[Spring MVC数据绑定大全](http://blog.csdn.net/lxf9601/article/details/5925810)

分类： [Sun JAVA](http://blog.csdn.net/lxf9601/article/category/311960)2010-10-07 21:24 6591人阅读 [评论](http://blog.csdn.net/lxf9601/article/details/5925810#comments)(2) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/lxf9601/article/details/5925810#report)

[spring](http://www.csdn.net/tag/spring)[mvc](http://www.csdn.net/tag/mvc)[class](http://www.csdn.net/tag/class)[input](http://www.csdn.net/tag/input)[user](http://www.csdn.net/tag/user)[string](http://www.csdn.net/tag/string)

刚开始用spring mvc 做web开发时,经常会不知道如何合适绑定页面数据.用惯struts2的朋友更认为spring mvc 绑定数据不如struts2方便(本人最开始也是这么认为),经过一段时间的应用和测试,总结出以下几种情况,希望对刚接触spring mvc 的朋友有所帮助.

[示例程序下载](http://download.csdn.net/source/2738877)

查看spring源码可以看出spring支持转换的数据类型已经很多了.

**org.springframework.beans.PropertyEditorRegistrySupport.java**

**[java]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. /\*\*
2. \* Actually register the default editors for this registry instance.
3. \*/
4. **private** **void** doRegisterDefaultEditors() {
5. **this**.defaultEditors = **new** HashMap<Class, PropertyEditor>(64);
6. // Simple editors, without parameterization capabilities.
7. // The JDK does not contain a default editor for any of these target types.
8. **this**.defaultEditors.put(Charset.**class**, **new** CharsetEditor());
9. **this**.defaultEditors.put(Class.**class**, **new** ClassEditor());
10. **this**.defaultEditors.put(Class[].**class**, **new** ClassArrayEditor());
11. **this**.defaultEditors.put(Currency.**class**, **new** CurrencyEditor());
12. **this**.defaultEditors.put(File.**class**, **new** FileEditor());
13. **this**.defaultEditors.put(InputStream.**class**, **new** InputStreamEditor());
14. **this**.defaultEditors.put(Locale.**class**, **new** LocaleEditor());
15. **this**.defaultEditors.put(Pattern.**class**, **new** PatternEditor());
16. **this**.defaultEditors.put(Properties.**class**, **new** PropertiesEditor());
17. **this**.defaultEditors.put(Resource[].**class**, **new** ResourceArrayPropertyEditor());
18. **this**.defaultEditors.put(TimeZone.**class**, **new** TimeZoneEditor());
19. **this**.defaultEditors.put(URI.**class**, **new** URIEditor());
20. **this**.defaultEditors.put(URL.**class**, **new** URLEditor());
21. // Default instances of collection editors.
22. // Can be overridden by registering custom instances of those as custom editors.
23. **this**.defaultEditors.put(Collection.**class**, **new** CustomCollectionEditor(Collection.**class**));
24. **this**.defaultEditors.put(Set.**class**, **new** CustomCollectionEditor(Set.**class**));
25. **this**.defaultEditors.put(SortedSet.**class**, **new** CustomCollectionEditor(SortedSet.**class**));
26. **this**.defaultEditors.put(List.**class**, **new** CustomCollectionEditor(List.**class**));
27. **this**.defaultEditors.put(SortedMap.**class**, **new** CustomMapEditor(SortedMap.**class**));
