INTRODUCTION TO CONSUMING RESTFUL WEB SERVICES USING JAVA

NYOUG – SEPTEMBER 14, 2016

SUZANNE PREZORSKI
LEAD PROGRAMMER ANALYST
ALTICE USA
INTRODUCTION

- REST is defined as an “architectural style”, and is short for Representational State Transfer.
- Created in 2000, by Roy Fielding at University of California, it is based on the HTTP protocol.
- APIs from many software systems employ REST, it is a way that you can exchange data with other software applications.
- In this crash course, brief tutorial you will learn:
  - How can I write java programs to work with REST APIs?
  - What are some software tools that I can use with REST APIs?
WADL – WEB APPLICATION DESCRIPTION LANGUAGE

- An XML based description of a HTTP-Based RESTful Web Service
- WADL is machine readable, and you can import it into API Testing tools. Doing so will automatically generate the HTTP actions that you can make on the web service.
- There are other ways to describe RESTful web services:
  - WSDL 2.0
  - RAML (raml.org)
<xml version="1.0" encoding="UTF-8" standalone="yes"/>
<application xmlns="http://research.sun.com/wesb/2006/10/">
<doc xmlns:jersey="http://jersey.dev.java.net/"
jersey:generateBy="Jersey: 1.0-SNAPSHOT 10/02/2008 12:17 PM">
<resources base="http://localhost:8996/storage/">
<resource path="/containers"/>
</resources>

<method name="GET" id="getContainers">
<response>
<representation mediaType="application/xml"/>
</response>
</method>

<resource path="/container"/>

<method name="PUT" id="putContainer">
<request>
<representation mediaType="application/xml"/>
</request>
</method>

<method name="DELETE" id="deleteContainer"/>

<method name="GET" id="getContainer">
<request>
<param xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
</request>
</method>

<resource path="/item/.*"/>

<method name="PUT" id="putItem">
<request>
<representation mediaType="/*"/>
</request>
</method>

<method name="DELETE" id="deleteItem"/>

<method name="GET" id="getItem">
<request>
<representation mediaType="/*"/>
</request>
</method>
</application>
HTTP METHODS & URIs

• When consuming RESTful Web Services from a client – The client interacts with resources, and HTTP verbs or methods, through URIs.

• URI stands for Uniform Resource Indicator – it is the address to a resource. For example: http://google.com/search?q=NYOUG

• POPULAR HTTP VERBS: (CRUD)
  • GET – The client will send a GET request to read a resource via the URI
  • PUT – In order to create a new resource, client will invoke a PUT
  • POST – Update an existing resource
  • DELETE – Client will call a DELETE a resource to remove it.
@GET
@Path("/get")
@Produces("text/html")
public String sayHello()
{
    return "Hello World";
}

http://localhost:8080/HelloWorld/get
7 API TESTING TOOL

- You can download an open source tool such as SOAP UI (from Smart Bear), and type in a resource and get enter requests: (SOAP UI works with SOAP and REST Web Services)
JSON and XML – Return Types

• JSON (pronounced: “Jason”), short for: Java Script Object Notation – Geared towards data exchange

• XML – Extensible Markup Language – Geared towards encoding documents

• In the API description of the Web Service, it should specify how the data will be returned to you. For example in a GET operation, the return type may either be XML or JSON.

• JSON is much more less verbose than XML
JSON VS XML

Attribute value pairs are used by JSON
JAX-RS – JAVA API FOR RESTFUL WEB SERVICES

- Uses Annotations to map to resources: @Path, @GET, @POST, @DELETE

- On the client side there are a few steps for a REST Client to make a call to a web service that supports REST:
  - Instantiate the Client Interface from javax.ws.rs.client.Client
  - Configure the client with the target that you want (i.e. the URI)
  - Make the request, specifying your target
  - Issue the request
package lib;
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.core.MediaType;
public class CallRestService {
  public static void main(String[] args) {
    Client client = ClientBuilder.newClient();
    String name = client.target("http://geo.groupkt.com/ip/68.193.182.43/json/").request(MediaType.APPLICATION_JSON).get(String.class);
    System.out.println(name);
    client.close();
  }
}
HTTP CODES

