Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Srovnávací analýza webových frameworků v jazyce Java

Bakalářská práce

Autor: Jan Zahradník
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Michal Macinka

Odborný konzultant: Ing. Jan Vondrouš, Ing. Jiří Novák

Hradec Králové Srpen 2020

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 11.8.2020 Jan Zahradník

Poděkování:

Děkuji vedoucímu bakalářské práce Ing. Michalu Macinkovi za metodické vedení práce a pomoc při návrhu testovací aplikace. Dále bych chtěl poděkovat Ing. Janu Vondroušovi a Ing. Jiřímu Novákovi za odborné znalosti a pomoc v konkrétních webových frameworcích.

Anotace

Cílem bakalářské práce bylo představit vybrané webové frameworky v programovacím jazyce Java a jejich následné testování.

V první části byl popsán a definován samotný pojem framework a webový framework. Byla krátce představena jejich historie. Byly vysvětleny typy architektury a princip webových frameworků. Představeny byly vybrané webové frameworky a jejich výhody a užití.

V druhé části byla ve vybraných webových frameworcích navržena jednoduchá testovací aplikace. Ta představovala jednoduchý rezervační systém do kina. Aplikace byla podrobně popsána a představena. Byly popsány hlavní rozdíly v návrhu aplikace v jednotlivých frameworcích.

V poslední části byla aplikace podrobena testováním výkonnosti a zátěže. Účelem bylo získat odpovídající výsledky a následně je porovnat. Otázkou také byla jejich pohodlnost při programování.

Annotation

Title: Comparative Analysis of Web Frameworks in Java

Goal of bachelor theses was to introduce selected web frameworks in Java and their subsequent testing.

In the first part framework and web framework concept was described. Their history was briefly introduced. Architecture and principals of web frameworks were explained. Benefits and uses of selected web frameworks were described.

In the second part simple testing application was designed in selected web frameworks. Designed application was simple cinema reservation system. Application was properly described and presented. Main differences in each framework for designed application were presented.

In the last part application was tested in performance testing. Purpose of that was to gather comparable results. Another question was their programming convenience.

Obsah

1 Úvod

Webové frameworky jsou od počátku 21. století rychle rostoucím a velice populárním nástrojem pro tvorbu webových aplikací. Počátky vývoje těchto nástrojů se datují již do první poloviny 90. let 20. století [1]. Od začátku 21. století lze hovořit o plně integrovaných webových frameworcích, které plně nahrazují předchozí návrhy pro tvorbu webových aplikací [1].

V principu se podle použití rozdělují na back-end a front-end. Všechny probírané frameworky v této práci jsou použitím zaměřeny na back-end v programovacím jazyce Java. Back-end nebo „server-side“ je v podstatě to, jak aplikace funguje, aktualizuje se a mění se. To se týká všeho, co uživatel v prohlížeči nevidí, jako jsou databáze a servery [2]. Moderní frameworky zjednodušují v aplikaci proces přístupu k datům v databázi.

Většina z nich využívá principu REST (Representational State Transfer). Tento způsob umožňuje přístup k datům pomocí standardních HTTP požadavků. Moderní frameworky pro vývoj server-side aplikací pomáhají vytváření REST rozhraní tím, že dokáží nadefinovat patřičné procedury pro všechny potřebné metody, takže vytvoření vlastního RESTful rozhraní je opravdu snadné [3].

Podle statistik o nejužívanějších webových frameworcích v programovacím jazyce Java od hotframeworks.com je nepoužívanějším Spring framework, jehož popularita dokonce narůstá a předpokládá se, že v jazyce Java se dnes jedná o standard. Apache Wicket je z hlediska popularity nejustálenějším a jeho popularita zůstává stejná. Apache Struts 2 se svojí popularitou sice spadá do první desítky stále nejpoužívanějších, ale jeho popularita klesá.

Otázkou tedy zůstává, zda si tyto vybrané zaslouží dané místo na žebříčku popularity z hlediska jejich výkonnosti.

2 Cíl práce

Smyslem této bakalářské práce je obeznámit čtenáře se základními principy webových frameworků jejich fungováním, historií, architekturou a představením vybraných pro programovací jazyk Java. Budou vybrány tři frameworky na základě jejich popularity. V nich bude navržena jednoduchá aplikace a poté budou otestovány a porovnány jejich výsledky.

Cílem tedy bude navržení jednoduchých aplikací ve vybraných frameworcích a jejich analýza. Otázkami jsou pohodlnosti při navrhování ve vybraných frameworcích, a hlavně testování jejich výkonnosti. Je Spring framework opravdu nejlepší? A o kolik se výkonnostně liší oproti konzistentnímu Apache Wicket, či staršímu a upadajícímu Apache Struts 2.

Analýza bude provedena formou testování výkonnosti daných frameworků. Naměřené testovací výsledky se budou vyhodnocovat vzájemně vůči každému frameworku. Na základě výsledků bude zřetelný nejvýkonnější webový framework v programovacím jazyce Java.

3 Webové frameworky

Webové frameworky jsou v dnešní době velice populárním nástrojem pro tvorbu webových aplikací. Obsahují množství knihoven specificky zaměřené na vývoj webové aplikace. Tím značně urychluji programátorovi práci a řeší za něj obecnější a známé problémy při tvorbě.

3.1 Framework

Frameworků existuje celá řada a v zásadě se rozdělují podle jejich účelu. Existují frameworky zaměřené na testování, pro webové aplikace, nebo jako nadstavba v daném programovacím jazyce. V principu však vždy slouží k lepšímu pochopení problematiky a usnadňují práci.

3.1.1 Definice, principy a účel

Framework znamená v českém překladu aplikační rámec a je to softwarová struktura, která v principu slouží k ulehčení programování aplikací, protože poskytují pro programátora podporu při řešení a vývoji softwarových a jiných projektů.

Spousta moderních frameworků obsahuje obrovské množství knihoven, které implementují spousty užitečných funkcí [4]. Tyto knihovny usnadňují programátorům práci, protože jsou v nich nadefinované např. metody, které oni sami nemusí vymýšlet, ale mohou je pouze využít. Tím se značně urychlí práce. Programátor se tedy při vývoji svého softwaru nemusí soustředit na typické problémy v dané oblasti, které jsou za něj vyřešeny. Tímto řešením může být i mimo jiné, zajištění přístupu k databázi. Programátor se tedy zabývá pouze řešením specifických problémů svého projektu. Jiné neobsahují tolik knihoven, ale díky svému návrhu se dají rozšířit o nové funkce [4]. Účelem frameworku je tedy poskytnout řešení pro známé a typické problémy v dané oblasti vývoje.

Problémem je, že programátor používá cizí postupy a kódy. To někdy může vést ke zmatku a celkovému nepochopení. Spousta populárních frameworků obsahuje rozsáhlou dokumentaci, která usnadňuje orientaci v daném frameworku. Pokud je dokumentace nedostatečná nebo špatně vytvořená, tak je programátor

nucen zdlouhavě v dokumentaci vyhledávat parametry funkcí a jiné problémy a práce se zpomaluje [4]. Při pochopení principu, metod a řešení, se ovšem práce značně zrychluje a ve výsledku tedy programátorovi ušetří spoustu času a problémů, kterými by se jinak musel zabývat.

Webové frameworky jsou frameworky, které se specializují na oblast vývoje webových aplikací. Krom podpůrných knihoven poskytují například knihovny pro připojení k databázi, tzv. session management a v první řadě podporují znovu použitelnost kódu [5]. Session management označuje proces bezpečného zacházení s tzv. session (relace, též seance). Důležitou součástí je komunikace ověřených uživatelů skrz zabezpečenou, zašifrovanou síť. Webové stránky a prohlížeče používají HTTP ke komunikaci a session je série HTTP požadavku a transakcí od jednoho uživatele, či entity [6]. Webové frameworky jsou často zaměřeny na dynamické webové stránky, ale lze je aplikovat i na statické webové stránky.
Příklad základní aplikace ve Spring Frameworku:

```
// importace Spring Frameworku do projektu
import org.springframework.context.annotation.Bean;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootAplication
public class SpringFile {
//hlavní spouštěcí metoda
    public static void main(String[] args){
    SpringBootApplication.run(SpringFile.class,args);
    }
// použití bean anotace => vytvoření beany
    @Bean
    public int Calculator (){
        return 0;
    }
```

Ukázka kódu 1: Ukázka kódu ve Spring frameworku. Zdroj: [autor]

3.1.2 Historie webových frameworků

Vývoj se datuje k počátkům World Wide Webu (WWW). V roce 1993 vznikl tzv. Common Gateway Interface (CGI) protokol, který stanovil standard komunikace mezi externími aplikacemi a webovými servery s účelem poskytnutí dynamických webových stránek, které by operovaly s uživatelskými vstupy [1]. Implementace z roku 1993 měla ovšem obrácený efekt na složitosti načítání na straně serveru, protože každý nový požadavek vytvářel další proces, čímž se zpomaloval celkový průběh komunikace mezi uživatelem, komunikujícím skrz externí aplikaci a webovým serverem. Pozdější implementace využívá trvalejších procesů pro co nejefektivnější komunikaci a co nejmenším využitím zdrojů web serveru, což způsobilo značné zrychlení.

V roce 1995 se objevují první plně integrované serverově jazykově orientované prostředí a jsou představeny jazyky určené pro tvorbu dynamických webových stránek jako je PHP, ColdFusion apod.

Ke konci 20. století se objevují plně integrované webové frameworky, které nabízejí a podporují řadu knihoven a shromažďují je do jednotného nástroje pro vývoj webových aplikací. Příkladem jsou ASP.NET, Java EE, Ruby On Rails atd. [7].

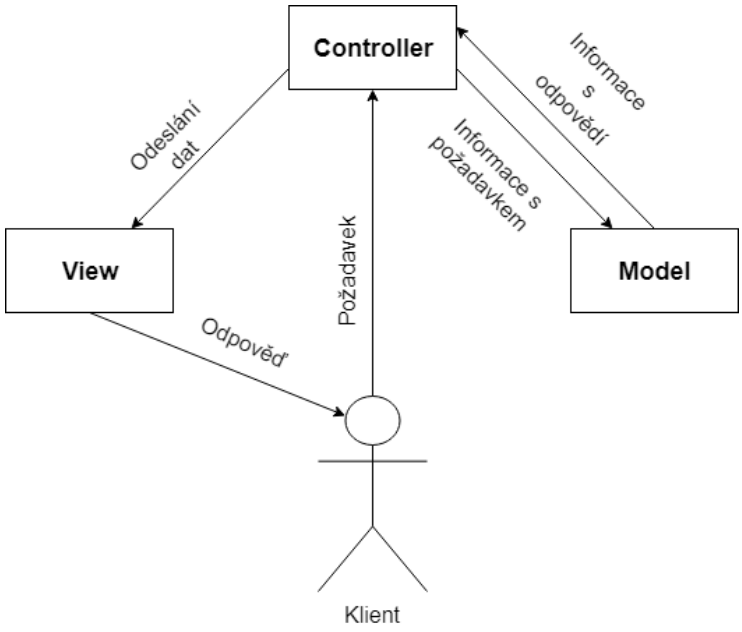
3.2 Architektura webových frameworků

Architektura, též architektonický vzor, tvoří základ pro všechny webové frameworky. Architektonické vzory zpravidla pracují se třemi základními vrstvami (databáze, řízení dat, uživatelské rozhraní) aplikace a snaží se o jejich efektivní spolupráci [8]. Nejpoužívanějším architektonickým vzorem je MVC (Model-View-Controller), který společně s třístupňovou architekturou využívá těchto vrstev.

3.2.1 MVC

V dnešní době více než 80 % frameworků používá architektonický vzor MVC (Model-View-Controller) [9]. Ten v principu rozděluje datový model aplikace (model), uživatelské rozhraní (view) a řídící logiku (controller) do tří nezávislých komponent. Tím umožňuje, že modifikace jedné z komponent má minimální vliv na zbylé dvě.

Architektonický vzor MVC poskytuje možnost paralelního vývoje na různých komponentách aplikace bez vzájemného blokování. Model je reprezentací dat, které se uživateli zobrazují [10]. Model obsahuje třídy, které přestavují jednotlivé částí stránek a stránky aplikace. View obsahuje veškerý HTML, CSS, Javascript kód, přes který je aplikace uživateli zobrazována a Controller představuje veškerou logiku aplikace. Chová se jako rozhraní, přes které spolu komunikují vrstvy Model a View [10]. Pokud si uživatel chce zobrazit svůj profil, vytvoří se požadavek, který se zašle na Controller. Ten provede odpovídající kód a zašle požadavek na Model, který ho provede a vrátí zpět požadovaná data. Ta jsou uživateli zpětně skrz Controller vrácena na View, kde je zobrazena odpověď, v tomto případě se zobrazí profil. Takto se práce při vývoji dělí i u třístupňové architektury. Následující Obrázek 1: Ilustrace architektonického vzoru MVC. Zdroj: [10] ilustruje architekturu MVC:



Obrázek 1: Ilustrace architektonického vzoru MVC. Zdroj: [10]

3.2.2 Třístupňová architektura

Třístupňová architektura (Three-tier architecture) je založena na třech základních vrstvách: prezentační, aplikace, databáze. Aplikační vrstva obsahuje veškerou logiku a komunikuje s klientem přes HTTP požadavky. Prezentační vrstva poté tvoří graficky znázorněné uživatelské rozhraní přístupné uživateli skrz

prohlížeč [11]. Tato architektura není zdaleka tak využívána jako předchozí návrhový vzor MVC.

Jednotlivé vrstvy se dají rozdělit i podle toho, kde vykonávají svojí funkci. Prezentační vrstva probíhá na počítači, laptopu, tabletu nebo chytrém telefonu přes webový prohlížeč či aplikaci. Aplikační vrstva by poté zahrnovala relační databázi nebo jiné typy databází, hostovaných buď přes cloud technologii nebo tzv. on-premise [11].

Takto rozdělená aplikace je při vývoji efektivnější, neboť se vývojáři mohou specializovat na vývoj jednotlivých vrstev, které znají a provádějí je efektivněji než nespecializovaný vývojář. Výhody této architektury jsou, jako u návrhového vzoru MVC, rychlost vývoje, výkon, dostupnost a škálovatelnost.

3.3 Webové frameworky v programovacím jazyce Java

Webové frameworky v programovacím jazyce Java jsou z většiny postaveny na návrhovém vzoru MVC, jsou multiplatformní (fungující na více operačních systémech, jako Microsoft Windows, mac OS atd.), open-source (s dostupným zdrojovým kódem) a obsahují podrobnou dokumentaci. Odlišují se od sebe v zásadě jinou implementací návrhového vzoru MVC.

