[[Spring MVC] -](http://www.cnblogs.com/HD/p/4107674.html) [SpringMVC的各种参数绑定方式](http://www.cnblogs.com/HD/p/4107674.html)

SpringMVC的各种参数绑定方式

**1. 基本数据类型(以int为例，其他类似)：**  
Controller代码：

@RequestMapping("saysth.do")

public void test(int count) {

}

表单代码：

<form action="saysth.do" method="post">

<input name="count" value="10" type="text"/>

......

</form>

表单中input的name值和Controller的参数变量名保持一致，就能完成数据绑定，如果不一致可以使用@RequestParam注解。需要注意的是，如果Controller方法参数中定义的是基本数据类型，但是从页面提交过来的数据为null或者”"的话，会出现数据转换的异常。也就是必须保证表单传递过来的数据不能为null或”"，所以，在开发过程中，对可能为空的数据，最好将参数数据类型定义成包装类型，具体参见下面的例子。

**2. 包装类型(以Integer为例，其他类似)：**  
Controller代码：

@RequestMapping("saysth.do")

public void test(Integer count) {

}

表单代码：

<form action="saysth.do" method="post">

<input name="count" value="10" type="text"/>

......

</form>

和基本数据类型基本一样，不同之处在于，表单传递过来的数据可以为null或”"，以上面代码为例，如果表单中num为”"或者表单中无num这个input，那么，Controller方法参数中的num值则为null。

**3. 自定义对象类型：**  
Model代码：

[复制代码](javascript:void(0);)

public class User {

private String firstName;

private String lastName;

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

}

[复制代码](javascript:void(0);)

Controller代码：

@RequestMapping("saysth.do")

public void test(User user) {

}

表单代码：

<form action="saysth.do" method="post">

<input name="firstName" value="张" type="text"/>

<input name="lastName" value="三" type="text"/>

......

</form>

非常简单，只需将对象的属性名和input的name值一一匹配即可。

**4. 自定义复合对象类型：**  
Model代码：

[复制代码](javascript:void(0);)

public class ContactInfo {

private String tel;

private String address;

public String getTel() {

return tel;

}

public void setTel(String tel) {

this.tel = tel;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

}

public class User {

private String firstName;

private String lastName;

private ContactInfo contactInfo;

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public ContactInfo getContactInfo() {

return contactInfo;

}

public void setContactInfo(ContactInfo contactInfo) {

this.contactInfo = contactInfo;

}

}

[复制代码](javascript:void(0);)

Controller代码：

[复制代码](javascript:void(0);)

@RequestMapping("saysth.do")

public void test(User user) {

System.out.println(user.getFirstName());

System.out.println(user.getLastName());

System.out.println(user.getContactInfo().getTel());

System.out.println(user.getContactInfo().getAddress());

}

[复制代码](javascript:void(0);)

表单代码：

[复制代码](javascript:void(0);)

<form action="saysth.do" method="post">

<input name="firstName" value="张" /><br>

<input name="lastName" value="三" /><br>

<input name="contactInfo.tel" value="13809908909" /><br>

<input name="contactInfo.address" value="北京海淀" /><br>

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

</form>

[复制代码](javascript:void(0);)

User对象中有ContactInfo属性，Controller中的代码和第3点说的一致，但是，在表单代码中，需要使用“属性名(对象类型的属性).属性名”来命名input的name。

**5. List绑定：**  
List需要绑定在对象上，而不能直接写在Controller方法的参数中。  
Model代码：

[复制代码](javascript:void(0);)

public class User {

private String firstName;

private String lastName;

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

}

public class UserListForm {

private List<User> users;

public List<User> getUsers() {

return users;

}

public void setUsers(List<User> users) {

this.users = users;

}

}

[复制代码](javascript:void(0);)

Controller代码：

[复制代码](javascript:void(0);)

@RequestMapping("saysth.do")

public void test(UserListForm userForm) {

for (User user : userForm.getUsers()) {

System.out.println(user.getFirstName() + " - " + user.getLastName());

}

}

[复制代码](javascript:void(0);)

表单代码：

[复制代码](javascript:void(0);)

<form action="saysth.do" method="post">

<table>

<thead>

<tr>

<th>First Name</th>

<th>Last Name</th>

</tr>

</thead>

<tfoot>

<tr>

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

</tr>

</tfoot>

<tbody>

<tr>

<td><input name="users[0].firstName" value="aaa" /></td>

<td><input name="users[0].lastName" value="bbb" /></td>

</tr>

<tr>

<td><input name="users[1].firstName" value="ccc" /></td>

<td><input name="users[1].lastName" value="ddd" /></td>

</tr>

<tr>

<td><input name="users[2].firstName" value="eee" /></td>

<td><input name="users[2].lastName" value="fff" /></td>

</tr>

</tbody>

</table>

</form>

[复制代码](javascript:void(0);)

其实，这和第4点User对象中的contantInfo数据的绑定有点类似，但是这里的UserListForm对象里面的属性被定义成List，而不是普通自定义对象。所以，在表单中需要指定List的下标。值得一提的是，Spring会创建一个以最大下标值为size的List对象，所以，如果表单中有动态添加行、删除行的情况，就需要特别注意，譬如一个表格，用户在使用过程中经过多次删除行、增加行的操作之后，下标值就会与实际大小不一致，这时候，List中的对象，只有在表单中对应有下标的那些才会有值，否则会为null，看个例子：

