RESTful Java client with Jersey client

**package** com.howtodoinjava.jersey.beans;

//https://howtodoinjava.com/jersey/jersey-restful-client-api-authentication-example/

**import** java.io.IOException;

**import** java.util.List;

**import** javax.ws.rs.client.Client;

**import** javax.ws.rs.client.ClientBuilder;

**import** javax.ws.rs.client.Invocation;

**import** javax.ws.rs.client.WebTarget;

**import** javax.ws.rs.core.MediaType;

**import** javax.ws.rs.core.Response;

**import** org.glassfish.jersey.client.ClientConfig;

**import** org.glassfish.jersey.client.authentication.HttpAuthenticationFeature;

**import** org.glassfish.jersey.jackson.JacksonFeature;

**public** **class** ConsumeRestAPI {

**public** **static** **void** main(String[] args) **throws** IOException

{

*httpGETCollectionExample*();

}

**private** **static** **void** httpGETCollectionExample()

{

/\*<dependency>

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

<artifactId>jersey-server</artifactId>

<version>${jersey2.version}</version>

</dependency>\*/

ClientConfig clientConfig = **new** ClientConfig();

//HttpAuthenticationFeature feature = HttpAuthenticationFeature.basic("howtodoinjava", "password");

HttpAuthenticationFeature authenticationfeature = HttpAuthenticationFeature.*basic*("admin", "admin");

clientConfig.register( authenticationfeature) ;

/\* <dependency>

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

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

<version>${jersey2.version}</version>

</dependency> \*/

//'org.glassfish.jersey.media:jersey-media-json-jackson:2.25.1'

clientConfig.register(JacksonFeature.**class**);

/\*

<dependency>

<groupId>javax.ws.rs</groupId>

<artifactId>javax.ws.rs-api</artifactId>

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

</dependency> \*/

Client client = ClientBuilder.*newClient*( clientConfig );

WebTarget webTarget = client.target("http://localhost:8080/SecureRestAPI/rest").path("employees");

//other scenario

//client.target("<http://localhost:8080/JerseyDemos/rest>").path("employees").path("//1");

Invocation.Builder invocationBuilder = webTarget.request(MediaType.*APPLICATION\_JSON*);

Response response = invocationBuilder.get();

System.*out*.println(response.getStatus());

System.*out*.println(response.getStatusInfo());

**if**(response.getStatus() == 200)

{

Employees employees = response.readEntity(Employees.**class**);

List<Employee> lists = employees.getEmployeeList();

**for** (Employee employee : lists) {

System.*out*.println(employee.getId() +" : " + employee.getName());

}

/\* List<Employee> listOfEmployees = employees.getEmployeeList();

System.out.println(Arrays.toString( listOfEmployees.toArray(new Employee[listOfEmployees.size()]) ));

\*/ }

//FOR POST OPERATION

//for POSt **import** javax.ws.rs.client.Entity;

Employee emp = **new** Employee();

emp.setId(1);

emp.setName("David Feezor");

Invocation.Builder invocationBuilderpost = webTarget.request(MediaType.*APPLICATION\_XML*);

response = invocationBuilderpost.post(Entity.*entity*(emp, MediaType.*APPLICATION\_XML*));

}

}

===========POM=======

<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>com.howtodoinjava.jersey</groupId>

<artifactId>JerseyDemos</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<repositories>

<repository>

<id>maven2-repository.java.net</id>

<name>Java.net Repository for Maven</name>

<url>http://download.java.net/maven/2/</url>

<layout>default</layout>

</repository>

</repositories>

<properties>

<jersey2.version>2.19</jersey2.version>

<jaxrs.version>2.0.1</jaxrs.version>

</properties>

<dependencies>

<!-- JAX-RS -->

<dependency>

<groupId>javax.ws.rs</groupId>

<artifactId>javax.ws.rs-api</artifactId>

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

</dependency>

<!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.0.1</version>

<scope>provided</scope>

</dependency>

<!-- Jersey 2.19 -->

<dependency>

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

<artifactId>jersey-container-servlet</artifactId>

<version>${jersey2.version}</version>

</dependency>

<dependency>

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

<artifactId>jersey-server</artifactId>

<version>${jersey2.version}</version>

</dependency>

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

<version>2.2.4</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.glassfish.jersey.media/jersey-media-json-jackson -->

