# 基本概念

REST是一种跨平台、跨语言的架构风格。基本实现形式：HTTP+URI+XML，但不是唯一的实现形式。具体而言，HTTP协议和URI用于统一接口和定位资源，文本、二进制流、XML和JSON等格式用来作为资源的表述，关注点更容易放在REST的核心概念和业务逻辑上。

JAX-RS（Java API for RESTful Web Services）是Java领域的REST式的Web服务的标准规范，是使用Java完成REST服务的基本约定。

# 特点

1、客户端-服务器的

2、无状态的

3、可缓存的

4、统一接口

5、分层系统

6、按需编码

# REST说明

Rest（representational state transfer, 表现层状态转移）是一种渐渐变成Web设计主流的设计理念。

（1）资源（**Resour**ce）：网络上一个实体（具体信息），每个资源都用一个URI来标识和定位，所有的资源都位于服务器中。

（2）表现层（**Representation**）：资源的表现形式。例如文本信息可以用Txt展现，也可以用HTML，XML，JSON格式表现，甚至是二进制格式。URI只代表资源实体，它的表现形式在Http请求头中用Accept和Content-Type字段指定，这两个字段才是对表现层的描述。客户端见到的所有东西都只是服务器上资源的表现层，客户端和服务器之间传递的也都是表现层（资源请求携带的参数，返回的JSON，TXT，JPG等MIME-TYPE）。

（3）状态转换（State Transfer）：客户端所有操作本质上就是用某种方法让服务器中的资源状态发生变化。客户端只能见到资源的表现层，所以服务器上资源状态的转换必然建立在表现层上。客户端让服务器资源发生状态变化的唯一方法就是使用HTTP请求，通过HTTP请求的不同方法（Method）实现对资源的不同的状态更改操作（如增删改查Create,Read,Update,Delete）。HTTP协议中设计的请求方法包括GET(获取)，POST(新增)，PUT(更新)，DELETE(删除)，HEAD，STATUS，OPTIONS等，不同方法代表了不同的操作，但是HTML只实现了GET和POST。

# JAX-RS2的目标

## 目标

1、基于POJO：JAX-RS2的API提供一组注解和相关的接口、类，并定义了POJO对象的生命周期和作用域，规定使用POJO来公布Web资源。

2、以HTTP为中心：JAX-RS2采用了HTTP协议，并提供了清晰的HTTP和资源定位元素来映射相关的API类和注解。

3、格式独立性：JAX-RS2对传输数据（HTTP Entity）的类型/格式的支持非常广泛，允许在标准风格之上使用额外的数据类型。

说明：HTTP Entity实体既可以是流也可以是字符串形式。

4、容器独立性：JAX-RS2的应用可以部署在各种Servlet容器中，比如Tomcat/Jetty，也可以部署在支持JAX-WS的容器中，比如：GlassFish。

5、内置于Java EE：JAX-RS2是Java EE规范的一部分，定义了在一个Java EE容器内的Web资源类的内部，如何使用Java EE的功能和组件。

## POJO

### POJO概念

POJO（Plain Old Java Objects）简单的Java对象，实际就是普通JavaBeans，是为了避免和EJB混淆所创造的简称。它通指没有使用Entity Beans的普通java对象，可以把POJO作为支持业务逻辑的协助类。

POJO有一些private的 [参数](http://baike.baidu.com/view/327406.htm)作为对象的属性。然后针对每个[参数](http://baike.baidu.com/view/327406.htm)定义了get和set方法作为访问的接口。例如：

public class User {

private long id;

private String name;

public void setId(long id) {

this. id = id;

}

public void setName(String name) {

this. name=name;

}

public long getId() {

return id;

}

public String getName() {

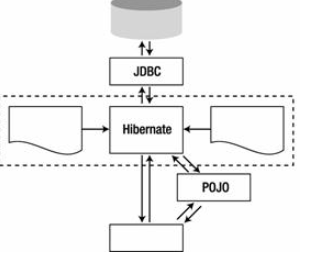
return name;

}

}

用来表示普通的Java对象，不是JavaBean, EntityBean 或者 SessionBean。POJO不担当任何特殊的角色，也不实现任何特殊的Java框架的接口如， [EJB](http://baike.baidu.com/view/3542.htm)， [JDBC](http://baike.baidu.com/view/25611.htm)等等。

即POJO是一个简单的普通的Java对象，它不包含业务[逻辑或](http://baike.baidu.com/view/674175.htm)持久逻辑等，但不是JavaBean、EntityBean等，不具有任何特殊角色和不继承或不实现任何其它Java框架的类或接口。



### POJO的意义

POJO让开发者可专注于业务逻辑和脱离框架的单元测试。除此之外， 由于POJO并不须要继承框架的类或实现其接口，开发者能够极其灵活地搭建继承结构和建造应用。

POJO的意义就在于它的简单而灵活性，因为它的简单和灵活，使得POJO能够任意扩展，从而胜任多个场合，也就让一个模型贯穿多个层成为现实。先写一个核心POJO，然后实现业务逻辑接口和持久化接口，就成了Domain Model； UI需要使用时，就实现数据绑定接口，变成VO（View Object）。

# JAX-RS元素

## 资源类

使用JAX-RS注解来实现相关Web资源的Java类。如果用MVC三层架构，那么资源类位于最前端，用于接收请求和返回响应。通常，我们使用resource作为包名，三层的包定义形如：resource-service-dao。

## 根资源类

使用@Path注解，提供资源类树的根资源及其子资源的访问。资源类分为根资源类和子资源类。

## 请求方法标识符

使用运行期注解@HttpMethod，用来标识处理资源的HTTP请求方法，该方法将使用资源类的相应方法处理，标准的方法包括DELETE、GET、HEAD、OPTIONS、POST、PUT。

## 资源方法

资源类中定义的方法使用了请求方法标识符，用来处理相关资源的请求。

## 子资源标识符

资源类中定义的方法，用来定位相关资源的子资源。

## 子资源方法

资源类中定义的方法，用来处理相关资源的子资源的请求。

## Provider

一种JAX-RS扩展接口的实现类，扩展了JAX-RS运行期的能力。

## Filter

一种用于过滤请求和响应的Provider。

## Entity Interceptor

一种用于处理拦截消息读写的Provider。

## Invocation

一种用于配置发布HTTP请求的客户端API对象。

## WebTarget

一种使用URI标识的Invocation容器对象。

## Link

一种携带元数据的URI，包括媒体类型、关系和标题等。

# Jersey项目概要

## 基本概念

Jersey是JAX-RS标准的参考实现，是Java领域中最纯正的REST服务开发框架，Jersey项目是GlassFish项目的一个子项目，专门用来实现JAX-RS标准，并提供了扩展特性。

## Jersey官网地址

https://jersey.github.io/

## Jersey项目的下载地址

https://jersey.github.io/download.html

## 源代码的托管地址

https://github.com/jersey/jersey

我们可以通过git命令，将jersey主干代码迁出到本地，示例：

git clone https://github.com/jersey/jersey.git

## Jersey的模块

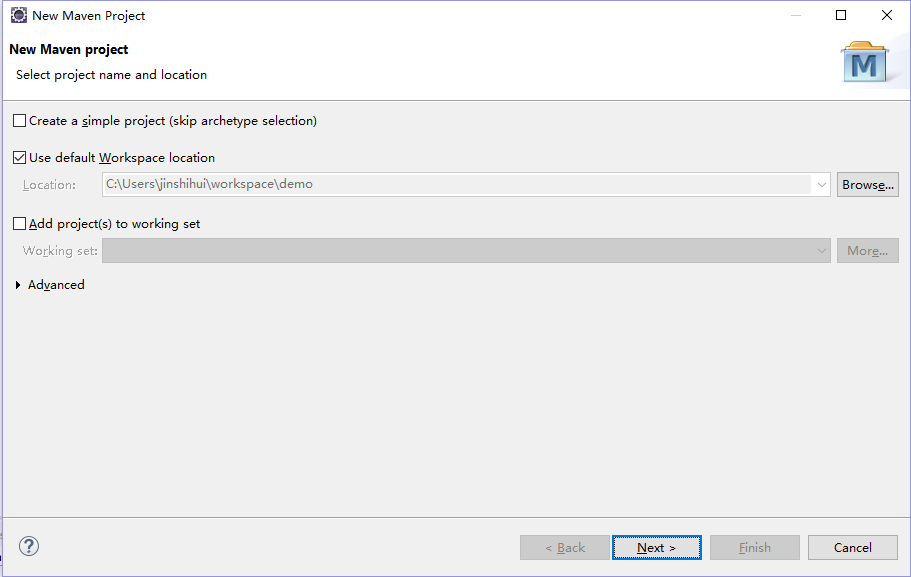
Jersey框架是由核心模块、容器模块、连接器模块、Media模块、扩展模块、测试框架模块、安全模块以及Glassfish Bundle模块等8个大的模块组成。jersey核心模块包括3个子模块，分别是通用包、服务器端实现包和客户端实现包，Jersey提供了3种HTTP容器，分别是Grizzly2、JDK-HTTP和SIMPLE-HTTP，Grizzly2同时提供了Servlet容器。Jersey客户端底层依赖于连接器来实现网络通信，如果标准的客户端模块功能不能满足业务需求，可以考虑引入Grizzly连接器包或者Apache连接器包。

