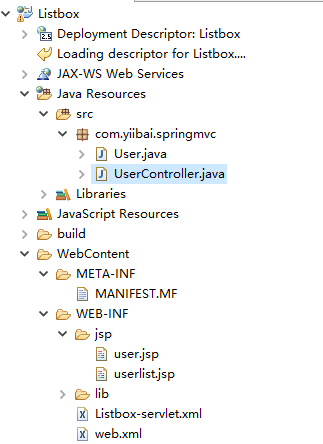
以下示例显示如何在使用**Spring Web MVC**框架的表单中使用列表框(Listbox)。首先使用Eclipse IDE来创建一个WEB工程，实现一个让用户可选择自己所善长的技术(多选)的功能。并按照以下步骤使用**Spring Web Framework**开发基于动态表单的Web应用程序：

1. 创建一个名称为 **Listbox** 的动态WEB项目。
2. 在 com.yiibai.springmvc 包下创建两个Java类User，UserController。
3. 在jsp子文件夹下创建两个视图文件：user.jsp，userlist.jsp。
4. 最后一步是创建所有源和配置文件的内容并运行应用程序，详细如下所述。

完整的项目文件目录结构如下所示 -



**User.java** 的代码如下所示 -

package com.yiibai.springmvc;

public class User {

private String username;

private String password;

private String address;

private boolean receivePaper;

private String [] favoriteFrameworks;

private String gender;

private String favoriteNumber;

private String country;

private String [] skills;

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public boolean isReceivePaper() {

return receivePaper;

}

public void setReceivePaper(boolean receivePaper) {

this.receivePaper = receivePaper;

}

public String[] getFavoriteFrameworks() {

return favoriteFrameworks;

}

public void setFavoriteFrameworks(String[] favoriteFrameworks) {

this.favoriteFrameworks = favoriteFrameworks;

}

public String getGender() {

return gender;

}

public void setGender(String gender) {

this.gender = gender;

}

public String getFavoriteNumber() {

return favoriteNumber;

}

public void setFavoriteNumber(String favoriteNumber) {

this.favoriteNumber = favoriteNumber;

}

public String getCountry() {

return country;

}

public void setCountry(String country) {

this.country = country;

}

public String[] getSkills() {

return skills;

}

public void setSkills(String[] skills) {

this.skills = skills;

}

}

Java

**UserController.java** 的代码如下所示 -

package com.yiibai.springmvc;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import org.springframework.stereotype.Controller;

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;

import org.springframework.ui.ModelMap;

@Controller

public class UserController {

@RequestMapping(value = "/user", method = RequestMethod.GET)

public ModelAndView user() {

User user = new User();

user.setFavoriteFrameworks((new String []{"Spring MVC","Struts 2"}));

user.setGender("M");

ModelAndView modelAndView = new ModelAndView("user", "command", user);

return modelAndView;

}

@RequestMapping(value = "/addUser", method = RequestMethod.POST)

public String addUser(@ModelAttribute("SpringWeb")User user,

ModelMap model) {

model.addAttribute("username", user.getUsername());

model.addAttribute("password", user.getPassword());

model.addAttribute("address", user.getAddress());

model.addAttribute("receivePaper", user.isReceivePaper());

model.addAttribute("favoriteFrameworks", user.getFavoriteFrameworks());

model.addAttribute("gender", user.getGender());

model.addAttribute("favoriteNumber", user.getFavoriteNumber());

model.addAttribute("country", user.getCountry());

model.addAttribute("skills", user.getSkills());

return "userlist";

}

@ModelAttribute("webFrameworkList")

public List<String> getWebFrameworkList()

{

List<String> webFrameworkList = new ArrayList<String>();

webFrameworkList.add("Spring MVC");

webFrameworkList.add("Struts 1");

webFrameworkList.add("Struts 2");

webFrameworkList.add("Apache Hadoop");

return webFrameworkList;

}

@ModelAttribute("numbersList")

public List<String> getNumbersList()

{

List<String> numbersList = new ArrayList<String>();

numbersList.add("1");

numbersList.add("2");

numbersList.add("3");

numbersList.add("4");

return numbersList;

}

@ModelAttribute("countryList")

public Map<String, String> getCountryList()

{

Map<String, String> countryList = new HashMap<String, String>();

countryList.put("US", "United States");

countryList.put("CH", "China");

countryList.put("SG", "Singapore");

countryList.put("MY", "Malaysia");

return countryList;

}

@ModelAttribute("skillsList")

public Map<String, String> getSkillsList()

{

Map<String, String> skillList = new HashMap<String, String>();

skillList.put("Hibernate", "Hibernate");

skillList.put("Spring", "Spring");

skillList.put("Apache Hadoop", "Apache Hadoop");

skillList.put("Struts", "Struts");

return skillList;

}

}

Java

这里的第一个服务方法user()，我们已经在ModelAndView对象中传递了一个名称为“command”的空User对象，因为如果在JSP文件中使用<form:form>标签，**spring**框架需要一个名称为“command”的对象。 所以当调用user()方法时，它返回user.jsp视图。

第二个服务方法addUser()将根据URL => Listbox/addUser 上的POST方法请求时调用。根据提交的信息准备模型对象。 最后从服务方法返回“userlist”视图，这将呈现userlist.jsp视图。

