



2 Часть. Spring MVC Framework SimpleFormController - обработка форм

[Статьи](#)

Наши спонсоры

Для обработки форм в Spring вы должны расширить ваш контроллер от SimpleFormController класса. Здесь мы будем создавать регистрационную форму для понимания как это все работает. Если вы используете Spring 3 и выше, то применяйте аннотации.

```
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.SimpleFormController;

import com.vaannila.domain.User;
import com.vaannila.service.UserService;

@SuppressWarnings("deprecation")
public class UserController extends SimpleFormController {

    private UserService userService;

    public UserController() {
        setCommandClass(User.class);
        setCommandName("user");
    }

    public void setUserService(UserService userService) {
        this.userService = userService;
    }

    @Override
    protected ModelAndView onSubmit(Object command) throws Exception {
        User user = (User) command;
        userService.add(user);
        return new ModelAndView("userSuccess", "user", user);
    }
}
```

Мы используем Spring 3. Здесь мы расширяем UserController от SimpleFormController, это позволяет классу контроллеру обрабатывать формы. Обычно форму связывают с каким-нибудь доменным объектом, в нашем случае это класс User. В Spring этот доменный объект имеет название command object (командный объект). Чтобы передать командный объект в jsp страницу мы должны установить command class (командный класс) с помощью setCommandClass() метода в конструкторе. User класс имеет свойство - имя, и чтобы передать его в jsp страницу вы должны будете использовать "command.name". Вы также можете менять это имя, используя setCommandName() метод. Для доступа к имени пользователя в jsp странице мы используем "user.name".

Мы должны иметь метод для управления формой, когда она считается отправленной. Для этих целей используется метод – onSubmit(). Этот метод у нас имеет доступ к command object User, который мы добавляем в сервисный класс и в конце возвращаем ModelAndView объект.

Все поля формы будут отправляться как String в контроллер формы (form controller). Spring имеет несколько зарегистрированных редакторов свойств для конвертирования String значений в подходящий тип данных.

```
public class User {
```

```
private String name;
private String password;
private String gender;
private String country;
private String aboutYou;
private String[] community;
private Boolean mailingList;

public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public String getPassword() {
    return password;
}
public void setPassword(String password) {
    this.password = password;
}
public String getGender() {
    return gender;
}
public void setGender(String gender) {
    this.gender = gender;
}
public String getCountry() {
    return country;
}
public void setCountry(String country) {
    this.country = country;
}
public String getAboutYou() {
    return aboutYou;
}
public void setAboutYou(String aboutYou) {
    this.aboutYou = aboutYou;
}
public String[] getCommunity() {
    return community;
}
public void setCommunity(String[] community) {
    this.community = community;
}
public Boolean getMailingList() {
    return mailingList;
}
public void setMailingList(Boolean mailingList) {
    this.mailingList = mailingList;
}
}
```

Это наш User Service интерфейс:

```
import com.vaannila.domain.User;

public interface UserService {

    public void add(User user);
}
```

Наша User Service реализация:

```
public class UserServiceImpl implements UserService {

    @Override
    public void add(User user) {
        //Persist the user object here.
        System.out.println("User added successfully");
    }
}
```

Теперь давайте создадим регистрационную форму, используя Spring form tags (теги формы). Перед тем как использовать эти теги мы должны импортировать Spring библиотеку с этими тегами.

```
<%@ taglib uri="http://www.springframework.org/tags/form"
prefix="form"%>
<html>
<head>
<title>Registration Page</title>
</head>
<body>
<form:form method="POST" commandName="user">
    <table>
        <tr>
            <td>User Name :</td>
            <td><form:input path="name" /></td>
        </tr>
        <tr>
            <td>Password :</td>
            <td><form:password path="password" /></td>
        </tr>
        <tr>
            <td>Gender :</td>
            <td><form:radio button path="gender" value="M" label="M" /> <form:radio button
                path="gender" value="F" label="F" /></td>
        </tr>
        <tr>
            <td>Country :</td>
            <td><form:select path="country">
                <form:option value="0" label="Select" />
                <form:option value="1" label="India" />
                <form:option value="2" label="USA" />
                <form:option value="3" label="UK" />
            </form:select></td>
        </tr>
        <tr>
            <td>About you :</td>
            <td><form:textarea path="aboutYou" /></td>
        </tr>
        <tr>
            <td>Community :</td>
            <td><form:checkbox path="community" value="Spring"
                label="Spring" /> <form:checkbox path="community" value="Hibernate"
                label="Hibernate" /> <form:checkbox path="community" value="Struts"
                label="Struts" /></td>
        </tr>
        <tr>
            <td></td>
            <td><form:checkbox path="mailingList"
                label="Would you like to join our mailinglist?" /></td>
        </tr>
        <tr>
            <td colspan="2"><input type="submit"></td>
        </tr>
    </table>
</form:form>
</body>
</html>
```

Здесь атрибут path используется для связывания полей формы с доменным объектом. Здесь мы используем HTTP Post метод при отправке формы. Для связывания полей формы с доменным объектом command object должен иметь одно и то же имя в jsp странице и в классе - контроллере. Здесь мы используем commandName form tag.

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
```

```

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
id="WebApp_ID" version="2.5">
<display-name>SpringExample6</display-name>
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.
        DispatcherServlet </servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>*.htm</url-pattern>
</servlet-mapping>
<welcome-file-list>
    <welcome-file>redirect.jsp</welcome-file>
</welcome-file-list>
</web-app>

```

Далее создаем конфигурационный файл Spring Bean.

```

<?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:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="viewResolver"
        class="org.springframework.web.servlet.view. InternalResourceViewResolver"
        p:prefix="/WEB-INF/jsp/" p:suffix=".jsp" />

    <bean id="userService" class="com.vaannila.service.UserServiceImpl" />

    <bean name="/userRegistration.htm" class="com.vaannila.web.UserController"
        p:userService-ref="userService" p:formView="userForm" p:successView="userSuccess" />

</beans>

```

Как вы могли заметить, мы использовали пространство имен "p". Это очень легко использовать. Оно позволяет устанавливать свойства не с помощью элементов, а с помощью атрибутов.

Когда HTTP GET запрос, форма будет визуализироваться. Когда форма отправлена (HTTP POST запрос) будет вызван метод onSubmit() в UserController классе. При удачной загрузке метода successView будет визуализирован. В случае ошибок formView автоматически все отобразит пользователю.

Пример нужно запускать через указание redirect.jsp файла. Ссылка redirect.jsp перенаправит нас к "userRegistration.htm".

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<% response.sendRedirect("userRegistration.htm"); %>

```

В итоге должна получиться следующая регистрационная страница:

Будет вызван `onSubmit()` метод у `UserController` класса, после чего мы перейдем на `userSuccess` view. Мы используем `InternalResourceViewResolver`, поэтому мы попадем на `userSuccess.jsp`. В `userSuccess.jsp` странице мы отобразим все детали о пользователе, используя `JSTL` тэги.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<html>
<head>
<title>Success Page</title>
</head>
<body>
User Details
<hr>User Name : ${user.name} <br />
Gender : ${user.gender} <br />
Country : ${user.country} <br />
About You : ${user.aboutYou} <br />
Community : ${user.community[0]} ${user.community[1]}
${user.community[2]}<br />
Mailing List: ${user.mailingList}
</body>
</html>
```

Artem Borodin (artem.borodin@gmail.com). Copyright © 2017