## 快速实现Java REST服务

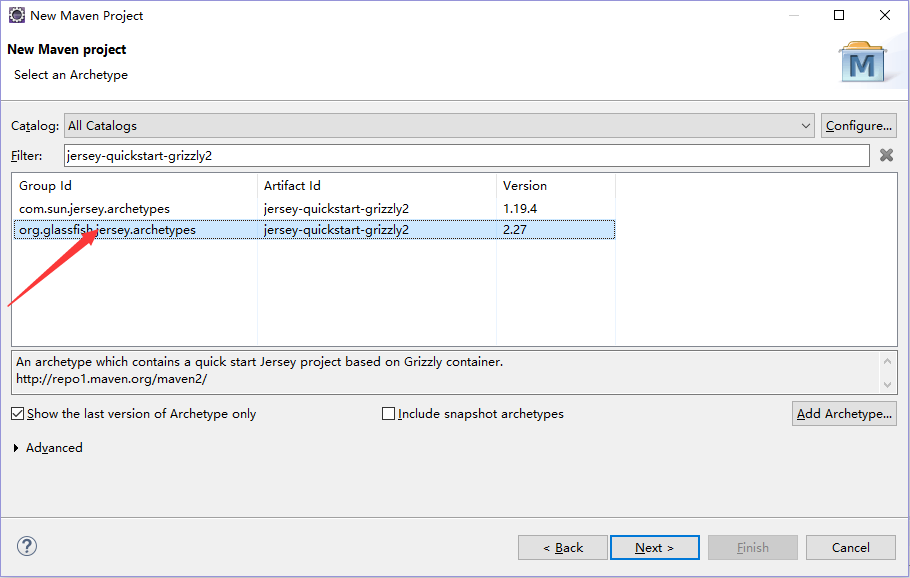
### 第一个REST服务

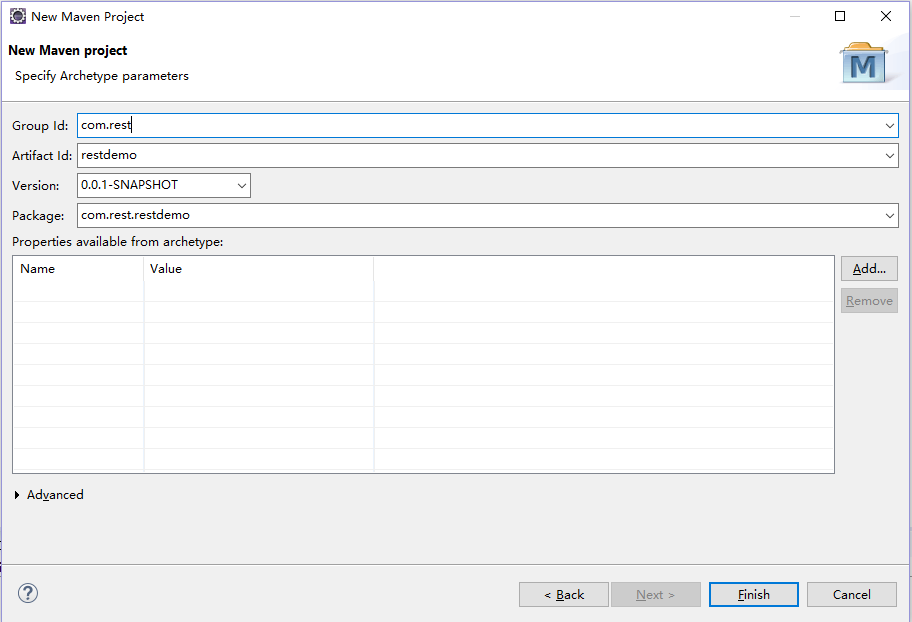
Jersey提供了Maven原型（archetype）来快速创建REST服务项目。

#### 创建项目



输入jersey-quickstart-grizzly2进行过滤





点击finish即可。

#### 运行服务

我们进入项目的根目录，并执行如下命令构建和启动服务。

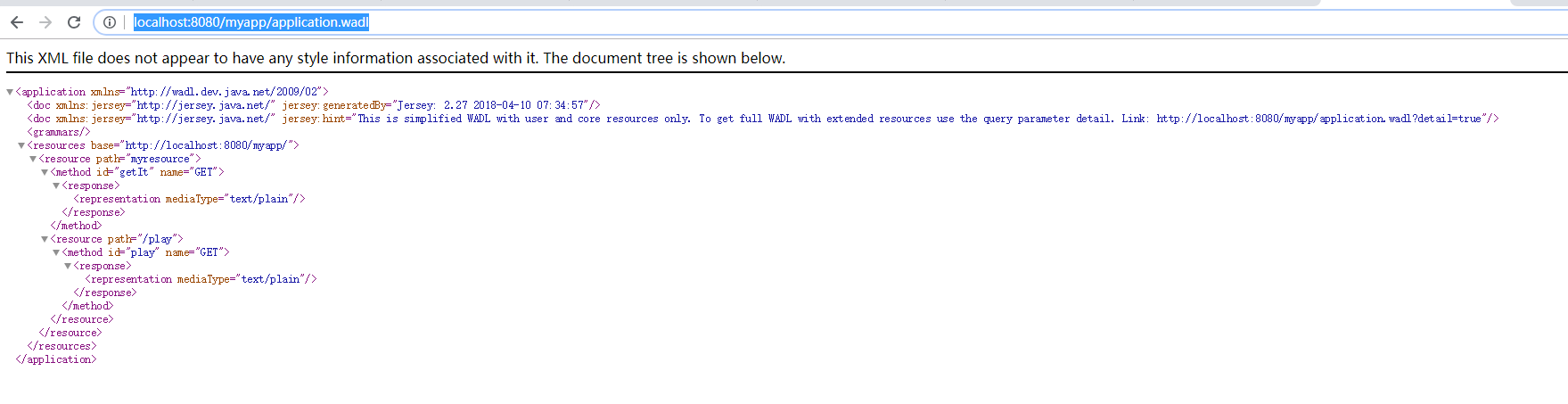
cd my-first-service

mvn package

mvn exec:java

或者直接运行Main.java类就可以启动服务。该命令启动了REST服务，端口是8080，我们可以随时通过回车键停止这个服务。同时该服务还提供了WADL，通过访问application.wadl，可以获取当前REST服务公布的接口。

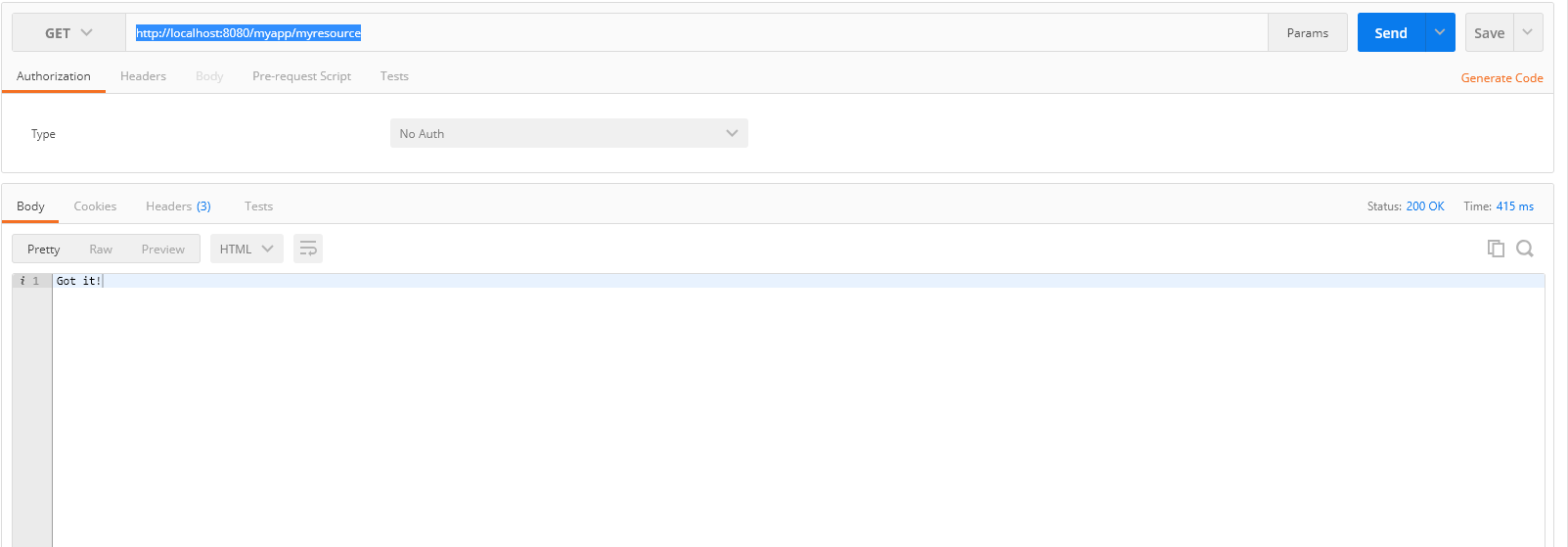
http://localhost:8080/myapp/application.wadl



