**Spring MVC Form example.**

В этом проекте показан базовый пример использования фреймворка Spring MVC 3+, а именно обработка форм.

**Запуск примера:**

Для запуска примера вам нужно иметь:

- Исходники проекта. Их можно скачать с Github репозитория: <https://github.com/Rudeg/Spring-MVC-Example>

**-** Одну из современных IDE для работы с Java. (IntelliJ IDEA, Eclipse, Netbeans).

- Tomcat версии 7.\*

Вам нужно скачать исходники проекта с репозитория. После этого, выполнить импорт проекта в IDE. Далее настроить автоматический деплой на аппликейшн сервер(Tomcat). После этих действий проект должен заработать.

Видео запуска проекта в разных IDE:

* Idea: <http://screencast.com/t/x1ISPAaEx>
* Eclipse: <http://screencast.com/t/huO1WyaLCD>
* Netbeans: <http://screencast.com/t/LYXUpCwADX>

*Запуск проекта без IDE:*

Для запуска проекта без IDE вам потребуется:

- Apache Maven версии 3.\*

- Tomcat версии 7.\*

Для начала нужно собрать исходники проекта в war файл. Это выполняется с помощью Mavena(команда mvn package). Чтобы это выполнить, нужно перейти в папку с исходниками проектами (в ней находится .pom файл) и выполнить команду “mvn package”.

После этого должна появиться папка Target, в которой будет war файл SpringExample.war.

Теперь этот файл нужно скопировать в папку Tomcata - webapp. Это делается для того, чтоб при запуске томката он автоматически развернул war файл и запустил его. Tomcat запускается скриптом startup из папки bin.

После всех этих действий, проект будет доступен по адрессу <http://localhost:port/warname> , где

port – порт на котором запускается Tomcat; warname – имя War файла.

**Создание нового проекта:**

Изначально создаем новый Maven модуль из уже существующего артефакта, с примером базового веб приложения:

org.apache.maven.archetypes: maven-archetype-webapp.

После создания проекта, мы должны изменить POM файл, добавив в него используемые библиотеки. После изменений POM файл должен стать таким:

<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 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>SpringExample</artifactId>

<packaging>war</packaging>

<version>1.0-SNAPSHOT</version>

<name>SpringExample</name>

<properties>

<spring.version>3.2.3.RELEASE</spring.version>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<!--Spring-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<!--JSP-->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

<scope>provided</scope>

</dependency>

</dependencies>

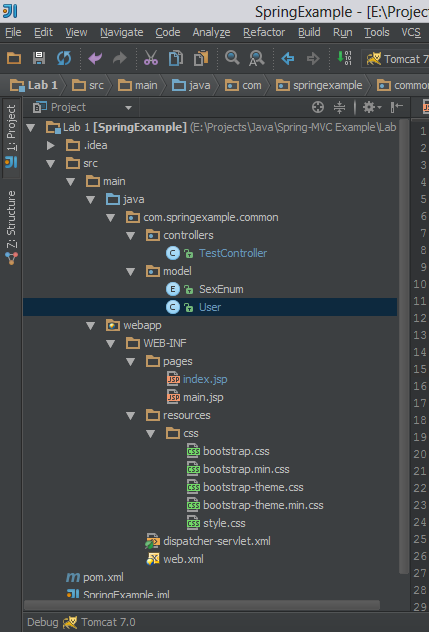
<build>

<finalName>SpringExample</finalName>

</build>

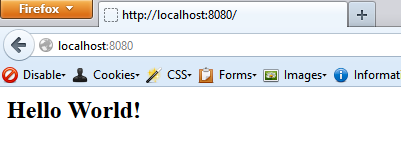
</project>

В окончательном варианте структура проекта должна получится приблизительно такой:



После этого мы можем настроить деплой нашего приложения на апликейшн сервер – Tomcat (настраивается в зависимости от используемой IDE).

Если всё прошло успешно, то при запуске приложения вы должны получить:



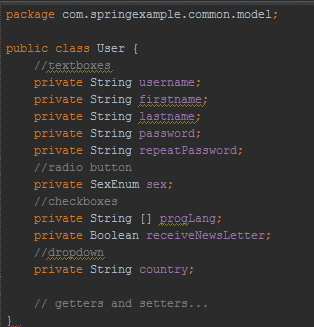
После этого создаем папку java, в которой будут храниться наши контроллеры и сущности.

