Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Кафедра «Информатика»

Практический пример №5
Spring MVC

Составил: Старший преподаватель кафедры «Информатика»
Черниговский Алексей Сергеевич

Красноярск 2020

Содержание

1 Создание Maven проекта

Создаем новый Maven проект в IDE Eclipse из архетипа webapp

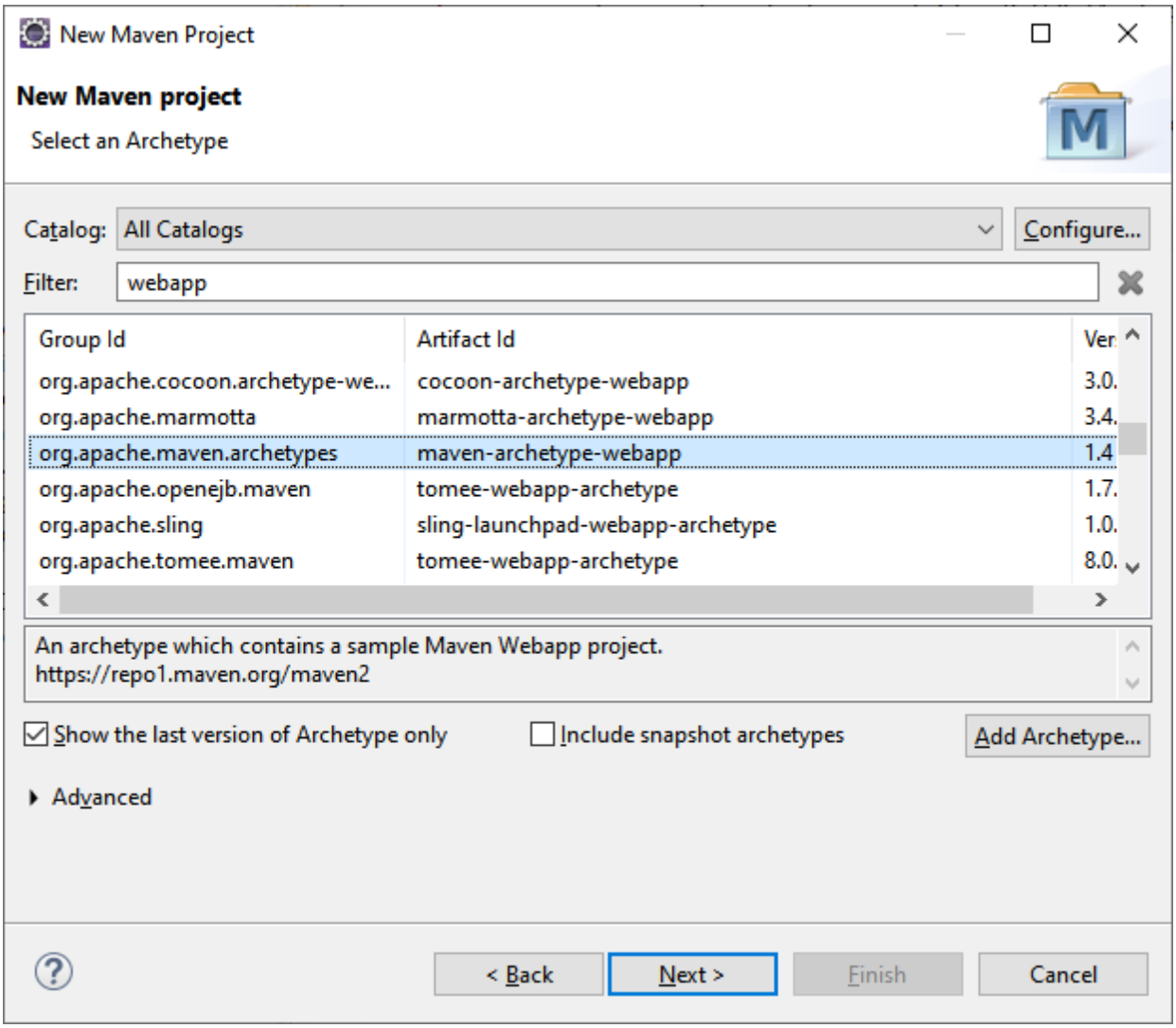

Рисунок 1 — webapp artifact

Зададим ему Artifact Id: spring-mvc-app.