#### 访问服务

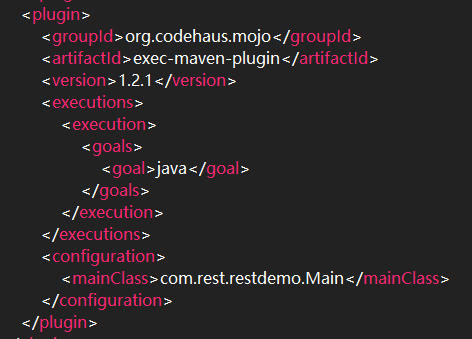
我们使用postman来访问REST服务公布的myresource资源方法getIt，示例如下：

http://localhost:8080/myapp/myresource

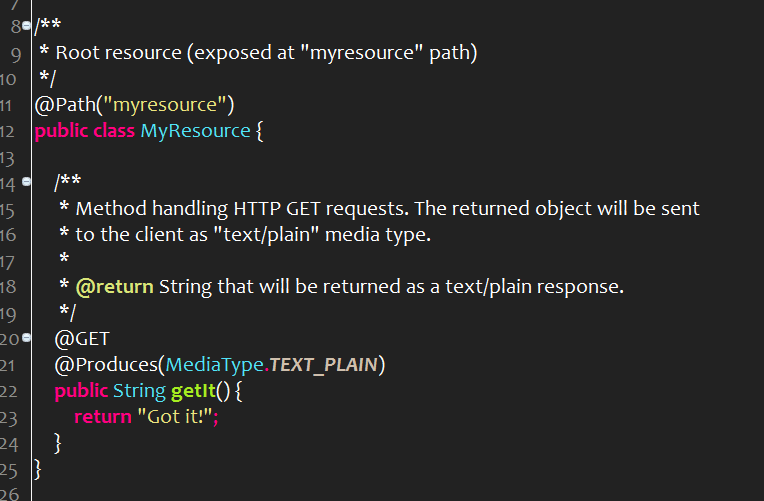


#### 分析项目

首先从启动服务的命令mvn exec:java入手，该命令调用了exec-maven-plugin插件中定义的一个值为java的goal，用以触发mainClass中的main函数。在pom.xml中，exec插件完整定义如下：



源代码中，还包括了资源类MyResource和他的单元测试类MyResourceTest，在资源类MyResource中，@Path中定义了资源路径，@GET中定义了GET方法getIt()，@Produces中定义了响应的类型是普通的字符串，代码示例：



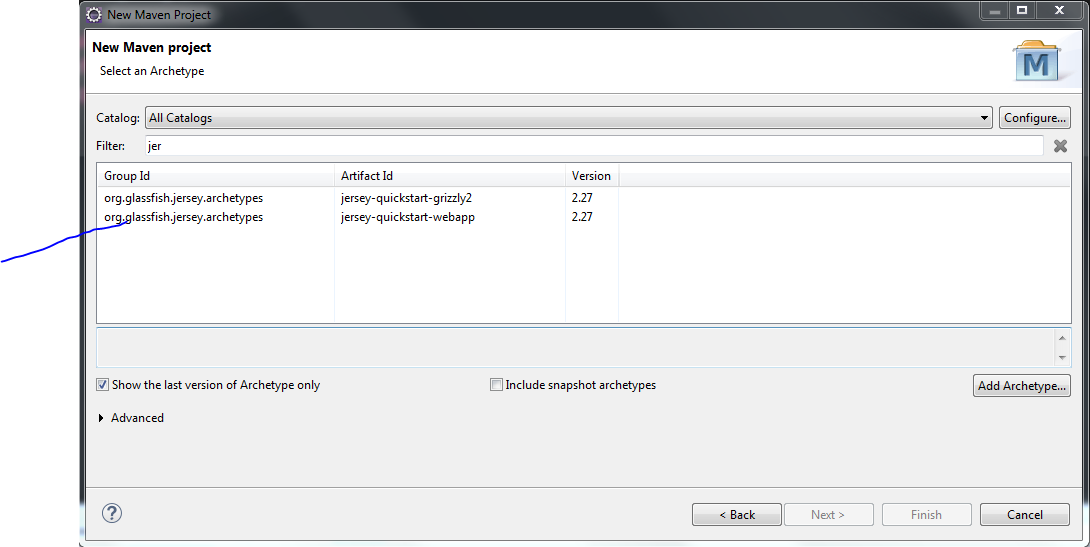
#### 单元测试

我们运行mvn test进行单元测试，或者使用IDE可以直接通过图像界面单击对该方法进行测试。

### 第一个Servlet容器服务（重点）

#### 创建项目

使用jersey-quickstart-webapp原型去创建项目



#### 添加jetty插件

我们在pom.xml文件中添加jetty插件

<plugin>

<groupId>org.eclipse.jetty</groupId>

<artifactId>jetty-maven-plugin</artifactId>

<version>9.3.5.v20151012</version>

</plugin>

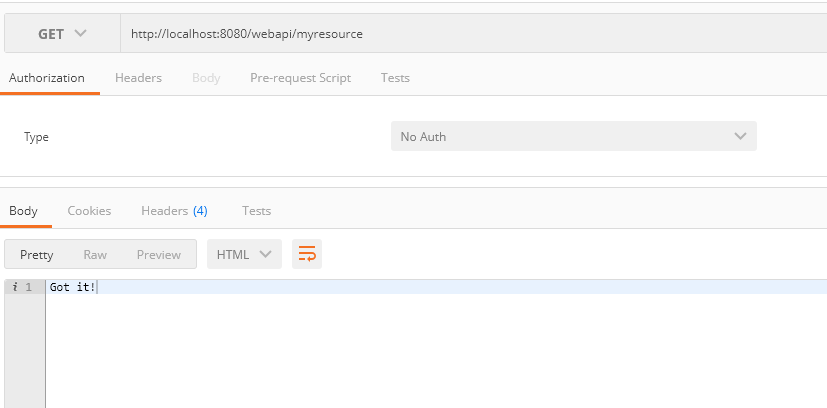
#### 运行服务

然后使用clean jetty:run命令去运行项目

#### 访问项目

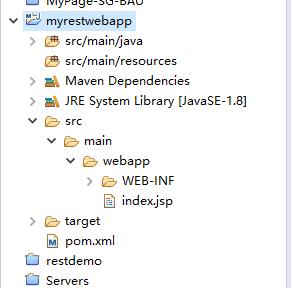
使用postman去请求地址

http://localhost:8080/webapi/myresource



#### 分析项目

这是一个标准的maven web项目，如图：



Web的根目录默认名称为webapp，通过的Servlet版本是2.5，需要使用WEB-INF/web.xml文件来配置REST服务。

说明：部署 Jersey 项目，Servlet 容器版本应该是不低于2.5，如果想支持更高的特性(比如 JAX-RS 2.0 Async Support) ，Servlet容器版本应该是不低于3.0。