3.3.1 Spring framework

Obrázek 2:Logo Spring Framework. Zdroj:[12]

Spring Framework je velmi populární a oblíbený k vývoji J2EE aplikací. Je to multiplatformní a open-source nástroj pro tvorbu aplikací. Poslední verze k datu 9. 6. 2020 je Spring Framework 5.2.7 a je spravován společností Pivotal.

Spring Framework je postaven na návrhovém vzoru Inversion of Control jako tzv. IoC kontejner. V tradičním programování je kód zpracován skrze knihovny. Tím dochází k provedení požadovaných procesů. Spring Framework naproti tomu zpracovává sám opačným směrem vytvořený kód [13]. Tento IoC kontejner je poté zodpovědný za životní cyklus specifických objektů v aplikaci vytvořených. To může představovat vytvoření těchto objektů, zavolání jejich inicializačních metod a spojování objektů. Takto vytvořené objekty se nazývají tzv. beans, což jsou v podstatě objekty, které obsahují více objektů najednou. Konfigurace tohoto kontejneru lze přizpůsobit pomocí XML souborů nebo anotacemi

Spring Framework od verze 2.5 začal využívat anotace. Jejich deklaracemi můžeme konfigurovat vztahy mezi objekty a chování webové aplikace napsané s využitím Spring Frameworku. Nejzákladnější anotací je `@Bean`, která vždy spolupracuje s anotací `@Configuration` a slouží k deklaraci komponent, již zmiňovaných tzv. beans. Další klíčovou a často využívanou anotací je `@Autowired`, která slouží k propojování jednotlivých částí aplikace a zajišťuje inicializaci komponent s anotací `@Beans` [14].

Objekty tak ve Spring Frameworku mohou být definovány a zpracovány pomocí tzv. dependency lookup, což znamená, že zavolání objektů v kontejneru může být provedeno pomocí jejich specifického jména či typu. Také lze využít tradičního dependency injection, kde kontejner předává objekty ostatním pomocí konstruktoru, vlastností či specifických metod. Kontejner Spring Frameworku tak poskytuje trvalý mechanismus ke konfiguraci webové aplikaci a integraci v prostředí využívající Javu

3.3.2 Apache Struts 2

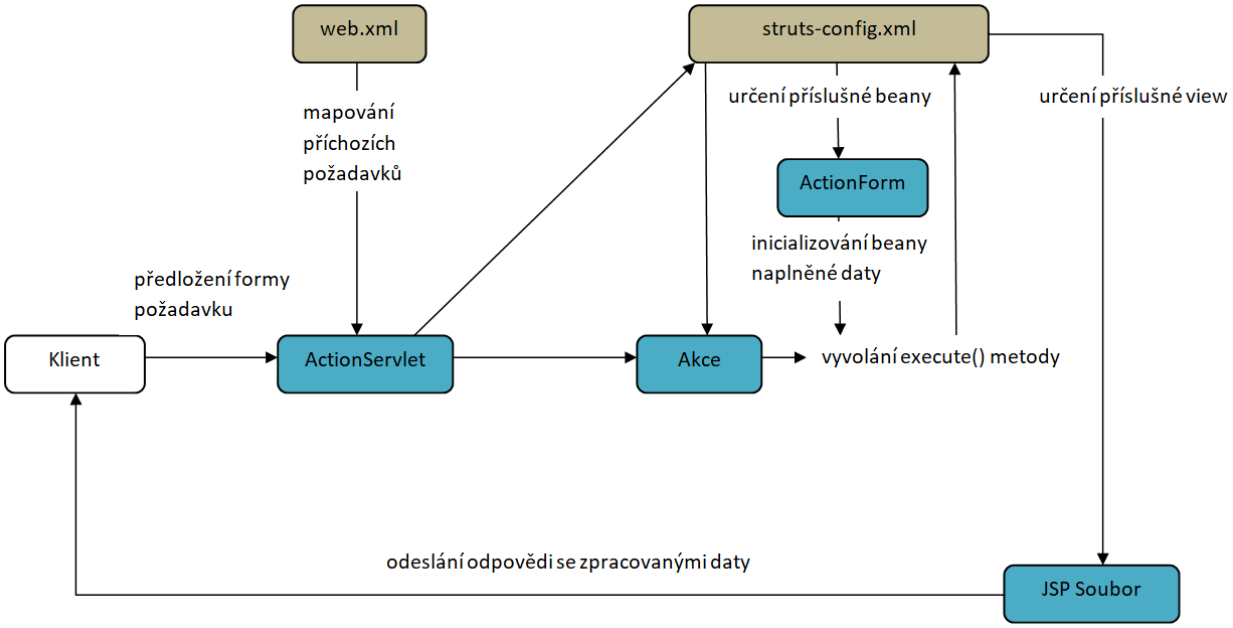
Obrázek 3: Logo Apache Struts. Zdroj:[15]

Struts nebo také Apache Struts ve své aktuální verzi Apache Struts 2 je navržen pro prostředí Java. Stejně jako Spring Framework se zde jedná o multiplatformní, open-source nástroj pro tvorbu webových aplikací. Poslední verze k datu 29. 11. 2019 je Apache Struts 2.5.22 a je spravován společností Apache Software Foundation. Podobně jako Spring Framework využívá Apache Struts 2 anotace tzv. JavaBeans třídy.

Apache Struts 2 podobně jako řada jiných je založen na návrhovém vzoru MVC, čímž aplikací dělí do tří vrstev. Prezentační vrstva zodpovědná za zobrazení a interakci s uživatelem je implementována pomocí Java Server Pages (JSP), což umožňuje vytvářet dynamické stránky pomocí HTML a jazyku Java. Řídící vrstva je reprezentována pomocí tzv. ActionServletu, který zachycuje požadavky od uživatele a rozhoduje o přidělení příslušného View z prezentační vrstvy na navrácení dat. V modelové vrstvě, kde se zajišťuje veškerá business logika a komunikace s úložištěm dat, se používají JavaBeans třídy s anotacemi. Zajištění vzájemného provázání vrstev je v Apache Struts 2 řešeno v konfiguračním souboru `struts-config.xml` [16].

Když vývojáři vyvíjejí svoji webovou aplikaci s webovým frameworkem Apache Struts 2 tak jim poskytuje řídící servlet, který je automaticky registrován v souboru `web.xml`. Řídící servlet využívá soubor `struts-config.xml` k mapování příchozích požadavků (request), Struts Action objektu a inicializování ActionForm objektů s akcí na dočasné uložení dat. Action objekt zpracuje požadavek použitím jeho `execute()` metody s použitím dočasně uložených dat. Poté, co Action objekt

zpracuje požadavek, uloží nová data do beany a odešle odpověď na příslušné view [16]. Následující Obrázek 4 ilustruje schéma fungování Apache Struts 2:



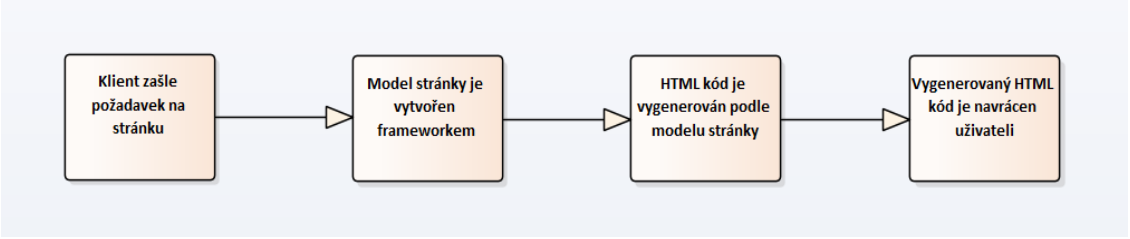
Obrázek 4: Schéma fungování Apache Struts 2 frameworku. Zdroj: [16]

3.3.3 Apache Wicket

Obrázek 5: Logo Apache Wicket frameworku. Zdroj: [17]

Apache Wicket je webový framework pro aplikace v prostředí Java. Stejně jako u předcházejících se jedná o multiplatformní, open-source nástroj, který vznikl v roce 2004. Poslední verze k datu 13. 10. 2019 je Apache Wicket 8.6.1, který je spravován společností Apache Software Foundation.

Apache Wicket je tzv. komponentově orientovaný a od klasických a předešlých se liší tím, že vybuduje celý model na straně serveru a vygeneruje HTML, které je zasláno klientovi [17]. Podobně orientované frameworky jsou např. GWT, JSF, Vaadin atd. Následující Obrázek 6 zobrazuje schéma Apache Wicket:



Obrázek 6: Schéma komponentově orientovaného frameworku Apache Wicket. Zdroj: [17]

Tím, že je komponentově orientovaný nabízí ve vývoji řadu výhod. Samotné webové stránky jsou zde zpracovávány jako instance objektu, což umožňuje využít výhod objektově orientovaného programování při návrhu stránek. Stavy komponent a stránek jsou zde řešeny jako Java objekty a je možné, aby udržovali stav a odkazovali na jiné objekty [17]. Další navazující výhodou je, že nad nimi lze spouštět základní a známe JUnit testování a navíc v základu obsahuje spousty utility tříd pro unit testování k simulaci chování uživatele na webové aplikaci [17].

4 Návrh testovací aplikace

Navrhovaná aplikace bude vytvořena jako jednodušší forma rezervačního systému na filmy do kina. Základním konceptem budou čtyři stránky. Stránka, která bude zobrazovat výpis lístků s možností smazání a změny, dále stránka pro přidání, úpravu rezervace s výpisem kapacity sálu a stránka, která v případě překročení kapacity sálu zobrazí chybu.

4.1 Princip navrhované aplikace a použité technologie

Principem navrhované testovací aplikace je testování z hlediska výkonnosti daného webového frameworku. Testovány budou celkem tři webové frameworky Spring framework, Apache Struts 2 a Apache Wicket. K co nejpřesnějšímu testování jednotlivých webových frameworků bude potřeba zajištění stále databáze, ke které budou mít jednotlivé webové frameworky přistup pro práci s daty. Z důvodu odstranění latence sítě bude databáze vytvořena lokálně pomocí nástroje HSQLDB.

HSQLDB je relační databázový systém v jazyce Java [18]. Poslední verze k datu 29. 6. 2020 je HSQLDB 2.5.1 a je spravován společností The HSQL Development Group. V navrhované aplikaci poslouží jako lokální databázový systém pro uchování dat.

Nástrojem pro automatizaci běhu bude Gradle. Gradle je open-source nástroj pro automatizaci běhu zaměřený na výkon a flexibilitu [19]. Poslední verze k datu 24. 3. 2020 je Gradle 6.3 a je spravován společnosti Gradle Inc. V navrhované aplikaci poslouží jako nástroj pro importaci potřebných knihoven a celkovou automatizací sestavení a běhu aplikace.

Apache Tomcat je webový server a servlet kontejner. Je vyvíjený jako open-source projekt [20]. Poslední verze k datu 11. 2. 2020 je Apache Tomcat 9.0.31 a je spravován společností Apache Software Foundation. V navrhované aplikaci poslouží jako server, na kterém aplikace poběží.

Spring Boot je konfigurační nástroj, který automatizuje veškerou konfiguraci pro běh webových aplikací. Spring Boot zjednodušuje vytváření samostatných

webových aplikací, které můžete „prostě spustit“ [21]. Poslední verze k datu 12. 6. 2020 je Spring Boot 2.3.1 a je spravován společností Pivotal. Jeho výhodou je že má v sobě již vložený server Apache Tomcat, na kterém navrhovanou aplikaci sám spustí a nemusíme ho tak složitě konfigurovat.

Hibernate je framework pro ORM (objektově-relační mapování) a funguje jako implementace JPA (Java Persistance API). JPA (Java Persistence API) je standard popisující programátorské rozhraní (API) a chování knihoven pro objektově-relační mapování [22]. V principu nám slouží k přemapování databázových objektů na Java objekty. Poslední stabilní verze k datu 7. 2. 2020 je Hibernate ORM 5.4.1 a je spravován společností Red Hat. V navrhované aplikaci poslouží jako implementace JPA pro snadné mapování na databázi.

Navrhovaná aplikaci bude cílit na testování back-end frameworků, a tudíž zde front-end nebude složitěji řešen. K lepšímu zobrazení front-endu aplikace bude využit Bootstrap a vlastní stylování v CSS.

Bootstrap je open-source sada nástrojů pro vývoj pomocí HTML, CSS a JavaScriptu [23]. Obsahuje řadu CSS knihoven pro snadnější tvorbu. Poslední verze k datu 28. 11. 2019 je Bootstrap 4.4.1 a je spravován Bootstrap Core Team. V navrhované aplikaci poslouží k snadnější stylizaci front-endu.

Celý projekt obsahující jednotlivé aplikace ve vybraných frameworcích je dostupný na https://github.com/users/ocasusMaximus/projects/2 .

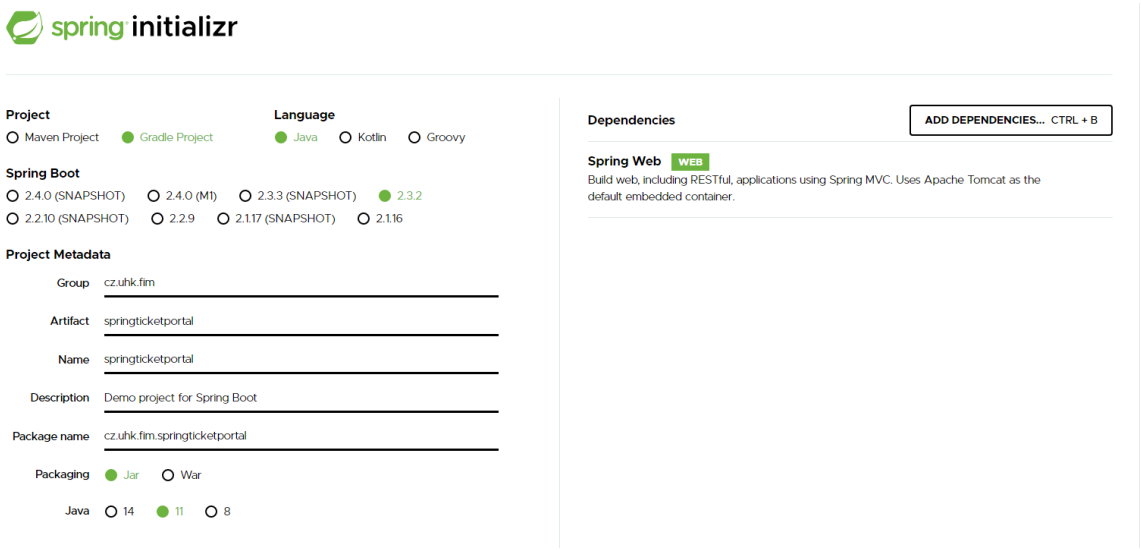
4.2 Navrhovaná aplikace se Spring frameworkem

K testování frameworku bude použita vždy aktuální verze. Pro Spring framework bude využito verze Spring framework 5.2.7.

