Jersey hello world example

<http://www.mkyong.com/webservices/jax-rs/jersey-hello-world-example/>

AuthenticationExample:

<http://javapapers.com/web-service/restful-services-http-basic-authentication/>

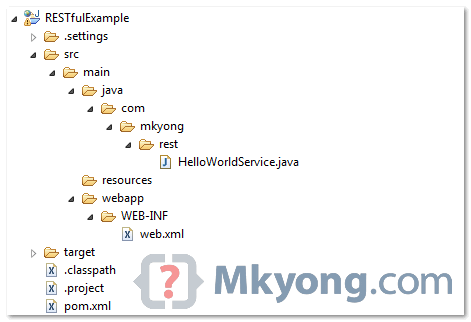
[Jersey](http://jersey.java.net/), reference implementation to develope RESTful web service based on the [JAX-RS (JSR 311)](http://jsr311.java.net/nonav/releases/1.1/index.html) specification.

In this tutorial, we show you how to develop a simple hello world REST web application with **Jersey**.

Technologies and Tools used in this article:

1. Jersey 1.8
2. JDK 1.6
3. Tomcat 6.0
4. Maven 3.0.3
5. Eclipse 3.6
6. **Note**  
   If you want to know what and how REST works, just search on Google, ton of available resources.

## 1. Directory Structure

1. This is the final web project structure of this tutorial.
2. 

## 2. Standard Web Project

1. Create a standard Maven web project structure.
2. mvn archetype:generate -DgroupId=com.mkyong.rest -DartifactId=RESTfulExample
3. -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false
4. **Note**  
   To support Eclipse, use Maven command :
5. mvn eclipse:eclipse -Dwtpversion=2.0

## 3. Project Dependencies

1. Jersey is published in Java.net Maven repository. To develop Jersey REST application , just declares “**jersey-server**” in Maven pom.xml.
2. File : pom.xml
3. <project ...>
4. <repositories>
5. <repository>
6. <id>maven2-repository.java.net</id>
7. <name>Java.net Repository for Maven</name>
8. <url>http://download.java.net/maven/2/</url>
9. <layout>default</layout>
10. </repository>
11. </repositories>
12. <dependencies>
13. <dependency>
14. <groupId>com.sun.jersey</groupId>
15. <artifactId>jersey-server</artifactId>
16. <version>1.8</version>
17. </dependency>
18. </dependencies>
19. </project>

## 4. REST Service

1. Simple REST service with Jersey.
2. package com.mkyong.rest;
3. import javax.ws.rs.GET;
4. import javax.ws.rs.Path;
5. import javax.ws.rs.PathParam;
6. import javax.ws.rs.core.Response;
7. @Path("/hello")
8. public class HelloWorldService {
9. @GET
10. @Path("/{param}")
11. public Response getMsg(@PathParam("param") String msg) {
12. String output = "Jersey say : " + msg;
13. return Response.status(200).entity(output).build();
14. }
15. }

## 5. web.xml

1. In web.xml, register “com.sun.jersey.spi.container.servlet.ServletContainer“, and puts your Jersey service folder under “**init-param**“, “com.sun.jersey.config.property.packages“.
2. File : web.xml
3. <web-app id="WebApp\_ID" version="2.4"
4. xmlns="http://java.sun.com/xml/ns/j2ee"
5. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6. xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
7. http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd">
8. <display-name>Restful Web Application</display-name>
9. <servlet>
10. <servlet-name>jersey-serlvet</servlet-name>
11. <servlet-class>
12. com.sun.jersey.spi.container.servlet.ServletContainer
13. </servlet-class>
14. <init-param>
15. <param-name>com.sun.jersey.config.property.packages</param-name>
16. <param-value>com.mkyong.rest</param-value>
17. </init-param>
18. <load-on-startup>1</load-on-startup>
19. </servlet>
20. <servlet-mapping>
21. <servlet-name>jersey-serlvet</servlet-name>
22. <url-pattern>/rest/\*</url-pattern>
23. </servlet-mapping>
24. </web-app>

## 6. Demo

1. In this example, web request from “**projectURL/rest/hello/**” will match to “**HelloWorldService**“, via @Path("/hello").
2. And the “**{any values}**” from “**projectURL/rest/hello/{any values}**” will match to parameter annotated with @PathParam.
3. URL : <http://localhost:8080/RESTfulExample/rest/hello/mkyong>
4. Deployment steps on Tomcat
5. <https://www.mkyong.com/maven/how-to-deploy-maven-based-war-file-to-tomcat/>
6. <https://stackoverflow.com/questions/5109112/how-to-deploy-a-war-file-in-tomcat-7>
7. 