RESTful webservice with Basic HTTP authentication

Prepared by:

Ashish Kumar Mondal

Contents

[Introduction 3](#_Toc416086260)

[Steps to create RESTful project 3](#_Toc416086261)

[Source Code 4](#_Toc416086262)

[Test the basic authentication in the webservice 5](#_Toc416086263)

[Test from browser 5](#_Toc416086264)

[Test Java client 5](#_Toc416086265)

[Troubleshooting for RESTful webservice setup 6](#_Toc416086266)

# Introduction

This document will help user to setup a RESTful webservice with Jersey framework.

**Assumption**: J2EE eclipse (e.g. Eclipse Kepler), JDK1.7 and maven is already available in the system

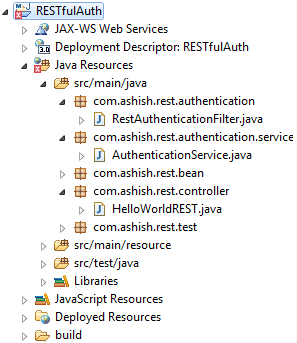
(If you need maven setup, follow my another document in Github "Maven build setup for your project.docx"

# Steps to create RESTful project

Best way to learn the RESTful web service is download a small helloworld project and check the pom.xml, web.xml, java file details as mentioned below

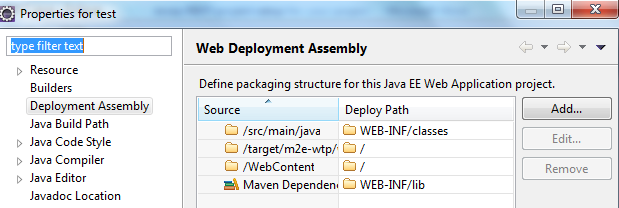
However, you can create a new project from scratch as mentioned below.

1. Create a Dynamic web project with module version 3.0 and java source directory must be src/main/java



1. Convert the project into maven project (right click on project-> Configure -> Convert to Maven project)
2. Create a package under java source (src/main/java): com.ashish.rest.controller
   1. Right click on project->properties->Deployment Assembly->Add->Java Build Path Entries->Maven Dependencies (Note: Deploy path should be WEB-INF/lib by default)

The meaning of the below entry is the dependent jars will go to WEB-INF/lib folder in the deployable



1. pom.xml should have below contains



1. Write the below java file to handle rest request



1. Write the below RestAuthenticationFilter Java files for Basic REST request authentication



1. Write the below service class for authenticate the user



1. Modify web.xml with the following content



1. You may need to update maven project by the following option (right click on the project->Maven->update maven project)

## Source Code



# Test the basic authentication in the webservice

## Test from browser

When I hit the below URL, I get **401** response as shown in the screenshot means your HTTP Basic authentication is working fine.

<http://localhost:8080/RESTfulAuth/rest/hello/getEmployee/123>



## Test Java client

Write the below attached Java client to test the below scenarios. Credential: **admin/admin** is being used as hardcoded



1. POST request without HTTP Basic authentication
2. GET request with HTTP Basic authentication
3. POST request with HTTP Basic authentication

java.lang.RuntimeException: Failed : HTTP error code : 404

at com.ashish.rest.test.TestClient.testPostWithoutBasicAuth(TestClient.java:98)

at com.ashish.rest.test.TestClient.main(TestClient.java:13)

Base64 encoded auth string: YWRtaW46YWRtaW4=

Response for the GET with HTTP Basic authentication request: {"empId":123,"address1":null,"address2":null,"address3":null,"pin":null,"salary":0.0,"name":"Ashish Mondal"}

=========================================================================

Base64 encoded auth string: YWRtaW46YWRtaW4=

Response for the POST with HTTP Basic authentication request: {"empId":0,"address1":null,"address2":null,"address3":null,"pin":null,"salary":1000.0,"name":"Ashish Mondal"}

=========================================================================

Below is the sample code in TestClient.java file as attached above

**private** **static** **void** testGETWithBasicAuth() {

**try** {

Client client = Client.*create*();

String name = "admin";

String password = "admin";

String authString = name + ":" + password;

String authStringEnc = **new** BASE64Encoder().encode(authString.getBytes());

System.*out*.println("Base64 encoded auth string: " + authStringEnc);

WebResource webResource = client.resource("http://localhost:8080/RESTfulAuth/rest/hello/getEmployee/123");

ClientResponse resp = webResource.accept("application/json")

.header("Authorization", "Basic " + authStringEnc)

.get(ClientResponse.**class**);

**if**(resp.getStatus() != 200){

System.*err*.println("Unable to connect to the server");

}

String output = resp.getEntity(String.**class**);

System.*out*.println("Response for the GET with HTTP Basic authentication request: "+output);

System.*out*.println("================================================================");

} **catch** (Exception e) {

e.printStackTrace();

}

}

# Troubleshooting for RESTful webservice setup

|  |  |  |
| --- | --- | --- |
| **SL No** | **Issue** | **Solution** |
| 1 | SEVERE: Servlet /JAXRS-HelloWorld threw load() exception  java.lang.ClassNotFoundException: com.sun.jersey.spi.container.servlet.ServletContainer | Right click on project->properties->Deployment Assembly->Add->Java Build Path Entries->Maven Dependencies (Note: Deploy path should be WEB-INF/lib by default) |
| 2 | com.sun.jersey.api.container.ContainerException: The ResourceConfig instance does not contain any root resource classes. | a) com.sun.jersey.config.property.packages doesn’t exist in your web.xml  <servlet>  <servlet-name>jersey-helloWorld-serlvet</servlet-name>  <servlet-class>  com.sun.jersey.spi.container.servlet.ServletContainer </servlet-class>  <init-param>  <param-name>  com.sun.jersey.config.property.packages  </param-name>  <param-value>com.ashish.rest.controller</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  b) com.sun.jersey.config.property.packages included a resource that doesn’t include any jersey services. In above case, "com.ashish.rest.controller” doesn’t contains any jersey services.  c) The project's java source directory must be under src/main/java folder as the project is of type Maven |
| 3 | Caused by: com.sun.jersey.api.MessageException: A message body writer for Java class com.ashish.rest.bean.Employee, and Java type class com.ashish.rest.bean.Employee, and MIME media type application/json was not found | Add below dependency in pom.xml  <dependency>  <groupId>com.sun.jersey</groupId>  <artifactId>jersey-json</artifactId>  <version>1.9.1</version>  </dependency>  <dependency>  <groupId>com.sun.jersey</groupId>  <artifactId>jersey-bundle</artifactId>  <version>1.18.1</version>  </dependency>  Add below entry in web.xml  <init-param>  <param-name>  com.sun.jersey.api.json.POJOMappingFeature  </param-name>  <param-value>true</param-value>  </init-param> |
| 4 | If you get below error during “mvn install” then see the next column for solution  [ERROR] /D:/ashish/PracticeWS/RESTfulAuth/src/main/java/com/ashish/rest/authentication/RestAuthenticationFilter.java:[5,21] package javax.servlet does not exist | Add the below dependency in pom.xml  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>javax.servlet-api</artifactId>  <version>3.0.1</version>  <scope>provided</scope>  </dependency> |