Díky podrobné dokumentaci a velmi dobře zpracovanému API bude za pomocí tzv. Spring initializr a použití Spring Web MVC vytvořena základní inicializace projektu. Spring initializr poskytuje rozšiřitelné API ke generování projektů založených na JVM (Java Virtual Machine) a ke kontrole metadat používaných ke generování projektů, např. vylistování závislostí a verzí [24]. Spring Web MVC obsahuje veškeré komponenty k jednoduchému inicializování a spuštění základní webové aplikace se Spring frameworkem. Formální název

„Spring Web MVC“ pochází z názvu jeho zdrojového modulu (spring-webmvc), ale je běžněji známý jako „Spring MVC“ [25].

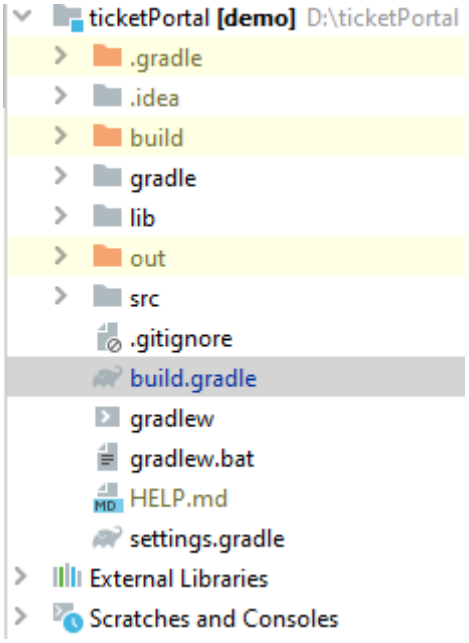
Následující Obrázek 7 zobrazuje základní nastavení projektu pomocí Spring initializr:



Obrázek 7: Nastavení ve Spring initializr. Zdroj:[24]

K sestavení projektu bude použit nástroj pro automatizaci sestavování programu (aplikace) Gradle. Zvoleným jazykem bude pochopitelně Java. K automatizaci běhu navrhované webové aplikace bude použit Spring Boot ve verzi 2.3.1.

Po inicializaci základního nastavení se vytvoří základní struktura projektu pro webovou aplikaci v prostředí Spring frameworku. Následující Obrázek 8 zobrazuje základní strukturu projektu pro navrhovanou aplikaci:



Obrázek 8: Struktura projektu pro Spring framework. Zdroj: [autor]

Při použití nástroje pro automatizaci běhu Gradle jsou automaticky vytvořeny potřebné soubory, z nichž nejdůležitější je soubor `build.gradle`. Ten slouží k deklarování závislostí a slouží jako základní konfigurační soubor pro Gradle. Konfigurace souboru `build.gradle` v jazyce Groovy pro navrhovanou aplikaci:

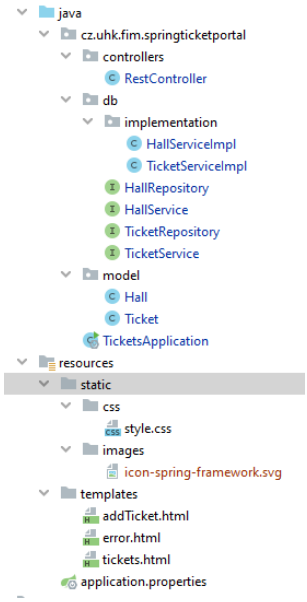
```
plugins {
    id 'org.springframework.boot' version '2.3.1.RELEASE'
    id 'io.spring.dependency-management' version '1.0.8.RELEASE'
    id 'java'
}
group = 'cz.uhk.fim'
version = '0.0.1-SNAPSHOT'

configurations {
    developmentOnly
    runtimeClasspath {
        extendsFrom developmentOnly
    }
}
repositories {
    mavenCentral()
}
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-
thymeleaf'
    implementation 'org.springframework.boot:spring-boot-starter-
web'
    implementation 'org.springframework.boot:spring-boot-starter-
data-jpa'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
    implementation 'org.hsqldb:hsqldb:2.5.1'
    testImplementation('org.springframework.boot:spring-boot-
starter-test') {
        exclude group: 'org.junit.vintage', module: 'junit-vintage-
engine'
    }
}
```

Ukázka kódu 2: Ukázka nastavení Gradle. Zdroj: [autor]

Jelikož Gradle využívá jazyka Groovy, jsou zde v této syntaxi uvedeny veškeré základní konfigurace. Důležitou součástí jsou tzv. dependencies (závislosti),

pomocí kterých si Gradle naimportuje z repozitáře Maven Central Repository soubory a implementuje je do projektu. Uvádí se zde veškeré potřebné závislosti od řadiče pro napojení do databáze po knihovny pro CSS stylování front-endu.

Pro dodržení architektonického vzoru MVC se ve Spring MVC dělí základní struktura na balíčky `java` a `resources`. Balíček `java` obsahuje veškerý Java kód a vytváří se zde modelová a řídící část projektu. Následující Obrázek 9 znázorňuje celkovou strukturu projektu s veškerým obsahem:



Obrázek 9: Celková struktura projektu pro Spring framework. Zdroj: [autor]

V modelové části budou vytvořeny potřebné entity, které se vytvářejí jako třídy v jazyce Java s anotacemi pro Spring MVC. Navrhovanou aplikací je rezervační systém do kina. Základní princip navrhované aplikace je ve vytváření lístku, proto je vytvořena třída `Ticket.java` (lístek) a `Hall.java` (sál). Tyto entity poté potřebují uchování dat. K tomu je určen balíček `db`, ve které každá z těchto entit má svoje rozhraní pro `repository` a `service`.

Rozhraní pro `repository` rozšiřuje JPA repository. Spring framework nabízí právě tyto generalizované třídy jako je `JPArepository` a díky tomu se nemusíme zabývat složitější konfigurací pomocí XML souborů. Nastavení rozhraní pro `Repository`, které rozšiřuje `JPARepository`:

```
package cz.uhk.fim.springticketportal.db;
import cz.uhk.fim.springticketportal.model.Ticket;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
@Repository
public interface TicketRepository extends JpaRepository<Ticket,
Integer> {
}
```

Ukázka kódu 3: Ukázka třídy TicketRepository.java. Zdroj: [autor]

Rozhraní pro `service` deklaruje základní metody/služby k práci s modelovací částí. Třída pro implementaci `service` využívá právě `JpaRepository`, které přináší základní metody pro nakládání s modelovacími entitami a blíže zde specifikujeme svoje metody pro zacházení databází. Celá business logika by zde měla být implementována.

Balíček `controller` poté obsahuje třídu `RestController.java`, která zastupuje controller a obsluhuje mapování na jednotlivé URL. URL (Uniform Resource Locator) je soubor znaků, který slouží k identifikaci přesného umístění informací na internetu. URL definuje doménovou adresu serveru, umístění zdroje na server a protokol [26]. Klasický controller z MVC architektury lze anotovat pomocí anotace `@Controller` [27]. V této třídě deklarujeme obsluhu jednotlivých požadavků HTTP protokolu. Spring MVC k tomu využívá metody označené anotacemi `@GetMapping` pro GET požadavek, `@PostMapping` pro POST požadavek atd. V dřívějších verzích byli označované anotacemi `@RequestMapping`, kde se v hodnotě method udával typ požadavku. U každé metody mapované na požadavek HTTP protokolu je v anotaci potřeba uvést hodnotu value, která nám mapuje URL, na kterou se přesměrovává v případě zavolání této metody a požadavku. Ukázka metody obsluhující HTTP GET požadavek pro zobrazení přidání lístku v navrhované aplikaci:

```
@GetMapping(value = "/addTicket")
public String getTickets(Model model) {
    model.addAttribute("ticket", new Ticket());
    model.addAttribute("halls", hallService.loadAllHalls());
    return "addTicket";
}
```

Ukázka kódu 4: Ukázka mapování GET požadavku ve Spring Frameworku. Zdroj: [autor]

Balíček `resources` (zdroje) obsahuje koncovou view z MVC architektonického vzoru, soubory pro CSS stylování a při použití Spring Boot i konfigurační soubor `application.properties`. V případě navrhované aplikace rezervačního systému do kina se zde vytvoří podle návrhu čtyři HTML soubory. Soubor `tickets.html`, který slouží k deklaraci zobrazení stránky pro výpis lístků. Soubor `addTicket.html` pak je určen pro zobrazení stránky k přidání lístků. Soubor `editTicket.html` slouží k zobrazení stránky pro úpravu lístků. Čtvrtým souborem je pak `error.html`, který se zobrazuje v případě chyby, jakou je překročení kapacity sálu.

Soubor `application.properties` slouží pro Spring Boot, který zajišťuje konfiguraci běhu navrhované aplikace. Deklarují se zde např. nastavení pro připojení k databázi HSQLDB s definovaným řadičem OJDBC, deklarace vstupu (portu) na kterém aplikace poběží, nebo nastavení JPA. Nastavení souboru application.properties pro projekt:

```
spring.datasource.url=jdbc:hsqldb:file:C:/db/
spring.datasource.username=SA
spring.datasource.password=
spring.datasource.driver-class-name=org.hsqldb.jdbc.JDBCDriver
# Change port
server.port=8080
# JPA Settings
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto=update
spring.jpa.hibernate.naming.implicit-
strategy=org.hibernate.boot.model.naming.ImplicitNamingStrategyLegac
yHbmImpl
spring.jpa.hibernate.naming.physical-
strategy=org.springframework.boot.orm.jpa.hibernate.SpringPhysicalNa
mingStrategy
spring.jpa.open-in-view=false
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.HSQLDi
alect
```

Ukázka kódu 5: Ukázka konfigurace v application.properties. Zdroj: [autor]

Výsledkem grafického zobrazení jsou potom čtyři HMTL stránky, které se vykreslují s použitím šablonovacího nástroje Thymeleaf. Díky němu se dá přistoupit k akcím definovaným v controlleru. Ukázka využití šablonovacího nástroje Thymeleaf pro odesílání formuláře:

```
<form id = "goback" th:action = "@{/addTicket}" >
    <button class="button_a" >Zpět</button>
</form>
```

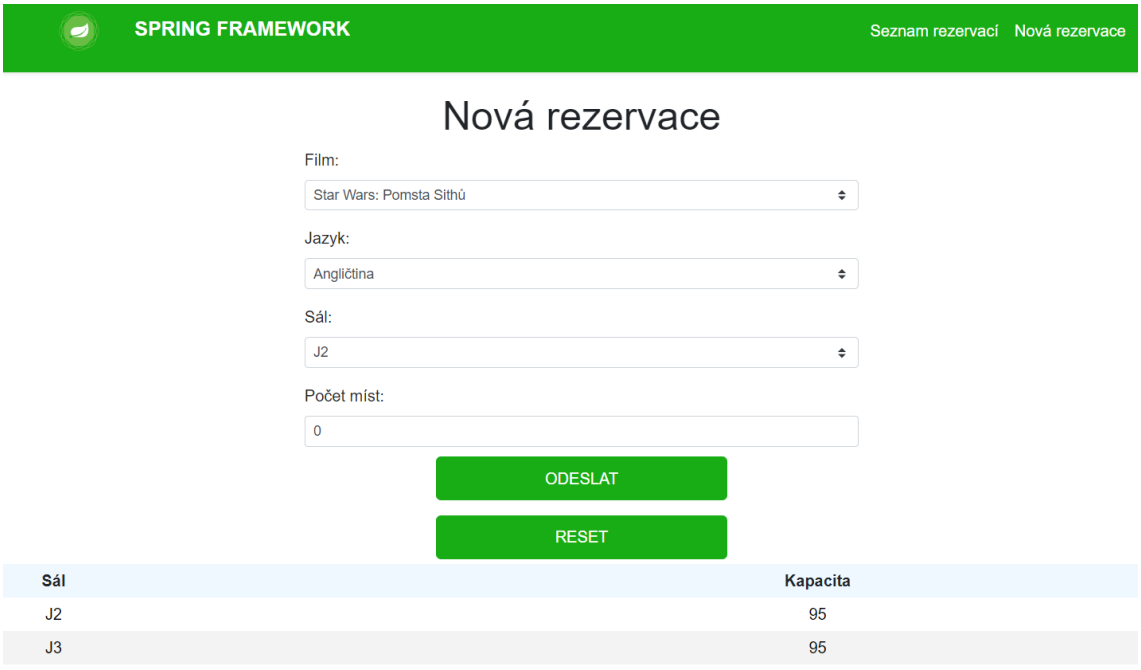
Ukázka kódu 6: Ukázka Thymeleafu. Zdroj: [autor]

Tato forma ukazuje provázání šablonovacího nástroje spolu se Spring MVC. Forma skrz Thymeleaf vyhledá akci na základě definované URL, která je totožná v controlleru ve Spring MVC. Výsledkem je, že po zmáčknutí tlačítka (button) se odešle forma, která zavolá akci v controlleru a prohlížeč nasměruje na stránku s přidáváním rezervací.