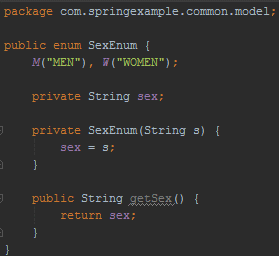
Создаем там 2 пакета:

com.springexample.common.controllers

com.springexample.common.model

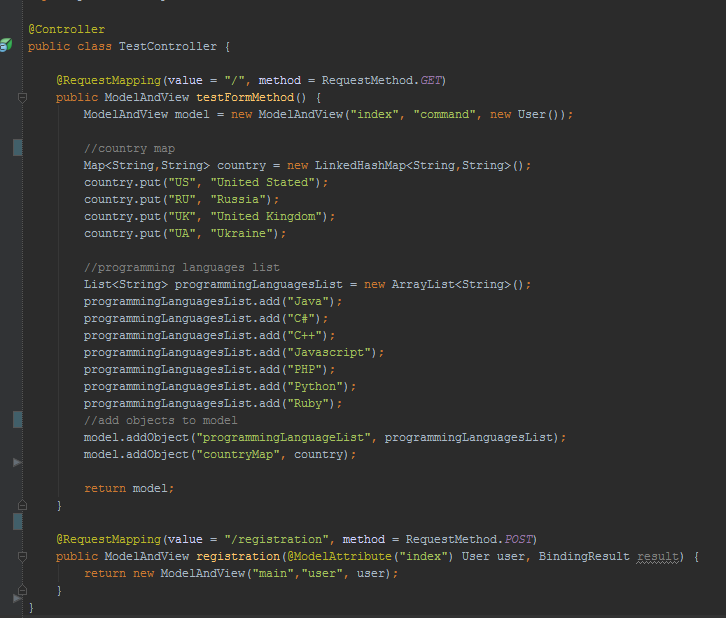
Для начала создадим в пакете model класс и энум для нашей основной entity:





В класс users нужно также добавить геттеры и сеттеры с помощь стандартных средств IDE.

После этого в пакете controllers создаем класс TestController, и исходным кодом:

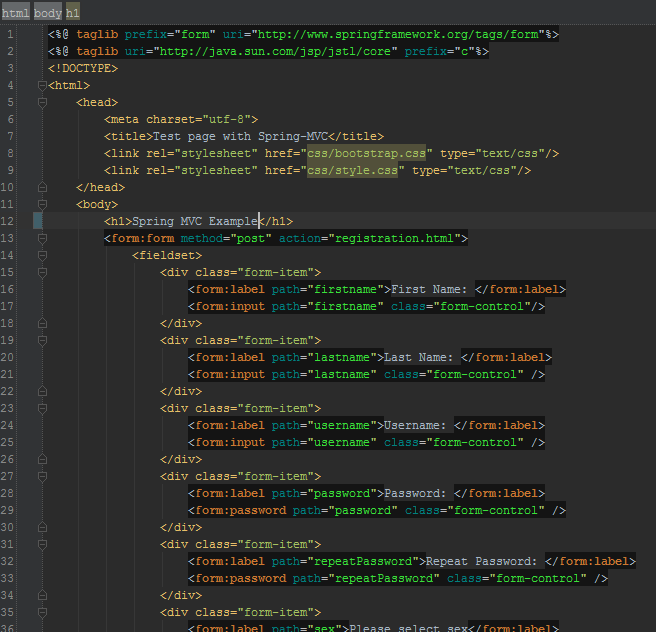


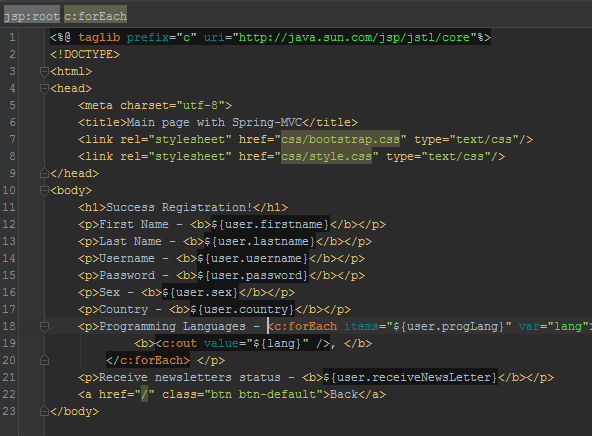
Этот класс отвечает за обработку запросов. С помощью аннотации @Controller мы указываем что этот класс – контроллер. А с помощью аннотаций @RequestMapping(…) – указываем какой запрос(путь + тип) будет обрабатывать данный метод.

После этих действий мы можем преступать к созданию наших View файлов. В папке WEB-INF создаем папку pages, в которой у нас будут лежать страницы. У нас будет 2 странички:

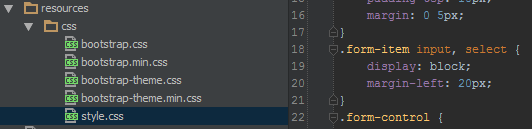
1. index.jsp – страница регистрации с формой
2. main.jsp – страница с данными после регистрации

Весь исходный код страниц можно посмотреть на гитхабе.



