## JAX-RS简介

JAX-RS是一套用JAVA实现REST服务的规范，提供了一些标注将一个资源类，一个POJO JAVA类，封装为Web资源。标注包括：

[1.@Path](mailto:1.@Path)，标注资源类或方法的相对路径

[2.@GET/@PUT/@POST/@DELETE](mailto:2.@GET/@PUT/@POST/@DELETE)，标注方法是用的HTTP请求的类型

[3.@Produces](mailto:3.@Produces)，标注返回的MIME媒体类型

[4.@Consumes](mailto:4.@Consumes)，标注可接受请求的MIME媒体类型

[5.@PathParam/@QueryParam/@HeaderParam/@CookieParam/@MatrixParam/@FormParam](mailto:5.@PathParam/@QueryParam/@HeaderParam/@CookieParam/@MatrixParam/@FormParam)，分别标注方法的参数来自于HTTP请求的不同位置，例如@PathParam来自于URL的路径，@QueryParam来自于URL的查询参数，@HeaderParam来自于HTTP请求的头信息，@CookieParam来自于HTTP请求的Cookie。

媒体类型：text/plain,文本类型

text/html,html类型

application/json,json类型

applicaiton/xml,xml类型

注意：这些类型后面建议加上字符类型(;charset=UTF-8),否则传输中文字符时，可能乱码

## 目前JAX-RS的实现包括：

* [Apache CXF](http://liugang594.iteye.com/wiki/Apache_CXF" \t "_blank" \o "Apache CXF)，开源的[Web服务](http://liugang594.iteye.com/wiki/Web%E6%9C%8D%E5%8A%A1" \o "Web服务" \t "_blank)框架。
* [Jersey](http://liugang594.iteye.com/w/index.php?title=Jersey&action=edit&redlink=1" \t "_blank" \o "Jersey)， 由[Sun](http://liugang594.iteye.com/wiki/Sun" \t "_blank" \o "Sun)提供的JAX-RS的参考实现。
* [RESTEasy](http://liugang594.iteye.com/w/index.php?title=RESTEasy&action=edit&redlink=1" \t "_blank" \o "RESTEasy)，[JBoss](http://liugang594.iteye.com/wiki/JBoss" \o "JBoss" \t "_blank)的实现。
* [Restlet](http://liugang594.iteye.com/w/index.php?title=Restlet&action=edit&redlink=1" \t "_blank" \o "Restlet)，由Jerome Louvel和Dave Pawson开发，是最早的REST框架，先于JAX-RS出现。
* [Apache Wink](http://liugang594.iteye.com/w/index.php?title=Apache_Wink&action=edit&redlink=1" \t "_blank" \o "Apache Wink)，一个[Apache软件基金会](http://liugang594.iteye.com/wiki/Apache%E8%BD%AF%E4%BB%B6%E5%9F%BA%E9%87%91%E4%BC%9A" \o "Apache软件基金会" \t "_blank)孵化器中的项目，其服务模块实现JAX-RS规范