Aplikace jako úvodní stránku nabízí přehled všech rezervací, které ovlivňují obsazenost sálů. Jednotlivé rezervace se dají měnit i mazat. Můžou se smazat i všechny rezervace najednou, čímž se uvolní veškerá kapacita sálů. Následující Obrázek 10 ukazuje úvodní stránku s přehledem rezervací:

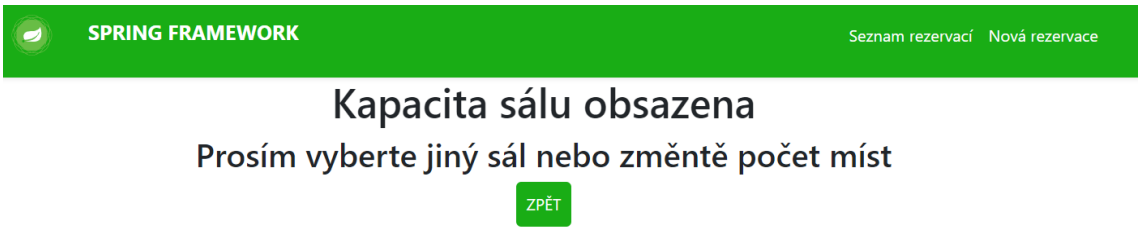
Obrázek 10: Spring framework stránka s výpisem seznamu rezervací. Zdroj: [autor]

Stránka pro přidávání nových rezervací nabízí možnost výběru rezervace pro daný film. U rezervace se dá vybrat film, jazyk pro daný film, sál a počet míst. Nelze vytvořit rezervaci s nulovým počtem míst. Pro uživatele je tu k dispozici přehled o kapacitě různých sálů a nelze si vytvořit rezervaci s počtem míst překračující dostupnou kapacitu sálu. Následující Obrázek 11 zobrazuje stránku pro přidávání rezervace:



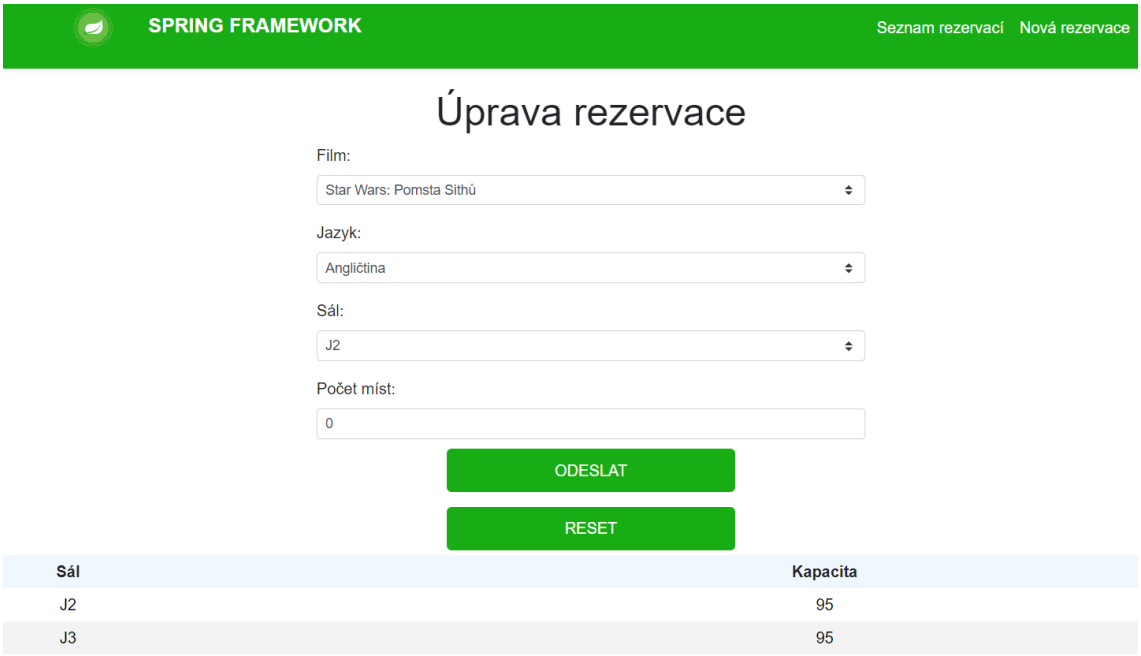
Obrázek 11: Spring framework stránka pro přidání rezervace. Zdroj: [autor]

Pokud uživatel zvolí rezervaci s počtem míst překračující kapacitu sálu, rezervace se nevytvoří. Místo toho se uživateli zobrazí stránka s varováním o obsazenosti kapacity sálu. Nabízí uživateli, aby se vrátil zpět na úvodní stránku s výpisem rezervací. Následující Obrázek 12 znázorňuje stránku s varováním o obsazenosti sálu:



Obrázek 12: Spring framework stránka pro varování s nedostatečnou kapacitou sálu. Zdroj: [autor]

Pokud uživatel na úvodní stránce u stávající rezervace stiskne tlačítko změnit, dostane se na stránku pro úpravu stávající rezervace. Zde může změnit film, jazyk, sál, nebo počet míst u stávající rezervace. Pokud opět překročí stávající kapacitu, bude přesměrován na chybovou stránku a změna rezervace nebude uložena. Následující Obrázek 13 zobrazuje stránku pro úpravu stávající rezervace:



Obrázek 13: Spring framework stránka pro úpravu stávající rezervace. Zdroj: [autor]

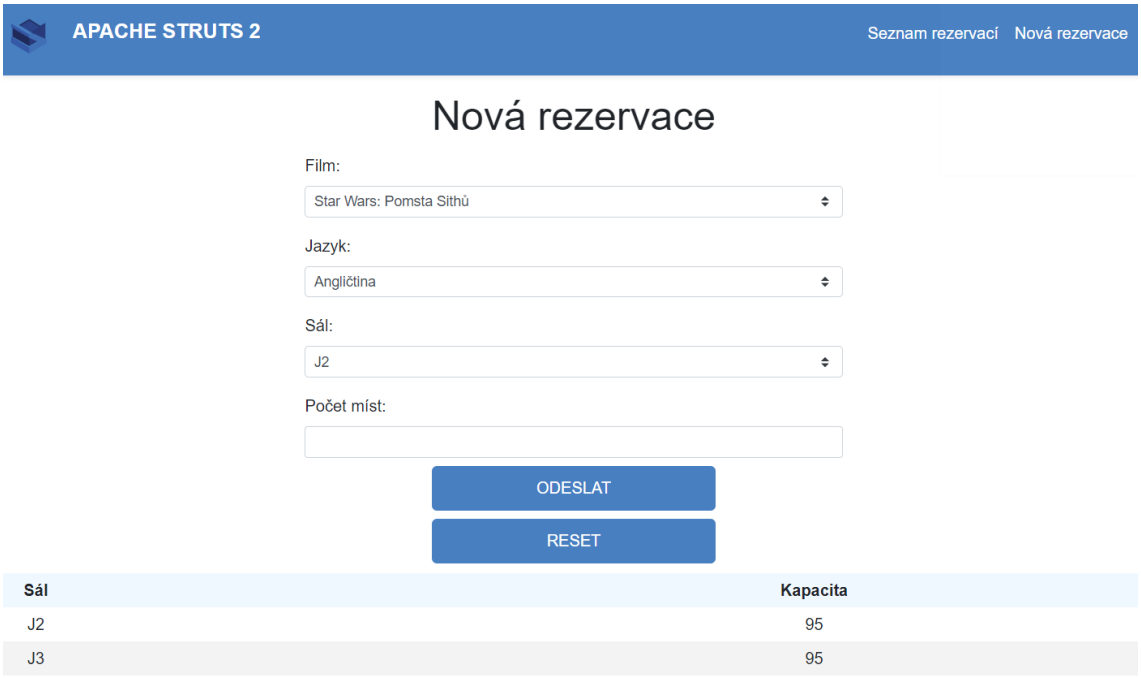
4.3 Navrhovaná aplikace s Apache Struts 2

K testování frameworku bude použita vždy aktuální verze. Pro Apache Struts 2 bude využito verze Apache Struts 2.5.22. Apache Struts 2 není tolik oblíbený a neposkytuje stejné inicializační nástroje jako např. Spring initializr. Spring boot poskytuje skvělé řešení pro nasazení a rozběhnutí vypracované aplikace na serveru Tomcat. Jeho kompatibilita s použitím Apache Struts 2 zajišťuje snadnější konfiguraci pro vystavění a běh aplikace. Navíc se díky použití stejných nástrojů dá dosáhnout co nejpřesnějších výsledků v testování.

Aplikace podobně jako s použitím Spring frameworku má úvodní stránku jako seznam rezervací do kina. Rezervace se dají měnit i mazat. V případě zrušení všech rezervací se dají smazat všechny rezervace najednou. Jako u Spring Frameworku každá stránka obsahuje identickou navigační lištu. Z úvodní stránky s výpisem rezervací se tedy dá dostat i na stránku pro přidávání nových rezervací. Následující Obrázek 14: Apache Struts 2 stránka pro zobrazení seznamu rezervací. Zdroj: [autor] znázorňuje stránku pro zobrazení seznamu rezervací:

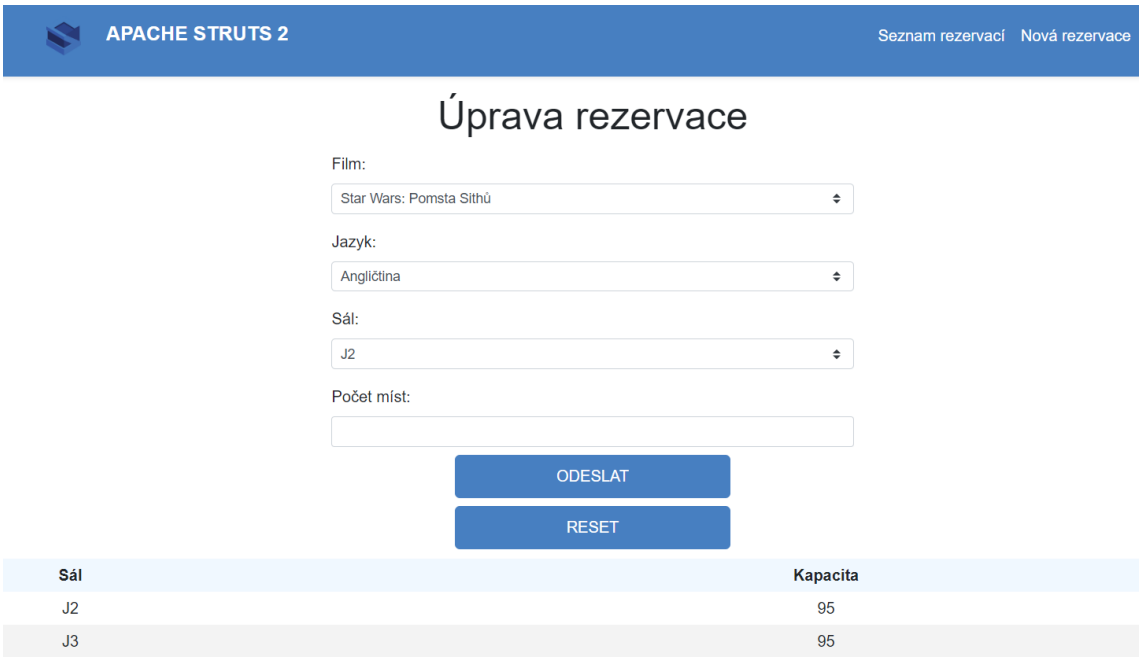
Obrázek 14: Apache Struts 2 stránka pro zobrazení seznamu rezervací. Zdroj: [autor]

Stránka pro přidávání nových rezervací nabízí i editaci stávající rezervace. U rezervace lze vybrat film, jazyk, sál a počet míst. Stránka obsahuje také výpis sálů a jejich aktuální kapacitu. Aplikace nedovoluje vytvořit novou rezervaci s nulovým počtem míst. Nelze provádět rezervaci, která by překračovala aktuální počet míst. Následující Obrázek 15: Apache Struts 2 stránka pro přidání nových rezervací. Zdroj: [autor] znázorňuje stránku pro přidávání nových rezervací:



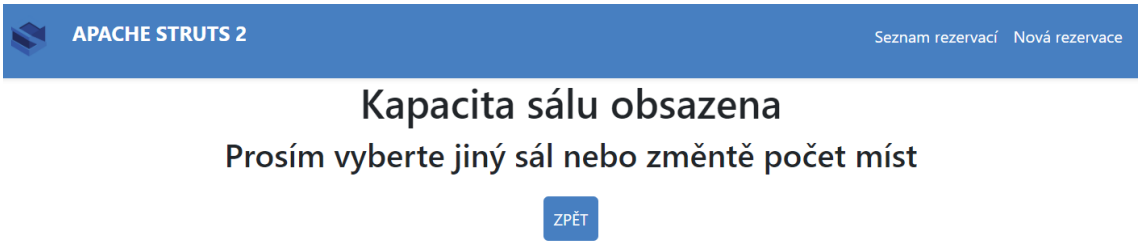
Obrázek 15: Apache Struts 2 stránka pro přidání nových rezervací. Zdroj: [autor]

Skrz tlačítko změnit u stávající rezervace je možnost upravovat stávající rezervaci. Uživatel je po stisknutí přesměrován na stránku s úpravou stávající rezervace. Zde je možné změnit film, jazyk, sál a počet míst. Pokud nově zadaná úprava bude překročovat kapacitu sálu, změna nebude uložena. Následující Obrázek 16 znázorňuje stránku pro přidávání nových rezervací:



Obrázek 16: Apache Struts 2 stránka pro úpravu stávající rezervace. Zdroj: [autor]

V případě vytvoření rezervace s překročením aktuální kapacity sálu aplikace uživatele přesměruje na chybovou stránku s upozorněním o překročení kapacity sálu. Požadovaná rezervace se v systému nevytvoří. Uživateli poté nabídne možnost vrátit se zpět na úvodní stránku, nebo mu dá možnost upravit stávající rezervaci, aby nepřekračovala aktuální kapacitu sálu. Následující Obrázek 17 znázorňuje chybovou stránku s překročenou kapacitou sálu:



Obrázek 17: Apache Struts 2 stránka pro varování s nedostatečnou kapacitou sálu. Zdroj: [autor]

4.4 Navrhované aplikace s Apache Wicket

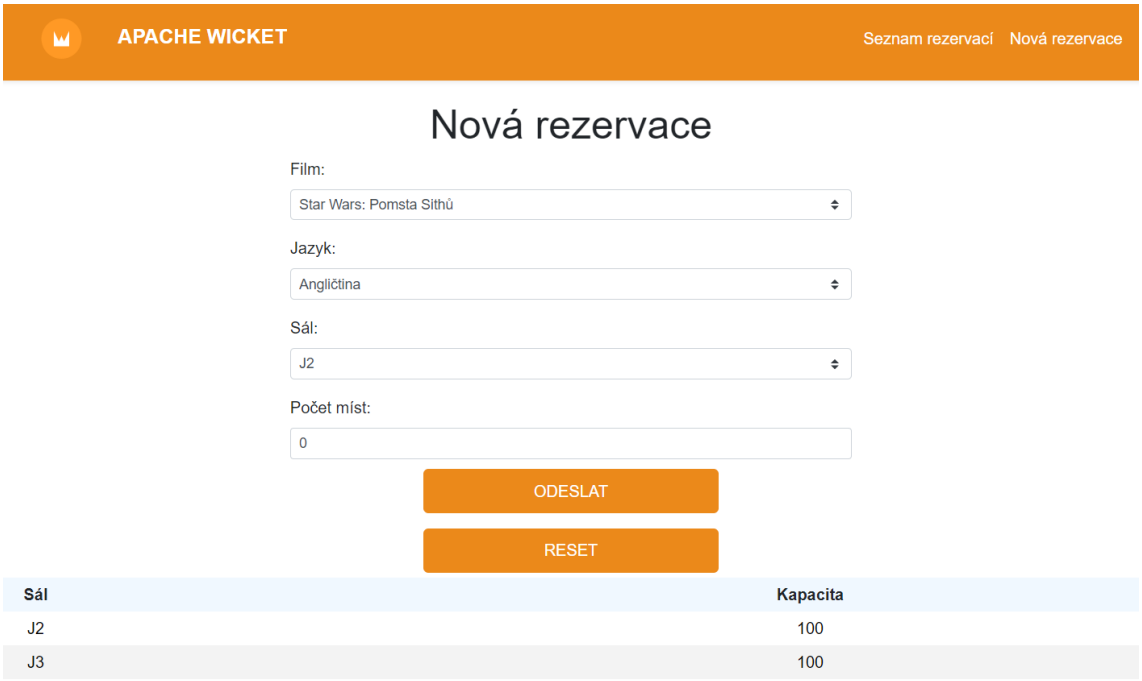
V navrhované aplikaci bude použita nejnovější stabilní verze Apache Wicket 8.6.1. Apache Wicket je méně populární než Spring framework, ale na rozdíl od upadajícího Apache Struts 2 si konzistentně drží podobnou míru užití v průběhu let. Zejména u nás se hojně používá. Na českém internetu jej používají například společnosti Vodafone, Air Bank a další [28].

