PROSELYTE

Записки задумчивого программиста

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).

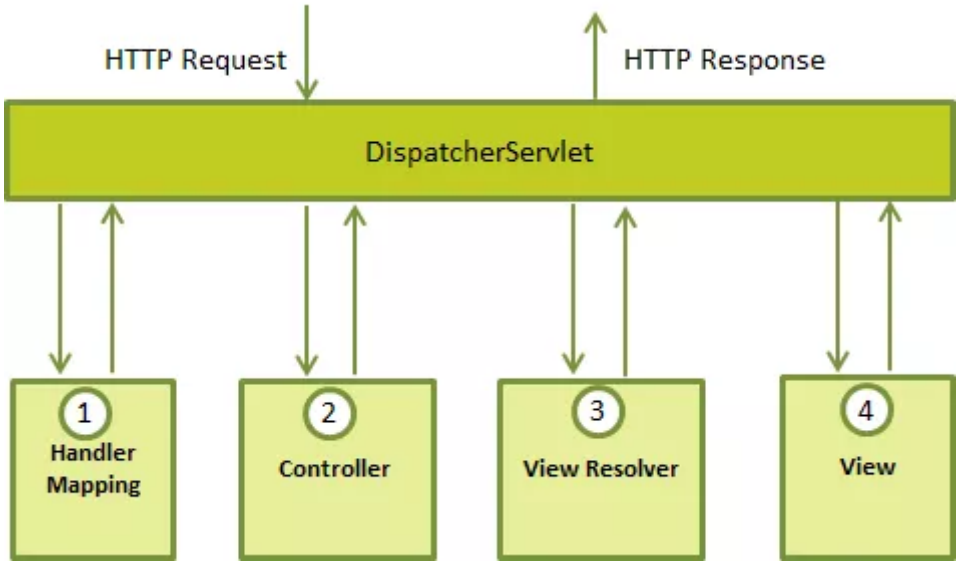
Представление (View)

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:



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. С 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 +

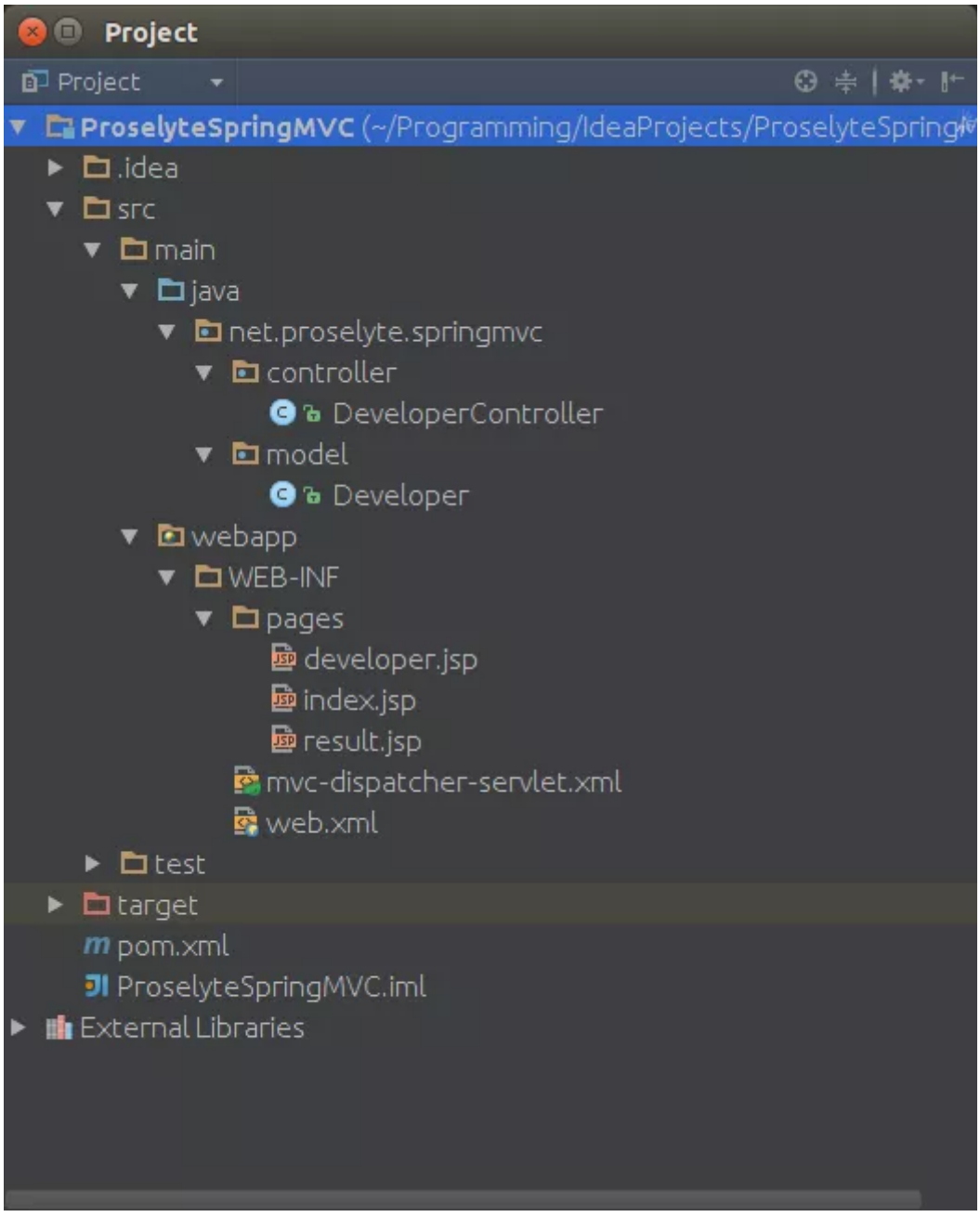
некоторые дополнительные функции.

