framework - Overview*. Dostupné tiež z: https://www.tutorialspoint.com/spring/spring_overview.htm.
2. *Introduction to Spring Framework*. Dostupné tiež z: https://www.geeksforgeeks.org/introduction-to-spring-framework/.
3. *Spring Framework*. Dostupné tiež z: https://www.techtarget.com/searchapparchitecture/definition/Spring-Framework.
4. *What is Spring Framework? An Unorthodox Guide*. Dostupné tiež z: https://www.marcobehler.com/guides/spring-framework.
5. *Spring – MVC Framework*. Dostupné tiež z: https://www.geeksforgeeks.org/spring-mvc-framework/.
6. *Difference Between Spring MVC and Spring Boot*. Dostupné tiež z: https://www.interviewbit.com/blog/difference-between-spring-mvc-and-spring-boot/.
7. *What is the Difference Between Spring MVC Spring Boot?* Dostupné tiež z: https://kodytechnolab.com/spring-boot-vs-spring-mvc.
8. *Web MVC framework*. Dostupné tiež z: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html.
9. *Spring Boot - Introduction*. Dostupné tiež z: https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm.
10. *What Is Spring Boot?* Dostupné tiež z: https://stackify.com/what-is-spring-boot/.
11. *A Comparison Between Spring and Spring Boot*. Dostupné tiež z: https://www.baeldung.com/spring-vs-spring-boot.
12. *Spring vs Spring Boot: Know The Difference*. Dostupné tiež z: https://www.interviewbit.com/blog/spring-vs-spring-boot/.
13. *Pros and Cons of Using Spring Boot*. Dostupné tiež z: https://bambooagile.eu/insights/pros-and-cons-of-using-spring-boot/.
14. *Spring vs. Spring Boot vs. Spring MVC*. Dostupné tiež z: https://www.javatpoint.com/spring-vs-spring-boot-vs-spring-mvc.