Get HTTP Header In JAX-RS

examples.javacodegeeks.com/enterprise-java/rest/get-http-header-in-jax-rs/

Nikos Maravitsas

In this example we are going to see how to get Request HTTP Header parameters in a JAX-RS REST Service. You can see the list of all Header fields in the HTTP 1.1 RFC.

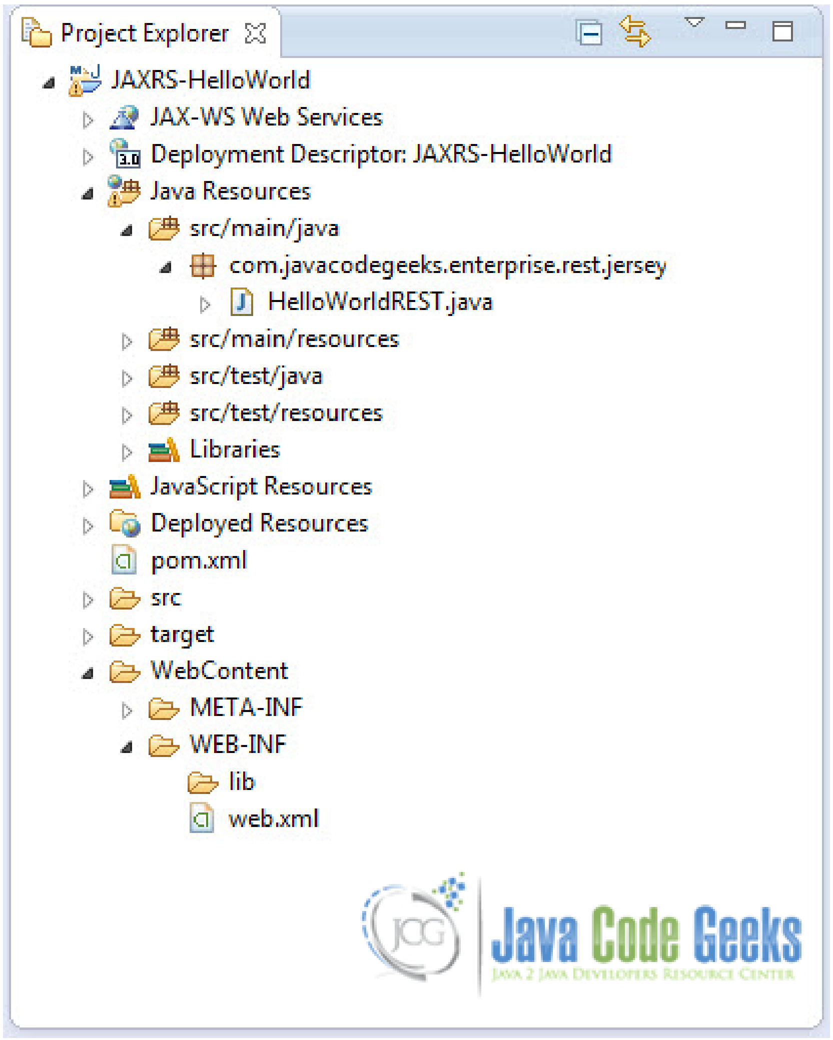
To be able to parse HTTP Header fields you can either use `@HeaderParam` to parse each Header field individually, or use `@Context` to inject an `HttpHeaders` object to the method an parse the Header fields you want from there.

In this example we are not going to focus on how to create JAX-RS application from top to bottom. So make sure you read carefully Jersey Hello World Example and RESTEasy Hello World Example, and pay attention to the sections concerning the creation of the project with Eclipse IDE as well as the deployment of the project in Tomcat.

The code of this tutorial is going to be based on Jersey Hello World Example. You can download the Eclipse project of this tutorial here : JAXRS-HelloWorld.zip

1. Project structure

Let's remind ourselves the structure of the project we are working on:



```
Project Explorer
JAXRS-HelloWorld
    JAX-WS Web Services
    Deployment Descriptor: JAXRS-HelloWorld
    Java Resources
        src/main/java
            com.javacodegeeks.enterprise.rest.jersey
                HelloWorldREST.java
        src/main/resources
        src/test/java
        src/test/resources
        Libraries
    JavaScript Resources
    Deployed Resources
    pom.xml
    src
    target
    WebContent
        META-INF
        WEB-INF
            lib
            web.xml
```

The code presented in this new tutorial will only concern `HelloWorldREST.java` file.

At this point you can also take a look at the `web.xml` file to see how the project is configured:

web.xml:

```
<?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>Restful Web Application</display-name>

 <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.javacodegeeks.enterprise.rest.jersey</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
 </servlet>

 <servlet-mapping>
  <servlet-name>jersey-helloworld-serlvet</servlet-name>
  <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

As you can see our servlet is mapped to `/rest/` URI pattern. So the basic structure of the URIs used in this example will have the form :

```
http://localhost:8080/JAXRS-HelloWorld/rest/....
```

2. Using @HeaderParam annotation

HelloWorldREST.java:

```
package com.javacodegeeks.enterprise.rest.jersey;

import javax.ws.rs.GET;
import javax.ws.rs.HeaderParam;
import javax.ws.rs.Path;
import javax.ws.rs.core.Response;

@Path("/members")
public class HelloWorldREST {

 @GET
 @Path("/info")
 public Response responseMsg(@HeaderParam("Content-Length") int length,
@HeaderParam("user-agent") String userAgent) {

   String output = "User agent :"+ userAgent +" sent :"+length +" bytes";
   return Response.status(200).entity(output).build();

 }
}
```

When you put on your browser:

```
http://localhost:8080/JAXRS-HelloWorld/rest/members/info
```

Outputs:

```
User agent :Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/30.0.1599.101 Safari/537.36 sent :0 bytes
```

2. Using @Context annotation

You can use `@Context` annotation usually to inject contextual Java types related to the request or response. In a JAX-RS application using servlet, `ServletConfig`, `ServletContext`, `HttpServletRequest` and `HttpServletResponse` objects are available using`@Context`. Let's see how you can use it ti inject an `UriInfo`, an interface that provides access to application and request URI information.

HelloWorldREST.java:

```
package com.javacodegeeks.enterprise.rest.jersey;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.HttpHeaders;
import javax.ws.rs.core.Response;

@Path("/members")
public class HelloWorldREST {

 @GET
 @Path("/info")
 public Response responseMsg(@Context HttpHeaders httpHeaders) {

  String host = httpHeaders.getRequestHeader("host").get(0);

  String agent = httpHeaders.getRequestHeader("user-
agent").get(0);

  String output = "User agent :"+ agent +" from host : "+host+".
```

Header fields : “; // list all header fields available through HttpHeaders for(String field : httpHeaders.getRequestHeaders().keySet()){ output += ” “+field; } return Response.status(200).entity(output).build(); } }

When you put on your browser:

```
http://localhost:8080/JAXRS-HelloWorld/rest/members/info
```

Outputs:

```
User agent :Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.2; Win64; x64;
Trident/7.0; MATMJS) from host : localhost:8080.

Header fields : accept accept-language user-agent ua-cpu accept-encoding host
connection
```

Download the Eclipse Project

This was an example on how to get HTTP Header in a JAX-RS REST Service. Download the Eclipse Project of this example : JAXRS-Headers.zip