УДК: 004.45(045) DOI:10.35803/1694-5298.2022.2.641-645

ТЕМИРОВ М.А., АБЫКЕЕВ К.ДЖ.

[1]КНУ им. Ж. Баласагына, Бишкек, Кыргызская Республика
[2]КГУСТА им. Н. Исанова, Бишкек, Кыргызская Республика

TEMIROV M.A., ABYKEEV K.DJ.
[1]KSU n.a.J.Balasagyn, Bishkek, Kyrgyz Republic
[2]KSUCTA n.a. N. Isanov, Bishkek, Kyrgyz Republic
karyshkyr79@mail.ru min.max@mail.ru

SPRING MVC ФРЕЙМВОРКУН КОЛДОНУП WEB ТИРКЕМЕ ИШТЕП ЧЫГУУДАГЫ АРТЫКЧЫЛЫКТАР ЖӨНҮНДӨ

ПРЕИМУЩЕСТВА РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ SPRING MVC FRAMEWORK

THE ADVANTAGES OF DEVELOPING WEB APPLICATIONS USING THE SPRING MVC FRAMEWORK

Макалада ар бир корпоративдик Java-тиркемелерди иштеп чыгууда окшош эле баптоолорду кайталап аткарууга туура келет. Web-тиркемелерди түзүүдө ***Spring Boot технологиясын колдонуу программист үчүн баптоо жана программдык кодду жазуу жагынан көп жеңилдиктерди берет. Макалада, spring boot фреймворкун колдонуп mvc моделинде web тиркеме түзүү жана анын башка технолгияларга салыштырмалуу артыкчылыктары каралды.***

Өзөк сөздөр: ***Spring Boot технологиясы;*** *MVC модели; Китепканалар; HTTP GET суроо-талабы; Фреймворк.*

В статье рассматриваются преимущества создания веб-приложения в модели mvc с использованием фреймворка spring boot и его преимущества перед другими технологиями. Использование технологии Spring Boot при создании веб-приложений дает программисту массу преимуществ с точки зрения настройки и написания программного кода. Вам потребуется повторять одни и те же настройки при разработке каждого корпоративного Java-приложения.

Ключевые слова: *технология Spring Boot; модели MVC; библиотеки; HTTP GET-запрос; Фреймворк.*

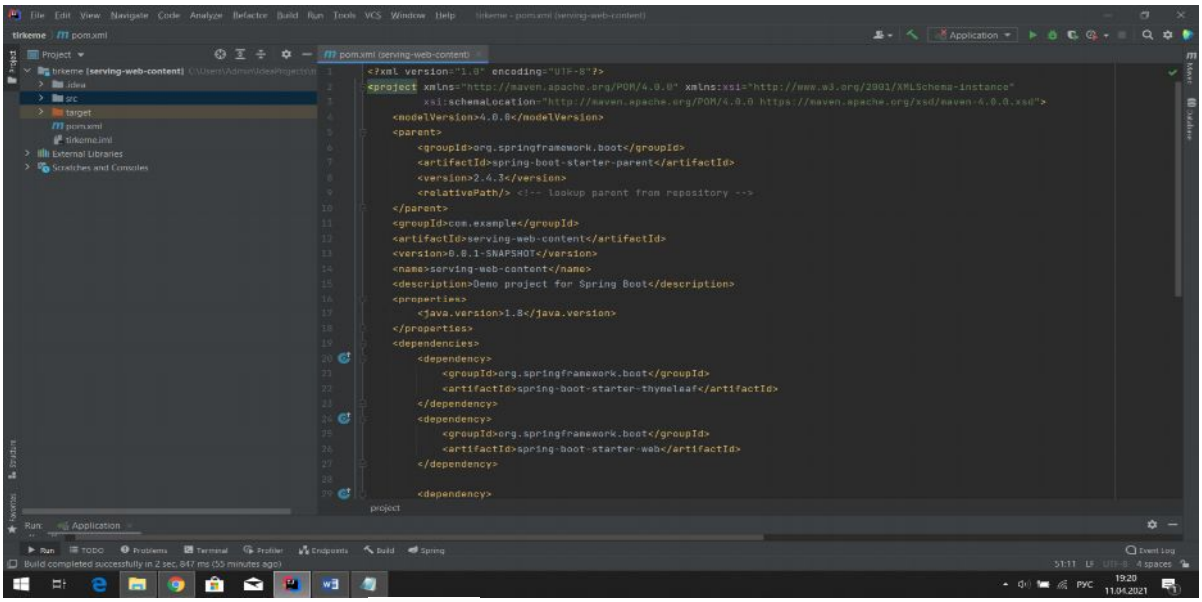
This article discusses the advantages of creating a web application in the mvc model using the spring boot framework and its advantages over other technologies. Using Spring Boot technology when creating web applications gives the programmer a lot of advantages in terms of customizing and writing program code. You will need to repeat the same settings every time you develop an enterprise Java application.