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*.

Структура проекта



Блок Model

Developer.java

```
package net.proselyte.springmvc.model;

public class Developer {
    private int id;
    private String name;
    private String specialty;
    private int experience;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSpecialty() {
        return specialty;
    }

    public void setSpecialty(String specialty) {
        this.specialty = specialty;
    }

    public int getExperience() {
        return experience;
```

```
        }

        public void setExperience(int experience) {
            this.experience = experience;
        }

        @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 {
    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String index() {
```

```
            return "/index";
        }
        @RequestMapping(value = "developer", method = RequestMethod.GET)
        public ModelAndView developer() {
            return new ModelAndView("developer", "command", new Developer());
        }

        @RequestMapping(value = "/addDeveloper", method = RequestMethod.POST)
        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";
        }
    }
```

web.xml

```
<web-app version="2.4"
    xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-inst
    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: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>
```

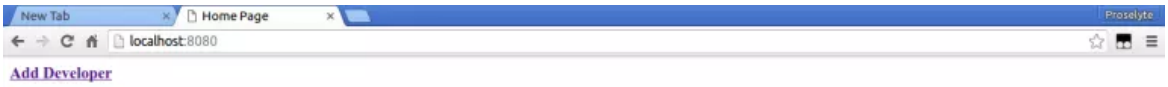
```
</body>
</html>
```

result.jsp

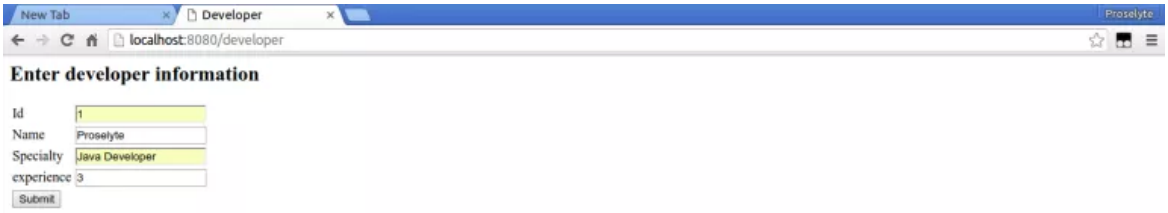
```
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<html>
<head>
  <title>Developer Info</title>
</head>
<body>

<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

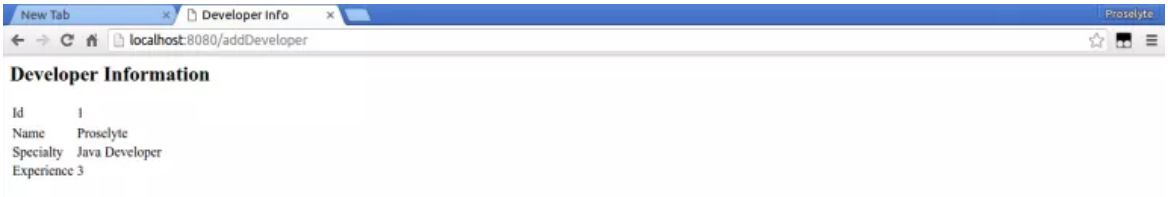
Главная страница



Добавление разработчика



Вывод информации



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

Поиск ...

Руководство по Spring

Полезности

Туториалы
Видео
Книги
Статьи

Немного о себе

Приветствую! Меня зовут Евгений. На этом сайте я пишу о вещах, которые мне интересны (программирование, книги, спорт), а иногда, и просто о жизни. Вы можете связаться со мной, отправив письмо на мой email:

proselytear@yahoo.com Имеет смысл, предварительно ознакомиться вот с этим FAQ разделом.

Недавние публикации

Полезность: руководство по Servlets

Полезность: руководство по Scala

Что такое AOT компиляция и с чем её едят?

Как не вылететь из IT через 5 лет. Часть 2 – Язык программирования.

Полезность: руководство по MongoDB