Hlavním přínosem, který Apache Wicket nabízí je, že je komponentově orientovaný. Ke každým HTML komponentům jako např. `<form>`, `<select>`, atd. nabízí svoje komponenty. To jsou v podstatě Java objekty. Apache Wicket se tedy snaží o co nejpřesnější přiblížení k objektově orientovanému programování v jazyce Java.

Navrhovaná aplikace v Apache Wicket bude mít stejný počet stránek jako u předchozích frameworků. Aplikace má úvodní stránku s výpisem rezervací do kina. Rezervace se dají měnit, nebo mazat. Dají se smazat i všechny rezervace najednou. Navigační lišta je kromě jiného barevného provedení totožná s předchozími frameworky. Z úvodní stránky s výpisem rezervací se poté dá dostat na stránku s přidáním nové rezervace. Následující Obrázek 18 znázorňuje stránku pro zobrazení seznamu rezervací:

Obrázek 18: Apache Wicket stránka pro zobrazení seznamu rezervací. Zdroj: [autor]

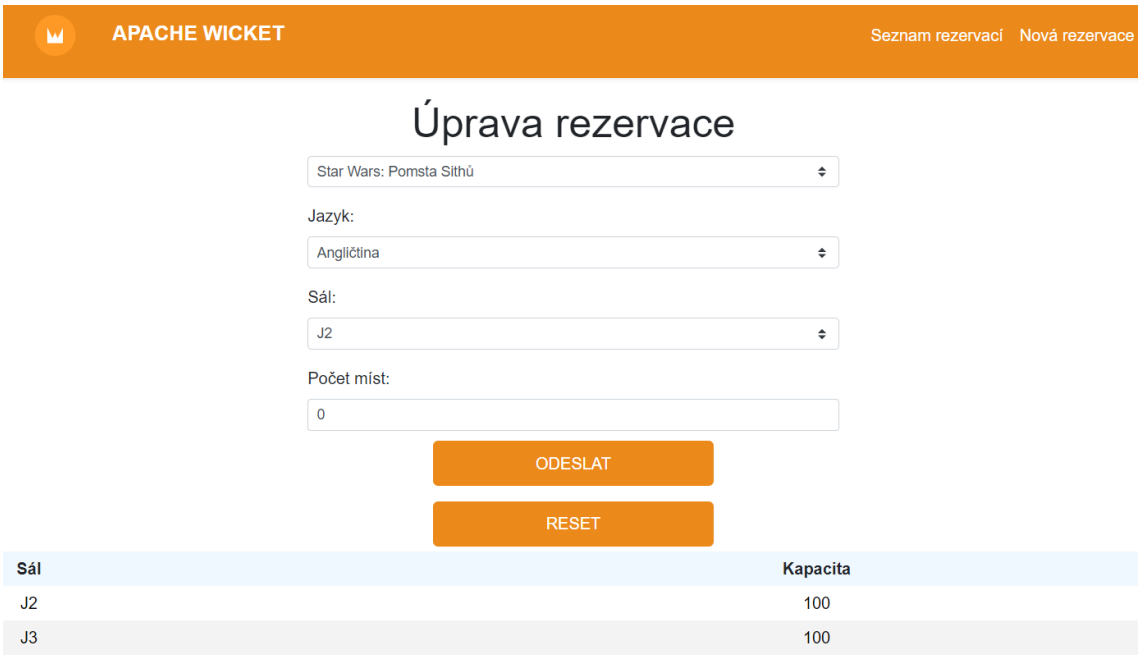
Stránka pro přidání nové rezervace uživateli zobrazuje výpis dostupných sálů a jejich aktuální kapacity. Uživatel zde má možnost vybrat si z formuláře nabízený film, který poté nabízí varianty jazykového provedení. Uživatel si může zvolit sál a počet míst. Pokud uživatel zadá rezervaci nepřekračující aktuální kapacitu, rezervace se uloží. Uživatel bude přesměrován na úvodní stránku s výpisem rezervací. Následující Obrázek 19 znázorňuje stránku pro přidávání nových rezervací:



Obrázek 19: Apache Wicket stránka pro přidání nových rezervací. Zdroj: [autor]

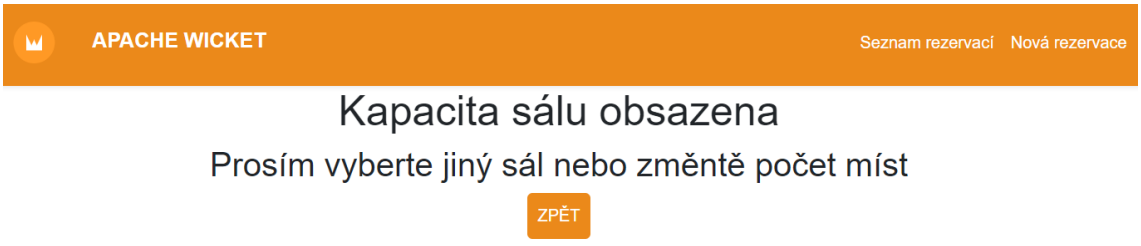
Z úvodní stránky s výpisem rezervací uživatel může měnit stávající rezervaci stisknutím tlačítka změnit. Tím se dostane na stránku pro úpravu stávající rezervace, kde může měnit jednotlivé atributy. Pokud uživatel opět změní

počet míst, který bude překračovat aktuální kapacitu sálu, změna rezervace se neuloží. Uživatel bude přesměrován na chybovou stránku. V opačném případě se provede změna stávající rezervace a uživatel bude přesměrován na úvodní stránku. Následující Obrázek 20 znázorňuje stránku pro úpravu stávající rezervace:



Obrázek 20: Apache Wicket stránka pro úpravu stávající rezervace. Zdroj: [autor]

Pokud u přidávání nové nebo úpravy stávající rezervace uživatel překročí aktuální kapacitu sálu, rezervace se neuloží. Uživatel místo toho bude přesměrován na chybovou stránku s obsazeností sálu. Z této stránky se poté může dostat tlačítkem zpět na úvodní stránku. Pomocí navigační lišty se může vracet i na stránku s přidáním nové rezervace. Následující Obrázek 21 znázorňuje chybovou stránku s překročenou kapacitou sálu:



Obrázek 21: Apache Wicket stránka pro varování s nedostatečnou kapacitou sálu. Zdroj: [autor]

5 Testování Frameworků

K testování daných frameworků bude využito tzv. testování výkonnosti a zátěže (performance testing). Cílem testování výkonnosti a zátěže je odhalit chyby a odstranit nebo minimalizovat je a tím vylepšit celkovou výkonnost aplikace. Výkonnosti je myšlena doba odezvy, spolehlivost, využití zdrojů, škálovatelnost a jiné [29].

To vývojářům poskytuje informace o rychlosti, stabilitě a škálovatelnosti jejich aplikace. Důležité přitom je, že odkrývá chyby aplikace jako např. pomalý běh, nekonzistentnost a tím její špatné využití.

K testování navržených aplikací bude využit nástroj Apache JMeter. Apache JMeter nebo JMeter je součástí projektu Jakarta od Apache. Jedná se o nástroj pro měření výkonnosti a vytváření zátěže (performance testing) pro webové aplikace [30]. Poslední verze k datu 15. 5. 2020 je Apache JMeter 5.3.0 a je spravován společností Apache Software Foundation. JMeter je napsán v jazyce Java a disponuje grafickým rozhraním. Jde o velmi populární nástroj pro vytváření zátěžových testů webových aplikací [30].

Navrhované aplikace jsou založené na programovacím jazyce Java, tudíž je jejich spouštění a běh součástí JRE (Java Runtime Enviroment). Veškeré aplikace vyvíjené v JRE běží na JVM (Java Virtual Machine). JVM funguje jako tzv. run-time engine (běhový modul) pro spouštění aplikací v jazyce Java. Je zodpovědný za spouštění hlavní metody každé aplikace v JRE [31].

Důležité pro testování výkonnosti aplikací spouštěných JVM je způsob jeho zpracování kódu a rozběhnutí aplikace. Poté, co jsou zkompilovány veškeré třídy aplikace, se v době startu všechny důležité třídy načtou do JVM paměti. Tím jsou za běhu zpřístupněné jako první. Ostatní třídy, které nejsou pro hlavní běh aplikace důležité, se poté načítají při jejich požadavcích. Tento způsob načítání tříd se nazývá tzv. lazy class loading (líné načítání tříd) a způsob kompilace tříd tzv. just-in-time (přesně na čas) [32].

Doba odezvy (response time) prvního požadavku na webovou aplikaci napsanou v jazyce Java, je tedy často mnohem pomalejší než průměrná doba odezvy při běhu. Doba odezvy je čas, za který uživatel při zaslání HTTP požadavku

dostane HTTP odpověď [33]. Při dalších požadavcích za běhu aplikace už jsou třídy v JVM paměti nahrané, a proto je doba odezvy rychlejší. Tento celý proces vylaďování JVM je znám jako zahřívání. Při výkonnostním testování je proto nutné brát v potaz proces zahřívání.

Běžná praxe při výkonnostním testování těchto aplikací je opakovaně posílat aplikaci HTTP požadavky, které provolávají třídy aplikace a tím se načítají do JVM paměti. Výsledky prvních provolávání se proto při výkonnostním testování ignorují, dokud se JVM nezahřeje.

5.1 Testování Spring frameworku

Testování Spring frameworku bude založeno na opakovaném provolávání HTTP požadavky na tzv. koncové body. V architektonickém vzoru MVC, na kterém je Spring framework postavený, se o obsluhu HTTP požadavků stará komponenta controllers. Ta zajišťuje řídící logiku a skrz ni se po provolání potřebných částí požadavku vrací odpověď známou jako HTTP response. Měřena bude doba odezvy a propustnost (throughput). Propustnost je počet HTTP operací zpracovaný za určitý čas [34]. V případě testování se bude jednat o počet HTTP operací za minutu.

V navrhované aplikaci ve Spring frameworku jde o metody třídy `RestController.java`. K získání informací od serveru Tomcat na kterém aplikace běží, jsou využity HTTP GET požadavky.

Metody třídy `RestController.java` k získání informací (HTTP GET požadavky) jsou v aplikaci čtyři. Metoda s obsluhou HTTP požadavku na URL „/“ slouží na zobrazení hlavní domovské stránky a jako HTTP odpověď vrací stránku s výpisem všech rezervací. Ukázka kódu:

```
@GetMapping(value = "/")
public String index(Model model) {
    model.addAttribute("tickets", ticketService.loadAllTickets());
    return "tickets";
}
```

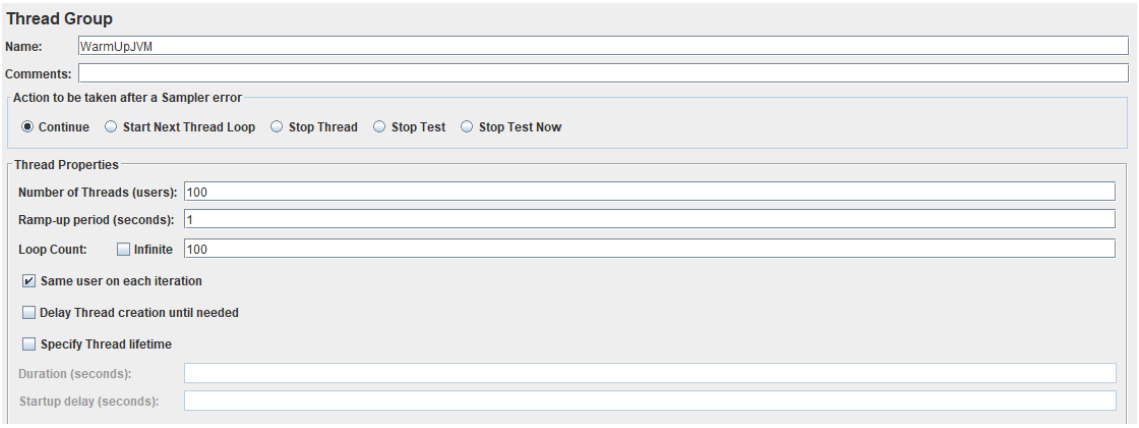
Ukázka kódu 7: Ukázka mapováni GET požadavku na úvodní stránku. Zdroj: [autor]

Metoda s obsluhou HTTP požadavku na URL „/addTicket“ slouží na zobrazení stránky s přidáním rezervace, kterou vrací jako odpověď. Ukázka kódu:

```
@GetMapping(value = "/addTicket")
public String getTickets(Model model) {
    model.addAttribute("ticket", new Ticket());
    model.addAttribute("halls", hallService.loadAllHalls());
    return "addTicket";
}
```

Ukázka kódu 8: Ukázka mapování GET požadavku na stránku s přidáním nové rezervace. Zdroj: [autor]

K co nejpřesnějším výsledkům bude před skutečným testováním provedeno zahřívání JVM. Zahřívání bude dosaženo opakovaným odesíláním HTTP požadavků. Bude provedena simulace sta uživatelů, kteří každý odešlou sto požadavků. Následující Obrázek 22: Nastavení JMeteru pro zahřátí JVM ve Spring frameworku. Zdroj: zobrazuje nastavení nástroje JMeter pro simulaci uživatelů k zahřátí JVM navrhované aplikace:

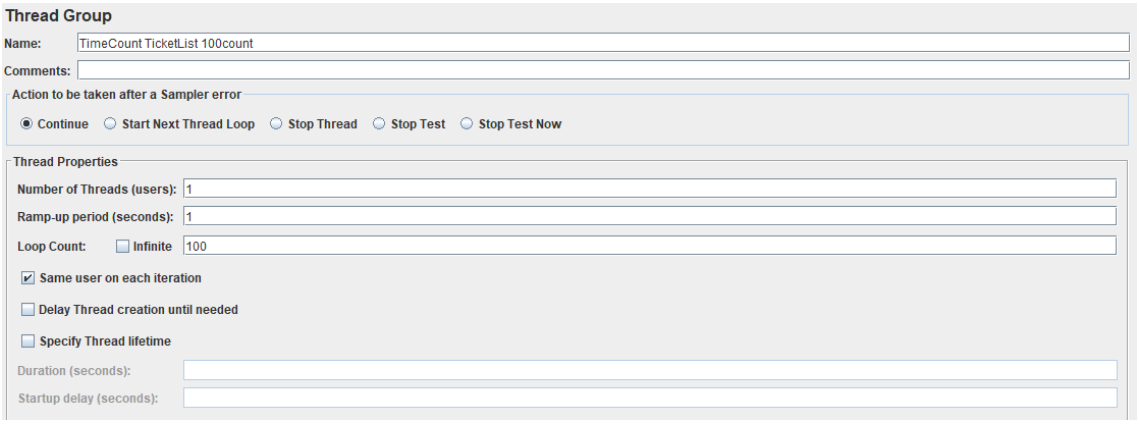


Obrázek 22: Nastavení JMeteru pro zahřátí JVM ve Spring frameworku. Zdroj: [autor]

Po zahřátí JVM bude provedeno skutečné měření. Skutečné měření se bude skládat z testů založených na testování doby odezvy a propustnosti. Nejlepším ukazatelem bude test HTTP GET požadavků. K lepší simulaci provozu bude testování úvodní stránky provedeno při registrovaných padesáti rezervacích.

