El concepto de JAX RS Security es uno de los que pronto o tarde nos encontramos cuando trabajamos con servicios REST . En muchos casos los servicios REST necesitan de algún tipo de autenticación y autorización , no puede acceder a ellos cualquiera. Vamos a diseñar un servicio que nos devuelva una lista de noticias y que necesitemos securizarle. Lo primero es crear el objeto de negocio que usará el servicio.

```
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;
 }
}
```

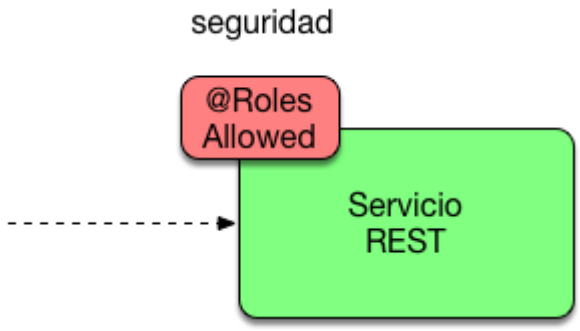
Creado el objeto de negocio el siguiente paso es crear el servicio que nos devuelve una lista

de noticias y securizarle.

```
package com.arquitecturajava;

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

import javax.annotation.security.RolesAllowed;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

import com.arquitecturajava.bo.Noticia;

@Path("/noticias")
public class ServicioNoticias {


 @GET
 @Produces (MediaType.APPLICATION_JSON)
 @RolesAllowed({"administrador"})
 public List<Noticia> getNoticias() {

 List<Noticia> noticias=new ArrayList<Noticia>();
 noticias.add(new Noticia("1","noticia1"));
 noticias.add(new Noticia("2","noticia2"));


 return noticias;
```

```
 }

}
```

En este caso todo es muy sencillo , hemos usado las anotaciones @Path y @GET para crear el servicio . Hemos decidido que devuelve los datos en JSON con @Produces y finalmente hemos usado una anotación de @RolesAllowed para asignar el rol que puede acceder al servicio. Acabamos de configurar la parte de autorización , es decir quien o quienes van a tener acceso al servicio.



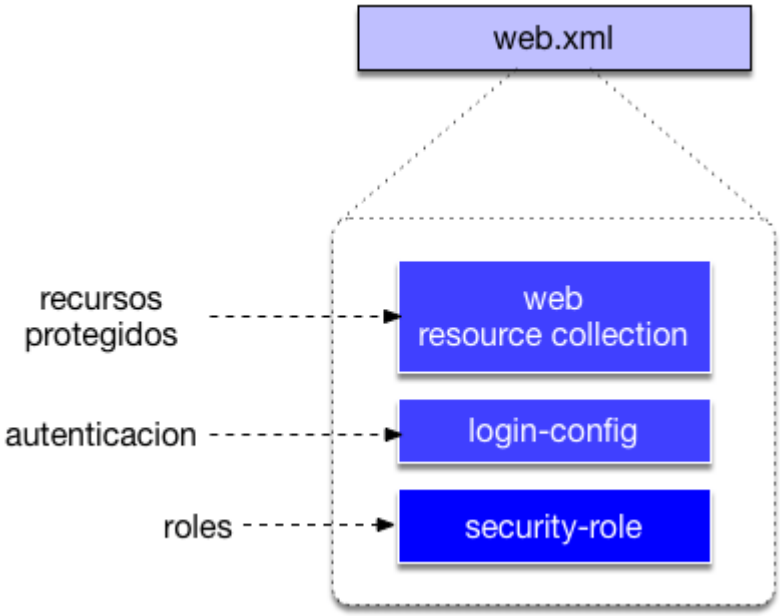
JAX RS Security y web.xml

El siguiente paso es usar el fichero web.xml y proteger el acceso a la aplicación.

```
<?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" version="3.0">
 <display-name>WebRESTSeguridad</display-name>
```

```
 <security-constraint>
 <web-resource-collection>
 <web-resource-name>recurso noticias</web-resource-name>
 <url-pattern>/</url-pattern>
 <http-method>GET</http-method>
 </web-resource-collection>
 <auth-constraint>
 <role-name>administrador</role-name>
 </auth-constraint>
 </security-constraint>

 <login-config>
 <auth-method>BASIC</auth-method>
 <realm-name>jaxrs</realm-name>
 </login-config>

 <security-role>
 <role-name>administrador</role-name>
 </security-role>


</web-app>
```

Este fichero define las url protegidas de nuestra aplicación. Define ademas que solo el rol de administrador tiene acceso a la aplicación junto el tipo de autenticación necesaria para acceder.



