**spring RestTemplate用法详解**

2016-01-07  [liang1234\_](http://www.360doc.com/userhome/835902)   阅 1498  转 5

[转藏到我的图书馆](javascript:void(0);)

[微信](javascript:void(0);)分享：

|  |
| --- |
| 前面介绍过Spring的MVC结合不同的view显示不同的数据，如：结合json的view显示json、结合xml的view显示xml文档。那么这些数据除了在WebBrowser中用JavaScript来调用以外，还可以用远程服务器的Java程序、C#程序来调用。也就是说现在的程序不仅在BS中能调用，在CS中同样也能调用，不过你需要借助RestTemplate这个类来完成。RestTemplate有点类似于一个WebService客户端请求的模版，可以调用http请求的WebService，并将结果转换成相应的对象类型。至少你可以这样理解！  Blog：<http://blog.csdn.net/IBM_hoojo>  <http://hoojo.cnblogs.com/>  **一、准备工作**  1、 下载jar包  spring各版本jar下载地址：<http://ebr.springsource.com/repository/app/library/detail?name=org.springframework.spring>  相关的依赖包也可以在这里找到：<http://ebr.springsource.com/repository/app/library>  2、 需要jar包如下  [clip_image002](http://hi.csdn.net/attachment/201106/10/0_1307675892eizQ.gif)  3、 当前工程的web.xml配置  xml version="1.0" encoding="UTF-8"?>  <web-app version="2.4"  xmlns="http://java.sun.com/xml/ns/j2ee"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee  http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd">      <servlet>  <servlet-name>dispatcherservlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServletservlet-class>  <init-param>  <param-name>contextConfigLocationparam-name>  <param-value>/WEB-INF/dispatcher.xmlparam-value>  init-param>  <load-on-startup>1load-on-startup>  servlet>    <servlet-mapping>  <servlet-name>dispatcherservlet-name>  <url-pattern>\*.dourl-pattern>  servlet-mapping>    <welcome-file-list>  <welcome-file>index.jspwelcome-file>  welcome-file-list>  web-app>  4、 WEB-INF中的dispatcher.xml配置  xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:util="http://www.springframework.org/schema/util"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd  http://www.springframework.org/schema/mvc  http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context-3.0.xsd  http://www.springframework.org/schema/util  http://www.springframework.org/schema/util/spring-util-3.0.xsd">    <context:component-scan base-package="com.hoo.\*">    <context:exclude-filter type="assignable" expression="com.hoo.client.RESTClient"/>  context:component-scan>      <bean id="handlerAdapter" class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter"/>      <bean name="xStreamMarshallingView" class="org.springframework.web.servlet.view.xml.MarshallingView">  <property name="marshaller">  <bean class="org.springframework.oxm.xstream.XStreamMarshaller">    <property name="autodetectAnnotations" value="true"/>  bean>  property>  bean>      <bean class="org.springframework.web.servlet.view.BeanNameViewResolver">  <property name="order" value="3"/>  bean>      <bean id="handlerMapping" class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping">  <property name="order" value="1" />  bean>    beans>  5、 启动后，可以看到index.jsp 没有出现异常或错误。那么当前SpringMVC的配置就成功了。  **二、REST控制器实现**  REST控制器主要完成CRUD操作，也就是对于http中的post、get、put、delete。  还有其他的操作，如head、options、trace。  具体代码：  package com.hoo.controller;    import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import org.springframework.web.servlet.ModelAndView;    /\*\*  \* **function:**SpringMVC REST示例  \* @author hoojo  \* @createDate 2011-6-9 上午11:34:08  \* @file RESTController.java  \* @package com.hoo.controller  \* @project SpringRestWS  \* @blog http://blog.csdn.net/IBM\_hoojo  \* @email hoojo\_@126.com  \* @version 1.0  \*/  @RequestMapping("/restful")  @Controller  public class RESTController {    @RequestMapping(value = "/show", method = RequestMethod.GET)  public ModelAndView show() {  System.out.println("show");  ModelAndView model = new ModelAndView("xStreamMarshallingView");  model.addObject("show method");  return model;  }    @RequestMapping(value = "/get/{id}", method = RequestMethod.GET)  public ModelAndView getUserById(@PathVariable String id) {  System.out.println("getUserById-" + id);  ModelAndView model = new ModelAndView("xStreamMarshallingView");  model.addObject("getUserById method -" + id);  return model;  }    @RequestMapping(value = "/add", method = RequestMethod.POST)  public ModelAndView addUser(String user) {  System.out.println("addUser-" + user);  ModelAndView model = new ModelAndView("xStreamMarshallingView");  model.addObject("addUser method -" + user);  return model;  }    @RequestMapping(value = "/edit", method = RequestMethod.PUT)  public ModelAndView editUser(String user) {  System.out.println("editUser-" + user);  ModelAndView model = new ModelAndView("xStreamMarshallingView");  model.addObject("editUser method -" + user);  return model;  }    @RequestMapping(value = "/remove/{id}", method = RequestMethod.DELETE)  public ModelAndView removeUser(@PathVariable String id) {  System.out.println("removeUser-" + id);  ModelAndView model = new ModelAndView("xStreamMarshallingView");  model.addObject("removeUser method -" + id);  return model;  }  }  上面的方法对应的http操作：  /show -> get 查询  /get/id -> get 查询  /add -> post 添加  /edit -> put 修改  /remove/id -> delete 删除  在这个方法中，就可以看到RESTful风格的url资源标识  @RequestMapping(value = "/get/{id}", method = RequestMethod.GET)  public ModelAndView getUserById(@PathVariable String id) {  System.out.println("getUserById-" + id);  ModelAndView model = new ModelAndView("xStreamMarshallingView");  model.addObject("getUserById method -" + id);  return model;  }  value=”/get/{id}”就是url中包含get，并且带有id参数的get请求，就会执行这个方法。这个url在请求的时候，会通过Annotation的@PathVariable来将url中的id值设置到getUserById的参数中去。 ModelAndView返回的视图是xStreamMarshallingView是一个xml视图，执行当前请求后，会显示一篇xml文档。文档的内容是添加到model中的值。  **三、利用RestTemplate调用REST资源**  代码如下：  package com.hoo.client;    import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Component;  import org.springframework.web.client.RestTemplate;    /\*\*  \* **function:**RestTemplate调用REST资源  \* @author hoojo  \* @createDate 2011-6-9 上午11:56:16  \* @file RESTClient.java  \* @package com.hoo.client  \* @project SpringRestWS  \* @blog http://blog.csdn.net/IBM\_hoojo  \* @email hoojo\_@126.com  \* @version 1.0  \*/  @Component  public class RESTClient {    @Autowired  private RestTemplate template;    private final static String url = "http://localhost:8080/SpringRestWS/restful/";    public String show() {  return template.getForObject(url + "show.do", String.class, new String[]{});  }    public String getUserById(String id) {  return template.getForObject(url + "get/{id}.do", String.class, id);  }    public String addUser(String user) {  return template.postForObject(url + "add.do?user={user}", null, String.class, user);  }    public String editUser(String user) {  template.put(url + "edit.do?user={user}", null, user);  return user;  }    public String removeUser(String id) {  template.delete(url + "/remove/{id}.do", id);  return id;  }  }  RestTemplate的getForObject完成get请求、postForObject完成post请求、put对应的完成put请求、delete完成delete请求；还有execute可以执行任何请求的方法，需要你设置RequestMethod来指定当前请求类型。  RestTemplate.getForObject(String url, Class responseType, String... urlVariables)  参数url是http请求的地址，参数Class是请求响应返回后的数据的类型，最后一个参数是请求中需要设置的参数。  template.getForObject(url + "get/{id}.do", String.class, id);  如上面的参数是{id}，返回的是一个string类型，设置的参数是id。最后执行该方法会返回一个String类型的结果。  下面建立一个测试类，完成对RESTClient的测试。代码如下：  package com.hoo.client;    import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.test.context.ContextConfiguration;  import org.springframework.test.context.junit38.AbstractJUnit38SpringContextTests;    /\*\*  \* **function:**RESTClient TEST  \* @author hoojo  \* @createDate 2011-6-9 下午03:50:21  \* @file RESTClientTest.java  \* @package com.hoo.client  \* @project SpringRestWS  \* @blog http://blog.csdn.net/IBM\_hoojo  \* @email hoojo\_@126.com  \* @version 1.0  \*/  @ContextConfiguration("classpath:applicationContext-\*.xml")  public class RESTClientTest extends AbstractJUnit38SpringContextTests {    @Autowired  private RESTClient client;    public void testShow() {  System.out.println(client.show());  }    public void testGetUserById() {  System.out.println(client.getUserById("abc"));  }    public void testAddUser() {  System.out.println(client.addUser("jack"));  }    public void testEditUser() {  System.out.println(client.editUser("tom"));  }    public void testRemoveUser() {  System.out.println(client.removeUser("aabb"));  }  }  我们需要在src目录下添加applicationContext-beans.xml完成对restTemplate的配置。restTemplate需要配置MessageConvert将返回的xml文档进行转换，解析成JavaObject。  xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context-3.0.xsd">    <context:component-scan base-package="com.hoo.\*"/>    <bean id="restTemplate" class="org.springframework.web.client.RestTemplate">  <property name="messageConverters">  <list>  <bean class="org.springframework.http.converter.xml.MarshallingHttpMessageConverter">  <property name="marshaller" ref="xStreamMarshaller"/>  <property name="unmarshaller" ref="xStreamMarshaller"/>  bean>  list>  property>  bean>    <bean id="xStreamMarshaller" class="org.springframework.oxm.xstream.XStreamMarshaller">  <property name="annotatedClasses">  <array>  array>  property>  bean>  beans>  上面配置了xStreamMarshaller是和RESTController中的ModelAndView的view对应的。因为那边是用xStreamMarshaller进行编组的，所以RestTemplate这边也需要用它来解组。RestTemplate还指出其他的MarshallingHttpMessageConverter； |