Testování doby odezvy se bude skládat ze simulace jednoho uživatele, který bude zasílat 100 požadavků. Následující Obrázek 23: Nastavení JMeteru pro testování doby odezvy úvodní stránky ve Spring frameworku. Zdroj: zobrazuje nastavení

nástroje JMeter pro simulaci uživatele zasílající požadavky na zobrazení úvodní stránky:



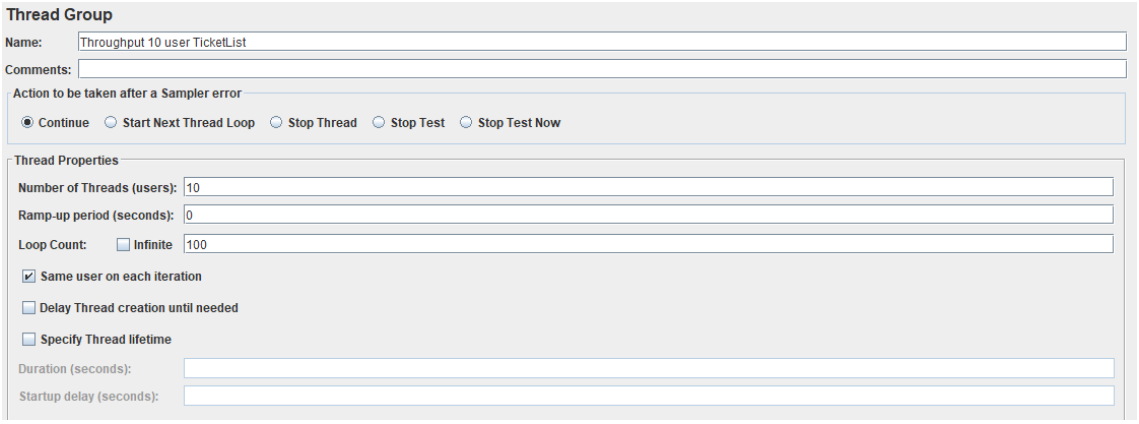
Obrázek 23: Nastavení JMeteru pro testování doby odezvy úvodní stránky ve Spring frameworku. Zdroj: [autor]

U každého HTTP požadavku bude nastavena doba čekání jedné sekundy. Tím bude zajištěna přesnější simulace uživatele. Následující Obrázek 24: Nastavení JMeteru pro časovou prodlevu mezi požadavky ve Spring frameworku. Zdroj: zobrazuje nastavení nástroje JMeter pro časovou prodlevu mezi HTTP požadavky:

Obrázek 24: Nastavení JMeteru pro časovou prodlevu mezi požadavky ve Spring frameworku. Zdroj: [autor]

Z výsledků bude měřena minimální, průměrná a maximální doba odezvy. Testovací nástroj JMeter nabízí u zobrazení výsledků testu i 90% percentil doby odezvy. Ten je přesnějším ukazatelem doby odezvy, protože odfiltruje ty nejdelší. Do výsledného měření bude zahrnut taky. Naprosto stejným způsobem bude testována i stránka pro přidávání nové rezervace.

Druhým testováním bude testování propustnosti navržené aplikace. Propustnost bude nejlépe měřitelná na úvodní stránce s výpisem rezervací. Testování proběhne simulací jednoho, deseti, padesáti a stech uživatelích. Každý uživatel bude zasílat deset požadavků. Mezi požadavky nebude žádná časová prodleva. Následující Obrázek 25: Nastavení JMeteru pro testování propustnosti ve Spring frameworku. Zdroj: zobrazuje nastavení nástroje JMeter pro testování propustnosti na zobrazení úvodní stránky při deseti uživatelích:



Obrázek 25: Nastavení JMeteru pro testování propustnosti ve Spring frameworku. Zdroj: [autor]

Podobné nastavení bude použito při testování propustnosti s více uživateli. Z výsledků testování bude patrné, jak dobře aplikace ve Spring frameworku zvládne dobu odezvy a propustnost.

5.2 Testování Apache Struts 2

Testování frameworku Apache Struts 2 bude podobně jako u Spring frameworku založeno na měření doby odezvy a propustnosti. Zde se ovšem konfigurace neprovádí pomocí anotací jako ve Spring frameworku.

Apache Struts 2 funguje pomocí akcí (actions), které obsluhují jednotlivé volání. Konfigurace se v navrhované aplikaci provádí v souboru `struts.xml` (dříve `struts-config.xml`). Zde jsou definovány jednotlivé akce obsluhující HTTP požadavky. Při provolání se zavolá příslušná metoda dané akce, které jsou definovány v balíčku `actions`. Apache Struts 2 tedy nevyužívá anotací a jeho řídící logika je rozdělena do `struts.xml` a definování samotné akce v definované třídě. Ukázka konfigurace v `struts.xml` pro zavolání metody k zobrazení domovské stránky s výpisem rezervací:

```
<action name="" method="listTickets"
class="cz.uhk.fim.actions.ShowPage">
    <result name="success">/WEB-INF/templates/tickets.jsp</result>
</action>
```

Ukázka kódu 9: Akce pro zavolání výpisu všech rezervací. Zdroj: [autor]

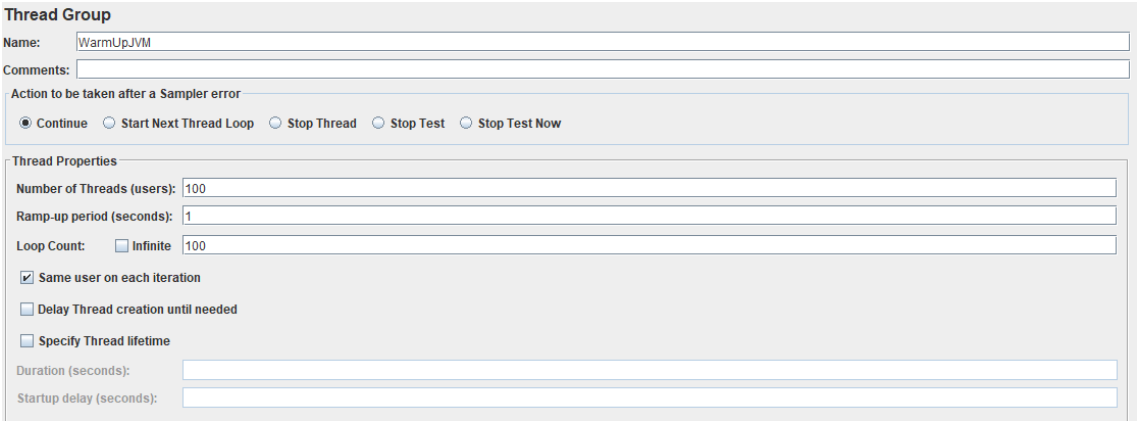
V třídě `ShowPage.java` se zavolá příslušná metoda `listTickets()`, která provede vylistování všech rezervací. Při úspěšném uložení vrací `struts.xml` textovou hodnotu SUCCESS, která v konfiguraci provolá výsledek, který uživateli zobrazí stránku s výpisem rezervací. Ukázka metody pro vytvoření nové rezervace v třídě `ShowPage.java`:

```
public String listTickets(){
    ticketService.loadAllTickets();
    ticketList = ticketService.loadAllTickets();
    return SUCCESS;
}
```

Ukázka kódu 10: Ukázka metody s načtením všech rezervací. Zdroj: [autor]

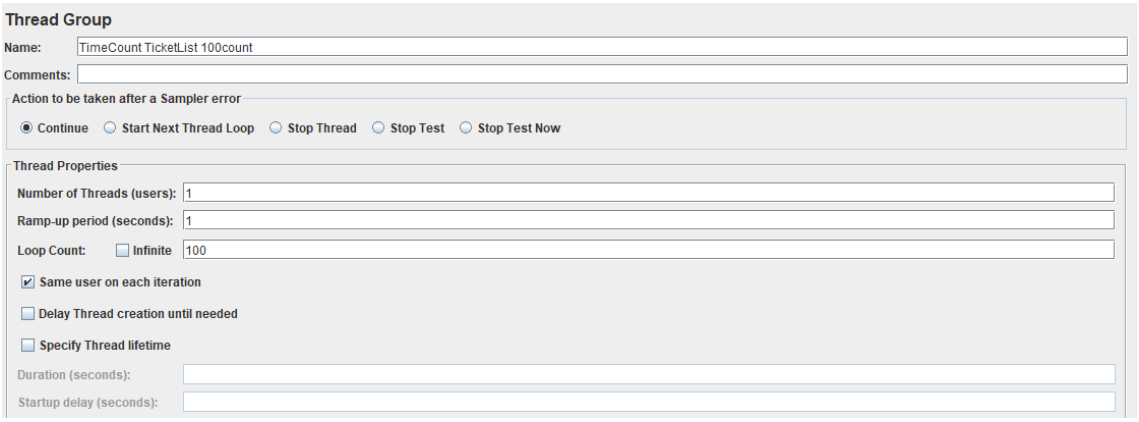
V navrhované aplikaci ve frameworku Apache Struts 2 podobně jako u Spring frameworku jsou využívány HTTP GET požadavky. Definice samotných metod akce pro obsluhu HTTP požadavků je totožné se Spring frameworkem. Rozdílem tedy je jejich správná konfigurace ve `struts.xml`, která je následně provolává. Do verze Spring 2.5 se ve Spring frameworku XML konfigurace využívali také [14].

Jelikož se opět jedná o aplikaci v jazyce Java, tak její běh v prostředí JRE zajišťuje JVM. K co nejpřesnějším výsledkům je před skutečným testováním potřeba JVM zahřát. Pro nejpřesnější srovnání bude konfigurace pro zahřátí JVM totožná. Bude tedy simulováno sta uživatelů, kteří každý odešlou sto požadavků. Následující Obrázek 26: Nastavení JMeter pro zahřatí JVM aplikace v Apache Struts 2. Zdroj: zobrazuje nastavení nástroje JMeter pro simulaci uživatelů k zahřátí JVM navrhované aplikace:



Obrázek 26: Nastavení JMeter pro zahřatí JVM aplikace v Apache Struts 2. Zdroj: [autor]

Stejně jako u Spring frameworku bude po zahřatí JVM provedeno skutečné měření. Skutečné měření bude probíhat testováním HTTP GET požadavků. Bude testována doba odezvy a propustnost aplikace. Pro porovnatelné výsledky bude nastavení pro testování doby odezvy totožné jako ve Spring frameworku. Bude tedy testována doba odezvy při jednom uživateli zasílající sto HTTP GET požadavků na úvodní stránku a stránku pro přidávání rezervací. Následující Obrázek 27: Nastavení JMeteru pro test doby odezvy úvodní stránky v Apache Struts 2. Zdroj: zobrazuje nastavení nástroje JMeter pro simulaci uživatele zasílající požadavky na zobrazení úvodní stránky:

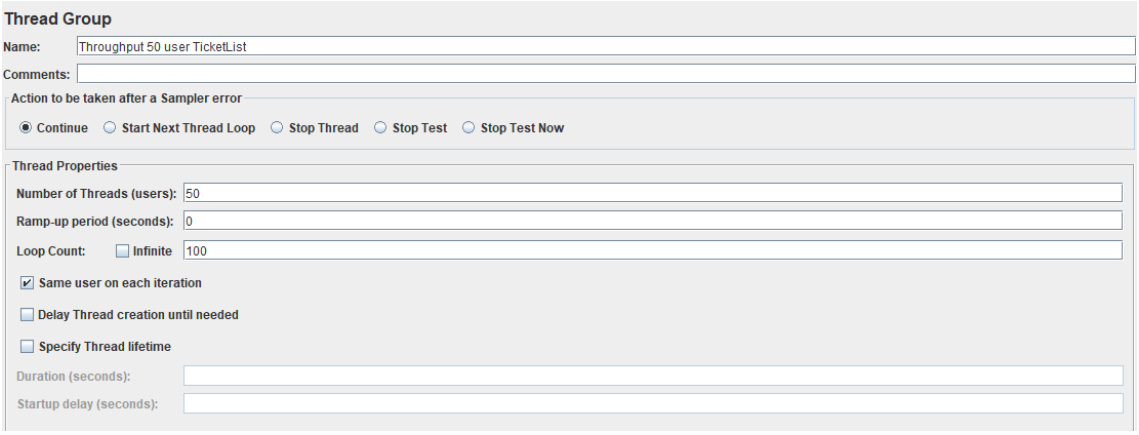


Obrázek 27: Nastavení JMeteru pro test doby odezvy úvodní stránky v Apache Struts 2. Zdroj: [autor]

Bude nastavena stejná konfigurace časové prodlevy jedné sekundy mezi požadavky. Z výsledků bude opět měřena minimální, průměrná a maximální doba

odezvy i 90% percentil. Naprosto stejným způsobem bude testována i stránka pro přidávání nové rezervace.