2 Подключение сервера Tomcat

Переходим по ссылке и скачиваем нужную вам версию. https://tomcat.apache.org/download-80.cgi

Здесь и далее будет также использоваться НЕ инсталлер, а разархивированные файлы Apache Tomcat.

Необходимо указать переменную среды CATALINA_HOME, которая бы указывала на папку куда был распакован архив Tomcat.

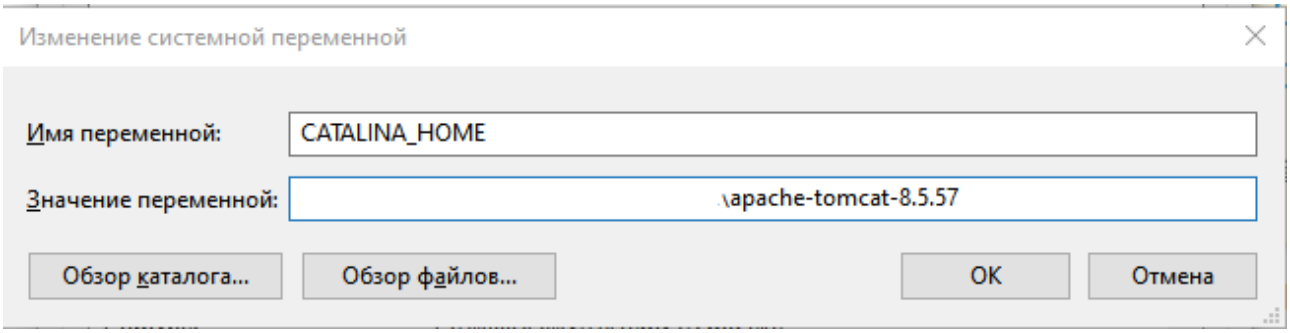

Рисунок 2 — Установка переменной CATALINA_HOME

И переменную JAVA_HOME, которая бы указывала на путь к исполняемой среде Java (JRE).

Зачастую порт 8080 бывает занят другими приложениями, для того чтобы изменить порт сервера по умолчанию перейдите в conf\server.xml и измените <Connector port="8080" на 8081.

В папке bin Tomcat запустите startup.bat, запустится сервер Tomcat. Чтобы убедиться, что все запущено успешно откройте в вашем веб-браузере следующую страницу http://localhost:8081 должна появиться страница следующего содержания:

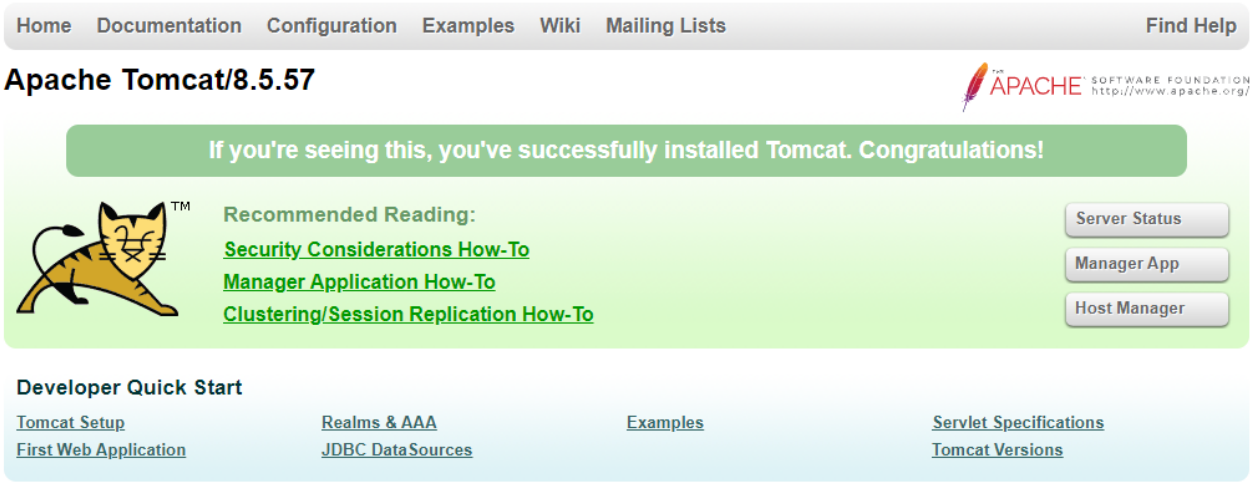

Рисунок 3 — Начальная страница Apache Tomcat

Это означает что Apache Tomcat запустился успешно.

