Руководство по Spring. Spring MVC Framework (основы).

Целью SpringMVC Framework является поддержка в Spring архитектуры модель-представление-контроллер (model-view-controller).  Spring обеспечивает готовые компоненты, которые могут быть использованы (и используются) для разработки веб-приложений.

Главной целью MVC является разделение объектов, бизнес-логики и внешнего вида приложения. Все эти компоненты слабо связаны между собой и при желании мы можем изменить, например, внешний вид приложения, не внося существенные изменения в остальные два компонента.

Итак, давайте пройдемся по каждому из этих блоков.

**Модель (Model)**

Этот блок инкапсулирует данные приложения. На практике это POJO-классы (Plain Old Java Objects – Простые старые Java-объекты).

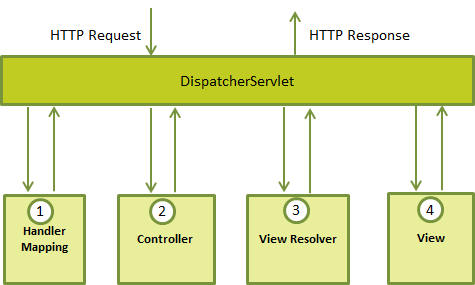
**Представление (View)**

Модуль представления отвечает за вывод данных пользователю. Обычно это JSP файл, который может быть опознан и интерпретирован браузером на пользовательской машине.

**Контроллер (Controller)**

Контроллер отвечает за обработку запросов пользователей и передачу данных модулю **View**для обработки.

**В основе Spring MVC Framework лежит DispatcherServlet, задача которого – обработка всех HTTP запросов и ответов.** В понимании DispatcherServlet нам поможет следующий рисунок:

[](https://i2.wp.com/proselyte.net/wp-content/uploads/2016/02/spring_dispatcherservlet.png)

После получения HTTP-запроса DispatcherServlet (далее – DS) выполняет следующие действия.

1. После получения HTTP-запроса DispatcherServlet даёт указание объекту Handling Mapping (обработка связывания), который вызывает следующий объект.
2. DS посылает запрос контроллеру и вызывает соответствующие методы, в основе которых лежат методы GET и POST. Эти методы возвращают объект, в соответствии с бизнес-логикой метода и передают название (название ссылки) обратно в DS.
3. C помощью View Resolver, DS подбирает необходимый вид для запроса.
4. И, когда внешний вид сформирован, DS передаёт эти данные в модуль View, который обрабатывается браузером пользователя.

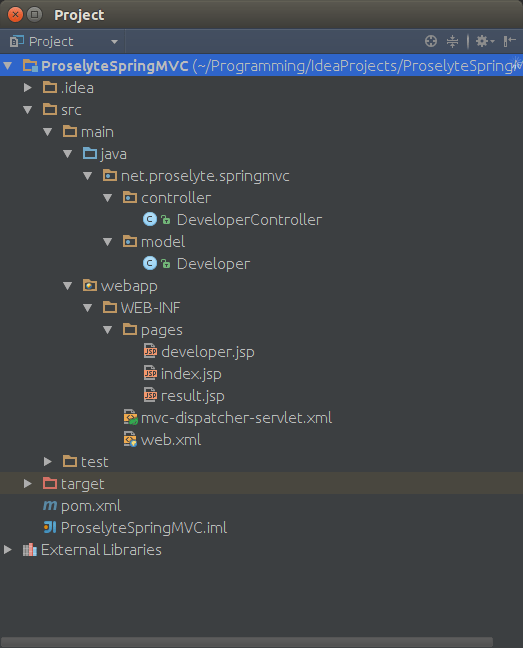
Все компоненты, указанные в рисунке, являются частями **WebApplicationContext**, который является расширением ApplicationContext + некоторые дополнительные функции.

Для того, чтобы понять, как всё это работает вместе, предлагаю рассмотреть пример простого приложения с использованием Spring MVC Framework.

Пример приложения:

Исходный код проекта можно скачать по [*ЭТОЙ ССЫЛКЕ*](https://proselyte.net/wp-content/uploads/2016/02/ProselyteSpringMVC.zip).

Структура проекта

[](https://i2.wp.com/proselyte.net/wp-content/uploads/2016/02/proselyteSpringMVCStructure.png)

*Блок Model*

**Developer.java**

package net.proselyte.springmvc.model;

public class Developer {

private int id;

private String name;

private String specialty;

private int experience;

getter/setter

@Override

public String toString() {

return "Developer{" +

"id=" + id +

", name='" + name + '\'' +

", specialty='" + specialty + '\'' +

", experience=" + experience +

'}';

}

}

*Блок Controller*

**DeveloperController**

package net.proselyte.springmvc.controller;

import net.proselyte.springmvc.model.Developer;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.servlet.ModelAndView;

@Controller

public class DeveloperController {

@GetMapping("/")

public String index() {

return "/index";

}

@GetMapping("developer")

public ModelAndView developer() {

return new ModelAndView("developer", "command", new Developer()); //тут параметры следующие: имя файла вида(\_\_.jsp, *а может быть \_.html*) куда передается атрибут, имя передаваемого атрибута, значение атрибута