## 创建项目

### 导入jar包

|  |
| --- |
| <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>org.redis</groupId>  <artifactId>redis\_dev</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>war</packaging>  <dependencies>  <!-- 时间操作组件 -->  <dependency>  <groupId>joda-time</groupId>  <artifactId>joda-time</artifactId>  <version>2.9</version>  </dependency>  <!-- Apache工具组件 -->  <dependency>  <groupId>org.apache.commons</groupId>  <artifactId>commons-lang3</artifactId>  <version>3.3.2</version>  </dependency>  <dependency>  <groupId>org.apache.commons</groupId>  <artifactId>commons-io</artifactId>  <version>1.3.2</version>  </dependency>  <dependency>  <groupId>commons-net</groupId>  <artifactId>commons-net</artifactId>  <version>3.3</version>  </dependency>  <!-- Jackson Json处理工具包 -->  <dependency>  <groupId>com.fasterxml.jackson.core</groupId>  <artifactId>jackson-databind</artifactId>  <version>2.8.5</version>  </dependency>  <!-- httpclient -->  <dependency>  <groupId>org.apache.httpcomponents</groupId>  <artifactId>httpclient</artifactId>  <version>4.4</version>  </dependency>  <!-- 单元测试 -->  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.12</version>  <scope>test</scope>  </dependency>  <!-- 日志处理 -->  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-log4j12</artifactId>  <version>1.6.4</version>  </dependency>  <!-- Mybatis -->  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis</artifactId>  <version>3.3.0</version>  </dependency>  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis-spring</artifactId>  <version>1.2.3</version>  </dependency>  <dependency>  <groupId>com.github.miemiedev</groupId>  <artifactId>mybatis-paginator</artifactId>  <version>1.2.15</version>  </dependency>  <dependency>  <groupId>com.github.pagehelper</groupId>  <artifactId>pagehelper</artifactId>  <version>4.1.6</version>  </dependency>  <!-- MySql -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.32</version>  </dependency>  <!-- 连接池 -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>druid</artifactId>  <version>1.0.9</version>  </dependency>  <!-- Spring -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>4.2.5.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  <version>4.2.5.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>4.2.5.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>4.2.5.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aspects</artifactId>  <version>4.2.5.RELEASE</version>  </dependency>  <!-- JSP相关 -->  <dependency>  <groupId>jstl</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>servlet-api</artifactId>  <version>2.5</version>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jsp-api</artifactId>  <version>2.0</version>  <scope>provided</scope>  </dependency>  <!-- 文件上传组件 -->  <dependency>  <groupId>commons-fileupload</groupId>  <artifactId>commons-fileupload</artifactId>  <version>1.3.1</version>  </dependency>  <!-- Redis客户端 -->  <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  <version>2.7.2</version>  </dependency>  <dependency>  <groupId>org.springframework.data</groupId>  <artifactId>spring-data-redis</artifactId>  <version>1.6.1.RELEASE</version>  </dependency>  <!-- spring session -->  <dependency>  <groupId>org.springframework.session</groupId>  <artifactId>spring-session</artifactId>  <version>1.2.2.RELEASE</version>  </dependency>  <!-- cxf -->  <dependency>  <groupId>org.apache.cxf</groupId>  <artifactId>cxf-rt-frontend-jaxws</artifactId>  <version>3.1.6</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.apache.cxf/cxf-core -->  <!-- 这个包是必须的，因为会使用拦截器等记录时，需要这个包 -->  <dependency>  <groupId>org.apache.cxf</groupId>  <artifactId>cxf-core</artifactId>  <version>3.1.6</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.apache.cxf/cxf-rt-transports-http -->  <dependency>  <groupId>org.apache.cxf</groupId>  <artifactId>cxf-rt-transports-http</artifactId>  <version>3.1.6</version>  </dependency>  <dependency>  <groupId>org.apache.cxf</groupId>  <artifactId>cxf-rt-transports-http-jetty</artifactId>  <version>3.1.6</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.apache.cxf/cxf-rt-ws-security -->  <dependency>  <groupId>org.apache.cxf</groupId>  <artifactId>cxf-rt-ws-security</artifactId>  <version>3.1.6</version>  </dependency>  <!-- cxf-rs -->  <!-- https://mvnrepository.com/artifact/org.apache.cxf/cxf-rt-frontend-jaxrs -->  <dependency>  <groupId>org.apache.cxf</groupId>  <artifactId>cxf-rt-frontend-jaxrs</artifactId>  <version>3.1.6</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.apache.cxf/cxf-rt-rs-client -->  <dependency>  <groupId>org.apache.cxf</groupId>  <artifactId>cxf-rt-rs-client</artifactId>  <version>3.1.6</version>  </dependency>  <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.jaxrs/jackson-jaxrs-json-provider -->  <dependency>  <groupId>com.fasterxml.jackson.jaxrs</groupId>  <artifactId>jackson-jaxrs-json-provider</artifactId>  <version>2.8.9</version>  </dependency>  <!-- freemarker -->  <dependency>  <groupId>org.freemarker</groupId>  <artifactId>freemarker</artifactId>  <version>2.3.22</version>  </dependency>  <dependency>  <groupId>com.fasterxml.jackson.core</groupId>  <artifactId>jackson-databind</artifactId>  <version>2.8.5</version>  </dependency>  <!-- jetty -->  <!-- https://mvnrepository.com/artifact/org.eclipse.jetty/jetty-util -->  <dependency>  <groupId>org.eclipse.jetty</groupId>  <artifactId>jetty-util</artifactId>  <version>9.2.15.v20160210</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.eclipse.jetty/jetty-server -->  <dependency>  <groupId>org.eclipse.jetty</groupId>  <artifactId>jetty-server</artifactId>  <version>9.2.15.v20160210</version>  </dependency>  <!-- https://mvnrepository.com/artifact/org.eclipse.jetty/jetty-http -->  <dependency>  <groupId>org.eclipse.jetty</groupId>  <artifactId>jetty-http</artifactId>  <version>9.2.15.v20160210</version>  </dependency>  </dependencies>  </project> |