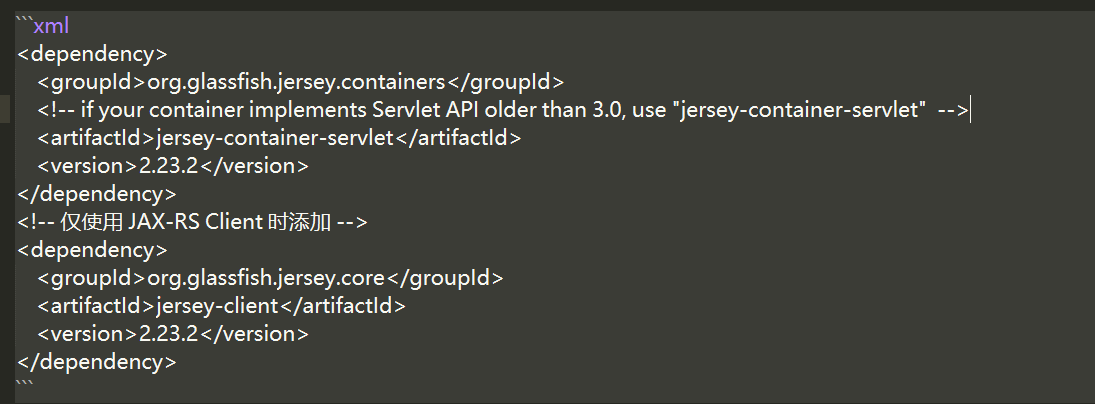
# Jersey Java版本的兼容性

2.6以前的版本，Jersey 由 Java SE 6编译。2.7 版本后发生了变化。现在几乎所有的 Jersey 组件用 Java SE 7目标编译。这意味着，如果要使用最新的 Jersey ，你将至少需要Java SE 7能够编译并运行你的应用程序。只有 'core-common' 和 'core-client' 模块仍然需要 Java SE 6编译。

# 常见jersey示例

## 基于 Servlet 的服务端应用

以下依赖可以应用于没有集成任何 JAX-RS 实现的应用服务器（servlet 容器）。需要在部署的应用里面包含 JAX-RS API 和 Jersey 的实现。



# Java REST服务

在Rest服务中，资源类是接收REST请求并完成响应的核心类，而资源类是由REST服务的“提供者”来调度的，这一概念类似其他框架中自定义的Servlet类，该类会将请求分派给指定的Controller/Action类来处理，这个提供者就是JAX-RS2中定义的Application以及Servlet。

Application类在JAX-RS2标准中定义为javax.ws.rs.core.Application，相当于JAX-RS2服务的入口，如果REST服务没有自定义Application的子类，容器将默认生成一个javax.ws.rs.core.Application类。

# Java领域的其他REST实现

## Spring MVC项目

Spring3.0开始提供了对REST式应用开发的支持，但Spring目前并没有也没有必要推出一个实现JAX-RS标准的模块，MVC模块提供的REST功能并没有采用JAX-RS提出的标准。本质上，Spring MVC控制流程是使用Controller处理Model在某种动词性的业务逻辑操作，而JAX-RS的控制流程是使用资源类Resource处理名称性的资源表述。

# REST API设计（重点）

## 根资源类

根资源类是带有@Path注解的，包含至少一个@Path注解的方法或者方法带有@GET @PUT @POST @DELETE资源方法指示器的POJO，资源方法是带有资源方法指示器注解的方法。

## 统一接口

REST使用HTTP协议的通用方法作为统一接口的标准词汇，REST服务所提供的方法信息都在HTTP方法里。要定义严谨的REST统一接口，就需要真正理解HTTP方法的安全性和幂等性。

安全性是指外系统对这个接口的访问，不会使服务器端资源的状态发生改变，幂等性是指外系统对同一REST接口的多次访问，得到的资源状态是相同的。

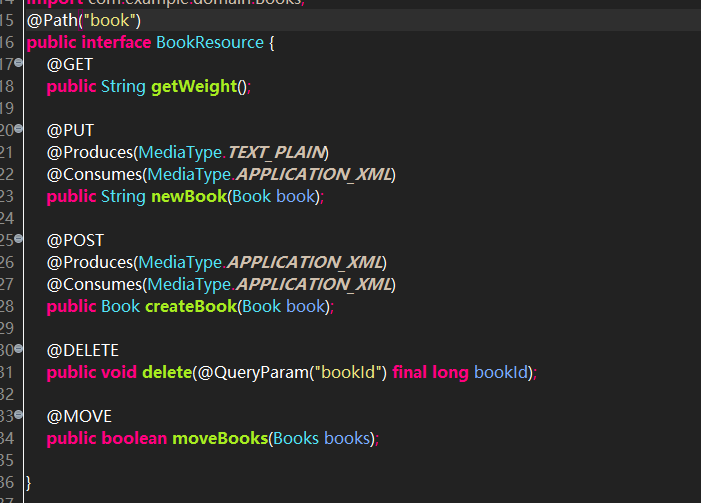
## GET方法

REST使用HTTP的GET方法获取服务提供的资源，GET方法是只读的。HTTP的GET方法用于读取资源，GET方法是幂等的，因为读取同一个资源，总是得到相同的数据，GET方法也是安全的，因为读取资源不会对其状态做改动。定义了@GET注解对资源方法定义，使得该方法用于处理GET请求。

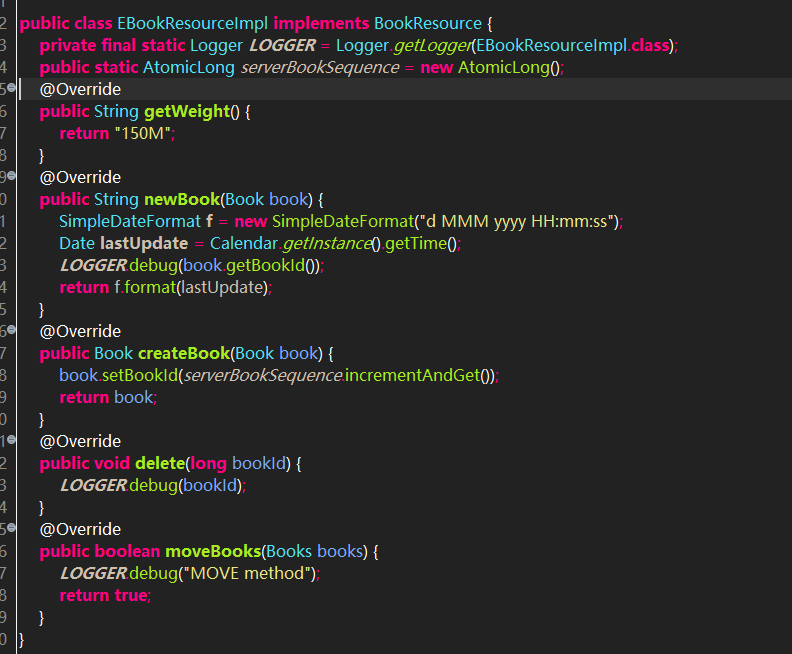
### 抽象层注解资源

JAX-RS2的HTTP方法注解可以定义在接口和POJO中，置于接口中的方法名更具抽象性和通用性。示例代码如下：

定义了BookResource接口



定义接口的实现类：



说明：在接口中抽象的定义了资源的请求方法类型后，其全部实现类都无须再定义，这使得编码更整洁和抽象。

测试：

## PUT方法

PUT方法是一种写操作的HTTP请求，REST使用HTTP的PUT方法更新或者添加资源。

PUT方法是幂等的，即多次插入或者更新同一份数据，在服务器端对资源状态所产生的改变时相同的，PUT方法不是安全的，有写动作的HTTP方法都不是安全的，我们知道由于使用同一份数据向服务器请求更新某一资源，得到的结果应该总是相同的。因此对于更新操作，使用PUT是没有问题的。

创建操作通常应当设计为POST方法的API，唯有一种场景应当使用PUT方法来设计API，即客户端在发起创建请求时，在同一份数据中总可以提供唯一的主键值，服务器不会对其进行修改，这样的创建请求确保了幂等性，不应再使用POST方法，定义了@PUT注解来定义相关资源的方法。

## POST方法

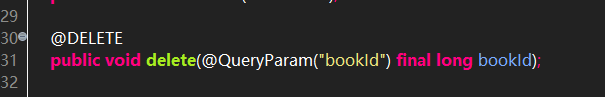
POST方法是一种写操作的HTTP请求，而REST只使用HTTP的POST方法添加资源。

定义为POST的REST接口用于写数据，POST方法的特性是既不幂等也不安全，由于请求会改变服务器端资源的状态，因此它是不安全的。由于每次请求对服务器端资源状态的改变并不是相同的，因此它不是幂等的。

## DELETE方法

DELETE方法是幂等的，即多次删除同一份数据（通常请求中传递的参数是数据的主键值），在服务器端产生的改变是相同的，JAX-RS定义了@DELETE注解来定义相关资源方法。

执行删除的资源方法，其返回值可以定义为void，即该方法没有返回值，之所以在删除资源的场景中可以采用这样的方式定义，是因为删除的前提是对该资源信息已经充分了解，没有必要再将其从服务器上传递回来。代码如下：



## 资源定位（重点）

REST使用URI实现资源定位，从这个角度上讲，对外提供REST式的Web服务的接口就是公布一系列的URI及其参数。在设计REST式的Web服务过程中，资源地址的设计是非常严谨的，如果设计不得体，不仅REST接口的风格无法统一，使系统的扩展性和易用性降低，也很难实现资源准确的被定位。值得注意的是一个URI资源地址唯一对应一个资源，但是一个资源可以拥有多个URI资源地址。

