RESTful教程

# REST环境安装

1. **jdk**
2. **安装eclipse IDE**
3. **设置Jersey框架库**
4. **安装Apache Tomcat**

# RESTful Web Services示例程序

# 创建Java项目

# 添加必需的库

# 第二步，让我们在项目中添加Jersey框架和它的依赖（库）。

# 创建源文件

package com.yiibai;

import java.util.List;

import javax.ws.rs.GET;

import javax.ws.rs.Path;

import javax.ws.rs.Produces;

import javax.ws.rs.core.MediaType;

@Path("/UserService")

public class UserService {

UserDao userDao = new UserDao();

@GET

@Path("/users")

@Produces(MediaType.APPLICATION\_XML)

public List<User> getUsers(){

return userDao.getAllUsers();

}

}

有以下两个要点需要注意下面的主程序，UserService.java:

1. 第一步是指定用于使用@Path注释到UserService的Web服务的路径。
2. 第二个步骤是，以指定用于使用@Path注释到UserService的方法的特定web服务方法的路径。

# 创建web.xml配置文件

需要创建一个Web XML配置文件，该文件是一个XML文件，用于指定Jersey框架的servlet应用。

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

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

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

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

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

id="WebApp\_ID" version="3.0">

<display-name>User Management</display-name>

<servlet>

<servlet-name>Jersey RESTful Application</servlet-name>

<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>

<init-param>

<param-name>jersey.config.server.provider.packages</param-name>

<param-value>com.yiibai</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>Jersey RESTful Application</servlet-name>

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

</servlet-mapping>

</web-app>

# 部署程序

一旦你使用创建源和Web配置文件完成，这一步是准备好编译和运行程序。要做到这一点，使用Eclipse，导出您的应用程序为war文件，并部署在Tomcat中。 要使用eclipse创建WAR文件，按照选项 File -> export -> Web > War File 最后选择项目UserManagement和目标文件夹。 要在Tomcat部署WAR文件，将用户Management.war在Tomcat的安装目录 > webapps 目录并启动Tomcat。

# 运行程序

我们使用Postman，一个Chrome扩展来测试Web服务。

提出一个请求到UserManagement，让所有的用户列表。http://localhost:8080/UserManagement/rest/UserService/users POSTMAN使用GET请求，看看下面的结果。

# RESTful资源

# 什么是资源

REST架构对待每一个内容都作为一种资源。这些资源可以是文本文件，HTML网页，图片，视频或动态业务数据。 REST服务器只是提供资源，REST客户端可访问和修改的资源。这里每个资源由URI标识/全局标识。 REST采用各种交涉代表的资源如文本，JSON，XML。 XML和JSON是资源的最流行的表示。

# 资源表示

在REST资源是在面向对象编程或数据库类似于实体类似的对象。一旦资源被标识则其表示是用一个标准的格式来决定，以便服务器可以发送资源上文所述的格式和客户端可以理解的格式。

# 好的资源表示

REST并没有对资源表示格式有任何限制。客户端可以请求JSON表示，其中作为另一种客户端可能会要求同一资源的XML表示到服务器等。它是REST服务器的负责传递客户端的资源到客户端可以理解的格式。

以下是在设计资源的表示形式在一个RESTful Web服务要考虑的重要因素。

* **易懂:**服务器和客户端应能够理解和使用的资源的表示格式。
* **完整:**格式应当能够完全代表一个资源。例如，一个资源可以包含其他资源。格式应该能够代表简单以及资源的复杂的结构。
* **可链接:** 资源可以有一个联动到另一个资源，一个格式应当能够处理这种情况。

然而，目前大多数的Web服务使用XML或JSON格式代表的资源。有很多可用的理解，分析，并修改XML和JSON数据库和工具。

# RESTful Web服务消息

# HTTP 请求



HTTP请求有五个主要部分：

* Verb- 表示HTTP方法，如GET，POST，DELETE，PUT等
* URI- 统一资源标识符(URI)来标识服务器上的资源
* HTTP Version- 表示HTTP版本，例如HTTP1.1版。
* Request Header- 包含元数据的HTTP请求消息作为键 - 值对。 例如，客户端(或浏览器)型，由客户端支持的格式，邮件正文的格式，缓存设置等。
* Request Body- 消息内容或资源的表示。