Key words: *Spring Boot technology; MVC models; libraries; HTTP GET request; Framework.*

Түзүлүп жаткан тиркемеге карата Spring-модулдардын керектүүсүн импорттоо керек. Мисалы, web-тиркеме түзүү учурунда web-контейнерлер китепканалары (библиотекалары) импорттолот. Эгер тышкы Hibernate, Jackson сыяктуу китепканалар импорттолсо, анда алар Spring технологиясынын версиясына туура келүүсү абзел.

MVC (англисче model-view-controller) моделинде spring boot фреймворкун колдонуп web тиркеме түзүү процессин жана этаптарын карап көрөлү. Тиркемени бул ыкма менен уюштурууда, программдык кодду, ар кандай маселелерди чечүүчү өзүнчө блокторго бөлүү сунушталат. Биринчи блок тиркеменин берилиштерине, экинчиси көрүнүшүнө, үчүнчүсү тиркеменин ишин көзөмөлдөөгө жооп берет.

Түзүлүп жаткан тиркеме, HTTP GET суроо-талабын http://localhost:8080/page дареги боюнча кабыл алган статикалык баракты камтыйт. Биринчиден, IntelliJIDEA аттуу иштеп чыгуу аймагынынын (IDE) жардамы менен Maven 3.2+ аспабын колдонуп проект түзөбүз. Ал эми, компилятор катары JDK 1.8 версиясынан өйдө колдоно берсек болот. Проект түзүлгөндөн кийин төмөнкүдөй көрүнүшкө ээ болот:



1-сүрөт. POM.XML файлынын түзүлүшү

1-сүрөттө pom.xml файлынын түзүлүшү көрүнүп турат. Бирок бул жерде кээ бир керектүү плагиндер жок. Spring Boot плагиндерин [1] жана керектүү баптоолорду жасагандан кийин pom.xml файлы төмөнкү көрүнүшкө ээ болот:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>2.4.3</version>
                <relativePath/> <!-- lookup parent from repository -->
        </parent>
        <groupId>com.example</groupId>
        <artifactId>serving-web-content</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>serving-web-content</name>
        <description>Demo project for Spring Boot</description>
        <properties> <java.version>1.8</java.version>
        </properties> <dependencies>          <dependency>
                        <groupId>org.springframework.boot</groupId>
```

ABBYY PDF Transformer 2.0 Click here to buy www.ABBYY.com

```
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>       <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>       <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>       <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>       </dependencies>
    <build> <plugins>   <plugin>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>       </plugins>      </build>
</project>
```

Ар бир жаңы плагин кошулгандан кийин IDE автоматтык түрдө жаңыланбаса анда, колго жаңылоого туура келет.

Эми MVC модели боюнча керектүү класстарды түзүү керек. Spring проектисинде HTTP-суроо-талаптар (запростор) текшергич (контроллер) аркылуу ишке ашат. Демек, текшергич классын түзөбүз жана аны - addController деп атайлы. Бул класс төмөнкү иерархия менен түзүлүшү керек: src/main/java/com/example/ servingwebcontent/appController.java. Төмөнкү листингде текшергичтин программасы келтирилген:

```
package com.example.servingwebcontent;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
@Controller
public class appController {
    @GetMapping("/page")
    public String page(@RequestParam(name="name", required=false, defaultValue="students of KSUCTA") String name, Model model) {
        model.addAttribute("name", name);
        return "index";     } }
```

Жогорудагы листингде көрүнүп тургандай appController классы, page ыкмасы үчүн GET суроо-талабын ишке ашырып жыйынтыгын көрүнүшкө кайтарып берет. Ал эми index.html файлы HTML маалыматты браузерге чыгарууга жооп берет. Текшергич классы жөнөкөй жана кыска жазылганы менен, көп кызмат аткарат. Эгер аткарган кызматына токтоло кетсек, бул жерде @GetMapping аннотациясы HTTP GET суроо-талабынын жыйынтыгы, page ыкмасында көрүнүүсүн камсыз кылат. Ал эми, @RequestParam аннотациясы page ыкмасынын "name" параметри менен саптык name суроо-талапбынын маанисин байланыштырат. Эгер бул параметрдин мааниси жок болсо анда, defaultValue (баштапкы маани) мааниси колдонулат. name параметринин мааниси Model объектисине берилет жана көрүнүү шаблонуна ачык, жеткиликтүү болот.

Жыйынтыктарды чыгаруу үчүн HTML баракчаны колдонобуз. Ал үчүн, resources папкасынына templates папкасын түзөбүз жана ушул папканын ичине HTML файл түзүп атын index.html деп атайбыз. Бул файлга төмөнкү кодду жазабыз:

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Getting Started: Serving Web Content</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
<p th:text="'Hello, ' + ${name} + '!'" />
</body>
</html>
```

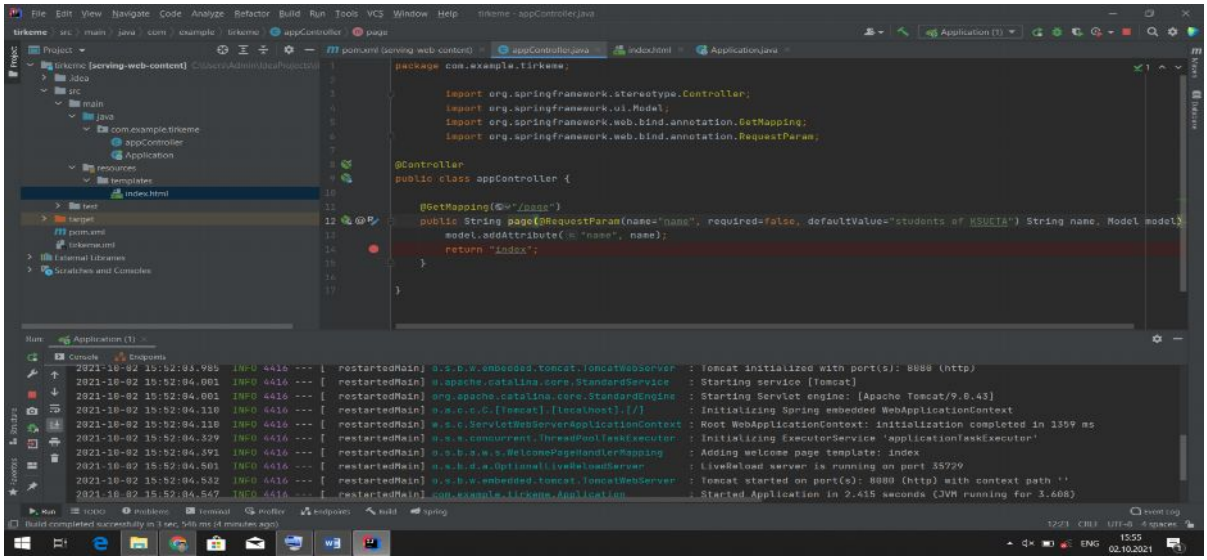
Сиздер байкагандай бул жерде кадимки HTML документтин түзүлүшү жазылган. Бирок, сервер тарабынан жиберилген жыйынтыкты HTMLде ишке ашырылыш ыкмалары Thymeleaf технологиясынын негизинде жүзгө ашырылууда. Бизде, контроллер классында name параметринин мааниси катары “students of KSUCTA” жазылып турат, ал эми абзацта th:text=’Hello’ деген маани бар. Демек, тиркемени иштеткенде браузерде “Hello students of KSUCTA” деген сап чыгат.