### 创建一个用于服务的接口

|  |
| --- |
| package tt.rs.service;  import javax.ws.rs.Consumes;  import javax.ws.rs.GET;  import javax.ws.rs.POST;  import javax.ws.rs.Path;  import javax.ws.rs.PathParam;  import javax.ws.rs.Produces;  import javax.ws.rs.QueryParam;  import tt.student.dto.Student;  /\*\*  \* 这里的@Path注解的值是一个相对的URI路径，这个路径指定了该Java类的位置，  \* 例如/helloworld。在这个URI中可以包含变量，例如可以获取用户的姓名然后作为参数传入URI中：/helloworld/{username}。  \*  \* @Produces:@Produces注解用来指定资源能够生成并发送给客户端的MIME媒体类型，例如“text/plain”.  \* 四种媒体类型：  \* 1.text/plain:文本类型  \* 2.text/html:html类型  \* 3.application/json:json类型  \* 4.application/xml:xml类型  \*  \* 注意：使用这四种类型时，建议在后面加上;charset=UTF-8，以防中文乱码  \*  \*  \* @author user  \*  \*/  @Path(value = "/say")  //@Produces("text/plain;charset=UTF-8")  @Produces({"application/json;charset=UTF-8"})  public interface RestFulService {  /\*\*  \* 这里的@Path注解的值是一个相对的URI路径，这个路径指定了该Java类的位置，  \* 例如/helloworld。在这个URI中可以包含变量，例如可以获取用户的姓名然后作为参数传入URI中：/helloworld/{username}。  \*  \* @GET:@GET注解是请求方法指示符，这个指示符注解的Java方法会处理HTTPGET请求。资源的行为由资源回应的HTTP方法决定。  \*  \* @QueryParam:@QueryParam注解是可以抽取并在资源类中使用的一类参数。Query参数是从请求URI的查询参数中抽取的。  \*  \* @param name  \* @return  \*/  @Path(value = "/hello")  @GET  String sayHi(@QueryParam(value = "name") String name);    /\*\*  \* @PathParam:@PathParam注解是可以抽取并用在资源类中的一类参数。URIpath参数是从请求的URI中抽取的，而且参数的名称和@Path注解中定义的变量名对应。  \* @param name  \* @return  \*/  @Path(value = "/sayHello/{name}")  @GET  String sayHello(@PathParam("name") String name);    /\*\*  \* @Consumes:@Consumes注解是用来指定资源能够接受的客户发送的MIME媒体类型。  \* @param student  \* @return  \*/  @Path(value = "/student")  @POST  @Consumes("application/json;charset=UTF-8")  Student getAll(Student student);    } |