### 资源路径概述

资源地址的路径变量是用来表达逻辑上的层次结构的，资源和子资源的形式是自左向右的、斜杠分隔的名词。

schema 协议名称

### 资源地址和作用域

在路径变量里可以使用标点符号以辅助增强逻辑清晰性。这些辅助符号作为资源地址的查询变量，用来表达算法的输入，实现对方法作用域的约束。

（1）问号？是用来分隔资源地址和查询字符串的，与符号&是用来分隔查询条件的参数的。示例代码：

GET /books?start=0&size=10

（2）逗号，是用来分隔有次序的作用域信息，需要注意的是逗号分隔的逻辑上的顺序信息，这种顺序可以是约定的，也可以是系统约定的。示例：

GET /books/01,2002-12,2004

连字符-是用来做逻辑上的辅助分隔的。

（3）分号；是用来分隔无次序的作用域信息，通常这些信息是逻辑上并列存在的，比如并列的查询条件。示例：

GET /books/restful;program=java;type=web

#### 资源地址设计

##### 添加/创建

POST /books

PUT /books/{id}

##### 删除

DELETE /books/{id}

##### 修改/更新

PUT /books/{id}

##### 查询全部

GET /books

##### 主键查询

GET /books/{id}

GET /books?id=123456

##### 分页作用域查询

GET /books/?start=0&size=10

GET /books/01,2002-12,2014

GET /books/restful;program=java;type=web

GET /books?limit=100&sort=bookname

### @QueryParam注解

查询条件决定了方法的作用域，查询参数组成了查询条件。定义了@QueryParam注解来定义查询参数。

代码示例：

public Yijing getByQuery(@QueryParam("id") final int seqId){

return ParamCache.find(""+seqId);

}

在这段代码中，使用@QueryParam定义了id参数。

### @PathParam注解

定义了@PathParam注解来定义路径参数，每个参数对应一个子资源。示例列表：

基本路径参数

/path-resource/Eric

结合查询参数

/path-resource/Eric?hometown=B

带有标点符号的资源路径

/path-resource/199-1999

子资源变长的资源路径

/path-resource/Asia/China/northest/liaoning

### @Path注解

@Path注解定义了资源路径，@Path接收了一个value参数来解析资源路径地址，该参数除了可以使用静态参数外，也可以使用动态变量的方式，其格式为：{参数名称：正则表达式}，示例代码如下：

@GET

@Path("{from:\\d+}-{to:\\d+}")

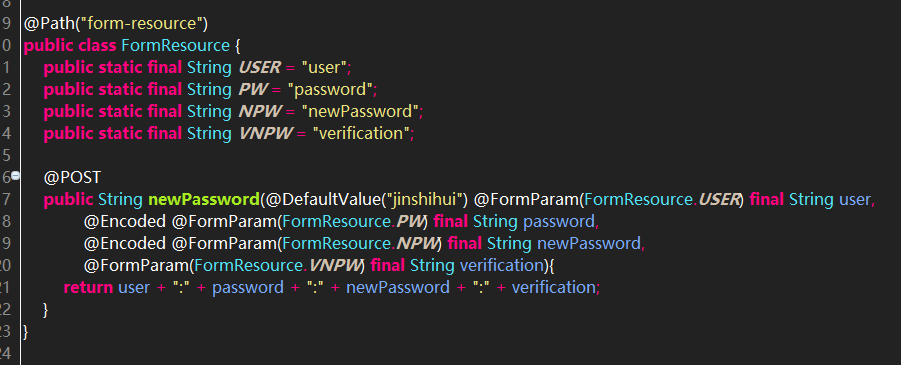
public String getByCondition(@PathParam("from") final Integer from,@PathParam("to") final Integer to){

}

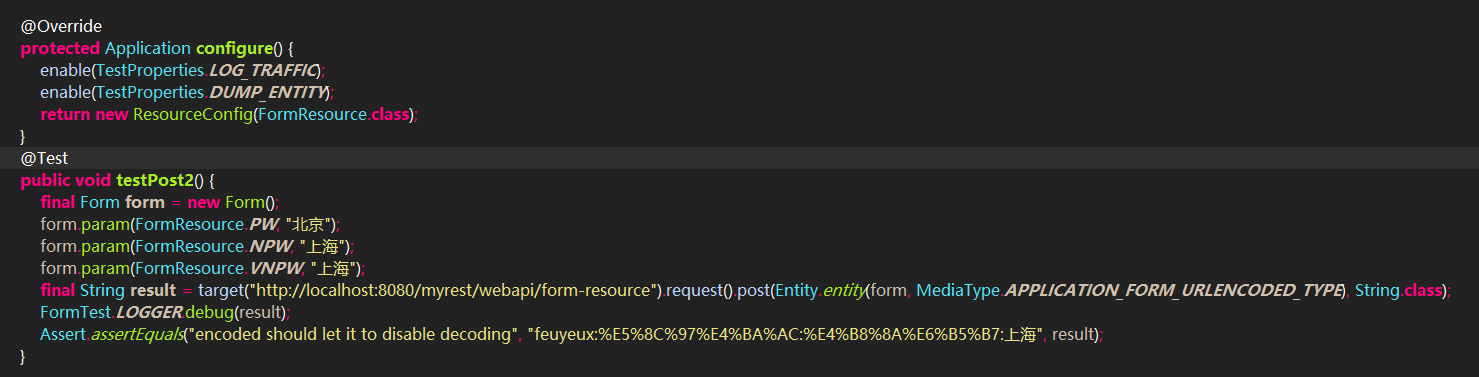
在这段代码中，使用@PathParam注解定义的两个参数from和to用以定义查询区间，正则表达式部分是\d+，表示数字，两个参数中间的连接符（-）是路径的格式信息。

### @FormParam注解

定义了@FormParam注解来定义表单参数，相应的REST方法用以处理请求实体媒体类型为Content-Type：application/x-www-form-urlencoded的请求，示例代码如下：



测试代码：



说明：@Enabled注解用以标识禁用自动解码，REST方法得到的参数值就不会解码，如果将其直接返回，那么客户端得到的值就会是处于编码状态的字符串。

## 传输格式

通常REST接口会以XML和JSON作为主要的传输格式，还支持其他的数据格式。

### 基本类型

包括4种整型（byte、short、int、long）、2种浮点类型（float、double）、Unicode编码的字符（char）和布尔类型（boolean），jersey支持全部的基本类型，还支持与之相关的引用类型。

代码示例：

@POST

@Path("b")

public String postBytes(final byte[] bs) {

for (final byte b : bs) {

LOGGER.debug(b);

}

return "byte[]:" + new String(bs);

}

资源方法postBytes()的输入参数是byte[]类型，输出参数是String类型。

### 文件类型

Jersey支持传输File类型的数据，以方便客户端直接传递File类实例给服务器端。文件类型的请求，默认使用的媒体类型是Content-Type：text/html。

### InputStream类型

Jersey支持Java的两大读写模式，即字节流和字符流。

### XML类型

Xml类型是使用最广泛的数据类型，Jersey对xml类型的数据处理，支持Java领域的两大标准，即JAXP（Java API for XML Processing）和JAXB（Java Architecture for XML Binding）。

JAXB只需要在POJO中定义相关的注解，使其和XML的schema对应，无须对XML进行程序式解析。JAXB通过序列化和反序列化实现了XML数据和POJO对象的自动转换过程。在运行时，JAXB通过编组(marshall)过程将POJO序列化成XML格式的数据，通过解编(unmarshall)过程将XML格式的数据反序列化为Java对象。REST风格以及敏捷思想的宗旨就是简单—开发过程简单化、执行逻辑简单化，JAXB更适合做REST开发。Jersey支持使用JAXBElement作为REST方法参数的形式，也支持直接使用POJO作为REST方法参数的形式，后一种更为常用。

### JSON类型（重点）

Json类型已经成为Ajax技术中数据传输的实际标准。

#### 使用Jackson处理JSON

Jackson是一种流行的json支持技术，提供了3种JSON解析方式。

（1）第一种是基于流式API的增量式解析/生成JSON的方式，读写JSON内容的过程时通过离散事件触发的。

（2）基于树型结构的内存模型，提供了一种不变式的JsonNode内存树模型，类似DOM树。

（3）第三种是基于数据绑定的方式，org.codehaus.jackson.map.ObjectMapper解析，使用JAXB的注解。

##### 代码示例

###### （1）定义依赖

<dependency>

<groupId>org.glassfish.jersey.media</groupId>

<artifactId>jersey-media-json-jackson</artifactId>

<version>${jersey.version}</version>

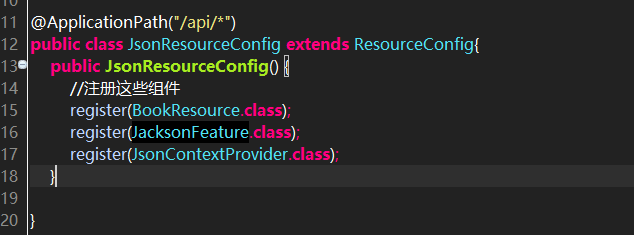
</dependency>

说明：这个依赖是jackson用于处理json类型的数据。

###### （2）定义Application

使用jackson的应用，需要在其Application中注册JacksonFeature，同时如果有必要，根据不同的实体类做详细的解析，可以注册ContextResolver的实现类。