28. // Default editors for primitive arrays.
29. **this**.defaultEditors.put(**byte**[].**class**, **new** ByteArrayPropertyEditor());
30. **this**.defaultEditors.put(**char**[].**class**, **new** CharArrayPropertyEditor());
31. // The JDK does not contain a default editor for char!
32. **this**.defaultEditors.put(**char**.**class**, **new** CharacterEditor(**false**));
33. **this**.defaultEditors.put(Character.**class**, **new** CharacterEditor(**true**));
34. // Spring's CustomBooleanEditor accepts more flag values than the JDK's default editor.
35. **this**.defaultEditors.put(**boolean**.**class**, **new** CustomBooleanEditor(**false**));
36. **this**.defaultEditors.put(Boolean.**class**, **new** CustomBooleanEditor(**true**));
37. // The JDK does not contain default editors for number wrapper types!
38. // Override JDK primitive number editors with our own CustomNumberEditor.
39. **this**.defaultEditors.put(**byte**.**class**, **new** CustomNumberEditor(Byte.**class**, **false**));
40. **this**.defaultEditors.put(Byte.**class**, **new** CustomNumberEditor(Byte.**class**, **true**));
41. **this**.defaultEditors.put(**short**.**class**, **new** CustomNumberEditor(Short.**class**, **false**));
42. **this**.defaultEditors.put(Short.**class**, **new** CustomNumberEditor(Short.**class**, **true**));
43. **this**.defaultEditors.put(**int**.**class**, **new** CustomNumberEditor(Integer.**class**, **false**));
44. **this**.defaultEditors.put(Integer.**class**, **new** CustomNumberEditor(Integer.**class**, **true**));
45. **this**.defaultEditors.put(**long**.**class**, **new** CustomNumberEditor(Long.**class**, **false**));
46. **this**.defaultEditors.put(Long.**class**, **new** CustomNumberEditor(Long.**class**, **true**));
47. **this**.defaultEditors.put(**float**.**class**, **new** CustomNumberEditor(Float.**class**, **false**));
48. **this**.defaultEditors.put(Float.**class**, **new** CustomNumberEditor(Float.**class**, **true**));
49. **this**.defaultEditors.put(**double**.**class**, **new** CustomNumberEditor(Double.**class**, **false**));
50. **this**.defaultEditors.put(Double.**class**, **new** CustomNumberEditor(Double.**class**, **true**));
51. **this**.defaultEditors.put(BigDecimal.**class**, **new** CustomNumberEditor(BigDecimal.**class**, **true**));
52. **this**.defaultEditors.put(BigInteger.**class**, **new** CustomNumberEditor(BigInteger.**class**, **true**));
53. // Only register config value editors if explicitly requested.
54. **if** (**this**.configValueEditorsActive) {
55. StringArrayPropertyEditor sae = **new** StringArrayPropertyEditor();
56. **this**.defaultEditors.put(String[].**class**, sae);
57. **this**.defaultEditors.put(**short**[].**class**, sae);
58. **this**.defaultEditors.put(**int**[].**class**, sae);
59. **this**.defaultEditors.put(**long**[].**class**, sae);
60. }
61. }

**1.基本数据类型**

很简单,该怎么写就怎么写.

controller代码

**[java]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. @RequestMapping
2. **public** **void** test1(String name, Integer age, Double income, Boolean isMarried, String[] interests)
3. {
4. System.out.println("简单数据类型绑定=========");
5. System.out.println("名字:" + name);
6. System.out.println("年龄:" + age);
7. System.out.println("收入:" + income);
8. System.out.println("已结婚:" + isMarried);
9. System.out.println("兴趣:");
10. **for** (String interest : interests)
11. {
12. System.out.println(interest);
13. }
14. System.out.println("====================");
15. }

jsp代码

**[xhtml]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. **<form** action="test1.do" method="post"**>**
2. **<div** class="f\_left"**>**姓名:**</div>**
3. **<div** class="f\_left"**><input** name="name" value="张三"**/></div>**
4. **<div** class="clear"**></div>**
5. **<div** class="f\_left"**>**年龄:**</div>**
6. **<div** class="f\_left"**><input** name="age" value="20"**/></div>**
7. **<div** class="clear"**></div>**
8. **<div** class="f\_left"**>**收入:**</div>**
9. **<div** class="f\_left"**><input** name="income" value="100000"**/></div>**
10. **<div** class="clear"**></div>**
11. **<div** class="f\_left"**>**结婚:**</div>**
12. **<div** class="f\_left"**>**
13. **<input** type="radio" name="isMarried" value="true" checked="checked"**/>**是
14. **<input** type="radio" name="isMarried" value="false"**/>**否**</div>**
15. **<div** class="clear"**></div>**
16. **<div** class="f\_left"**>**兴趣:**</div>**
17. **<div** class="f\_left"**>**
18. **<input** type="checkbox" name="interests" value="听歌" checked="checked"**/>**听歌
19. **<input** type="checkbox" name="interests" value="书法" checked="checked"**/>**书法
20. **<input** type="checkbox" name="interests" value="看电影" checked="checked"**/>**看电影
21. **</div>**
22. **<div** class="clear"**></div>**
23. **<div><input** type="submit" value="提交表单"**/></div>**
24. **</form>**

**2.简单对象类型**

与基本类型相拟,只不过绑定到对象上更加简洁.(类拟struts的ActionForm)

controller代码

**[java]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. **@RequestMapping**
2. **public void test2(User user)**
3. **{**
4. **System.out.println("简单对象类型绑定=========");**
5. **System.out.println("名字:" + user.getName());**
6. **System.out.println("年龄:" + user.getAge());**
7. **System.out.println("收入:" + user.getIncome());**
8. **System.out.println("已结婚:" + user.getIsMarried());**
9. **System.out.println("========================");**
10. **}**

user代码

**[java]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. **public** **class** User {
2. **private** String name;
3. **private** Integer age;
4. **private** Boolean isMarried;
5. **private** Double income;
6. **private** String[] interests;
8. **public** String getName() {
9. **return** name;
10. }
11. **public** **void** setName(String name) {
12. **this**.name = name;
13. }
14. **public** Integer getAge() {
15. **return** age;
16. }
17. **public** **void** setAge(Integer age) {
18. **this**.age = age;
19. }
20. **public** Boolean getIsMarried() {
21. **return** isMarried;
22. }
23. **public** **void** setIsMarried(Boolean isMarried) {
24. **this**.isMarried = isMarried;
25. }
26. **public** Double getIncome() {
27. **return** income;
28. }
29. **public** **void** setIncome(Double income) {
30. **this**.income = income;
31. }
32. **public** String[] getInterests() {
33. **return** interests;
34. }
35. **public** **void** setInterests(String[] interests) {
36. **this**.interests = interests;
37. }
38. }

jsp代码

**[xhtml]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. **<form action="test2.do" method="post">**
2. **<div class="f\_left">姓名:</div>**
3. **<div class="f\_left"><input name="name" value="lxf"/></div>**
4. **<div class="clear"></div>**
5. **<div class="f\_left">年龄:</div>**
6. **<div class="f\_left"><input name="age" value="20"/></div>**
7. **<div class="clear"></div>**
8. **<div class="f\_left">收入:</div>**
9. **<div class="f\_left"><input name="income" value="100000"/></div>**
10. **<div class="clear"></div>**
11. **<div class="f\_left">结婚:</div>**
12. **<div class="f\_left">**
13. **<input type="radio" name="isMarried" value="true" checked="checked"/>是**
14. **<input type="radio" name="isMarried" value="false"/>否</div>**
15. **<div class="clear"></div>**
16. **<div class="f\_left">兴趣:</div>**
17. **<div class="f\_left">**
18. **<input type="checkbox" name="interests" value="听歌" checked="checked"/>听歌**
19. **<input type="checkbox" name="interests" value="书法" checked="checked"/>书法**
20. **<input type="checkbox" name="interests" value="看电影" checked="checked"/>看电影**
21. **</div>**
22. **<div class="clear"></div>**
23. **<div><input type="submit" value="提交表单"/></div>**
24. **</form>**