3 Интеграция Eclipse с Tomcat

В вашем проекте в нижней части Eclipse переходим на вкладку Servers, нажимаем по ссылке создания нового сервера (Риунок 4)

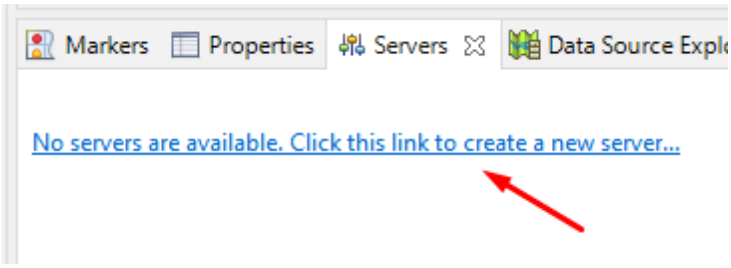

Рисунок 4 — Создание нового сервера

Выбираем Apache Tomcat v8.5 Server

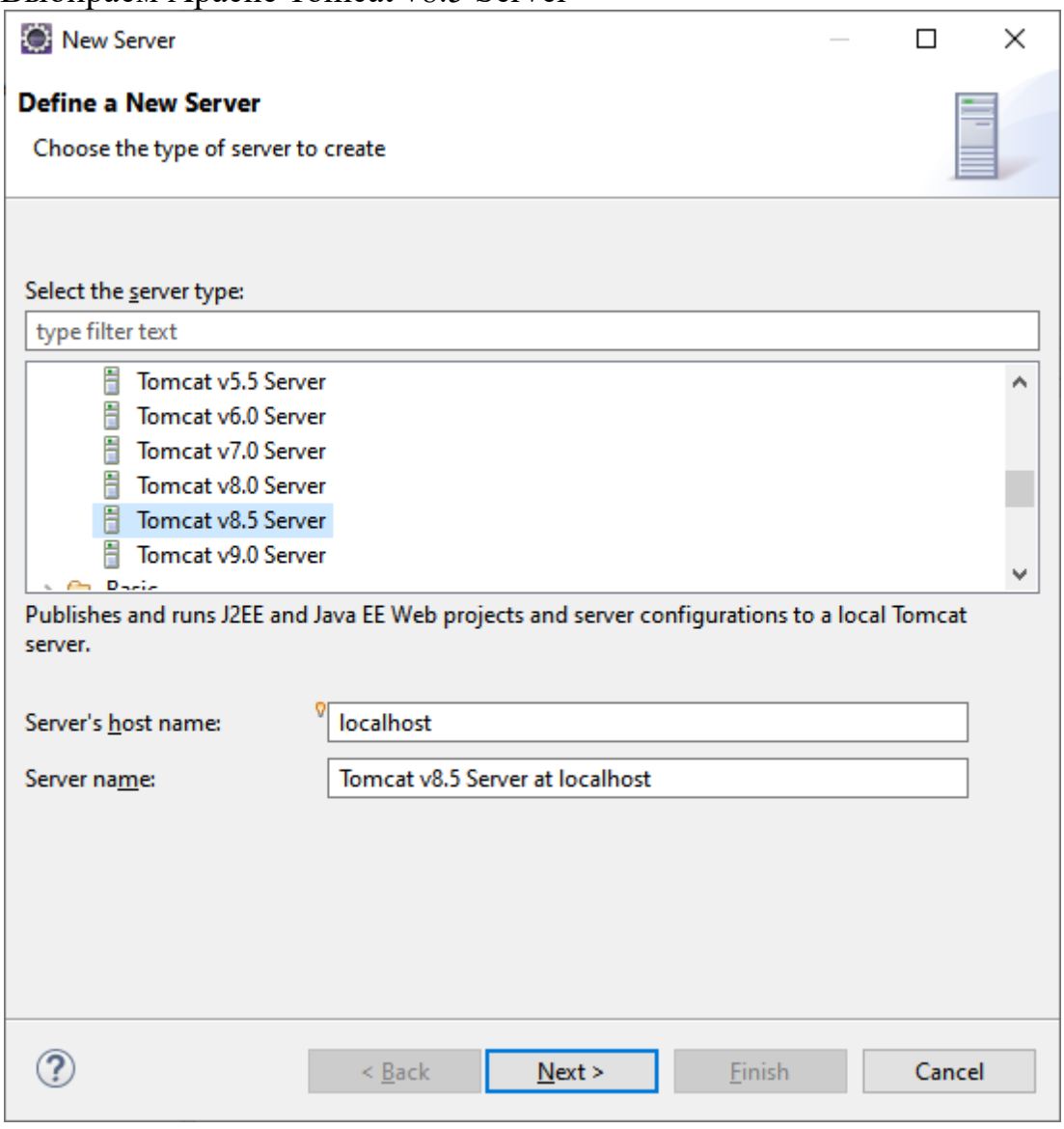

Рисунок 5 — Apache Tomcat v8.5 Server

Указываем путь до папки с Apache Tomcat и JRE.