### 接口的实现类

|  |
| --- |
| package tt.rs.service.impl;  import tt.rs.service.RestFulService;  import tt.student.dto.Student;  public class RestFulServiceImpl implements RestFulService {  @Override  public String sayHi(String name) {  String str = name + ",你好,我想对你说Hello World!";  return str;  }  @Override  public String sayHello(String name) {  return name + ",Hello World!";  }  @Override  public Student getAll(Student student) {  // System.out.println(student.getName() + ", welcome username you");  Student stu = new Student();  stu.setName(student.getName());  stu.setAge(student.getAge());  stu.setAddress(student.getAddress());  return stu;  }  } |

### 发布服务

|  |
| --- |
| package tt.rs.test;  import java.util.ArrayList;  import java.util.List;  import org.apache.cxf.jaxrs.JAXRSServerFactoryBean;  import com.fasterxml.jackson.jaxrs.json.JacksonJsonProvider;  import tt.rs.service.impl.StudentRestServiceImpl;  public class PubishTest {  public static void main(String[] args) {  //添加provider  List<Object> providerList = new ArrayList<Object>();  providerList.add(new JacksonJsonProvider());    //发布REST任务  JAXRSServerFactoryBean sf = new JAXRSServerFactoryBean();  sf.setResourceClasses(StudentRestServiceImpl.class); //设置资源类，一定要使用实现类  sf.setAddress("http://localhost:8080/"); //这里注意只需要声明地址和端口号即可  sf.setProviders(providerList); //用于实现 JSON 数据的序列化与反序列化  sf.create();  }  } |

## 客户端调用（这里的调用全部用HttpUrlConnection进行调用）

### GET方法调用

|  |
| --- |
| package com.cxf.rs.test;  import java.io.ByteArrayOutputStream;  import java.io.InputStream;  import java.net.HttpURLConnection;  import java.net.URL;  public class ClientTest {  public static void main(String[] args) {  try {  URL url = new URL("http://localhost:8080/say/hello?name=lisi");    //打开连接  HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();  System.out.println(urlConnection.getResponseCode());    if(200 == urlConnection.getResponseCode()){  //得到输入流  InputStream is = urlConnection.getInputStream();  ByteArrayOutputStream baos = new ByteArrayOutputStream();  byte[] buffer = new byte[1024];  int len = 0;  while(-1 != (len = is.read(buffer))){  baos.write(buffer,0,len);  baos.flush();  }  System.out.println(baos.toString("utf-8"));  }  } catch (Exception e) {  e.printStackTrace();  }  }  } |

### POST方法调用

|  |
| --- |
| package com.cxf.rs.test;  import java.io.BufferedInputStream;  import java.io.ByteArrayOutputStream;  import java.io.PrintWriter;  import java.net.HttpURLConnection;  import java.net.URL;  public class ClientPostTest {  public static void main(String[] args) {  URL url = null;    try {  url = new URL("http://localhost:8080/select/getName2");  HttpURLConnection httpUrlConnection = (HttpURLConnection) url.openConnection();  httpUrlConnection.setRequestMethod("POST"); //提交方式  // httpUrlConnection.setConnectTimeout(10000);//连接超时 单位毫秒  // httpUrlConnection.setReadTimeout(2000);//读取超时 单位毫秒  // 发送POST请求必须设置如下两行  httpUrlConnection.setDoOutput(true);  httpUrlConnection.setDoInput(true);  httpUrlConnection.setRequestProperty("Content-Type", "application/json");  httpUrlConnection.setUseCaches(false);  httpUrlConnection.setConnectTimeout(30000);  httpUrlConnection.setReadTimeout(30000);  httpUrlConnection.setInstanceFollowRedirects(false);    // 获取URLConnection对象对应的输出流  PrintWriter printWriter = new PrintWriter(httpUrlConnection.getOutputStream());  // 发送请求参数  String post = "{\"id\":\"1\",\"name\":\"zhangsan\",\"sex\":\"男\",\"age\":\"20\",\"address\":\"上海市\"}";  printWriter.write(post);//post的参数 xx=xx&yy=yy  // flush输出流的缓冲  printWriter.flush();  //开始获取数据  BufferedInputStream bis = new BufferedInputStream(httpUrlConnection.getInputStream());  ByteArrayOutputStream bos = new ByteArrayOutputStream();  int len;  byte[] arr = new byte[1024];  while((len=bis.read(arr))!= -1){  bos.write(arr,0,len);  bos.flush();  }  bos.close();    System.out.println(bos.toString("utf-8"));      } catch (Exception e) {  e.printStackTrace();  }  }  } |