编写JsonResourceConfig类，继承ResourceConfig类

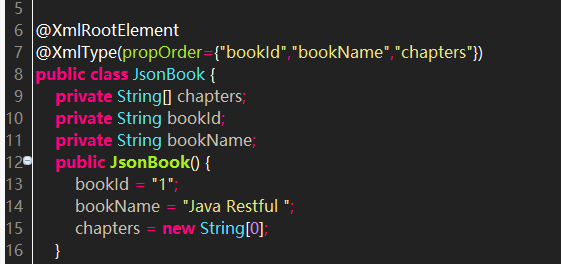


在这段代码中，注册了ContextResolver的实现类JsonContextProvider，用于提供JSON数据的上下文。

###### （3）定义POJO

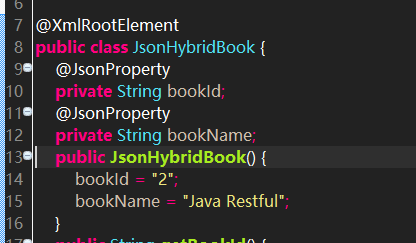
定义3种不同方式的POJO，用于表示Jackson处理JSON的多种方式，分别是JsonBook，JsonHybridBook，JsonNoJaxbBook。

第一种方式仅用JAXB注解的普通POJO。



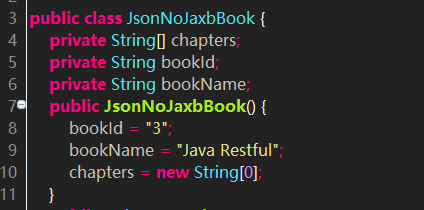
省略了get set方法。

第二种方式是将JAXB注解和Jackson提供的注解混合使用的POJO



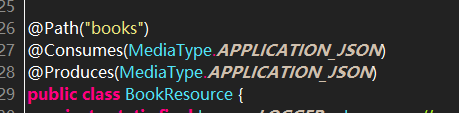
省略了get set方法

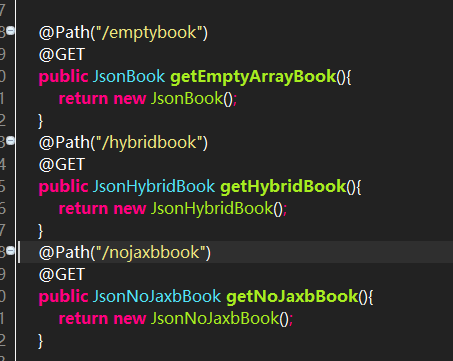
第三种是不使用任何注解的POJO



###### （4）定义资源类

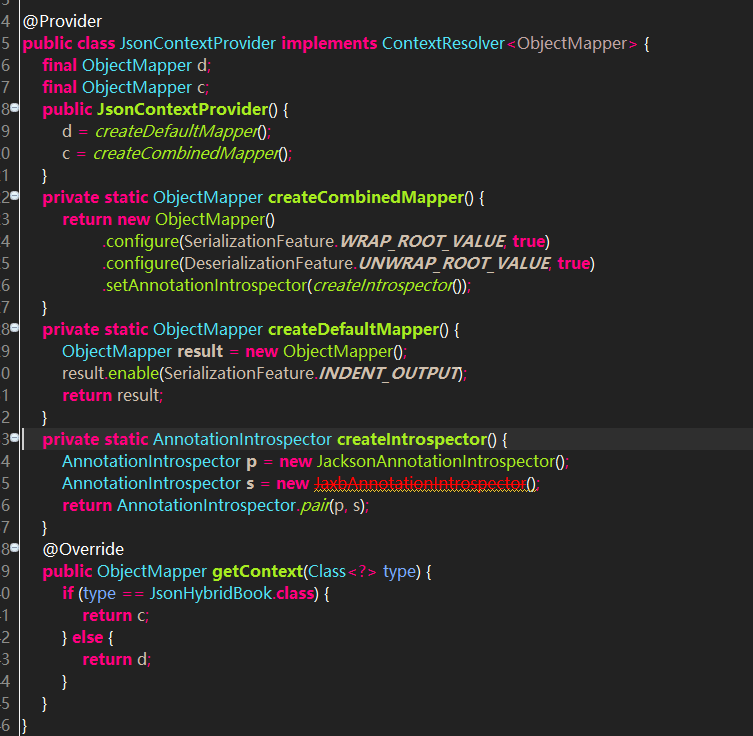
用于演示Jackson对上述3种不同POJO的支持





###### （5）上下文解析实现类

JsonContextProvider是ContextResolver接口（上下文解析器）的实现类，其作用是根据上下文提供的POJO类型，分别提供两种解析方式，第一种是默认的方式，第二种是混合使用Jackson和Jaxb。



通过这样的实现，当流程获取JSON上下文时，可以使用Jackson依赖包完成对相关POJO的处理。

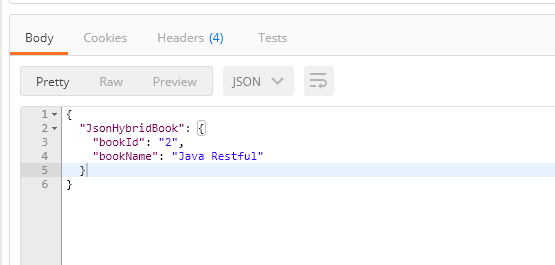
###### （6）集成测试

启动项目，使用postman去测试。

输入http://localhost:8080/myrest/api/books/emptybook



输入http://localhost:8080/myrest/api/books/hybridbook



输入http://localhost:8080/myrest/api/books/nojaxbbook



## ResourceConfig类（重点）

这个类是为了配置web应用程序的资源配置。

### 方法

#### public ResourceConfig register(final Class<?> componentClass)

在这个可配置上下文的范围内配置JAX-RS组件类

## 处理响应

REST的响应处理结果应包括响应头中HTTP状态码，响应实体中媒体参数类型和返回值类型，以及异常情况处理。JAX-RS2支持4种返回值类型的响应，分别是无返回值、返回Response类实例、返回GenericEntity类实例和返回自定义类实例。

### 返回类型

#### void

在返回值类型是void的响应中，其响应实体为空，HTTP状态码为204，看一下这种类型的资源方法。代码如下：

@DELETE

@Path("{s}")

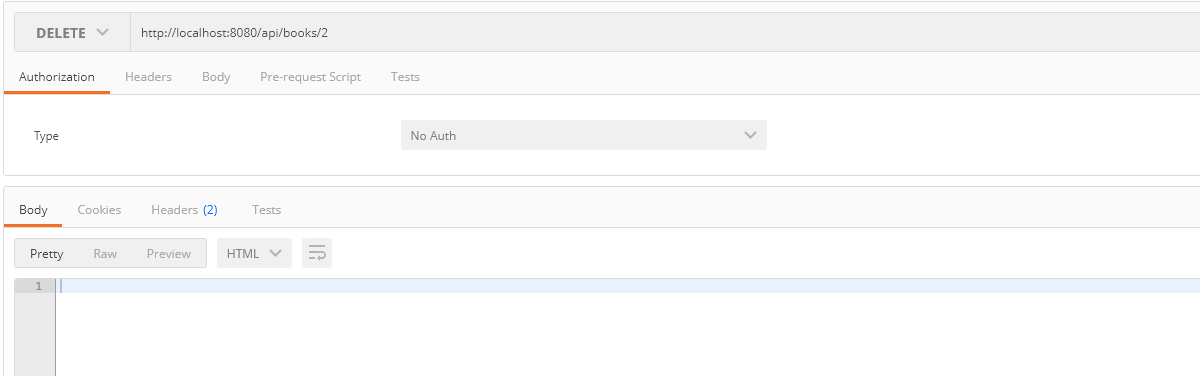
public void delete(@PathParam("s") final String s) {

LOGGER.debug(s);