- 1xx: Meta – Used when communicating with server
- 2xx: Success – example 200 (“OK”)
- 3xx: Redirection – examples: 301 ("Moved Permanently")
- 4xx: Client Side Error – infamous 404 (“Not Found”); 401 (“Unauthorized”)
- 5xx: Server Side Errors - example 500 (“Internal Server Error”)

```java
Response vres = client.target("http://geo.groupkt.com/ip/68.193.182.43/json/").request().get();
Integer responseCode = vres.getStatus();
System.out.println(responseCode);
```
13 HTTP HEADERS

- Both HTTP requests and response send back headers

Example of a response header

```
HTTP/1.1 200 OK
Server: openresty
Date: Mon, 12 Sep 2016 01:28:02 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 436
Connections: keep-alive
x-cache-key: /data/2.5/weather?q=manhattan%2Cny
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: GET, POST
```

Each API may have different requirements on what you may need to send in the request header. You can send multiple header lines in an HTTP Request.

Header fields are in a name : value format: For example: Authorization: Basic <base64 string>
REST & AUTHENTICATION

- HTTPs is a secure version of HTTP. Called “REST over HTTPS”;
- Traffic is encrypted using SSL/TLS.
- Some APIs will require the use of a “token”. In this case you will be required to authenticate in the HTTP Header. Usually authentication handled by user name password. Base64 encode the `user:password` string, and pass the encoding in:
  
  ```
  Authorization: Basic czZCaGRSa3F0Mzo3RmpmcDBaQnlxS3REUmJuZlZkbUl3
  ```

- Some APIs will ask you to fetch a token first before calling other resources.
- Once you obtain the token, you pass it into the HTTP Header, using Authorization : Bearer `<token>`
15 OAUTH AUTHENTICATION

- OAUTH is a popular authentication method for RESTful Web services
- An open authentication framework
- www.oauth.net
- JAX-RS supports OAUTH authentication.
- Example using Jersey JAX-RS implementation using OAUTH:

https://github.com/jersey/jersey/commit/e8aff74b4493e73dd25eec1f6adfbe3685bd8d9d
### 16 API KEYS

- Some API providers would like you to register and get an API key, which is required to pass into the request:

```json
{
    "cod": 401,
    "message": "Invalid API key. Please see http://openweathermap.org/faq#error401 for more info."
}
```

* If you forget the API Key, you may get a HTTP Code 401
If you are authenticating with a user name and password, you usually base64 encode the string.

With a simple line of code, you can encode the string:

```java
package lib;
import org.glassfish.jersey.internal.util.Base64;
public class EncodeString64 {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String ConnectString;
        ConnectString="nyoug:pwd123";
        System.out.println(Base64.encodeAsString("nyoug:pwd123"));
    }
}
```
API KEYS – CODE EXAMPLE, PASSING IN QUERY PARAMETERS

Client client = ClientBuilder.newClient();

String name = client.target("http://api.openweathermap.org/data/2.5/weather/").queryParam("q","London,UK").queryParam("appid","01960008703e730395c88f7e946386g6")
.request(MediaType.APPLICATION_JSON)
.get(String.class);

System.out.println(name);
client.close();
Client client=ClientBuilder.newClient();

Response vres = client.target("http://geo.groupkt.com/ip/68.193.182.43/json/").request().get();

Integer responsecode= vres.getStatus();

System.out.println(responsecode);

client.close();
SOFTWARE DOWNLOADS FOR DEMOS

• Link to Download: Oracle Enterprise Pack for Eclipse (12.2.1.3.1) – (Neon)

• https://jersey.java.net/download.html

• http://smarbear.com
  (API Testing Tool)
2.1 SOME REST API EXAMPLES

- programmableweb.com (Directory of Web Services)
- Github (https://developer.github.com/v3/)
- Twitter (https://dev.twitter.com/rest/public)
- PayPal (https://developer.paypal.com/docs/api/)
- OpenWeather (api.openweathermap.org)
REFERENCES


• W3C : WADL Submission : https://www.w3.org/Submission/wadl/

• JAX-RS documentation: https://jersey.java.net/documentation/latest/client.html

• RESTful Web Services. Leonard Richardson & Sam Ruby. O’REILLY.

• Oracle Java Documentation (docs.oracle.com)