**user.jsp** 的代码如下所示 -

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

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

<html>

<head>

<title>Spring MVC表单处理(列表框)</title>

</head>

<body>

<h2>用户信息</h2>

<form:form method="POST" action="/Listbox/addUser">

<table>

<tr>

<td><form:label path="username">用户名：</form:label></td>

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

</tr>

<tr>

<td><form:label path="password">密码：</form:label></td>

<td><form:password path="password" /></td>

</tr>

<tr>

<td><form:label path="address">地址：</form:label></td>

<td><form:textarea path="address" rows="5" cols="30" /></td>

</tr>

<tr>

<td><form:label path="receivePaper">是否订阅：</form:label></td>

<td><form:checkbox path="receivePaper" /></td>

</tr>

<tr>

<td><form:label path="favoriteFrameworks">喜欢的框架/技术：</form:label></td>

<td><form:checkboxes items="${webFrameworkList}"

path="favoriteFrameworks" /></td>

</tr>

<tr>

<td><form:label path="gender">性别：</form:label></td>

<td><form:radiobutton path="gender" value="M" label="男" /> <form:radiobutton

path="gender" value="F" label="女" /></td>

</tr>

<tr>

<td><form:label path="favoriteNumber">喜欢的数字：</form:label></td>

<td><form:radiobuttons path="favoriteNumber"

items="${numbersList}" /></td>

</tr>

<tr>

<td><form:label path="country">所在国家：</form:label></td>

<td><form:select path="country">

<form:option value="NONE" label="请选择..." />

<form:options items="${countryList}" />

</form:select></td>

</tr>

<tr>

<td><form:label path="skills">技术：</form:label></td>

<td><form:select path="skills" items="${skillsList}"

multiple="true" /></td>

</tr>

<tr>

<td colspan="2"><input type="submit" value="提交" /></td>

</tr>

</table>

</form:form>

</body>

</html>

HTML

这里使用<form:select /> 及其属性 multiple=true标签来呈现HTML列表多选框。 例如 -

<form:select path="skills" items="${skillsList}" multiple="true" />

HTML

它将呈现以下HTML内容。

<select id="skills" name="skills" multiple="multiple">

<option value="Struts">Struts</option>

<option value="Hibernate">Hibernate</option>

<option value="Apache Wicket">Apache Hadoop</option>

<option value="Spring">Spring</option>

</select>

<input type="hidden" name="\_skills" value="1"/>

HTML

**userlist.jsp** 的代码如下所示 -

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

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

<html>

<head>

<title>Spring MVC表单处理(列表框)</title>

</head>

<body>

<h2>提交用户信息 -</h2>

<table>

<tr>

<td>用户名：</td>

<td>${username}</td>

</tr>

<tr>

<td>密码：</td>

<td>${password}</td>

</tr>

<tr>

<td>地址：</td>

<td>${address}</td>

</tr>

<tr>

<td>是否订阅：</td>

<td>${receivePaper}</td>

</tr>

<tr>

<td>喜欢的技术/框架：</td>

<td>

<%

String[] favoriteFrameworks = (String[]) request.getAttribute("favoriteFrameworks");

for (String framework : favoriteFrameworks) {

out.println(framework);

}

%>

</td>

</tr>

<tr>

<td>性别：</td>

<td>${(gender=="M"? "男" : "女")}</td>

</tr>

<tr>

<td>喜欢的数字：</td>

<td>${favoriteNumber}</td>

</tr>

<tr>

<td>国家：</td>

<td>${country}</td>

</tr>

<tr>

<td>技术：</td>

<td>

<%

String[] skills = (String[]) request.getAttribute("skills");

for (String skill : skills) {

out.println(skill);

}

%>

</td>

</tr>

</table>

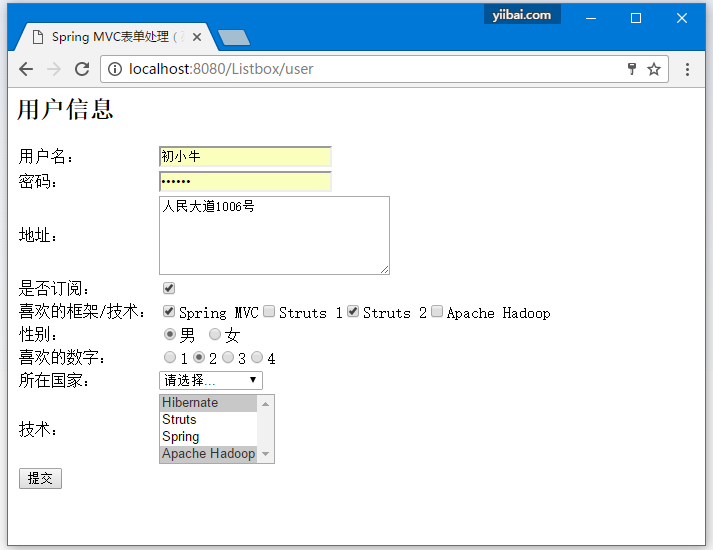
</body>

</html>

HTML

完成创建源和配置文件后，发布应用程序到Tomcat服务器。

现在启动Tomcat服务器，现在尝试访问URL => [http://localhost:8080/Listbox/user](http://localhost:8080/Listbox/user" \t "_blank) ，如果**Spring Web**应用程序没有问题，应该看到以下结果：



提交所需信息后，点击提交按钮提交表单。 如果**Spring Web**应用程序没有问题，应该看到以下结果：