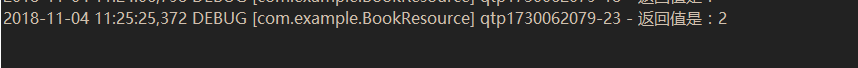
}

delete操作无须返回更多的关于资源表述的信息，因此该方法没有返回值，即返回值类型为void。

postman测试如下：

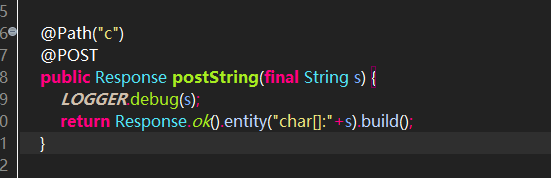


后台结果：

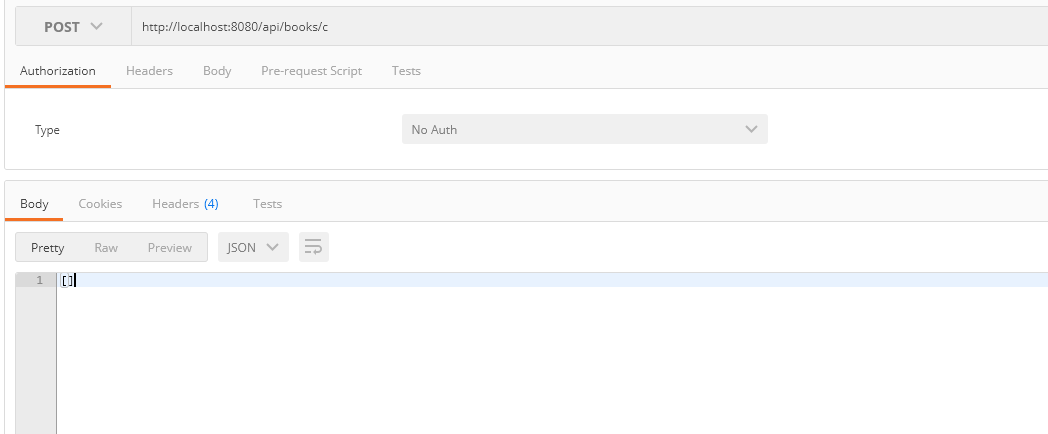


#### Response

在返回值类型为Response的响应中，响应实体为Response类的entity()方法定义的实体类实体。如果该内容为空，则HTTP状态码是204，否则HTTP状态码是200OK，代码如下：

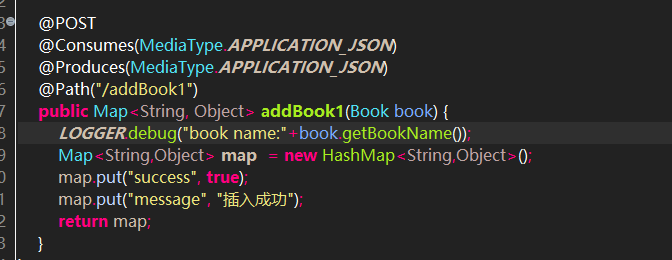


postman测试如下：

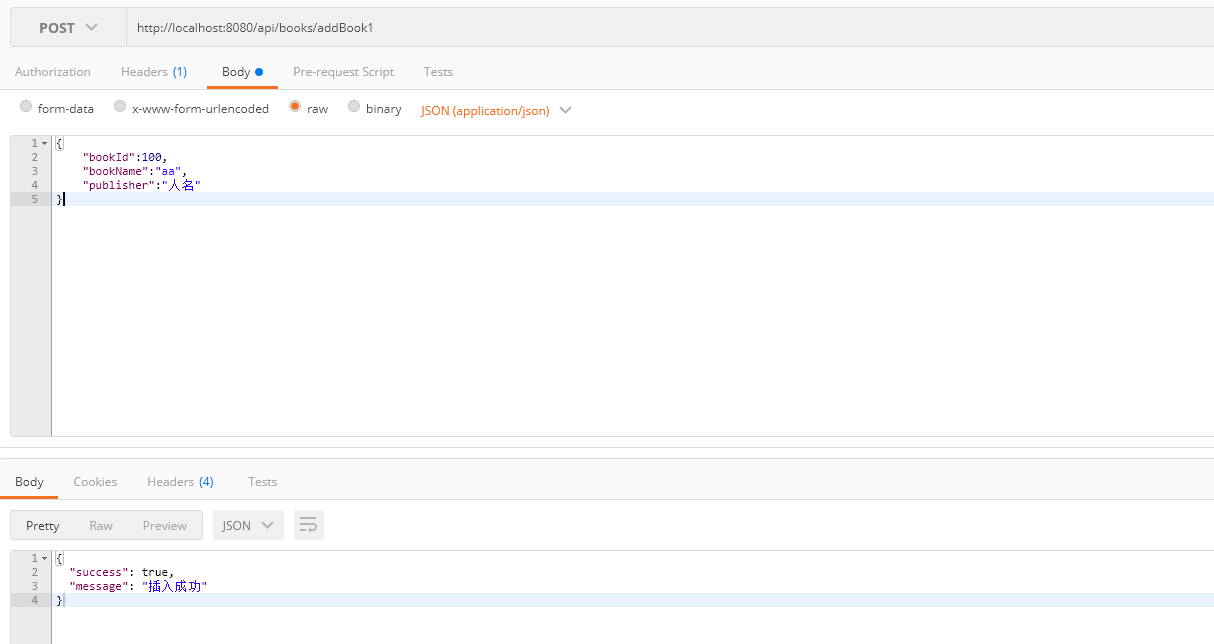


#### 自定义类型

JDK中的类都可以作为返回值类型，更常用的是返回自定义的POJO类型，代码如下：



postman测试如图：



### 处理异常

实现REST的资源方法时应该具有良好的异常处理能力，这包括异常的定义和错误状态码的正确返回。

#### 处理状态码

常用的状态码这里就不写了，网上查询就可以。应该在处理异常的同时，为REST请求的客户端提供对应的错误码。

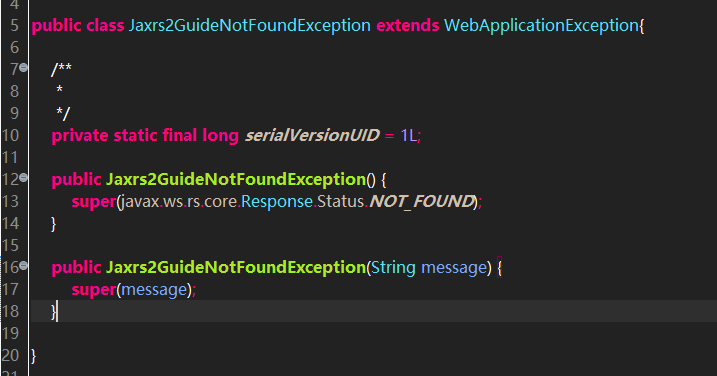
JAX-RS2规定的REST式的Web服务的基本异常类型为运行时异常WebApplicationException类，该类包含3个主要的子类分别对应如下内容：

（1）HTTP状态码为3xx的重定向类RedirectionException

（2）HTTP状态码为4xx的请求错误类ClientErrorException

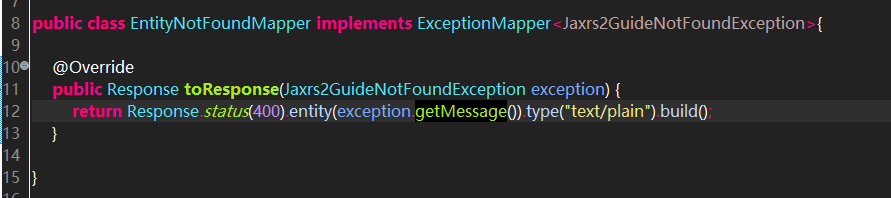
（3）HTTP状态码为5xx的服务器错误类ServerErrorException

除了Jersey提供的标准异常类型，我们也可以根据业务需要自定义相关的业务异常类，代码如下：



#### ExceptionMapper

Jersey框架为我们提供了更为通用的异常处理方式，通过实现ExceptionMapper接口并使用@Provider注解将其定义为一个Provider，可以实现通用的异常的面向切面处理，而非针对某一个资源方法的异常处理。代码如下：



说明：定义EntityNotFoundMapper类并实现ExceptionMapper接口，并提供了泛型类型，当响应中发生了Jaxrs2GuideNotFoundException类型的异常时，响应流程就会被拦截并补充HTTP状态码和异常信息，已文本作为媒体格式返回给客户端。

## 内容协商

一个资源可以有不同格式的表述，表述（即响应实体）的内容是人类可识别的信息，服务器很难使用一种表述来适应所有用户。conneg(HTTP Content Negotiation 内容协商)是指在服务器提供的多种表述中，为特定的请求选择最好的一种表述的处理过程。服务器和客户端/浏览器之间往复通信来协商用于交换数据的内容格式等信息，达成一致即为最好。