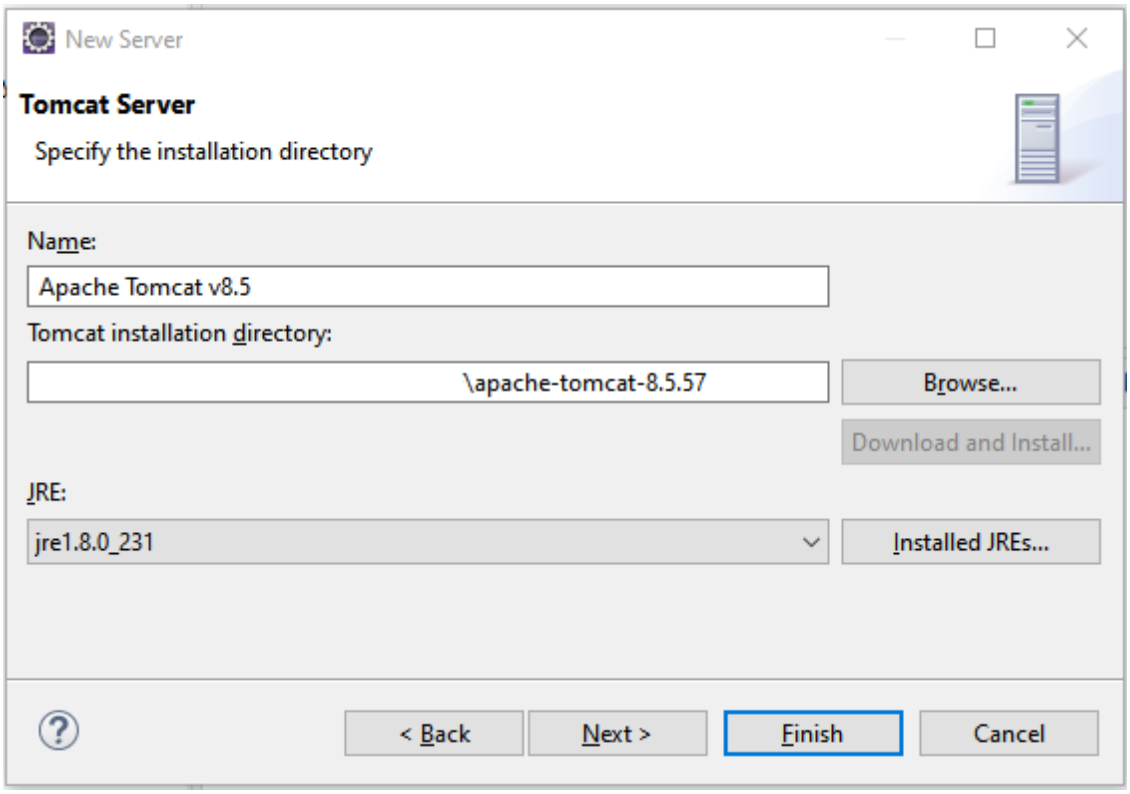

Рисунок 6 — Настройка путей для сервера

Теперь сервер можно запускать по кнопке.

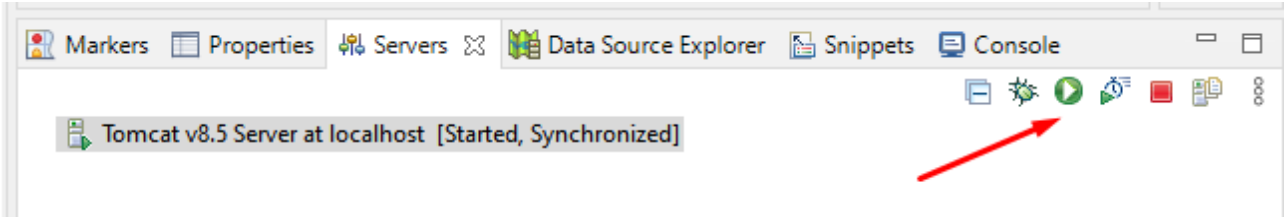

Рисунок 7 — Запуск сервера

Теперь, для запуска нашего Spring Web MVC проекта достаточно кликнуть правой кнопкой по проекту и запустить его на сервере