Testování propustnosti bude mít stejnou konfiguraci. Bude se tedy testovat při jednom, deseti, padesáti uživatelích, kteří každý budou zasílat sto HTTP požadavků. K přesnějšímu testování propustnosti nebude mezi požadavky nastavena žádná časová prodleva. Následující Obrázek 28: Nastavení JMeteru pro testování propustnosti úvodní stránky v Apache Struts 2. Zdroj: zobrazuje nastavení nástroje JMeter pro testování propustnosti na zobrazení úvodní stránky při padesáti uživatelích:



Obrázek 28: Nastavení JMeteru pro testování propustnosti úvodní stránky v Apache Struts 2. Zdroj: [autor]

Podobné nastavení bude použito při dalším testování propustnosti s jinými počty uživatelů. Z výsledků testování bude patrné, jak dobře aplikace v Apache Struts 2 zvládne dobu odezvy a propustnost.

5.3 Testování Apache Wicket

Testování Apache Wicket bude stejné jako u frameworků Spring framework a Apache Struts 2. Bude testována doba odezvy a propustnost. Apache Wicket se ovšem liší od předchozích frameworků, protože je komponentově orientovaný.

Ke každému HTML souboru je nutné mít stejně pojmenovanou třídu v jazyce Java. V té se potom vytváří komponenty z Apache Wicket frameworku odpovídající komponentám v HTML souboru. Ty se poté ztotožňují skrz HTML označení `<wicket:id></wicket:id>`. Ukázka označení formuláře v HTML souboru:

```
<form wicket:id="goback">
    <button class="button_a" >Zpět</button>
</form>
```

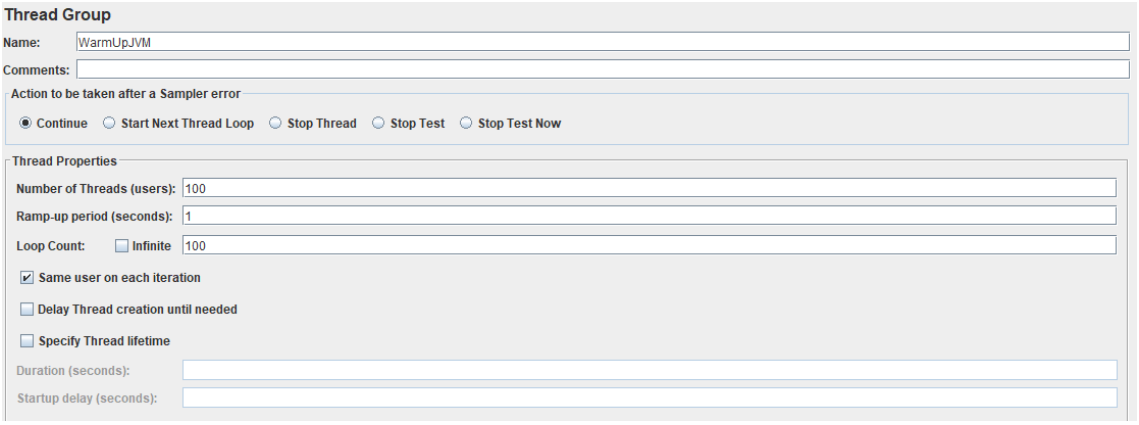
Ukázka kódu 11: Ukázka označení formuláře v HTML. Zdroj: [autor]

Komponenta se stejným id se poté musí vyskytovat v odpovídající Java třídě. Díky tomu Apache Wicket může s komponentami pracovat jako s Java třídami. To přináší veškeré výhody programování v jazyce Java. Ukázka zápisu odpovídající komponenty v Java třídě:

```
Form<Any> form = new Form<Any>("goback") {
        @Override
        protected void onSubmit() {
            setResponsePage(Tickets.class);
        }
    };
    add(form);
}
```

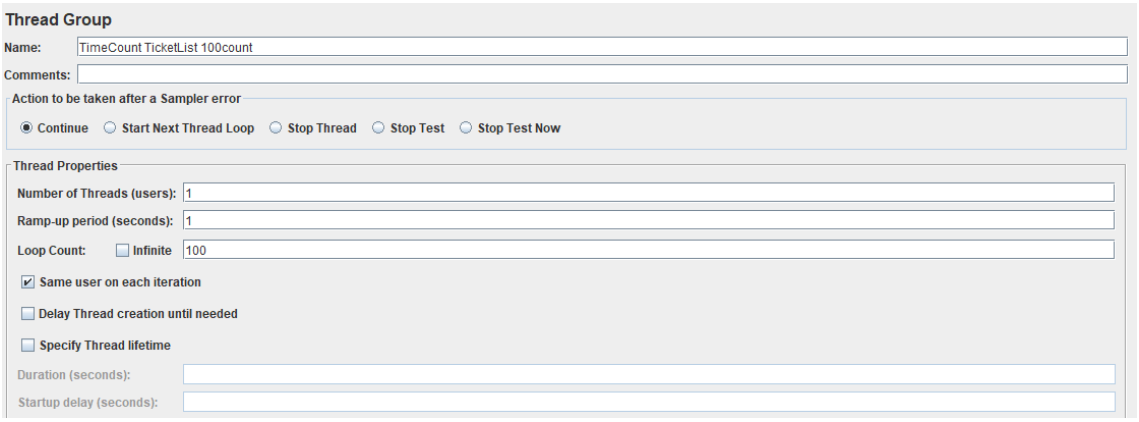
Ukázka kódu 12: Ukázka komponenty Form v Apache Wicket, Zdroj: [autor]

Každá z těchto tříd je poté mapována na danou URL. U úvodní stránky je např. namapovaná na URL „/“. Díky tomu bude možné otestovat Apache Wicket voláním HTTP požadavky na příslušné URL. Opět pro co nejpřesnější výsledky bude před samotným testováním provedeno zahřívání JVM. Bude zde obdobně simulováno 100 uživatelů zasílající 100 HTTP požadavků na úvodní stránku a stránku pro přidání nové rezervace. Následující Obrázek 29 zobrazuje nastavení nástroje JMeter pro simulaci uživatelů k zahřátí JVM navrhované aplikace:



Obrázek 29: Nastavení JMeter pro zahřatí JVM aplikace v Apache Wicket. Zdroj: [autor]

Po zahřátí JVM bude provedeno skutečné měření. Měření se bude opět zakládat na testování doby odezvy a propustnosti. Pro testování doby odezvy bude simulováno 100 požadavků zaslané jedním uživatelem na úvodní stránku. Následující Obrázek 30 zobrazuje nastavení nástroje JMeter pro simulaci uživatele zasílající požadavky na zobrazení úvodní stránky:

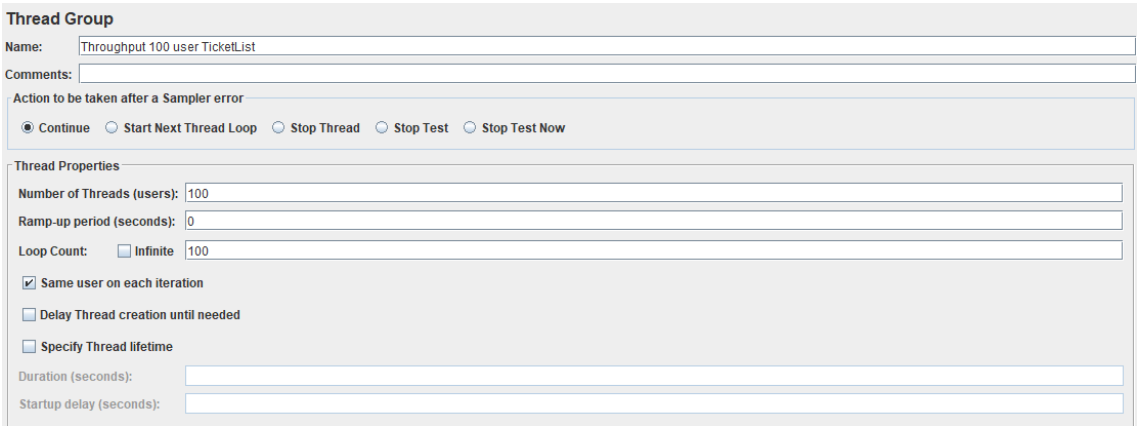


Obrázek 30: Nastavení JMeteru pro test doby odezvy úvodní stránky v Apache Wicket. Zdroj: [autor]

Mezi požadavky bude opět nastavena časová prodleva jedné sekundy. Z výsledků bude opět měřena minimální, průměrná a maximální doba odezvy i 90% percentil. Stejně tak bude testována i stránka pro přidání nové rezervace.

Testování propustnosti bude provedeno stejně jako u předchozích frameworků. Bude se testovat při jednom, deseti, padesáti a sto uživatelích, kteří každý zašlou 100 požadavků. Při testování propustnosti nebude mezi požadavky nastavena žádná časová prodleva. Následující Obrázek 31 zobrazuje nastavení

nástroje JMeter pro testování propustnosti na zobrazení úvodní stránky při sto uživatelích:



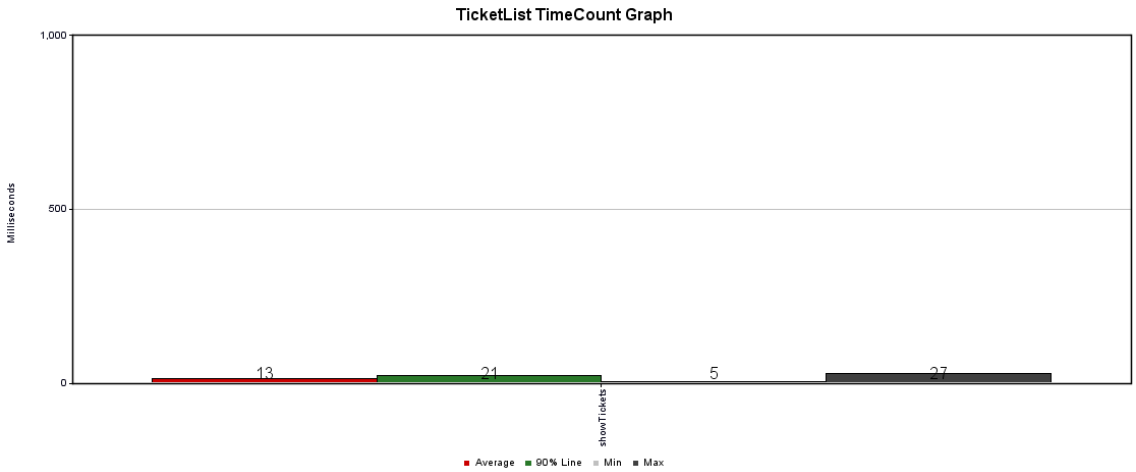
Obrázek 31: Nastavení JMeteru pro test propustnosti úvodní stránky v Apache Wicket. Zdroj: [autor]

Podobné nastavení bude použito při jiném počtu uživatelů. Výsledky ukážou, jak dobře si framework Apache Wicket povede v testování vůči zbývajícím frameworkům.

6 Shrnutí výsledků

Testování výkonnosti a zátěže daných webových frameworků spočívalo v měření jejich zpracování HTTP požadavků. Po zahřátí JVM, kde se výsledky ignorovali, bylo provedeno skutečné měření.

První byla testována doba odezvy při simulaci zátěže na aplikaci v daném webovém frameworku. Bylo simulováno sto HTTP požadavků od jednoho uživatele. Z výsledků měření byla vybrána minimální, průměrná, maximální doba odezvy a doba odezvy v 90% percentilu měření. Následující Obrázek 32: Výsledný graf doby odezvy ve Spring frameworku. Zdroj: zobrazuje výsledky měření doby odezvy úvodní strany ve Spring frameworku:



Obrázek 32: Výsledný graf doby odezvy ve Spring frameworku. Zdroj: [autor]

Jelikož cílem bylo testování zpracování samotných webových frameworků, nebylo zapotřebí navrhované aplikace napojovat na vzdálenou databázi. Proto byla databáze spouštěna lokálně pomocí HSQLDB a tím se odstranila latence sítě. Díky tomu se výsledky měření výrazně snížili o řádově stovky milisekund. Jak je z grafu vidět, tak při testu doby odezvy úvodní stránky byla průměrná doba odezvy ve Spring frameworku pouhých 13 milisekund. Tyto výsledky byli při zaregistrovaných padesáti rezervací a při žádné registrované rezervaci by doba odezvy byla určitě menší. Následující Tabulka 1 znázorňuje výsledky testování doby odezvy úvodní stránky s padesáti zaregistrovanými rezervacemi:

Tabulka 1- Výsledky testování doby odezvy úvodní stránky. Zdroj: [autor]

	Spring framework	Apache Struts 2	Apache Wicket
Minimální (ms)	5	37	10
Průměrná (ms)	13	51	19
90% percentil (ms)	21	67	31
Maximální (ms)	27	107	44

Z výsledků testování doby odezvy úvodní stránky je patrné, že Spring framework je schopen HTTP požadavky zpracovávat rychleji než Apache Struts 2. Zejména zpracování tabulky s výpisem rezervací bylo frameworkem Apache Struts 2 provedeno o dost pomaleji. Apache Wicket je v tomto testování podobně výkonný, jako Spring framework. Z tohoto testování tedy nejlépe vyšel Spring framework, který je v průměru skoro čtyřikrát rychlejší než Apache Struts 2. Nejméně výkonný v tomto testování je Apache Struts 2, který zde hodně zaostává.

