**МИНИСТЕРСТВО ТРАНСПОРТА И КОММУНИКАЦИЙ РЕСПУБЛИКИ БЕЛАРУСЬ**

**Учреждение образования**

**«Белорусский государственный университет транспорта»**

**Кафедра «Управление эксплуатационной работой и охрана труда**

**Отчёт  
по лабораторным работам**

по дисциплине «Средства и технологии анализа и разработки информационных систем»

Выполнил Проверил

Студент группы ГИ-31 м.т.н., ст.пр.

Долинова В. А. Козлов В. Г.

Гомель 2024

**ЛАБОРАТОРНАЯ РАБОТА №8  
ТЕМА: РАЗРАБОТКА ЭФФЕКТИВНЫХ ВЕБ-ПРИЛОЖЕНИЙ (SPRING MVC)**

**Цель работы:** разработка веб-приложений с реализацией шаблона MVC.

**Задание**

А) Создать проект Spring MVC и подключить к нему все необходимые зависимости.

Б) Создать дополнительный контроллер и представление (JSP) с отображением полей объекта класса из лабораторной работы №7.

В) Прокомментировать код программы.

**Листинг**

//pom.xml

<!-->Добавление зависимостей для работы со Spring-MVC<-->  
<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>4.3.9.RELEASE</version>  
</dependency>

//web.xml

<!-->HTTP-запросы перенаправляются на сервелет, создание сервелета<-->  
 <servlet>  
 <servlet-name>dispatcher</servlet-name>  
 <!-->класс, к-ый используется для создания сервелета из библиотеки спринг<-->  
 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
 <init-param>  
 <!-->указание в качестве параметра пути, в к-ом находится спринг конфигурация<-->  
 <param-name>contextConfigLocation</param-name>  
 <param-value>/WEB-INF/applicationContextMVC.xml</param-value>  
 </init-param>  
 <load-on-startup>1</load-on-startup>  
 </servlet>

//applicationContextMVC.xml

<!-->файл, в к-ом лежит спринг приложение в папке WEB-INF<-->  
 <bean class = "org.springframework.web.servlet.view.InternalResourceViewResolver">  
 <!-->указание того, где лежит представление (/music)<-->  
 <property name = "prefix" value = "/WEB-INF/music/" />  
 <!-->указание расширения представления (.jsp)<-->  
 <property name = "suffix" value = ".jsp" />  
 </bean>

//music.jsp

<%-- Представление в виде html-документа, которое получает данные от Контроллера и отображает их в браузере --%>  
<html style="background-color: #FFCCFF; text-align: center;">  
<head>  
 <title>Музыкальные произведения</title>  
</head>  
<body>  
 <h2 style="color: #666666; font-size:42px; font-family: Monospace;">Музыкальные колллекции</h2>  
 <%-- отображение значений по заданному ключу в контроллере %-->  
 <c:forEach items="${musiccarrier}" var="music">  
 <div style=" border: 2px solid #666699; font-family: Arial;">  
 <h3 style="color: #9999FF; font-size:34px;">${music.number} Альбом</h3>  
 <p style="color: #330066; font-size:24px;">Название: ${music.name}</p>  
 <p style="color: #330066; font-size:24px;">Автор: ${music.author}</p>  
 <p style="color: #330066; font-size:24px;">Жанр: ${music.genre}</p>  
 <p style="color: #330066; font-size:24px;">Год выпуска: ${music.year}</p>  
 </div>  
 </c:forEach>  
</body>  
</html>