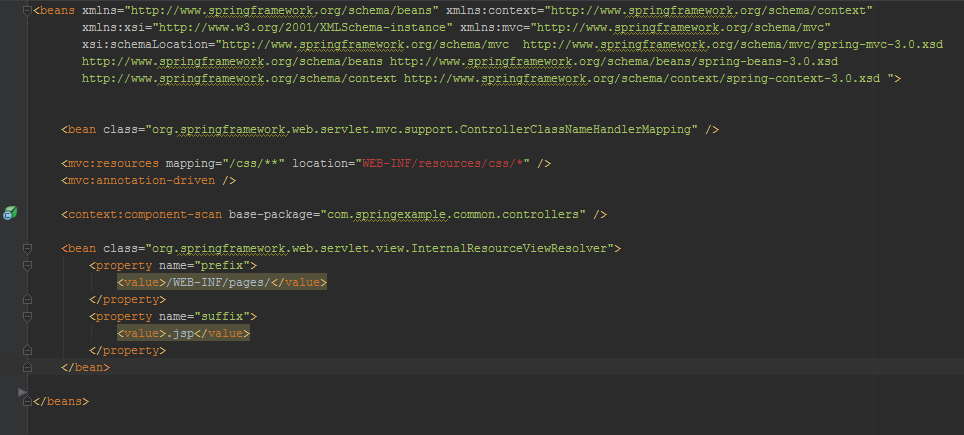


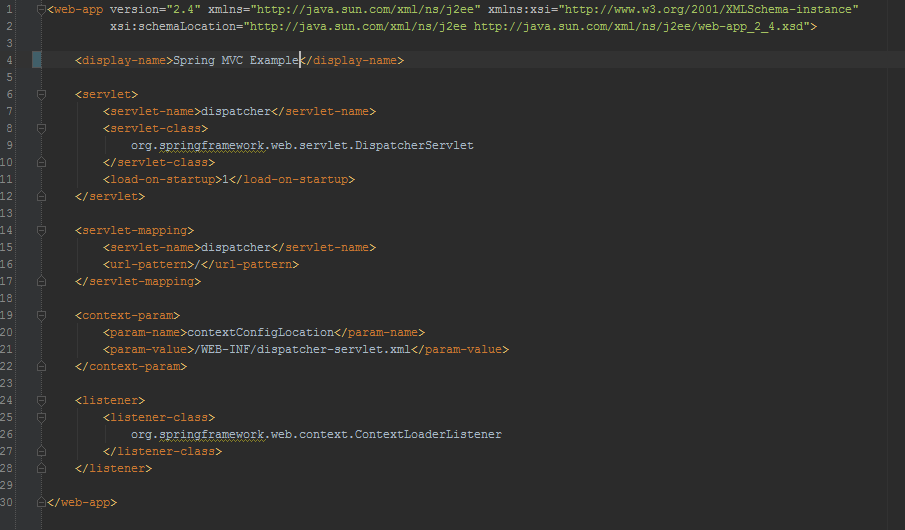
Далее создаем папку Resources, которая будет содержать все подгружаемые в html файлы ресурсы, такие как css стили и javascript файлы.



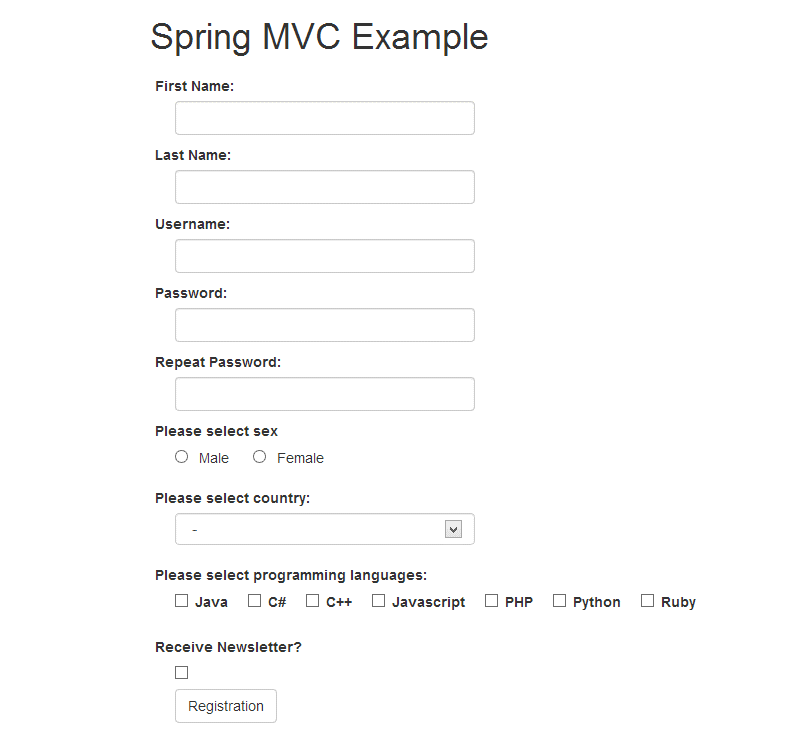
В примере используется фреймворк bootstrap для более красивой стилизации. Его использование по желанию.

После всего этого в папке WEB-INF нужно создать конфигурационный файл для фреймворка Spring – dispatcher-servlet.xml. В нём мы указываем путь к файлам jsp, путь к ресурсам, и также что контроллеры у нас указаны с помощью аннотаций.



После этих действий, осталось изменить конфигурационный файл web.xml. В нём мы указываем класс, который будет выполнять роль Dispatcher Servleta и путь к нашему конфигурационному файлу Spring фреймворка. 

После всех этих действий, можно пробовать запустить приложение. При правильной настройке всего, у вас должно получится что-то по типу:



И после сабмита формы, вы должны получить вот такой результат:

