[springMVC本地化](http://blog.csdn.net/wutbiao/article/details/7454345)

标签： [语言](http://www.csdn.net/tag/%e8%af%ad%e8%a8%80)[class](http://www.csdn.net/tag/class)[application](http://www.csdn.net/tag/application)[browser](http://www.csdn.net/tag/browser)[spring](http://www.csdn.net/tag/spring)[session](http://www.csdn.net/tag/session)

2012-04-12 16:20 4959人阅读 [评论](http://blog.csdn.net/wutbiao/article/details/7454345#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/wutbiao/article/details/7454345#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

springMVC（3） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

springMVC的本地化和国际化

   spring框架的大部分都支持国际化，就像springMVC一样。DispatcherServlet使你能够动态的通过客户端的本地语言进行配置。这是通过LocaleResolver完成的。  
   当一个请求到达，DispatcherServlet就会寻找LocaleResolver，如果找到就会试图通过它进行本地化。通过RequestContext.getLocale()方法我们可以得到locale resolver解释的本地化语言。  
   Locale resolvers和拦截器定义在org.springframework.web.servlet.i18n包中，并且可以在application context中进行配置。下面是本地化解析器的配置部分。

一、AcceptHeaderLocaleResolver  
   这个Locale resolvers依据 accept-language请求头信息进行解析处理，通常这个头信息包含客户端操作信息的本地标示。  
二、CookieLocaleResolver  
   这个解析器通过cookie寻找客户端特定的语言信息。如果找到，则使用这个配置。使用这个Locale resolvers的一些属性，我们可以指定这个cookie的名称甚至是最大存活时间。下面是一个配置实例：  
e.g.

**[html]** [view plain](http://blog.csdn.net/wutbiao/article/details/7454345) [copy](http://blog.csdn.net/wutbiao/article/details/7454345)

1. **<bean** id="localeResolver" class="org.springframework.web.servlet.i18n.CookieLocaleResolver"**>**
3. **<property** name="cookieName" value="clientlanguage"**/>**
5. <!-- in seconds. If set to -1, the cookie is not persisted (deleted when browser shuts down) -->
6. **<property** name="cookieMaxAge" value="100000"**>**
8. **</bean>**

下面是CookieLocaleResolver的属性列表  
 property default   description  
      cookieName classname+LOCAL  cookie的名称  
 cookieMaxAge Integer.MAX\_INT  最大存活时间  
 cookiePath /   限制可以访问cookie的程序路径。

三、 SessionLocaleResolver  
   这个解析器允许你通过session解析本地设置。  
四、LocaleChangeInterceptor  
   通过配置LocaleChangeInterceptor，我们可以动态改变本地语言。它会检测请求中的参数并且改变地区信息。它调用LoacalResolver.setLocal()进行配置。下面的例子显示了，调用所有的\*.view资源包含一个名字叫做siteLanguage的变量，来改变地区信息。所以下面的URL：<http://www.sf.net/home.view?siteLanguage=nl> 将会改变站点语言为荷兰语。  
e.g.

**[html]** [view plain](http://blog.csdn.net/wutbiao/article/details/7454345) [copy](http://blog.csdn.net/wutbiao/article/details/7454345)

1. **<bean** id="localeChangeInterceptor"
2. class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor"**>**
3. **<property** name="paramName" value="siteLanguage"**/>**
4. **</bean>**
6. **<bean** id="localeResolver"
7. class="org.springframework.web.servlet.i18n.CookieLocaleResolver"**/>**
9. **<bean** id="urlMapping"
10. class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping"**>**
11. **<property** name="interceptors"**>**
12. **<list>**
13. **<ref** bean="localeChangeInterceptor"**/>**
14. **</list>**
15. **</property>**
16. **<property** name="mappings"**>**
17. **<value>**/\*\*/\*.view=someController**</value>**
18. **</property>**
19. **</bean>**

### [扩展SpringMVC以支持更精准的数据绑定1](http://jinnianshilongnian.iteye.com/blog/1717180)

**博客分类：**

* [springmvc杂谈](http://jinnianshilongnian.iteye.com/category/277825)
* [跟开涛学SpringMVC](http://jinnianshilongnian.iteye.com/category/231099)
* [java开发常见问题分析](http://jinnianshilongnian.iteye.com/category/283252)

最新版请点击查看[FormModelMethodArgumentResolver.java](https://github.com/zhangkaitao/es/blob/master/common/src/main/java/com/sishuok/es/common/web/bind/method/annotation/FormModelMethodArgumentResolver.java)

问题描述：

### [springMVC 数据绑定 多个对象 如何准确绑定?](http://www.iteye.com/problems/89942)

**Java代码  [收藏代码](javascript:void())**

1. <form>
3. <input name="student.name" value="Kate" />
5. <input name="student.type" value="自费" />
7. <input name="teacher.name" value="Gavin" />
9. <input name="teacher.level" value="2" />
11. </form>

**Java代码  [收藏代码](javascript:void())**

1. @RequestMapping("/school.do")
3. **public** String school(Student student, Teacher teacher) {
5. **return** "school";
7. }

如果还是想刚才的jsp那些写表单，是不能封装参数的，必须把“student.”和“teacher.”去掉，但是这样封装就不能准确封装了。

这个问题最近老是有人问，所以写一个扩展很容易解决这个问题，springmvc和spring一样，预留的扩展点足够多。

我们都知道struts2默认就是这种方案，这是因为struts2采用了OGNL，并通过栈（根对象）进行操作的，而且栈中默认有action实例，所以很自然的没有这种问题。

springmvc不同，没有根对象的概念，而且本身很难来解决这个问题，因此大家在使用时最好避免这种方式或者使用类似于struts1的FormBean组合对象来解决。

解决方案：

扩展spring的HandlerMethodArgumentResolver以支持自定义的数据绑定方式。

1、请下载附件的代码，放到工程中；

2、在RequestMappingHandlerAdapter添加自定义HandlerMethodArgumentResolver Bean；

**Java代码  [收藏代码](javascript:void())**

1. <bean **class**="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter">
2. <!--线程安全的访问session-->
3. <property name="synchronizeOnSession" value="true"/>
4. <property name="customArgumentResolvers">
5. <list>
6. <bean **class**="cn.javass.spring.mvc.method.annotation.RequestJsonParamMethodArgumentResolver"/>
7. <bean **class**="cn.javass.spring.mvc.method.annotation.FormModelMethodArgumentResolver"/>
8. </list>
9. </property>
10. </bean>

 //customArgumentResolvers用于注入自定义的参数解析器，此处我们注了FormModelMethodArgumentResolver；FormModelMethodArgumentResolver我直接修改的org.springframework.web.servlet.mvc.method.annotation.ServletModelAttributeMethodProcessor；

3、使用方式

**Java代码  [收藏代码](javascript:void())**

1. **public** String user(@FormModel("student") Student student, @FormModel("teacher") Teacher teacher)

4、测试控制器

**Java代码  [收藏代码](javascript:void())**

1. **package** cn.javass.chapter6.web.controller.formmodel;
3. **import** java.util.Arrays;
4. **import** java.util.List;
6. **import** org.springframework.stereotype.Controller;
7. **import** org.springframework.web.bind.annotation.RequestMapping;
9. **import** cn.javass.chapter6.model.UserModel;
10. **import** cn.javass.spring.mvc.bind.annotation.FormModel;
11. **import** cn.javass.spring.mvc.util.MapWapper;

14. @Controller
15. @RequestMapping("/formmodel")
16. **public** **class** FormModelController {
18. //ok   http://localhost:9080/springmvc-chapter6/formmodel/user?user.username=zhang&user.password=123
19. @RequestMapping("/user/{user.realname}")
20. **public** String user(@FormModel("user") UserModel user) {
21. System.out.println(user);
22. **return** "redirect:/success";
23. }
25. //ok   http://localhost:9080/springmvc-chapter6/formmodel/array1?array[0]=zhang&array[1]=li
26. @RequestMapping("/array1")
27. **public** String array1(@FormModel("array") String[] array) {
28. System.out.println(Arrays.toString(array));
29. **return** "redirect:/success";
30. }
32. //ok   http://localhost:9080/springmvc-chapter6/formmodel/array2?array[0].username=zhang&array[0].password=123&array[1].username=li
33. @RequestMapping("/array2")
34. **public** String array2(@FormModel("array") UserModel[] array) {
35. System.out.println(Arrays.toString(array));
36. **return** "redirect:/success";
37. }

40. //ok   http://localhost:9080/springmvc-chapter6/formmodel/list1?list[0]=123&list[1]=234
41. @RequestMapping("/list1")
42. **public** String list1(@FormModel("list") List<Integer> list) {
43. System.out.println(list);
44. **return** "redirect:/success";
45. }
47. //ok   http://localhost:9080/springmvc-chapter6/formmodel/list2?list[0].username=zhang&list[1].username=li
48. @RequestMapping("/list2")
49. **public** String list2(@FormModel("list") List<UserModel> list) {
50. System.out.println(list);
51. **return** "redirect:/success";
52. }
54. //ok   http://localhost:9080/springmvc-chapter6/formmodel/map1?map['0']=123&map["1"]=234
55. @RequestMapping("/map1")
56. **public** String map1(@FormModel("map") MapWapper<String, Integer> map) {
57. System.out.println(map);
58. **return** "redirect:/success";
59. }
60. //ok   http://localhost:9080/springmvc-chapter6/formmodel/map2?map['0'].password=123&map['0'].username=123&map["1"].username=234
61. @RequestMapping("/map2")
62. **public** String map2(@FormModel("map") MapWapper<Integer, UserModel> map) {
63. System.out.println(map);
64. **return** "redirect:/success";
65. }
66. }

具体使用可以下载之前springmvc第六章源代码<http://jinnianshilongnian.iteye.com/blog/1683388>

将附件中的FormModel.rar解压放到src下进行测试。

支持的spring版本：

springmvc 3.1.x，暂不支持3.0。为什么不支持呢？springmvc 3.1 和 3.0 从架构上发生了变化，而且springmvc3.1更容易扩展。

支持绑定的数据：

模型、集合、数组、MapWapper（Map的一个包装器，通过getInnerMap获取真实Map）

缺点：

spring自定义的参数解析器会放在默认解析器之后，不能指定order，因此如果我们@FormModel("map") Map map，此map会变成Model（请参考<http://jinnianshilongnian.iteye.com/blog/1698916> 第六部分、Model Map ModelMap），希望未来的版本支持自定义顺序来解决这个问题；此处我们使用MapWapper解决，可以通过MapWapper.getInnerMap()拿到我们需要的Map

其他方案：

[**[SpringMVC]修改源码使之能够更加智能的自动装配request请求参数.**](http://www.iteye.com/topic/1124433)（不建议修改源代码解决）

[@**rainsoft**](http://rainsoft.iteye.com/) 也给出了类似的方案， <http://www.iteye.com/topic/1124433#2357830>

如果你使用的是mvc:annotation-driven，请这样配置

**Java代码  [收藏代码](javascript:void())**

1. <mvc:annotation-driven>
2. <mvc:argument-resolvers>
3. <bean **class**="com.sishuok.es.common.web.bind.method.annotation.FormModelMethodArgumentResolver"/>
4. </mvc:argument-resolvers>
5. </mvc:annotation-driven>

### [扩展SpringMVC以支持绑定JSON格式的请求参数](http://jinnianshilongnian.iteye.com/blog/1719952)

**博客分类：**

* [springmvc杂谈](http://jinnianshilongnian.iteye.com/category/277825)
* [跟开涛学SpringMVC](http://jinnianshilongnian.iteye.com/category/231099)
* [java开发常见问题分析](http://jinnianshilongnian.iteye.com/category/283252)

上一篇：[《扩展SpringMVC以支持更精准的数据绑定》](http://jinnianshilongnian.iteye.com/blog/1717180" \t "_self)

此方案是把请求参数（JSON字符串）绑定到java对象，，@RequestBody是绑定内容体到java对象的。

问题描述：

**Java代码  [收藏代码](javascript:void())**

1. <span style="font-size: x-small;">你好，对于如下的json数据，springmvc的数据绑定该如何做？
2. accessionDate   2012-11-21
3. deptIds [{"deptId":4,"isPrimary":**true**}]
4. email   ewer@dsfd.com
5. fax 3423432
6. gender  **true**
7. 其实就是我前台一表单提交的数据，extjs form提交的，关键在于deptIds的映射，数组不知怎么解决，还有就是springmvc的数据绑定好像要求javabean与json中的数据一一对应，否则也会出错</span>

 如请求参数是deptIds={"deptId":4,"isPrimary":true}]  是一个json数组，此时需要绑定为一个对象，该如何实现呢？

解决方案：

类似于之前写的[《扩展SpringMVC以支持更精准的数据绑定》](http://jinnianshilongnian.iteye.com/blog/1717180)，扩展spring的HandlerMethodArgumentResolver以支持自定义的数据绑定方式。

1、请下载附件的代码，放到工程中；

2、在RequestMappingHandlerAdapter添加自定义HandlerMethodArgumentResolver Bean；

**Java代码  [收藏代码](javascript:void())**

1. <span style="font-size: x-small;">     <bean **class**="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter">
2. <!--线程安全的访问session-->
3. <property name="synchronizeOnSession" value="true"/>
4. <property name="customArgumentResolvers">
5. <list>
6. <bean **class**="cn.javass.spring.mvc.method.annotation.RequestJsonParamMethodArgumentResolver"/>
7. <bean **class**="cn.javass.spring.mvc.method.annotation.FormModelMethodArgumentResolver"/>
8. </list>
9. </property>
10. </bean>  </span>

  //customArgumentResolvers用于注入自定义的参数解析器，此处我们注入了RequestJsonParamMethodArgumentResolver。

3、使用方式

**Java代码  [收藏代码](javascript:void())**

1. <span style="font-size: x-small;">@RequestMapping("/list")
2. **public** String list(@RequestJsonParam("list") List<Integer> list) </span>

4、测试控制器

**Java代码  [收藏代码](javascript:void())**

1. <span style="font-size: x-small;">**package** cn.javass.chapter6.web.controller.jsonparam;
3. **import** java.util.Arrays;
4. **import** java.util.List;
5. **import** java.util.Set;
7. **import** org.springframework.stereotype.Controller;
8. **import** org.springframework.web.bind.annotation.RequestMapping;
10. **import** cn.javass.chapter6.model.DataBinderTestModel;
11. **import** cn.javass.chapter6.model.UserModel;
12. **import** cn.javass.spring.mvc.bind.annotation.RequestJsonParam;
13. **import** cn.javass.spring.mvc.util.MapWapper;

16. @Controller
17. @RequestMapping("/jsonparam")
18. **public** **class** JsonParamController {
20. //ok   http://localhost:9080/springmvc-chapter6/jsonparam/list?list=[1,2,34]
21. //fail http://localhost:9080/springmvc-chapter6/jsonparam/list?list=[1,2,a]
22. @RequestMapping("/list")
23. **public** String list(@RequestJsonParam("list") List<Integer> list) {
24. System.out.println(list);
25. **return** "redirect:/success";
26. }
28. //ok   http://localhost:9080/springmvc-chapter6/jsonparam/set?set=[1,2,34]
29. //fail http://localhost:9080/springmvc-chapter6/jsonparam/set?set=[1,2,a]
30. @RequestMapping("/set")
31. **public** String set(@RequestJsonParam("set") Set<Integer> set) {
32. System.out.println(set);
33. **return** "redirect:/success";
34. }

37. //ok   http://localhost:9080/springmvc-chapter6/jsonparam/array?array=[1,2,3]
38. //fail http://localhost:9080/springmvc-chapter6/jsonparam/array?array=[1,2,a]
39. @RequestMapping("/array")
40. **public** String list(@RequestJsonParam("array") **int**[] array) {
41. System.out.println(Arrays.toString(array));
42. **return** "redirect:/success";
43. }

46. //ok   http://localhost:9080/springmvc-chapter6/jsonparam/map?map={"a":1, "b":2}
47. //fail http://localhost:9080/springmvc-chapter6/jsonparam/map?map={"a":1, "b":a}
48. @RequestMapping("/map")
49. **public** String map(@RequestJsonParam(value = "map", required=**false**) MapWapper<String, Integer> map) {
50. System.out.println(map);
51. **return** "redirect:/success";
52. }

55. //UserModel[]
56. //ok   http://localhost:9080/springmvc-chapter6/jsonparam/array2?array=[{"username":"123"},{"username":"234"}]
57. @RequestMapping("/array2")
58. **public** String array2(@RequestJsonParam(value = "array") UserModel[] array) {
59. System.out.println(Arrays.toString(array));
60. **return** "redirect:/success";
61. }
63. //List<UserModel>
64. //ok   http://localhost:9080/springmvc-chapter6/jsonparam/list2?list=[{"username":"123"},{"username":"234"}]
65. @RequestMapping("/list2")
66. **public** String list2(@RequestJsonParam(value = "list") List<UserModel> list) {
67. System.out.println(list);
68. **return** "redirect:/success";
69. }
71. //Set<UserModel>
72. //ok   http://localhost:9080/springmvc-chapter6/jsonparam/set2?set=[{"username":"123"},{"username":"234"}]
73. @RequestMapping("/set2")
74. **public** String set2(@RequestJsonParam(value = "set") Set<UserModel> set) {
75. System.out.println(set);
76. **return** "redirect:/success";
77. }
79. //Map<String, UserModel>
80. //ok   http://localhost:9080/springmvc-chapter6/jsonparam/map2?map={"a":{"username":"123"},"b":{"username":"234"}}
81. //暂不支持 Map<UserModel, UserModel>
82. @RequestMapping("/map2")
83. **public** String map2(@RequestJsonParam(value = "map") MapWapper<String, UserModel> map) {
84. System.out.println(map);
85. **return** "redirect:/success";
86. }

89. //ok   http://localhost:9080/springmvc-chapter6/jsonparam/model1?model={"username":123,"password":234,"realname":"zhang","workInfo":{"city":"abc","job":"abc","year":"abc"}, "schoolInfo":{"schoolType":"1","schoolName":"1","specialty":"1"}}
90. //没有realname1
91. //fail http://localhost:9080/springmvc-chapter6/jsonparam/model1?model={"username":123,"password":234,"realname1":123}
92. @RequestMapping("/model1")
93. **public** String model1(@RequestJsonParam(value = "model", required=**true**) UserModel user) {
94. System.out.println(user);
95. **return** "redirect:/success";
96. }
98. //ENUM
99. //ok   http://localhost:9080/springmvc-chapter6/jsonparam/model2?model={"state":"normal"}
100. //List<基本类型>
101. //ok   http://localhost:9080/springmvc-chapter6/jsonparam/model2?model={"hobbyList":["film", "music"]}
102. //Map<基本类型，基本类型>
103. //ok   http://localhost:9080/springmvc-chapter6/jsonparam/model2?model={"map":{"key":"value", "a":"b"}}
104. @RequestMapping("/model2")
105. **public** String model2(@RequestJsonParam(value = "model", required=**true**) DataBinderTestModel model) {
106. System.out.println(model);
107. **return** "redirect:/success";
108. }
110. //List<UserModel>
111. //ok   http://localhost:9080/springmvc-chapter6/jsonparam/model3?model={"userList":[{"username":"1"},{"username":"2"}]}
112. //Map<String，UserModel>
113. //ok   http://localhost:9080/springmvc-chapter6/jsonparam/model3?model={"userMap":{"1":{"username":"1"},"2":{"username":"2"}}}
115. //暂不支持 类似于 Map<UserModel, UserModel> 形式
116. @RequestMapping("/model3")
117. **public** String model3(@RequestJsonParam(value = "model") DataBinderTestModel model) {
118. System.out.println(model);
119. **return** "redirect:/success";
120. }
122. }
123. </span>

支持的spring版本：

springmvc 3.0 和 3.1.x。

支持绑定的数据：

模型、集合、数组、MapWapper（Map的一个包装器，通过getInnerMap获取真实Map）

暂时不支持JSR-303数据校验.

缺点：

1、和[《扩展SpringMVC以支持更精准的数据绑定》](http://jinnianshilongnian.iteye.com/blog/1717180" \t "_self)一样的缺点。

spring自定义的参数解析器会放在默认解析器之后，不能指定order，因此如果我们@FormModel("map") Map map，此map会变成Model（请参考<http://jinnianshilongnian.iteye.com/blog/1698916> 第六部分、Model Map ModelMap），希望未来的版本支持自定义顺序来解决这个问题；此处我们使用MapWapper解决，可以通过MapWapper.getInnerMap()拿到我们需要的Map。

2、支持的jackson版本是1.4.2，版本比较老，spring3.2里程碑版已升级到2.x。

3、暂时没有提供JSR303数据校验。