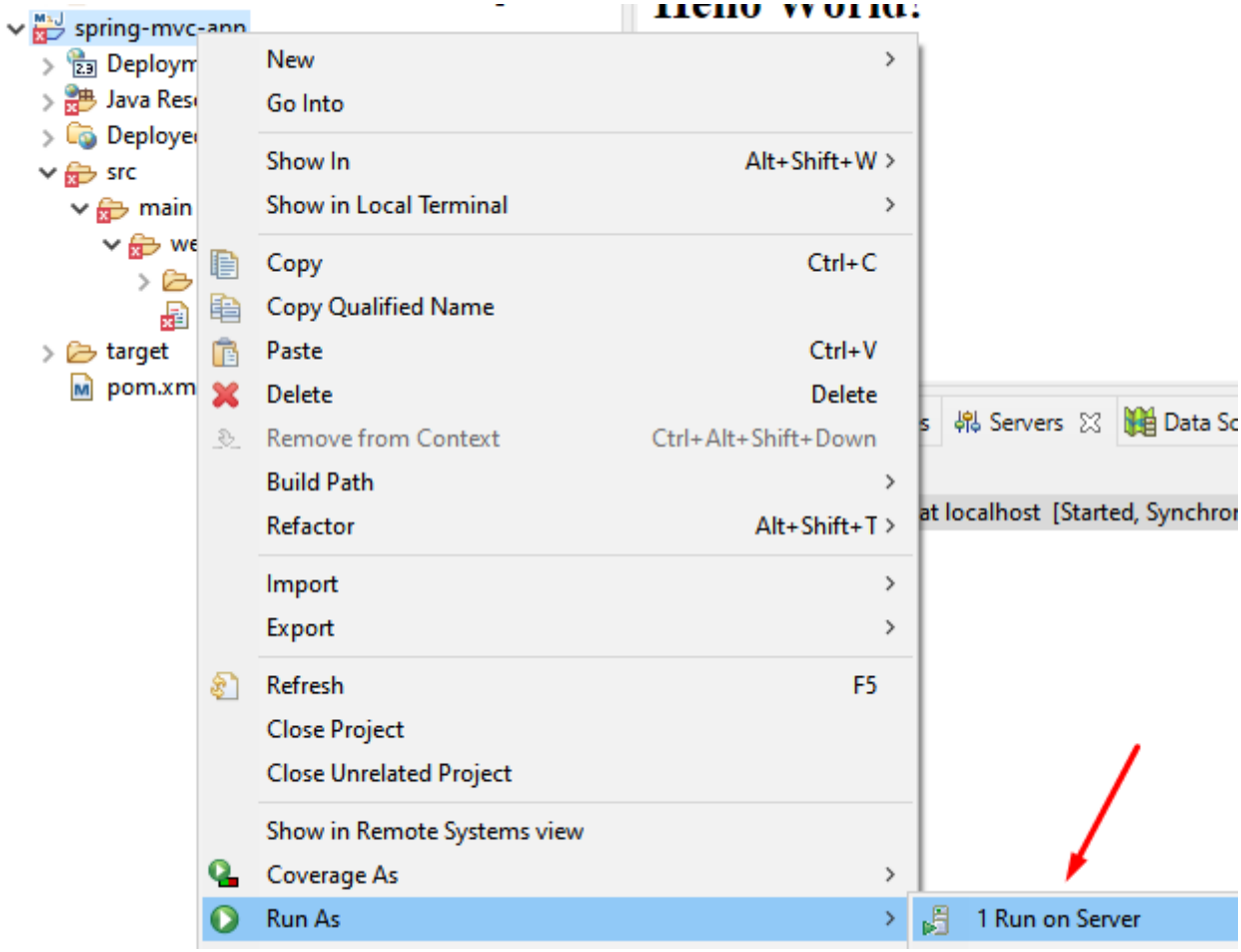

Рисунок 8 — Запуск проекта на сервера

В открывшемся окне сразу же нажимаем Finish и видим результат запуска нашего приложения:

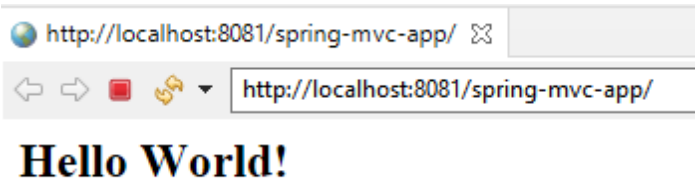

Рисунок 9 — Результат, простейшее веб-приложение

4 Зависимости

Добавим в pom.xml зависимости Spring Core, Spring Context, Spring Web, Spring Web MVC и Apache Freemarker (он нам понадобится в дальнейшем).

