<https://blog.csdn.net/Nabcdefgh/article/details/78367244>

Restful风格的API是一种软件架构风格，设计风格而不是标准，只是提供了一组设计原则和约束条件。它主要用于客户端和服务器交互类的软件。基于这个风格设计的软件可以更简洁，更有层次，更易于实现缓存等机制。

在Restful风格中，用户请求的url使用同一个url而用请求方式：get，post，delete，put...等方式对请求的处理方法进行区分，这样可以在前后台分离式的开发中使得前端开发人员不会对请求的资源地址产生混淆和大量的检查方法名的麻烦，形成一个统一的接口。

在Restful风格中，现有规定如下：

* GET（SELECT）：从服务器查询，可以在服务器通过请求的参数区分查询的方式。
* POST（CREATE）：在服务器新建一个资源，调用insert操作。
* PUT（UPDATE）：在服务器更新资源，调用update操作。
* PATCH（UPDATE）：在服务器更新资源（客户端提供改变的属性）。(目前jdk7未实现，tomcat7也不行)。
* DELETE（DELETE）：从服务器删除资源，调用delete语句。

了解这个风格定义以后，我们举个例子：

如果当前url是 http://localhost:8080/User

那么用户只要请求这样同一个URL就可以实现不同的增删改查操作，例如

http://localhost:8080/User?\_method=get&id=1001　　这样就可以通过get请求获取到数据库 user 表里面 id=1001 的用户信息

http://localhost:8080/User?\_method=post&id=1001&name=zhangsan　　这样可以向数据库 user 表里面插入一条记录

http://localhost:8080/User?\_method=put&id=1001&name=lisi　　这样可以将 user表里面 id=1001 的用户名改为lisi

http://localhost:8080/User?\_method=delete&id=1001　　这样用于将数据库 user 表里面的id=1001 的信息删除

这样定义的规范我们就可以称之为restful风格的API接口，我们可以通过同一个url来实现各种操作。

接下来我们讲解spring-mvc中是如何实现restful风格API接口的，并且对其中出现的问题进行解决！（java web对 put 和 delete 请求的不支持问题）

首先我们搭建好spring mvc的项目接口，并按照restful风格写好控制器，这里我写了一个User控制器类和一个User "Action" ->



这里的controller 和 action的url地址是按照restful风格编写的 访问地址 /User/User 用method区分请求方法

我们的前台使用的是jquery ajax进行请求->









有人会问了？ 为什么delete 和put用的也是post的请求，这里就要说说java里面对put和delete的不支持了->

java里面原本是对put和delete请求进行过滤掉的（不知道为什么要这么做），而且在servlet里面还有doGet，doPost，doDelete,doPut的对应方法，但是不可以使用（尴尬不尴尬），同样spring mvc里面也有对应的method=RequestMethod.PUT 和Delete,但是ajax里面type写成Put、Delete是可以访问到对应的方法的，但是参数却无法传递过去，所有传递过去的参数都是null（郁闷不郁闷）！C#就不会这样，C#的API编程需要开启一下PUT和Delete就可以了，并不需要java里面这么复杂，说到这里我们解决一下这个问题->

首先在springMVC 项目的Web.xml里面增加一个过滤器filter

按 Ctrl+C 复制代码



按 Ctrl+C 复制代码

当然有些新手不知道这段代码加在哪里，那么我就将我的web.xml一并粘贴在此处（我也搞这个半天...）

复制代码

1 <?xml version="1.0" encoding="UTF-8"?>

2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

3 xmlns="http://java.sun.com/xml/ns/javaee"

4 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"

5 version="3.0">

6

7 <!-- 浏览器不支持put,delete等method,由该filter将/xxx?\_method=delete转换为标准的http delete方法 -->

8 <filter>

9 <filter-name>hiddenHttpMethodFilter</filter-name>

10 <filter-class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-class>

11 </filter>

12 <filter-mapping>

13 <filter-name>hiddenHttpMethodFilter</filter-name>

14 <url-pattern>/\*</url-pattern>

15 </filter-mapping>

16

17 <!--这段代码如果不用上面的的话是可以实现put的-->

18 <!--<filter>

19 <filter-name>HttpMethodPutFilter</filter-name>

20 <filter-class>org.springframework.web.filter.HttpPutFormContentFilter</filter-class>

21 </filter>

22 <filter-mapping>

23 <filter-name>HttpMethodPutFilter</filter-name>

24 <url-pattern>/\*</url-pattern>

25 </filter-mapping>-->

26

27

28 <welcome-file-list>

29 <welcome-file>/index.jsp</welcome-file>

30 </welcome-file-list>

31 <!-- Spring MVC配置 -->

32 <servlet>

33 <servlet-name>spring</servlet-name>

34 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

35

36 <!-- load-on-startup元素标记容器是否在启动的时候就加载这个servlet(实例化并调用其init()方法) -->

37 <load-on-startup>1</load-on-startup>

38 </servlet>

39

40 <servlet-mapping>

41 <servlet-name>spring</servlet-name>

42 <url-pattern>/</url-pattern>

43 </servlet-mapping>

44

45 <!-- Spring配置 -->

46 <listener>

47 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

48 </listener>

49

50 <!-- 指定Spring Bean的配置文件所在目录。默认配置在WEB-INF目录下 -->

51 <context-param>

52 <param-name>contextConfigLocation</param-name>

53 <param-value>classpath:applicationContext.xml</param-value>

54 </context-param>

55 </web-app>

复制代码

这里我们将过滤器配置好了，我有一段注释掉了，如果用下面这个配置文件->

复制代码

1 <!--这段代码如果不用上面的的话是可以实现put的-->

2 <filter>

3 <filter-name>HttpMethodPutFilter</filter-name>

4 <filter-class>org.springframework.web.filter.HttpPutFormContentFilter</filter-class>

5 </filter>

6 <filter-mapping>

7 <filter-name>HttpMethodPutFilter</filter-name>

8 <url-pattern>/\*</url-pattern>

9 </filter-mapping>

复制代码

这个配置项如果写在这里的话是可以支持PUT请求的，但是DELETE请求依然不可以，那么我只能选择第一种方法了

复制代码

1 <filter>

2 <filter-name>hiddenHttpMethodFilter</filter-name>

3 <filter-class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-class>

4 </filter>

5 <filter-mapping>

6 <filter-name>hiddenHttpMethodFilter</filter-name>

7 <url-pattern>/\*</url-pattern>

8 </filter-mapping>

复制代码

这一段的方法是用 org.springframework.web.filter.HiddenHttpMethodFilter 的内置过滤器类进行对http请求的标准化。这样让我们可以自己声明请求的方式。

配置完成这个以后，我们在ajax里面需要传递一个参数\_method:"PUT" 和 \_method:"DELETE"，但是请求方式仍然是POST



这样配置的话，我们已经可以实现对DELETE修饰的方法进行访问，同样\_method:'PUT'我们可以对PUT修饰的方法进行访问，这样我们上面定义的控制器类已经可以实现了。

转载请注明出处：<http://www.cnblogs.com/qixiaoyizhan/> 且在文章页面明显位置给出原文链接，否则保留追究法律责任的权利