//MusicController.java

//Аннотация, которая помечает класс контроллер (в таком классе можно обрабатывать поступающие HTTP-запросы)  
//Контроллер получает данные Модели MisicCarrier и отдает их в Представление music.jsp  
@Controller  
//Аннтоация для сопоставления HTTP-запросов с исполняемыми методами  
@RequestMapping("/music") //обращение по URL /music  
public class MusicConroller {  
 //маппинг, к-ый говорит о том, что в методе будут обрабатываться поступившие запросы на адрес music  
 @RequestMapping(method = RequestMethod.*GET*)  
 //метод, к-ый обращается к созданной модели и получает данные (параметры указанные внедряют модель в метод)  
 public String displayMusic(ModelMap model) {  
 List<MisicCarrier> musiccarrier = new ArrayList<MisicCarrier>();  
 musiccarrier.add(new MisicCarrier("The Slim Shady LP", "Eminem", "Хип-хоп", 1999, 1));  
 musiccarrier.add(new MisicCarrier("Starboy", "The Weekend", "Поп", 2016, 2));  
 //обращение к модели по принципу ключ значение  
 model.addAttribute("musiccarrier", musiccarrier);  
 //возврат представления в методе  
 return "music";  
 }  
}

//MisicCarrier.java

//java класс - модель, которая отдает данные контроллеру  
public class MisicCarrier {  
 //создание полей  
 private int number;  
 private String name;  
 private String author;  
 private String genre;  
 private int year;  
  
//создание конструктора класса  
 public MisicCarrier(String name, String author, String genre, int year, int number) {  
 this.name = name;  
 this.author = author;  
 this.genre = genre;  
 this.year = year;  
 this.number = number;  
 }  
  
 //реализация сеттеров и геттеров для получения и установки значений  
  
 public int getNumber() {  
 return number;  
 }  
  
 public void setNumber(Integer number) { this.number = number;}  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getAuthor() {  
 return author;  
 }  
  
