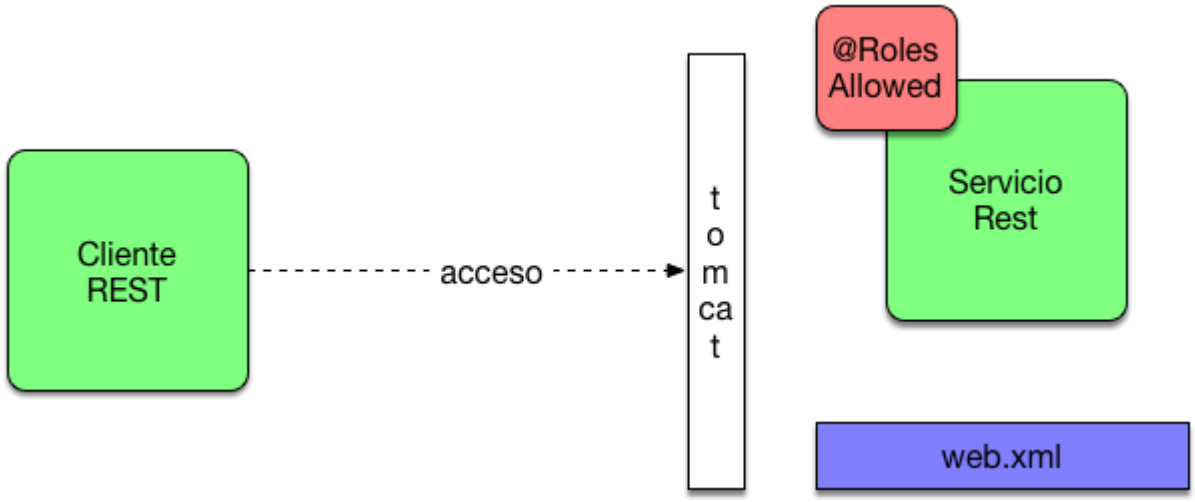
Vamos a ver un ejemplo de JAX RS Client Security. En el artículo anterior hemos protegido un recurso REST mediante autenticación básica . Es momento de conectarnos a él usando un cliente Java .



El primer paso es configurar las dependencias que nuestra aplicación de consola necesita usando Maven:

```
&amp;amp;lt;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"&amp;amp;gt;
&amp;amp;lt;modelVersion&amp;amp;gt;4.0.0&amp;amp;lt;/modelVersion&amp
;amp;gt;
&amp;amp;lt;groupId&amp;amp;gt;com.arquitecturajava&amp;amp;lt;/groupI
d&amp;amp;gt;
&amp;amp;lt;artifactId&amp;amp;gt;ClienteREST&amp;amp;lt;/artifactId&a
mp;amp;gt;
 &amp;amp;lt;version&amp;amp;gt;0.0.1-
SNAPSHOT&amp;amp;lt;/version&amp;amp;gt;
 &amp;amp;lt;dependencies&amp;amp;gt;
```

```
 &amp;amp;lt;dependency&amp;amp;gt;
&amp;amp;lt;groupId&amp;amp;gt;org.glassfish.jersey.core&amp;amp;lt;/g
roupId&amp;amp;gt;
 &amp;amp;lt;artifactId&amp;amp;gt;jersey-
client&amp;amp;lt;/artifactId&amp;amp;gt;
 &amp;amp;lt;version&amp;amp;gt;2.25.1&amp;amp;lt;/version&amp;amp;gt;
 &amp;amp;lt;/dependency&amp;amp;gt;
 &amp;amp;lt;dependency&amp;amp;gt;
&amp;amp;lt;groupId&amp;amp;gt;org.glassfish.jersey.media&amp;amp;lt;/
groupId&amp;amp;gt;
 &amp;amp;lt;artifactId&amp;amp;gt;jersey-media-json-
jackson&amp;amp;lt;/artifactId&amp;amp;gt;
 &amp;amp;lt;version&amp;amp;gt;2.25&amp;amp;lt;/version&amp;amp;gt;
 &amp;amp;lt;/dependency&amp;amp;gt;
 &amp;amp;lt;/dependencies&amp;amp;gt;
&amp;amp;lt;/project&amp;amp;gt;
```