# HTTP 响应

# 1509401946-1.jpg

HTTP响应有四个主要部分：

* **Status/Response Code** - 表示对所请求的资源服务器状态。例如404表示未找到资源，200表示响应正常。
* **HTTP Version**- 表示HTTP版本，例如HTTP1.1版。
* **Response Header**- 包含元数据的HTTP响应消息作为键 - 值对。 例如，内容长度，内容类型，响应时间，服务器类型等
* **Response Body**- 响应消息的内容或资源表示。

# RESTful Web服务地址

REST架构中的每个资源都由其URI，统一资源标识符。 URI是以下格式：

<protocol>://<service-name>/<ResourceType>/<ResourceID>

一个URI的目的是要找到承载Web服务的服务器上的资源。请求的另一个重要属性是动词，标识要在资源上执行的操作。例如，在REST Web服 第一应用教程， URI 就是http://localhost:8080/UserManagement/rest/UserService/users 和动词是GET。

# 构建一个标准的URI

以下是要考虑在设计一个URI要点：

* **使用复数名词** - 使用复数名词来定义的资源。例如，我们已经使用的用户识别用户的资源。
* **避免使用空格** - 利用下划线（\_）或连字符（ - ），使用一个长的资源的名称，例如，使用authorized\_users代替authorized%20users。
* **使用小写字母** - 虽然URI是区分大小写，这是很好的做法，以保持网址只有小写字母。
* **保持向后兼容** - 由于Web服务是一种公共服务，URI一旦做出公开应始终可用。在某些情况下URI更新，使用HTTP状态码，300表示旧的URI重定向到新的URI。
* **使用HTTP动词** - 始终使用HTTP动词像GET，PUT和DELETE做业务上的资源。这是不好用操作名字URI。

**例子**：

<http://localhost:8080/UserManagement/rest/UserService/getUser/1>

<http://localhost:8080/UserManagement/rest/UserService/users/1>

# RESTful Web Services方法

正如我们讨论至今认为RESTful web服务使得重用HTTP动词，以确定要执行所指定的资源(多个)的操作。 下表使用HTTP动词常用状态的例子。

|  |  |
| --- | --- |
| **S.N.** | **HTTP方法，URI和操作** |
| 1 | **GET** http://localhost:8080/UserManagement/rest/UserService/users 获取用户列表 (只读) |
| 2 | **GET** http://localhost:8080/UserManagement/rest/UserService/users/1 获取ID为1的用户 (只读) |
| 3 | **PUT** http://localhost:8080/UserManagement/rest/UserService/users/2 使用ID为2插入用户 (等幂) |
| 4 | **POST** http://localhost:8080/UserManagement/rest/UserService/users/2 更新ID为2的用户 (N/A) |
| 5 | **DELETE** http://localhost:8080/UserManagement/rest/UserService/users/1 删除ID为2用户 (等幂) |
| 6 | **OPTIONS** http://localhost:8080/UserManagement/rest/UserService/users 列出Web服务支持的操作 (只读) |
| 7 | **HEAD** http://localhost:8080/UserManagement/rest/UserService/users 仅返回HTTP头，没有主体。 (只读) |

下面是要考虑的重要问题：

* **GET**仅是读操作并且是安全的。
* **PUT** 和 **DELETE** 操作幂等意味着他们的结果总是相同的，无论多少次，这些操作可被调用。
* **PUT** 和 **POST** 动作几乎相同，区别仅位于在结果其中PUT操作是等幂，POST操作可能导致不同的结果。

# RESTful Web Services缓存

缓存是指存储服务器响应客户端本身，使客户端无须做出了对同样的资源服务器的请求连连。服务器的响应有关于缓存是如何做到使客户端缓存的响应一段时间或永远捕捉服务器响应信息。

|  |  |
| --- | --- |
| **S.N.** | **消息头和说明** |
| 1 | **Date** 日期和创建时的资源的时间。 |
| 2 | **Last Modified** 资源的日期和时间，它的最后修订。 |
| 3 | **Cache-Control** 主要头控制缓存。 |
| 4 | **Expires** 到期日和缓存时间 |
| 5 | **Age** 在几秒钟时间，从资源服务器获取时。 |