**3.List类型绑定**

网上很多人都给不出答案,关键在于,List需要绑定在对象(ActionForm),直接写在request-mapping函数的参数是不行的,更重要的一点是要创建对象(ArrayList).

controller代码

**[java]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. @RequestMapping
2. **public** **void** test3(Club club)
3. {
4. System.out.println("list类型绑定=========");
5. System.out.println("会员:");
6. **for** (User user : club.getUsers())
7. {
8. System.out.println(user.getName());
9. }
10. System.out.println("========================");
11. }

Club代码

**[java]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. **public** **class** Club {
2. **private** List<User> users = **new** ArrayList<User>();
4. **public** List<User> getUsers() {
5. **return** users;
6. }
8. **public** **void** setUsers(List<User> users) {
9. **this**.users = users;
10. }
11. }

jsp代码

**[xhtml]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. **<form** action="test3.do" method="post"**>**
2. **<div** class="f\_left"**>**会员1号:**</div>**
3. **<div** class="f\_left"**>**
4. **<input** name="users[0].name" value="张三"**/>**
5. **</div>**
6. **<div** class="clear"**></div>**
7. **<div** class="f\_left"**>**会员2号:**</div>**
8. **<div** class="f\_left"**>**
9. **<input** name="users[1].name" value="李四"**/>**
10. **</div>**
11. **<div** class="clear"**></div>**
12. **<div** class="f\_left"**>**会员3号:**</div>**
13. **<div** class="f\_left"**>**
14. **<input** name="users[2].name" value="王五"**/>**
15. **</div>**
16. **<div** class="clear"**></div>**
17. **<div><input** type="submit" value="提交表单"**/></div>**
18. **</form>**

**4.Set类型绑定**

与List类型相似,经测试表明,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.

controller代码

**[java]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. @RequestMapping
2. **public** **void** test4(Club2 club)
3. {
4. System.out.println("set类型绑定=========");
5. System.out.println("会员:");
6. **for** (User user : club.getUsers())
7. {
8. System.out.println(user.getName());
9. }
10. System.out.println("========================");
11. }

Club2代码

**[java]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. **public** **class** Club2 {
2. **private** Set<User> users = **new** LinkedHashSet<User>();
4. **public** Club2() {
5. users.add(**new** User());
6. users.add(**new** User());
7. users.add(**new** User());
8. }
10. **public** Set<User> getUsers() {
11. **return** users;
12. }
13. **public** **void** setUsers(Set<User> users) {
14. **this**.users = users;
15. }
16. }

jsp代码

**[xhtml]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. **<form action="test4.do" method="post">**
2. **<div class="f\_left">会员1号:</div>**
3. **<div class="f\_left">**
4. **<input name="users[0].name" value="张三"/>**
5. **</div>**
6. **<div class="clear"></div>**
7. **<div class="f\_left">会员2号:</div>**
8. **<div class="f\_left">**
9. **<input name="users[1].name" value="李四"/>**
10. **</div>**
11. **<div class="clear"></div>**
12. **<div class="f\_left">会员3号:</div>**
13. **<div class="f\_left">**
14. **<input name="users[2].name" value="王五"/>**
15. **</div>**
16. **<div class="clear"></div>**
17. **<div><input type="submit" value="提交表单"/></div>**
18. **</form>**

**5.Map类型绑定**

最灵活的一种方式,可无限绑定未定义的参数,注意必须绑定在对象下(ActionForm).

controller代码

**[java]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. **@RequestMapping**
2. **public void test5(UserForm userForm)**
3. **{**
4. **System.out.println("Map类型绑定=========");**
5. **System.out.println("名字:" + userForm.getUser().get("name"));**
6. **System.out.println("年龄:" + userForm.getUser().get("age"));**
7. **System.out.println("收入:" + userForm.getUser().get("income"));**
8. **System.out.println("已结婚:" + userForm.getUser().get("isMarried"));**
9. **System.out.println("========================");**
10. **}**