Программанын кирүү чекитин аныктаган классты түзүүгө киришели. Бул классты Application деп атап жана java/com.example.tirkeme/ иерархия боюнча жайгаштырып, андан соң, төмөнкү кодду жазалы:

```
package com.example.tirkeme;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

main() ыкмасы тиркемени иштетиш үчүн SpringApplication.run() ыкмасын колдонот.

Тиркемени иштетүү үчүн intellijIdea ишеп чыгуу аймагынын башкы тандоо сабынан Run баскычын басып жана Application классын көрсөтүшүбүз керек. Проект ийгиликтүү компиляциядан өткөндөн кийин, бизге төмөнкү маалыматты көрсөтөт:

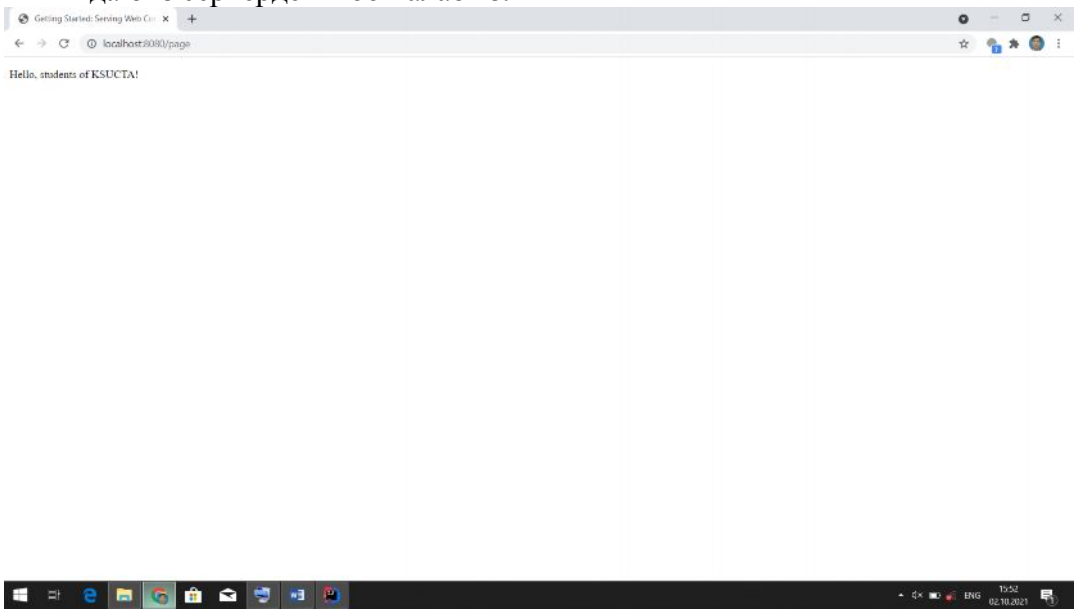


2-сүрөт. Tomcat сервери

2-сүрттөгү акыркы эки сап Tomcat сервери port(s): 8080 портунда ишке кириштиги жана тиркеме 2.415 секундда жүктөлдү деген билдирүүнү берип жатат (JVM running for 3.608).

Бул маалыматтарды алгандан кийин, браузерди ачып даректик сапка http://localhost:8080/page деп жазабыз. Бул жерде, localhost: деген жазуу сервер локалдык

экенин жана анын кирүүчү суроо-талаптарды кабыл алуучу портунун номуру 8080 экенидигин билдирет. Ал эми, page болсо жогоруда биз түзгөн ыкманын аты. Жыйынтыгында биз серверден жооп алабыз:



3- сүрөт. Серверден жооп

3-сүрөттө көрүнүп тургандай, серверден жооп ийгиликтүү алынды. Демек, проект туура түзүлдү, калган иш ар бир проектин өзүнүн талабына, логикасына карата өнүгүп, программдык коддор жазылат.

Сиздер байкагандай, проекти ишке ашырууда бир дагы XML программдык сап жазылган жок. Ошондой эле web.xml файлын дагы түзгөн жокпуз. Башка технологияларды колдонуп web тиркемелерди түзүүдө сервер үчүн XML программдык коддорду жазууга туура келмек. Жогорудагы web тиркеме 100% таза Java кодунан турат жана бул жерде проекттин түзүлүшүн тургузууда жана сервердик жагын баптоодо көнүмүш коддорду кайталап жазуу менен алек болуунун кажети жок. Бул программистке, ыңгайлуулукту тартуулап, дароо тиркеменин программасын жазууга тез киришүү мүмкүнчүлүгүн түзөт.

Адабияттар тизмеси

1. Официальный интернет ресурс фреймворка spring boot. [Электронный ресурс] режим доступа: URL: https://spring.io/
2. Джошуа Блох. Java эффективное программирование [Текст] / Джошуа Блох. - Киев: Лори, 2014. –435 с.
3. Гупта Арун. Java EE 7. [Текст] / Гупта Арун. - М.: Вильямс, 2014. - 336 с.
4. Гарнаев А. WEB-программирование на Java и JavaScript [Текст] / А. Гарнаев. - Москва: СПб. [и др.] : Питер, 2017. - 718 с.