<dependency>

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

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

<version>${jersey2.version}</version>

</dependency>

</dependencies>

<build>

<finalName>JerseyDemos</finalName>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.6</source>

<target>1.6</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

By [mkyong](http://www.mkyong.com/author/mkyong/) | July 19, 2011 | Updated : August 29, 2012 | Viewed : 740,499 times +3,002 pv/w

This tutorial show you how to use **Jersey client APIs** to create a RESTful Java client to perform “**GET**” and “**POST**” requests to REST service that created in this “[Jersey + Json](http://www.mkyong.com/webservices/jax-rs/json-example-with-jersey-jackson/)” example.

1. Jersey Client Dependency

To use Jersey client APIs, declares “**jersey-client.jar**” in your pom.xml file.

*File : pom.xml*

<dependency>

<groupId>com.sun.jersey</groupId>

<artifactId>jersey-client</artifactId>

<version>1.8</version>

</dependency>

2. GET Request

Review last REST service.

@Path("/json/metallica")

public class JSONService {

@GET

@Path("/get")

@Produces(MediaType.APPLICATION\_JSON)

public Track getTrackInJSON() {

Track track = new Track();

track.setTitle("Enter Sandman");

track.setSinger("Metallica");

return track;

}

//...

Jersey client to send a “GET” request and print out the returned json data.

package com.mkyong.client;

import com.sun.jersey.api.client.Client;

import com.sun.jersey.api.client.ClientResponse;

import com.sun.jersey.api.client.WebResource;

public class JerseyClientGet {

public static void main(String[] args) {

try {

Client client = Client.create();

WebResource webResource = client

.resource("http://localhost:8080/RESTfulExample/rest/json/metallica/get");

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

.get(ClientResponse.class);

if (response.getStatus() != 200) {

throw new RuntimeException("Failed : HTTP error code : "

+ response.getStatus());

}

String output = response.getEntity(String.class);

System.out.println("Output from Server .... \n");

System.out.println(output);

} catch (Exception e) {

e.printStackTrace();

}

}

}

Output…

Output from Server ....

{"singer":"Metallica","title":"Enter Sandman"}

3. POST Request

Review last REST service.

@Path("/json/metallica")

public class JSONService {

@POST

@Path("/post")

@Consumes(MediaType.APPLICATION\_JSON)

public Response createTrackInJSON(Track track) {

String result = "Track saved : " + track;

return Response.status(201).entity(result).build();

}

//...

Jersey client to send a “POST” request, with json data and print out the returned output.

package com.mkyong.client;

import com.sun.jersey.api.client.Client;

import com.sun.jersey.api.client.ClientResponse;

import com.sun.jersey.api.client.WebResource;

public class JerseyClientPost {

public static void main(String[] args) {

try {

Client client = Client.create();

WebResource webResource = client

.resource("http://localhost:8080/RESTfulExample/rest/json/metallica/post");

String input = "{\"singer\":\"Metallica\",\"title\":\"Fade To Black\"}";

ClientResponse response = webResource.type("application/json")

.post(ClientResponse.class, input);

if (response.getStatus() != 201) {

throw new RuntimeException("Failed : HTTP error code : "

+ response.getStatus());

}

System.out.println("Output from Server .... \n");

String output = response.getEntity(String.class);

System.out.println(output);

} catch (Exception e) {

e.printStackTrace();

}

}

}

Output…

Output from Server ....

Track saved : Track [title=Fade To Black, singer=Metallica]