UserForm.java代码

**[java]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. **public class UserForm {**
2. **private Map<String, String> userMap = new LinkedHashMap<String, String>();**
3. **private User user = new User();**
5. **public Map<String, String> getUserMap() {**
6. **return userMap;**
7. **}**
8. **public void setUserMap(Map<String, String> userMap) {**
9. **this.userMap = userMap;**
10. **}**
11. **public User getUser() {**
12. **return user;**
13. **}**
14. **public void setUser(User user) {**
15. **this.user = user;**
16. **}**
18. **}**

jsp代码

**[xhtml]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. **<form** action="test5.do" method="post"**>**
2. **<div** class="f\_left"**>**姓名:**</div>**
3. **<div** class="f\_left"**><input** name="user['name']" value="张三"**/></div>**
4. **<div** class="clear"**></div>**
5. **<div** class="f\_left"**>**年龄:**</div>**
6. **<div** class="f\_left"**><input** name="user['age']" value="20"**/></div>**
7. **<div** class="clear"**></div>**
8. **<div** class="f\_left"**>**收入:**</div>**
9. **<div** class="f\_left"**><input** name="user['income']" value="100000"**/></div>**
10. **<div** class="clear"**></div>**
11. **<div** class="f\_left"**>**结婚:**</div>**
12. **<div** class="f\_left"**>**
13. **<input** type="radio" name="user['isMarried']" value="true" checked="checked"**/>**是
14. **<input** type="radio" name="user['isMarried']" value="false"**/>**否**</div>**
15. **<div** class="clear"**></div>**
16. **<div><input** type="submit" value="提交表单"**/></div>**
17. **</form>**

**6.复合类型绑定**

controller代码

**[java]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. **@RequestMapping**
2. **public void test6(UserForm userForm)**
3. **{**
4. **System.out.println("复合类型绑定=========");**
5. **System.out.println("名字:" + userForm.getUser().getName());**
6. **System.out.println("年龄:" + userForm.getUser().getAge());**
7. **System.out.println("收入:" + userForm.getUser().getIncome());**
8. **System.out.println("已结婚:" + userForm.getUser().getIsMarried());**
9. **System.out.println("========================");**
10. **}**

UserForm代码

**[java]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. **public** **class** UserForm {
2. **private** Map<String, String> userMap = **new** LinkedHashMap<String, String>();
3. **private** User user = **new** User();
5. **public** Map<String, String> getUserMap() {
6. **return** userMap;
7. }
8. **public** **void** setUserMap(Map<String, String> userMap) {
9. **this**.userMap = userMap;
10. }
11. **public** User getUser() {
12. **return** user;
13. }
14. **public** **void** setUser(User user) {
15. **this**.user = user;
16. }
18. }

jsp代码

**[xhtml]** [view plaincopy](http://blog.csdn.net/lxf9601/article/details/5925810)

1. **<form** action="test6.do" method="post"**>**
2. **<div** class="f\_left"**>**姓名:**</div>**
3. **<div** class="f\_left"**><input** name="user.name" value="张三"**/></div>**
4. **<div** class="clear"**></div>**
5. **<div** class="f\_left"**>**年龄:**</div>**
6. **<div** class="f\_left"**><input** name="user.age" value="20"**/></div>**
7. **<div** class="clear"**></div>**
8. **<div** class="f\_left"**>**收入:**</div>**
9. **<div** class="f\_left"**><input** name="user.income" value="100000"**/></div>**
10. **<div** class="clear"**></div>**
11. **<div** class="f\_left"**>**结婚:**</div>**
12. **<div** class="f\_left"**>**
13. **<input** type="radio" name="user.isMarried" value="true" checked="checked"**/>**是
14. **<input** type="radio" name="user.isMarried" value="false"**/>**否**</div>**
15. **<div** class="clear"**></div>**
16. **<div><input** type="submit" value="提交表单"**/></div>**
17. **</form>**