*Table 1-1. Sample HTTP status codes and their meanings*

Status code In English Meaning

200 OK Request OK

303 See Other Redirect

400 Bad Request Request malformed

401 Unauthorized Authentication errors

403 Forbidden Request refused

404 Not Found Resource not found

405 Method Not Allowed Method not supported

415 Unsupported Media Type Content type not recognized

500 Internal Server Error Request processing failed

*Table 1-2. HTTP verbs and their CRUD operations*

HTTP verb CRUD operation

POST Create

GET Read

PUT Update

DELETE Delete

http:*//localhost:8080/predictions/*

If this URL is typed into a browser’s window, the browser generates a request similar to:

GET /predictions/ HTTP/1.1

User-Agent: Mozilla/5.0 (X11; Linux x86\_64) Chrome/24.0.1312.56

Host: localhost:8080

Accept: text/html

The web service APIs include:

• HttpServlet and its equivalents (e.g., JSP scripts)

• JAX-RS, which has various implementations

• Restlet, which is similar in style to JAX-RS

• JAX-WS @WebServiceProvider, which is a relatively low-level API

**What are idempotent and/or safe methods?**

Safe methods are HTTP methods that do not modify resources. For instance, using GET or HEAD on a resource URL, should NEVER change the resource. However, this is not completely true. It means: it won't change the resource representation. It is still possible, that safe methods do change things on a server or resource, but this should not reflect in a different representation.

Safe methods are methods that can be cached, prefetched without any repercussions to the resource.

Idempotent methods

An idempotent HTTP method is a HTTP method that can be called many times without different outcomes. It would not matter if the method is called only once, or ten times over. The result should be the same. Again, this only applies to the result, not the resource itself. This still can be manipulated (like an update-timestamp, provided this information is not shared in the (current) resource representation.

|  |  |  |
| --- | --- | --- |
| **HTTP** | **Idempotent** | **Safe** |
| OPTIONS | Y | Y |
| GET | Y | Y |
| HEAD | Y | Y |
| PUT | Y | N |
| POST | N | N |
| DELETE | Y | N |
| PATCH | N | N |

Securing RESTful services and accessing secure RESTful service

https://docs.oracle.com/cd/E19226-01/820-7627/bncbk/index.html

web.xml

<security-constraint>

<auth-constraint>

<role-name>member</role-name>

</auth-constraint>

<web-resource-collection>

<url-pattern>/backend/user/service/getUserInfo</url-pattern>

<http-method>GET</http-method>

</web-resource-collection>

<user-data-constraint>

<transport-guarantee>CONFIDENTIAL</transport-guarantee> <!-- The other parameter is INTEGRAL -->

</user-data-constraint>

</security-constraint>

**<!-- SECURITY CONSTRAINT #1 -->**

<security-constraint>

<web-resource-collection>

<web-resource-name>wholesale</web-resource-name>

<url-pattern>/acme/wholesale/\*</url-pattern>

<http-method>GET</http-method>

<http-method>POST</http-method>

</web-resource-collection>

<auth-constraint>

<role-name>PARTNER</role-name>

</auth-constraint>

<user-data-constraint>

<transport-guarantee>CONFIDENTIAL</transport-guarantee>

<!-- The other parameter is INTEGRAL -->

</user-data-constraint>

</security-constraint>

**<!-- SECURITY CONSTRAINT #2 -->**

<security-constraint>

<web-resource-collection>

<web-resource-name>retail</web-resource-name>

<url-pattern>/acme/retail/\*</url-pattern>

<http-method>GET</http-method>

<http-method>POST</http-method>

</web-resource-collection>

<auth-constraint>

<role-name>CLIENT</role-name>

</auth-constraint>

<user-data-constraint>

<transport-guarantee>CONFIDENTIAL</transport-guarantee>

</user-data-constraint>

</security-constraint>

<login-congig>

<auth-method>BASIC </auth-method>

</login-congig>

restful web-auth methods

basic - user name/password would be sent in header.It will be mapped with the users.xml in the tomcat server. role also will be mapped. In web browser, a popup window will appear asking for user name and password.

digest

form

client

SSL Config

Server - In server side, the <user-data-constraint> has to be added. In the server.xml, the <connector> tag for the https port should have the file path of the key and certificate.

Client - There is no config required. In Jersey, create a SSLContext, a TrustManager impl of X509TrustManager to validate the kind of certificates we receive during handshake, add the context to the WebResource.

in IBM, its not required to use the jersey. there is a easier config related method. see this link

https://www-01.ibm.com/support/knowledgecenter/SSEQTP\_8.5.5/com.ibm.websphere.wlp.doc/ae/twlp\_sec\_ws\_ssl.html?cp=SSEQTP\_8.5.5&lang=en

BASIC auth-method server.

Client - JAX-RS filter

HTTPBasicAuthFilter , a jersey api to pass user name and password

In Tomcat, its server.xml. In Websphere, its ibm-ws-bnd.xml for every war file would have the security contraint related details. the server.xml of was would have connector related details.

Spring REST client for Digest login

<http://www.baeldung.com/resttemplate-digest-authentication>