Una vez hecho esto vamos a crear el programa que se encarga de conectarse al servicio REST. Este programa incluirá una clase de negocio y un cliente.

```
package com.arquitecturajava.bo;

import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name="noticia")
public class Noticia {

        private String id;
        private String titulo;
```

```
        public String getId() {
                return id;
        }
        public void setId(String id) {
                this.id = id;
        }
        public String getTitulo() {
                return titulo;
        }
        public void setTitulo(String descripcion) {
                this.titulo = descripcion;
        }
}
```

```
package com.arquitecturajava.cliente;

import java.util.ArrayList;
import java.util.List;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.GenericType;
import javax.ws.rs.core.MediaType;

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

import com.arquitecturajava.bo.Noticia;
```

```
public class Principal {

        public static void main(String[] args) {

                Client client = ClientBuilder.newClient();
                WebTarget target =
client.target("http://localhost:8080/WebRESTSeguridad/noticias");

GenericType&amp;amp;lt;List&amp;amp;lt;Noticia&amp;amp;gt;&amp;amp;gt;
noticias = new
GenericType&amp;amp;lt;List&amp;amp;lt;Noticia&amp;amp;gt;&amp;amp;gt;
() {
                };

                List&amp;amp;lt;Noticia&amp;amp;gt; listaNoticia =
target.request(MediaType.APPLICATION_JSON).get(noticias);

                for (Noticia n : listaNoticia) {

                        System.out.println(n.getId());

                        System.out.println(n.getTitulo());

                }
        }

}
```

En este caso estamos usando JAX RS y el API de cliente para acceder al servicio REST que creamos anteriormente . Ahora bien no estamos pasando ningún tipo de credencial. Por lo

tanto el servicio nos devolverá una excepción de acceso no autorizado:

```
Problems  @ Javadoc  Declaration  Console  Debug  Servers
<terminated> Principal [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_71.jdk/Contents/Home/bin/java (17 feb. 201
Exception in thread "main" javax.ws.rs.NotAuthorizedException: HTTP 401 No Autorizado
        at org.glassfish.jersey.client.JerseyInvocation.convertToException(JerseyInvocation.java:1
        at org.glassfish.jersey.client.JerseyInvocation.translate(JerseyInvocation.java:883)
        at org.glassfish.jersey.client.JerseyInvocation.access$800(JerseyInvocation.java:92)
        at org.glassfish.jersey.client.JerseyInvocation$3.call(JerseyInvocation.java:724)
        at org.glassfish.jersey.internal.Errors.process(Errors.java:315)
        at org.glassfish.jersey.internal.Errors.process(Errors.java:297)
        at org.glassfish.jersey.internal.Errors.process(Errors.java:228)
        at org.glassfish.jersey.process.internal.RequestScope.runInScope(RequestScope.java:444)
        at org.glassfish.jersey.client.JerseyInvocation.invoke(JerseyInvocation.java:720)
        at org.glassfish.jersey.client.JerseyInvocation$Builder.method(JerseyInvocation.java:430)
        at org.glassfish.jersey.client.JerseyInvocation$Builder.get(JerseyInvocation.java:321)
        at com.arquitecturajava.cliente.Principal.main(Principal.java:30)
```

JAX RS Client Security

Para solventar este problema lo único que nos falta es modificar el cliente de JAX-RS y añadir la autenticación básica.

```
 HttpAuthenticationFeature autentificacion =
HttpAuthenticationFeature.basic("cecilio", "cecilio");
 Client client = ClientBuilder.newClient();
 client.register(autentificacion);
```

Acabamos de registrar el tipo de Autenticación, ahora ya no tendremos problemas y accederemos al servicio sin errores:

```
1
noticia1
2
noticia2
```

Nos acabamos de conectar con JAX RS Client Security. Cada framework de JAX-RS utiliza su propio sistema de autenticación en la parte cliente ya que la especificación lo deja muy

abierto. En este caso hemos abordado la autenticación más básica , la cual no incluye encriptación del canal.

Otros artículos relacionados:

JAX-RS Client y Servicios REST

El porqué de los MicroServicios

Seguridad y JAAS