En este caso hemos usado TommEE y un Realm o zona de seguridad sencilla basado en ficheros XML al cual denominamos jaxrs. Recordemos que para configurar el Realm a nivel te Tomcat es suficiente modificar el clásico fichero tomcat-users.xml

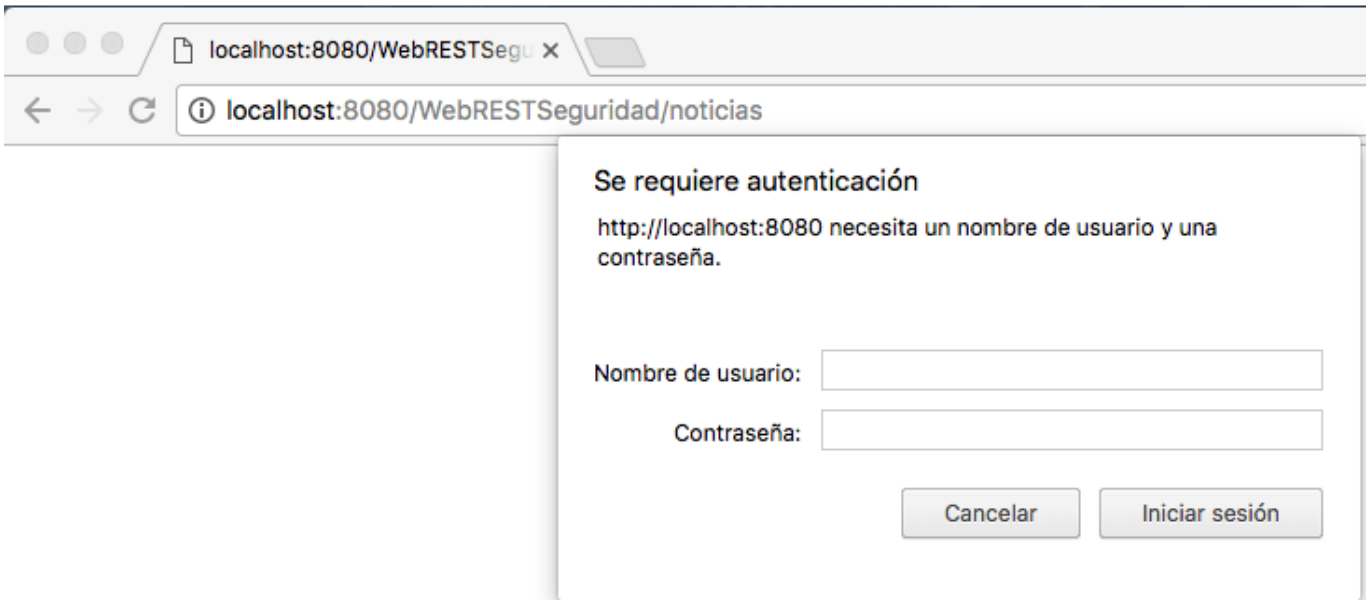
```
<role rolename="administrador"/>
<user username="cecilio" password="cecilio" roles="administrador"/>
```

Hemos dado de alta un único usuario con ese rol. Para que todo funcione correctamente en TomEE hay que añadir un fichero de resources.xml que configura el formato de los datos JSON que el servicio genera junto con un fichero de open-ejb.xml

```
<resources>
 <Service id="jsonProvider" class-
name="org.apache.cxf.jaxrs.provider.json.JSONProvider">
 dropRootElement=true
 supportUnwrapped = true
```

```
 dropCollectionWrapperElement=true
 serializeAsArray = true
 </Service>
</resources>


<openejb-jar xmlns="http://www.openejb.org/openejb-jar/1.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.openejb.org/openejb-jar/1.1">
 <pojo-deployment class-name="jaxrs-application">
 <properties>
 cxf.jaxrs.providers = jsonProvider
 </properties>
 </pojo-deployment>
</openejb-jar>
```

En principio ya tenemos todo configurado. Es momento de desplegar el servicio y ejecutarlo. Al invocarlo el navegador nos bloqueará el acceso a través de autenticación básica (la más sencilla de usar) solicitándonos usuario y clave.



Introducimos “cecilio” ,”cecilio” y accederemos.

Evidentemente no es la forma mas segura de autenticarse , pero ya tenemos construido el lado de servidor con JAX RS Security En el siguiente artículo veremos como conectarnos desde un cliente JAX-RS usando autenticación básica.

Otros artículos relacionados : JAX-RS Client y Servicios REST , Java Security y anotaciones JAAS ,Java EE MicroServices con Payara