表单代码：

[复制代码](javascript:void(0);)

<form action="saysth.do" method="post">

<table>

<thead>

<tr>

<th>First Name</th>

<th>Last Name</th>

</tr>

</thead>

<tfoot>

<tr>

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

</tr>

</tfoot>

<tbody>

<tr>

<td><input name="users[0].firstName" value="aaa" /></td>

<td><input name="users[0].lastName" value="bbb" /></td>

</tr>

<tr>

<td><input name="users[1].firstName" value="ccc" /></td>

<td><input name="users[1].lastName" value="ddd" /></td>

</tr>

<tr>

<td><input name="users[20].firstName" value="eee" /></td>

<td><input name="users[20].lastName" value="fff" /></td>

</tr>

</tbody>

</table>

</form>

[复制代码](javascript:void(0);)

这个时候，Controller中的userForm.getUsers()获取到List的size为21，而且这21个User对象都不会为null，但是，第2到第19的User对象中的firstName和lastName都为null。打印结果：

[复制代码](javascript:void(0);)

aaa - bbb

ccc - ddd

null - null

null - null

null - null

null - null

null - null

null - null

null - null

null - null

null - null

null - null

null - null

null - null

null - null

null - null

null - null

null - null

null - null

null - null

eee - fff

[复制代码](javascript:void(0);)

**6. Set绑定：**  
Set和List类似，也需要绑定在对象上，而不能直接写在Controller方法的参数中。但是，绑定Set数据时，必须先在Set对象中add相应的数量的模型对象。  
Model代码：

[复制代码](javascript:void(0);)

public class User {

private String firstName;

private String lastName;

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

}

public class UserSetForm {

private Set<User> users = new HashSet<User>();

public UserSetForm() {

users.add(new User());

users.add(new User());

users.add(new User());

}

public Set<User> getUsers() {

return users;

}

public void setUsers(Set<User> users) {

this.users = users;

}

}

[复制代码](javascript:void(0);)

Controller代码：

[复制代码](javascript:void(0);)

@RequestMapping("saysth.do")

public void test(UserSetForm userForm) {

for (User user : userForm.getUsers()) {

System.out.println(user.getFirstName() + " - " + user.getLastName());

}

}

[复制代码](javascript:void(0);)

表单代码：

[复制代码](javascript:void(0);)

<form action="saysth.do" method="post">

<table>

<thead>

<tr>

<th>First Name</th>

<th>Last Name</th>

</tr>

</thead>

<tfoot>

<tr>

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

</tr>

</tfoot>

<tbody>

<tr>

<td><input name="users[0].firstName" value="aaa" /></td>

<td><input name="users[0].lastName" value="bbb" /></td>

</tr>

<tr>

<td><input name="users[1].firstName" value="ccc" /></td>

<td><input name="users[1].lastName" value="ddd" /></td>

</tr>

<tr>

<td><input name="users[2].firstName" value="eee" /></td>

<td><input name="users[2].lastName" value="fff" /></td>

</tr>

</tbody>

</table>

</form>

[复制代码](javascript:void(0);)

基本和List绑定类似。  
需要特别提醒的是，如果最大下标值大于Set的size，则会抛出org.springframework.beans.InvalidPropertyException异常。所以，在使用时有些不便。

**7. Map绑定：**  
Map最为灵活，它也需要绑定在对象上，而不能直接写在Controller方法的参数中。  
Model代码：

[复制代码](javascript:void(0);)

public class User {

private String firstName;

private String lastName;

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

}

public class UserMapForm {

private Map<String, User> users;

public Map<String, User> getUsers() {

return users;

}

public void setUsers(Map<String, User> users) {

this.users = users;

}

}

[复制代码](javascript:void(0);)

Controller代码：

[复制代码](javascript:void(0);)

@RequestMapping("saysth.do")

public void test(UserMapForm userForm) {

for (Map.Entry<String, User> entry : userForm.getUsers().entrySet()) {

System.out.println(entry.getKey() + ": " + entry.getValue().getFirstName() + " - " +

entry.getValue().getLastName());

}

}

[复制代码](javascript:void(0);)

表单代码：

[复制代码](javascript:void(0);)

<form action="saysth.do" method="post">

<table>

<thead>

<tr>

<th>First Name</th>

<th>Last Name</th>

</tr>

</thead>

<tfoot>

<tr>

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

</tr>

</tfoot>

<tbody>

<tr>

<td><input name="users['x'].firstName" value="aaa" /></td>

<td><input name="users['x'].lastName" value="bbb" /></td>

</tr>

<tr>

<td><input name="users['y'].firstName" value="ccc" /></td>

<td><input name="users['y'].lastName" value="ddd" /></td>

</tr>

<tr>

<td><input name="users['z'].firstName" value="eee" /></td>

<td><input name="users['z'].lastName" value="fff" /></td>

</tr>

</tbody>

</table>

</form>

[复制代码](javascript:void(0);)

打印结果：

x: aaa - bbb

y: ccc - ddd

z: eee - fff