}

@PostMapping("/addDeveloper")

public String addStudent(@ModelAttribute("mvc-dispatcher") Developer developer,

ModelMap model) {

model.addAttribute("id", developer.getId());

model.addAttribute("name", developer.getName());

model.addAttribute("specilaty", developer.getSpecialty());

model.addAttribute("experience", developer.getExperience());

return "result"; //тут "result" это имя файла вида (*result.jsp*)

}

}

**web.xml**

<web-app version="2.4"

xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee

http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd">

<display-name>Spring MVC Application</display-name>

<servlet>

<servlet-name>mvc-dispatcher</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>mvc-dispatcher</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

</web-app>

**mvc-dispactcher-servlet.xml**

<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"

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">

<context:component-scan base-package="net.proselyte.springmvc.controller"/>

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/WEB-INF/pages/"/>

<property name="suffix" value=".jsp"/>

</bean>

</beans>

*Блок View*

**index.jsp**

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>

<head>

<title>Home Page</title>

</head>

<body>

<h3><a href="/developer">Add Developer</a></h3>

</body>

</html>

**developer.jsp**

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<html>

<head>

<title>Developer</title>

</head>

<body>

<h2>Enter developer information</h2>

<form:form method="post" action="addDeveloper">

<table>

<tr>

<td><form:label path="id">Id</form:label></td>

<td><form:input path="id" /></td>

</tr>

<tr>

<td><form:label path="name">Name</form:label></td>

<td><form:input path="name" /></td>

</tr>

<tr>

<td><form:label path="specialty">Specialty</form:label></td>

<td><form:input path="specialty" /></td>

</tr>

<tr>

<td><form:label path="experience">experience</form:label></td>

<td><form:input path="experience" /></td>

</tr>

<tr>

<td colspan="2">

<input type="submit" value="Submit"/>

</td>

</tr>

</table>

</form:form>

</body>

</html>

**result.jsp**

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<html>

<head>

<title>Developer Info</title>

</head>

<body>

<h2>Developer Information</h2>

<table>

<tr>

<td>Id</td>

<td>${id}</td>

</tr>

<tr>

<tr>

<td>Name</td>

<td>${name}</td>

</tr>

<tr>

<td>Specialty</td>

<td>${specilaty}</td>

</tr>

<tr>

<td>Experience</td>

<td>${experience}</td>

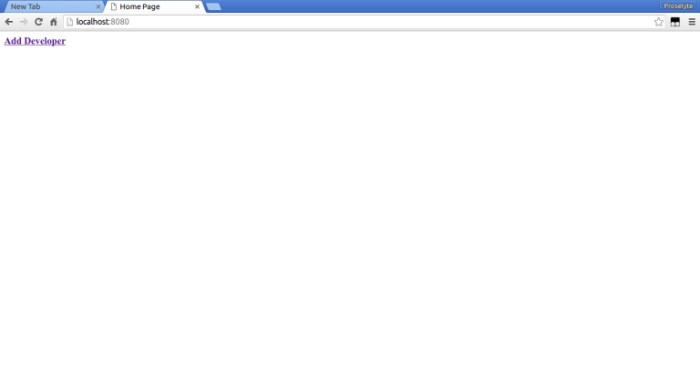
</tr>

</table>

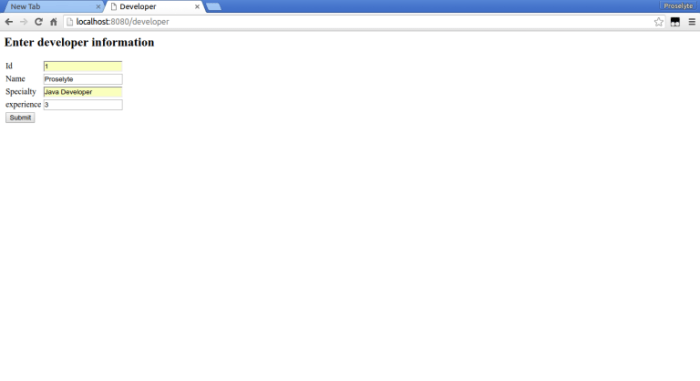
</body>

</html>

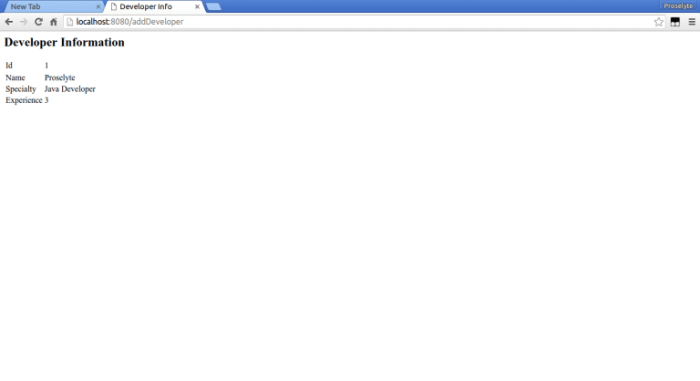
**Результат работы программы**

Главная страница  
[](https://i0.wp.com/proselyte.net/wp-content/uploads/2016/02/index.jsp_.png)

Добавление разработчика

[](https://i2.wp.com/proselyte.net/wp-content/uploads/2016/02/addDeveloper.png)

Вывод информации

[](https://i0.wp.com/proselyte.net/wp-content/uploads/2016/02/developerInfo.png)

В этой статье мы ознакомились с основами Spring MVC Framework и создали небольшое веб-приложение.