## 与spring / mybatis的整合

### 创建web.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns="http://xmlns.jcp.org/xml/ns/javaee"  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"  id="WebApp\_ID" version="3.1">  <display-name>redis\_dev</display-name>  <!-- 加载spring容器 -->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring/applicationContext\*.xml</param-value>  </context-param>  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <!-- 解决post乱码 -->  <filter>  <filter-name>CharacterEncodingFilter</filter-name>  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <init-param>  <param-name>encoding</param-name>  <param-value>utf-8</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>CharacterEncodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>    <filter>  <filter-name>spring-session</filter-name>  <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>  <async-supported>true</async-supported>  <init-param>  <param-name>targetBeanName</param-name>  <param-value>springSession</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>spring-session</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>    <!-- springmvc的前端控制器 -->  <servlet>  <servlet-name>mvc</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <!-- contextConfigLocation不是必须的， 如果不配置contextConfigLocation， springmvc的配置文件默认在：WEB-INF/servlet的name+"-servlet.xml" -->  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:springmvc/springmvc.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>mvc</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>    <!--定义一个cxf的servlet-->  <servlet>  <servlet-name>CXFServlet</servlet-name>  <servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>  </servlet>  <servlet-mapping>  <servlet-name>CXFServlet</servlet-name>  <url-pattern>/rsWebService/\*</url-pattern>  </servlet-mapping>    <welcome-file-list>  <welcome-file>login</welcome-file>  </welcome-file-list>  </web-app> |

### 创建spring的xml文件

略

### 创建发布jax-rs所需的xml文件

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:context="http://www.springframework.org/schema/context" xmlns:p="http://www.springframework.org/schema/p"  xmlns:aop="http://www.springframework.org/schema/aop" xmlns:tx="http://www.springframework.org/schema/tx"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:jaxws="http://cxf.apache.org/jaxws"  xmlns:jaxrs="http://cxf.apache.org/jaxrs"  xmlns:cxf="http://cxf.apache.org/core"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd  http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd  http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd  http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.0.xsd  http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd  http://cxf.apache.org/jaxrs http://cxf.apache.org/schemas/jaxrs.xsd  http://cxf.apache.org/core http://cxf.apache.org/schemas/core.xsd">    <!--ws的调用记录feature-->  <bean id="ttInvokeFeature" class="tt.itf.interceptor.ws.feature.TtInvokeFeature"></bean>    <cxf:bus>  <cxf:features>  <ref bean="ttInvokeFeature"></ref>  </cxf:features>  </cxf:bus>  <bean id="helloWorldRest" class="tt.rs.service.impl.RestFulServiceImpl"/>    <bean id="studentRest" class="tt.rs.service.impl.StudentRestServiceImpl"/>  <jaxrs:server id="userWebCXFService" address="/rs">  <jaxrs:serviceBeans>  <ref bean="helloWorldRest"/>  <ref bean="studentRest"/>  </jaxrs:serviceBeans>    <jaxrs:providers>  <bean class="com.fasterxml.jackson.jaxrs.json.JacksonJsonProvider"/>  </jaxrs:providers>  </jaxrs:server>  </beans> |

注意：其中的红色字体，建议加上，因为如果用到拦截器进行记录的话，需要用到蓝色字体，如果去掉的话会用不了

创建服务的接口和实现类和上面的步骤一样，之后部署项目就可以使用了，调用接口的话也可以用上面客户端的调用方法

**讲完，祝愉快!!!**