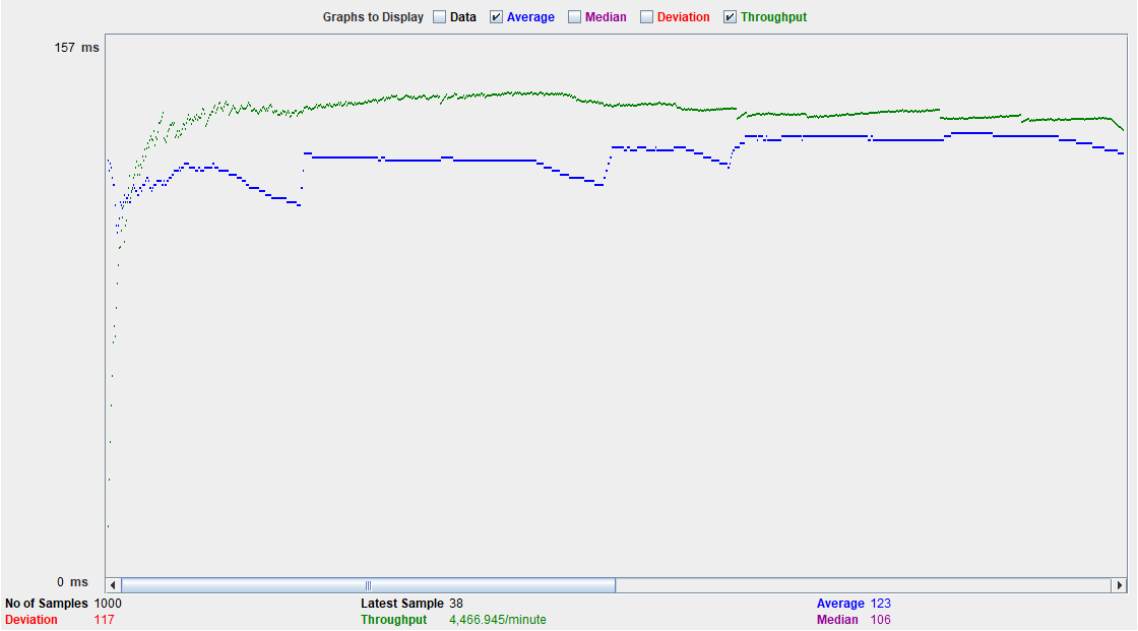
Po testování doby odezvy úvodní stránky bylo provedeno testování doby odezvy stránky pro přidávání nových rezervací. Samotná stránka s přidáváním nových rezervací není tolik rozsáhlá a díky tomu byla doba odezvy o dost menší. Následující Tabulka 2 znázorňuje výsledky testování doby odezvy stránky pro přidávání nové rezervace:

Tabulka 2- Výsledky testování doby odezvy stránky pro přidávání nových rezervací. Zdroj: [autor]

	Spring framework	Apache Struts 2	Apache Wicket
Minimální (ms)	2	2	1
Průměrná (ms)	6	7	4
90% percentil (ms)	9	17	7
Maximální (ms)	67	21	9

Z výsledků je patrné, že aplikace ve Spring framework se opět může pyšnit lepším výsledkem. Nicméně Apache Struts 2 v tomto testování nezaostává o tolik, jako při testování úvodní stránky. Překvapením v tomto testování je Apache Wicket. Ten testováním prochází dokonce ještě s lepšími výsledky než Spring framework. Rozdíly jsou, ale opravdu minimální a nemají až tak velký vliv. Nicméně Apache Struts 2 zde zaostává a prochází tímto testováním jako nejméně výkonný. Nejlépe se tedy umístil Apache Wicket po něm Spring framework a poslední je Apache Struts 2.

Druhým provedeným testováním bylo testování propustnosti. Kolik požadavků je navržená aplikace v daném webovém frameworku schopna zpracovat za jednu minutu. Následující Obrázek 33: Graf propustnosti při deseti uživatelích v Apache Struts 2. Zdroj: zobrazuje výsledky měření propustnosti úvodní strany v Apache Struts 2 při deseti uživatelích:



Obrázek 33: Graf propustnosti při deseti uživatelích v Apache Struts 2. Zdroj: [autor]

Z obrázku lze vidět propustnost při deseti uživatelích, kteří se snaží zasílat sto HTTP požadavků. Je názorně vidět, že propustnost aplikace byla na začátku nižší a postupem času se ustálila. Jelikož aplikace se snaží obsloužit více HTTP požadavků najednou, tak se tím prodlužuje doba odezvy jednotlivého HTTP

požadavku. Při takovéto zátěží vzrostla doba odezvy v průměru až na 123 ms. Je však nutné zmínit, že mezi požadavky nebyla simulována žádná časová prodleva, a proto je v tomto případě doba odezvy pouze orientační. Následující Tabulka 3 znázorňuje výsledky testování propustnosti úvodní stránky při padesáti registrovaných rezervacích:

Tabulka 3 - Výsledky testování propustnosti úvodní stránky. Zdroj: [autor]

	Spring framework (pož./min.)	**Apache Struts 2 (pož./min.)**	**Apache Wicket (pož./min.)**
1 uživatel / 100 požadavků	10 050	1 386	4 963
10 uživatelů / 100 požadavků	10 588	4 467	12 744
50 uživatelů / 100 požadavků	35 390	5 142	11 650
100 uživatelů / 100 požadavků	36 125	5 085	13 816

Propustnost Spring frameworku se opět ukazuje jako mnohem větší, než je tomu v případě Apache Struts 2. Znamená to tedy, že je schopen obsloužit mnohem více HTTP požadavků najednou. Tím se z hlediska propustnosti řadí na první pozici. Stejným výsledek oproti Spring frameworku v tomto testování dosahuje Apache Wicket, který má ale průměrně lepší propustnost než Spring framework.

Z hlediska propustnosti je tedy Spring framework značně výkonnější než zbývající testované frameworky. Při vyšší zátěži je dokonce schopen obsloužit mnohem větší počet požadavků. Druhým nejvýkonnějším je v tomto testování Apache Wicket, který je v průměru i dvakrát výkonnější než Apache Struts 2.

Z celkových výsledků, lze jasně vidět, že z hlediska testování výkonnosti a zátěže, se Spring framework řadí na první pozici. Jediné, kde lehce zaostával, bylo testování doby odezvy stránky pro přidání nové rezervace. Druhé místo pak patří frameworku Apache Wicket. Ten byl při testování doby odezvy stránky pro přidání nové rezervace dokonce výkonnější, ale s propustností si nevedl lépe než Spring

framework. Poslední, a tedy nejméně výkonný je Apache Struts 2. Ten nabízí nejmenší propustnost a průměrně nejdelší dobu odezvy.

7 Závěr

Bakalářská práce spočívala v představení a analýze webových frameworků. Webové frameworky byly představeny a v každém z nich byla navržena testovací aplikace. Ta spočívala v jednoduchém rezervačním systému do kina. Každá z těchto aplikací byla následně otestována z hlediska výkonnosti a zátěže.

Cílem bylo zjistit, jak si z hlediska výkonnosti a zátěže dané frameworky povedou. Jestli si zaslouží svojí pozici z hlediska popularity.

Spring framework v mnoha ohledech předčí zbývající testované frameworky. V testování výkonnosti a zátěže si kolikrát vedl i třikrát lépe. Z hlediska programátorského navíc nabízí výborné inicializační nástroje, jako Spring Initiliazr a celkově je velice pohodlným na programování. V testování obsadil první místo.

Apache Wicket se liší od zbývajících frameworku. Jeho programování je založeno na známém objektově orientovaném programování v jazyce Java. Pro programátory zvyklé na klasické objektově orientované programování je jednodušší na pochopení. V testování výkonnosti a zátěže přesto obsazuje druhé místo.

Apache Struts 2 obsadil třetí pozici. Z hlediska výkonnosti a zátěže i v testování propustnosti je nejhorším. Pohodlnost při programování je značně složitější oproti Spring frameworku. Apache Struts 2 navíc stále používá starší konfiguraci pomocí XML souborů. Neobsahuje žádné inicializační nástroje a dokumentace není tak přehledná.

8 Seznam použité literatury

[1] *CGI: Common Gateway Interface* [online]. 9. duben 2009 [vid. 2019-10-01]. Dostupné z: https://web.archive.org/web/20090409213905/http://hoohoo.ncsa.uiuc.edu/cgi/intro.html

[2] *Difference Between Front-End & Back-End Developer* [online]. [vid. 2020-03-08]. Dostupné z: https://www.pluralsight.com/blog/film-games/whats-difference-front-end-back-end

[3] MALÝ, Martin. REST: architektura pro webové API. *Zdroják* [online]. 2. srpen 2009 [vid. 2020-03-08]. Dostupné z: https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/

[4] DANĚK, Petr. Velký test PHP frameworků. *Root.cz* [online]. [vid. 2019-10-01]. Dostupné z: https://www.root.cz/clanky/velky-test-php-frameworku-2008/

[5] *Web application framework - DocForge* [online]. 23. červenec 2015 [vid. 2019-10-01]. Dostupné z: https://web.archive.org/web/20150723163302/http://docforge.com/wiki/Web_application_framework

[6] Session management. *Veracode* [online]. 16. duben 2018 [vid. 2019-11-18]. Dostupné z: https://www.veracode.com/security/session-management

[7] RAIBLE, Matt. HISTORY OF WEB FRAMEWORKS. In: [online]. Technology. B.m. 01:20:05 UTC [vid. 2019-11-18]. Dostupné z: https://www.slideshare.net/mraible/the-future-of-web-frameworks/11-HISTORY_OFWEB_FRAMEWORKS

[8] *Zend Framework: Documentation: Zend Framework & MVC Introduction - Zend Framework Manual* [online]. 4. listopad 2011 [vid. 2019-10-06]. Dostupné z: https://web.archive.org/web/20111104151457/http://framework.zend.com/manual/en/learning.quickstart.intro.html#learning.quickstart.intro.mvc

[9] *Web application framework: What it is, how it works, and why you need it* [online]. [vid. 2019-11-19]. Dostupné z: https://www.scnsoft.com/blog/web-application-framework

[10] MARGAM, Girish. Model-View-Controller (MVC). *Medium* [online]. 20. červenec 2018 [vid. 2019-09-29]. Dostupné z: https://medium.com/datadriveninvestor/model-view-controller-mvc-75bcb0103d66

[11] 3-Tier Architecture: A Complete Overview. *JReport* [online]. [vid. 2019-10-07]. Dostupné z: https://www.jinfonet.com/resources/bi-defined/3-tier-architecture-complete-overview/

[12] *spring.io* [online]. [vid. 2019-09-29]. Dostupné z: https://spring.io/

[13] Introduction to the Spring Framework. *TheServerSide.com* [online]. [vid. 2019-09-29]. Dostupné z: https://www.theserverside.com/news/1364527/Introduction-to-the-Spring-Framework

[14] JT. Spring Framework Annotations. *Spring Framework Guru* [online]. 20. září 2017 [vid. 2019-09-29]. Dostupné z: https://springframework.guru/spring-framework-annotations/

[15] DDOS. Apache Struts 2.5.19 release: Add support for Java 11 & Fix bugs. *InfoTech News* [online]. 1. leden 2019 [vid. 2019-11-19]. Dostupné z: https://meterpreter.org/apache-struts-mvc-framework/

[16] +YATIN. Struts2 Action Mapping Example. *Examples Java Code Geeks* [online]. [vid. 2019-09-29]. Dostupné z: https://examples.javacodegeeks.com/enterprise-java/struts-2/struts2-action-mapping-example/

[17] DEL BENE, Andrea, Martin GRIGOROV, Carsten HUFE, Christian KROEMER, Daniel BARTL, Paul BORS, Tobias SOLOSCHENKO a Joachim ROHDE. *Apache Wicket User Guide 6.x* [online]. 23. červenec 2019 [vid. 2019-09-29]. Dostupné z: https://ci.apache.org/projects/wicket/guide/6.x/guide/single.html#introduction

[18] *HSQLDB* [online]. [vid. 2020-07-16]. Dostupné z: http://hsqldb.org/

[19] *Gradle User Manual* [online]. [vid. 2020-03-03]. Dostupné z: https://docs.gradle.org/current/userguide/userguide.html

[20] *Apache Tomcat® - Welcome!* [online]. [vid. 2020-03-03]. Dostupné z: http://tomcat.apache.org/

[21] Spring. *Spring* [online]. [vid. 2020-03-03]. Dostupné z: https://www.spring.io

[22] *JPA (Java Persistence API) - Vojtěch Hordějčuk* [online]. [vid. 2020-03-02]. Dostupné z: http://voho.eu/wiki/java-jpa/

[23] CONTRIBUTORS, Mark Otto, Jacob Thornton, and Bootstrap. *Bootstrap* [online]. [vid. 2020-03-03]. Dostupné z: https://getbootstrap.com/

[24] *Spring Initializr Reference Guide* [online]. [vid. 2020-03-02]. Dostupné z: https://docs.spring.io/initializr/docs/current/reference/html/

[25] *Web on Servlet Stack* [online]. [vid. 2020-03-02]. Dostupné z: https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html

[26] ANTSTUDIO.CZ. Co je URL? | ANT studio. *ANT studio - mravenci na online* [online]. [vid. 2020-03-03]. Dostupné z: https://www.antstudio.cz/slovnik/co-je-url.htm

[27] BAELDUNG. The Spring @Controller and @RestController Annotations. *Baeldung* [online]. 2. duben 2018 [vid. 2020-03-03]. Dostupné z: https://www.baeldung.com/spring-controller-vs-restcontroller

[28] VITA. *Lekce 1 - Wicket - Seznámení a nastavení* [online]. [vid. 2020-08-02]. Dostupné z: https://www.itnetwork.cz/java-wicket-seznameni-a-nastaveni

[29] *Performance Testing Tutorial: What is, Types, Metrics & Example* [online]. [vid. 2020-07-10]. Dostupné z: https://www.guru99.com/performance-testing.html

[30] *JMeter | Testování softwaru* [online]. [vid. 2020-07-10]. Dostupné z: http://testovanisoftwaru.cz/automatizovane-testovani/jmeter/

[31] How JVM Works - JVM Architecture? *GeeksforGeeks* [online]. 20. říjen 2016 [vid. 2020-07-11]. Dostupné z: https://www.geeksforgeeks.org/jvm-works-jvm-architecture/

[32] BAELDUNG. How to Warm Up the JVM. *Baeldung* [online]. 23. červen 2017 [vid. 2020-07-11]. Dostupné z: https://www.baeldung.com/java-jvm-warmup

[33] 5 Ways to Reduce Server Response Times. *Rigor* [online]. 29. září 2017 [vid. 2020-07-14]. Dostupné z: https://rigor.com/blog/best-ways-reduce-server-response-times/

[34] *Latency and Throughput - Web Performance Tuning, 2nd Edition [Book]* [online]. [vid. 2020-07-14]. Dostupné z: https://www.oreilly.com/library/view/web-performance-tuning/059600172X/ch04s02.html

9 Seznam obrázků

10 Seznam tabulek

11 Seznam ukázek kódu

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2019/2020

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Zahradník Jan	E. Beneše 1433, Hradec Králové - Nový Hradec Králové	I1700159

TÉMA ČESKY:

Srovnávací analýza webových frameworků v jazyce Java

TÉMA ANGLICKY:

Comparative Analysis of Web Frameworks in Java

VEDOUCÍ PRÁCE:

Ing. Michal Macinka - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cílem této práce je seznámit čtenáře s webovými frameworky v jazyce Java a analýzou jejich efektivity při vývoji aplikací a jejich výslednou výkonností.

Osnova: 1. Úvod, 2. Webové frameworky, 3. Návrh testovací aplikace, 4. Testování frameworků, 5. Porovnání výsledků, 6. Závěr

SEZNAM DOPORUČENÉ LITERATURY:

Podpis studenta: ..... **Datum:** 5. 3. 19

Podpis vedoucího práce: ..... **Datum:** 5. 3. 19