客户端/浏览器通过使用HTTP Accept、Accept-Charset、Accept-Language和Accept-Encoding头来定义接收头的信息，将其所期待的格式或MIME类型告诉服务器，服务器根据协商算法，返回客户端/浏览器可接受的数据信息，内容协商不只是数据格式协商，还包括语言、编码、字符集等信息。Accept用于数据类型协商，Accept-Language用于语言协商，Accept-Charset用于字符集协商，Accept-Encoding用于压缩算法协商。

JAX-RS2对内容协商的支持，是通过@Produces实现的，其他协商没有从架构上提供支持，可以通过编码从请求头中获取信息并处理。

### @Produces注解

注解@Produces用于定义方法的响应实体的数据类型，可以定义一个或多个，同时可以为每种类型定义质量因素，质量因素是取值范围从0到1的小数值，如果不定义质量因素，那么该类型的质量因素默认为1。

### @Consumes注解

注解@Consumes用于定义方法的请求实体的数据类型，和@Produces不同的是，@Consumes的数据类型的定义只用于JAX-RS2匹配请求处理的方法，不做内容协商使用。如果匹配不到，服务器会返回HTTP状态码415。

# REST请求处理

## Jersey的AOP机制

AOP对增强REST服务的功能性、安全性和可扩展性等方面都具有深远意义，完整的REST风格的框架都从容器级别支持AOP功能。Jersey自身支持AOP，可以不依赖于Spring等支持AOP的框架。

AOP的典型应用场景有权限管理、日志记录、统计记录、事务以及异常处理等，其实现原理是代理被调用的方法，在其被执行的方法前后，增加额外业务功能。AOP的实现机制是通过注解或者XML配置，依据这些配置，动态生成字节码，使被调用代码对应的字节码被环绕注入新的功能，或者使用Java的动态代理机制，完成对被调用方法的增强。

Jersey提供的REST过滤器和拦截器为开发者提供了很贴心的切面扩展点，开发者无须像在Spring中为了针对某个类的方法进行AOP扩展而写配置文件，在Jersey中，只要实现相应扩展点的接口，即可实现REST请求流程中特定事件点的拦截、扩展，其他工作由底层的HK2帮我们做，典型的应用包括请求和响应的过滤和读写过滤。

## Providers详解

Providers是JAX-RS2定义的一种辅助接口，其实现类用于辅助REST框架完成过滤和读写拦截等功能。使用@Provider注解来标注这些实现类，可以被JAX-RS2的运行时自动探测、加载，Provider实例可以通过@Context注解被依赖注入到其他实例中。Providers接口定义了4个方法，分别用来获取MessageBodyReader、MessageBodyWriter、ExceptionMapper、ContextResolver实例。

## 实体Providers

Jersey之所以支持那么多种表述的类型，即响应实体的传输格式，是因为其底层实体Providers具备的对不同格式的处理能力，Jersey内部提供了非常丰富的MessageBodyReader接口和MessageBodyWriter接口实现类，用于处理不同格式的表述，比如字节数组、XML、文件和流等。

### MessageBodyReader接口

消息体读处理器接口MessageBodyReader<T>用于将传输流转换为Java类型的对象，MessageBodyReader接口定义了一个泛型，接口的实现类为这个泛型定义一个具体类型，该类型即是该实现类所支持的转换类型。实现类被业务系统启用有两种方式，一是使用注解@Provider定义实现类，业务系统在启动时自动探测并加载。另一种方式是通过编码注册到Application类或其子类中，业务系统在启动时，加载Application类或其子类时一并加载。

#### 方法

##### boolean isReadable(Class<?> type, Type genericType,Annotation[] annotations, MediaType mediaType)

用来判断实现类是否支持将当前请求的数据类型反序列化，以读取字节数组实体实现类。

# REST测试

自动化测试是软件质量保证的必要手段，项目源代码的代码覆盖率即指测试代码中对源代码的公有方法的覆盖情况。

## Jersey测试框架

# 注解

## @Path

该注解可以标记在类名之上，也可以标记在方法名上，该注解接收一个value参数，表示定义资源的地址，另外资源地址相同，但是HTTP方法不同的两个方法是完全不同的REST接口，HTTP方法和资源地址相结合在一起才可以完成对一个资源的定位。

属性：value

## @GET

该注解标记的方法表明用于处理GET请求，GET方法是幂等且安全的。这个注解用在方法上。

## @POST

该注解标记的方法表明处理POST请求，POST方法表明是一种创建操作的方法，POST方法是一种写操作的HTTP请求，RPC中的写操作均是使用post方法，而在REST中我们只是使用POST方法来添加资源。这个注解用在方法上。

## @PUT

该注解标记的方法表明处理PUT请求，用于更新操作。这个注解用在方法上。

## @DELETE

该注解标记的方法表明这个方法执行的是一个删除资源的操作，其返回值可以是void，即没有返回值。这个注解用在方法上。

## @Consumes

在服务器端，@Consumes(MediaType.APPLICATION\_XML)定义了服务器端要消费的媒体类型，也就是说消费客户端请求实体的媒体类型，可以是xml、json等类型。这个注解可以定义在类上，方法上。

## @Produces

定义了服务器端生产的媒体类型，即服务器端产生的响应实体的媒体类型，同样也可以是xml、json等媒体类型。这个注解可以定义在类上，方法上。

## @QueryParam

该注解表明是一个查询条件，查询条件决定了方法的作用域，查询参数组成了查询条件。Jersey中使用@QueryParam注解来定义查询参数。另外注解@QueryParam还可以和注解@DefaultVaule一起使用，注解@DefaultValue的作用则是预置一个默认值，当请求中不包括该参数时使用这个默认值。这个注解用在参数上、方法上、字段上。

## @PathParam

该注解定义路径的参数信息，每一个参数都对应一个子资源，这个注解用在参数上、方法上、字段上。

## @FormParam

该注解表示用来定义表单的参数。这个注解用在参数上、方法上、字段上。

## @Context

该注解用来解析上下文参数。这个注解用在参数上、方法上、字段上。

## @Provider

标记扩展接口的实现，该接口应该在提供者扫描阶段由JAX-RS运行时发现。这个注解用在类上。

## @ApplicationPath

这个注解用于声明Jersey的全局配置类，也即整个Jersey框架的程序入口。这个注解用在类上，该类需要满足下面条件：

（1）@ApplicationPath注解该类，并且指定该类对应的路径。

（2）继承 org.glassfish.jersey.server.ResourceConfig 类

（3）在该类中配置Jersey的配置，例如声明资源包路径，配置拦截器等。



说明：这个注解只有value这个属性，用于配置路径地址。

# JAXB解析XML

开发思路：

## 1、新建Hotel类

定义id、name、RoomTypeVOs变量，生成get set方法



说明：

@XmlRootElement：将类或枚举类型映射到 XML 元素。JAXB中的注解，用来根据java类生成xml内容。用于生成xml文件的根元素。

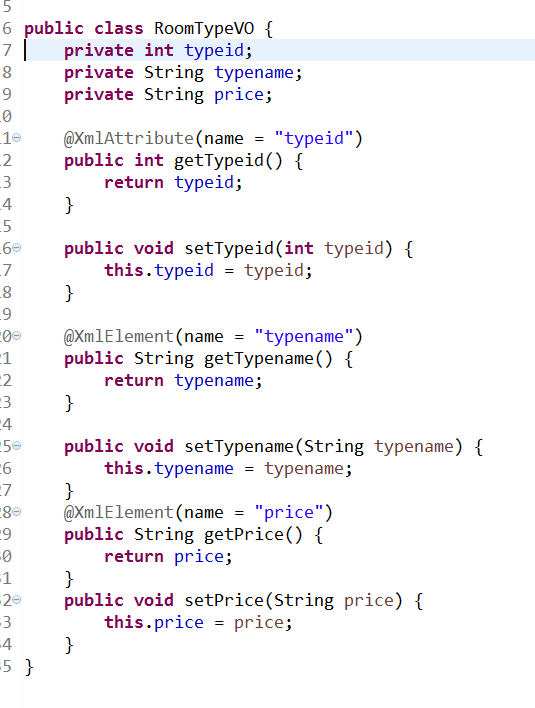
@XmlElement：用于生成xml文件中的元素。

@XmlAttribute：用于生成xml文件中元素的属性

@XmlElementWrapper：表示生成一个包装器元素，该注解支持两种形式的序列化。仅允许出现在集合属性上。使用@XmlElementWrapper注解后，将会在原xml结点上再包装一层xml。

## 2、新建RoomTypeVO

定义typeid、typename、price变量，生成get set方法



## 3、新建JaxbUtil类

新建一个JAXBContext对象。