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>5.2.8.RELEASE</version>
```

```
      </dependency>

    <dependency>
          <groupId>org.springframework</groupId>
          <artifactId>spring-context</artifactId>
          <version>5.2.8.RELEASE</version>
      </dependency>

      <dependency>
          <groupId>org.springframework</groupId>
          <artifactId>spring-web</artifactId>
          <version>5.2.8.RELEASE</version>
      </dependency>

      <dependency>
          <groupId>org.springframework</groupId>
          <artifactId>spring-webmvc</artifactId>
          <version>5.2.8.RELEASE</version>
      </dependency>
      <dependency>
          <groupId>org.freemarker</groupId>
          <artifactId>freemarker</artifactId>
          <version>2.3.30</version>
      </dependency>
```

5 Конфигурация приложения

Теперь необходимо сконфигурировать web.xml. Сервер при запуске считывает содержимое этого файла, здесь мы описываем то что будет делать сервер. Вставим следующее содержимое:

```
      <?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         id="WebApp_ID" version="3.1">

    <display-name>spring-mvc-app</display-name>

    <servlet>
        <servlet-name>dispatcher</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:config/applicationContextMVC.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>dispatcher</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

      </web-app>
```

Создадим applicationContextMVC.xml, где будет находиться конфигурация Spring приложения. Добавим код следующего содержания:

```
      <?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="
             http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd
      http://www.springframework.org/schema/context
      http://www.springframework.org/schema/context/spring-context.xsd
      http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd">

    <mvc:annotation-driven/>

    <context:component-scan base-package="controllers"/>

    <mvc:annotation-driven/>

   <bean id="freeMarkerConfig" class="org.springframework.web.servlet
             .view.freemarker.FreeMarkerConfigurer">
             <property name="templateLoaderPath" value="/WEB-INF/views"/>
             <property name="defaultEncoding" value="UTF-8"/>
             <property name="freemarkerSettings">
                    <props>
                           <prop key="default_encoding">UTF-8</prop>
                    </props>
             </property>
      </bean>

    <bean id="templateResolver" class="org.springframework.web.servlet.
             view.freemarker.FreeMarkerViewResolver">
         <property name="suffix" value=".ftl"/>
         <property name="contentType" value="text/html; charset=utf-8"/>
         <property name="cache" value="false"/>
    </bean>
```

Тег context:component-scan уже должен быть вам известен.

Тег mvc:annotation-driven включает необходимые аннотации для Spring MVC приложения.

Добавим папку для нашего Java кода (как в первой практике), в ней создадим новый класс HelloController в пакете ru.testmvcapp со следующим содержимым:

```
      import org.springframework.stereotype.Controller;
      import org.springframework.web.bind.annotation.GetMapping;

      @Controller
      public class HelloController {
      @GetMapping("/hello-world")
      public String sayHello() {
```

```
        return "hello_world!";
}
}
```

Теперь создадим представление. Создадим папку views в папке WEB-INF, а в ней hello_world.ftl.

Со следующим содержимым:

```
<html>
<body>
        <h2>Привет мир!</h2>
</body>
</html>
```

Теперь перезапустим наш сервер, перейдем по url /hello-world:

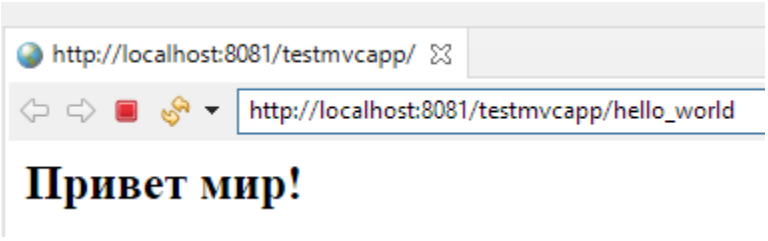

Рисунок 10 — запуск простейшего приложения

Обо всем и многом другом читайте в конспекте лекции!