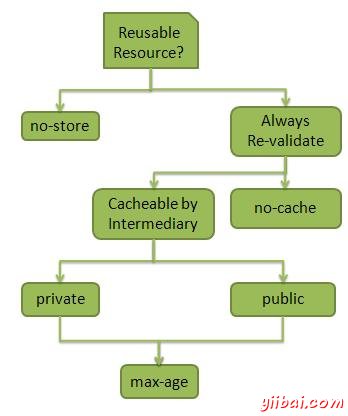
# Cache-Control头

以下是Cache-Control头的细节

|  |  |
| --- | --- |
| **S. N.** | **指令和说明** |
| 1 | **Public** 指示资源是可缓存任何组件。 |
| 2 | **Private** 指示资源是可缓存只有客户端和服务器，无中介可以缓存资源。 |
| 3 | **no-cache/no-store** 表示资源是不可缓存 |
| 4 | **max-age** 表示缓存有效期为max-age以秒为单位。在此之后，客户端必须使另一个请求。 |
| 5 | **must-revalidate** 指示服务器来验证资源，如果最大到期时间已经过去了。 |

## 最佳实践

* 始终保持静态内容，如图像，CSS，JavaScript的缓存，以2〜3天有效期。
* 不要让过期日期太长。
* 动态内容应只能缓存几个小时。



# RESTful Web服务安全

正如RESTfulWeb服务以HTTP URL的路径工作，所以这是非常重要的，在相同的方式维护一个RESTful Web服务的网站是安全的。以下是在设计一个RESTful Web服务应遵循的最佳实践。

* 验证 - 验证服务器上的所有输入。防止SQL或NoSQL注入攻击你的服务器。
* 基于会话的认证 - 使用基于会话的认证，每当一个请求到Web服务的方法来验证用户的身份。
* URL没有敏感数据 - 千万不要使用用户名，密码或会话令牌的URL，这些值应通过POST方法传递给Web服务。
* 方法执行限制 - 允许限制使用像GET，POST，DELETE等方法。 GET方法不应该用于删除数据。
* 验证格式错误 XML/JSON - 检查以及表单输入传递到Web服务方法。
* 抛出通用错误信息 - Web服务方法应使用HTTP错误信息像403，显示禁止访问等。

# RESTful Services Java (JAX-RS)

JAX-RS代表RESTful Web服务的Java API。JAX-RS是一款基于JAVA的编程语言和API规范，为创建基于RESTful Web服务提供支持。它的2.0版本发布于2013年5月24日。JAX-RS大量使用从Java SE5注释可用来简化基于Java Web服务创建和部署开发。 它还提供支持创建客户端RESTful Web服务。

|  |  |
| --- | --- |
| **S.N.** | **注解 & 描述** |
| 1 | **@Path** 资源类/方法的相对路径。 |
| 2 | **@GET** HTTP GET请求，用于获取资源。 |
| 3 | **@PUT** HTTP PUT请求，用于建立资源。 |
| 4 | **@POST** HTTP POST请求，用于创建/更新资源。 |
| 5 | **@DELETE** HTTP DELETE请求，用于删除的资源。 |
| 6 | **@HEAD** HTTP HEAD请求，用来获取方法可用状态。 |
| 7 | **@Produces** 状态通过Web服务产生的HTTP响应，例如APPLICATION/XML, TEXT/HTML, APPLICATION/JSON等。 |
| 8 | **@Consumes** 状态的HTTP请求类型，例如 application/x-www-form-urlencoded 接受POST请求的过程中HTTP主体表单数据。 |
| 9 | **@PathParam** 结合传递到方法，在路径中的值的参数。 |
| 10 | **@QueryParam** 结合传递给方法的路径中的查询参数的参数。 |
| 11 | **@MatrixParam** 结合传递给方法的路径中的HTTP矩阵参数的参数。 |
| 12 | **@HeaderParam** 结合传递到方法到HTTP头中的参数。 |
| 13 | **@CookieParam** 结合传递给方法一个Cookie参数。 |
| 14 | **@FormParam** 结合传递到方法的一种形式值的参数。 |
| 15 | **@DefaultValue** 分配一个默认值，以传递一个参数给方法。 |
| 16 | **@Context** 上下文中资源，例如HTTP请求的上下文。 |