 public void setAuthor(String author) {  
 this.author = author;  
 }  
  
 public String getGenre() {  
 return genre;  
 }  
  
 public void setGenre(String genre) {  
 this.genre = genre;  
 }  
  
 public int getYear() {  
 return year;  
 }  
  
 public void setYear(Integer year) { this.year = year;}  
  
}

**Результат**

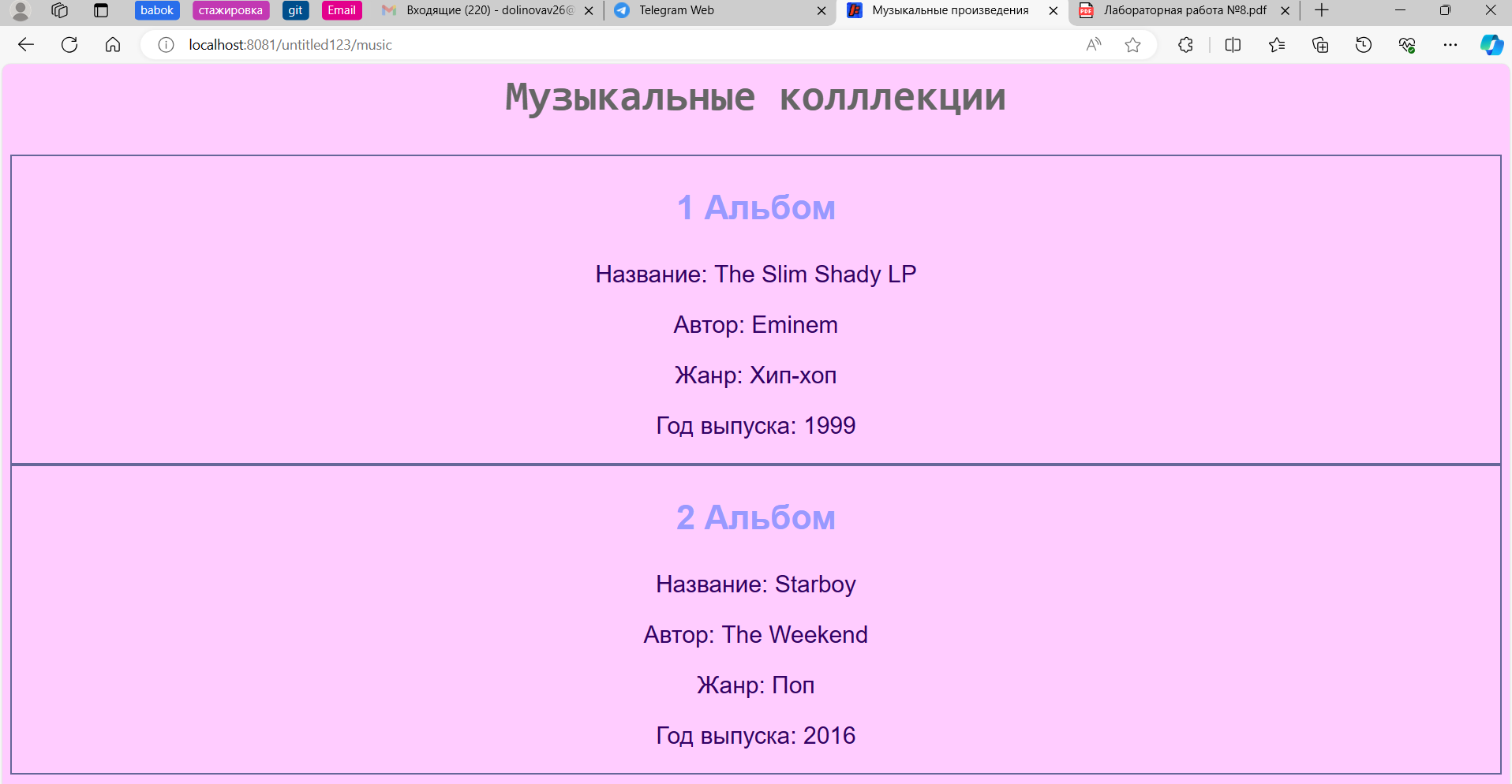
****

Рисунок 1 – **Разработанное веб-приложение**

На рисунке 1 представлено веб-приложение, которое создано на базе фреймворка SpringMVC, опираясь на шаблон модель (Model) – представление (View) – контроллер (Controller).

Как уже ранее упоминалось, разработанное приложение, которое отображает данные о музыкальных коллекциях, а именно: название альбома, его автора, жанр и год выпуска, – содержит модель MisicCarrier.java, представление music.jsp и контроллер MusicController.java.

Так, структура созданного приложения отображена на рисунке 2.

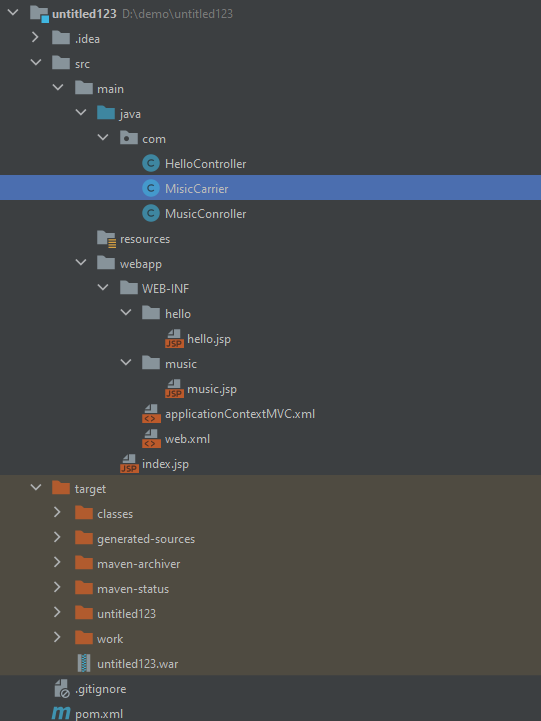


Рисунок 2 – **Структура веб-приложения**

Как видно из рисунка 2, приложение содержит все необходимые компоненты для реализации шаблона MVC.

**Вывод:** Таким образом, разработано веб-приложение с реализацией шаблона MVC. В ходе работы создан проект Spring MVC и подключены все необходимые зависимости. Кроме того, для реализации шаблона MVC был создан контроллер, реализована модель и разработано представление. Также были рассмотрены различные аннотации, которые упрощают разработку веб-приложения.