Todos hemos usado JAAS y el fichero web.xml para configurar la seguridad en una aplicación web Java. A partir de Servlets 3.0 podemos configurar la seguridad utilizando un conjunto de anotaciones en vez de xml que nos permiten integrar todo dentro del propio Servlet de forma más cómoda. Vamos a verlo en un ejemplo, supongamos que partimos del siguiente bloque del web.xml que define un web-resource-collection y nos protege de acceso un Servlet.

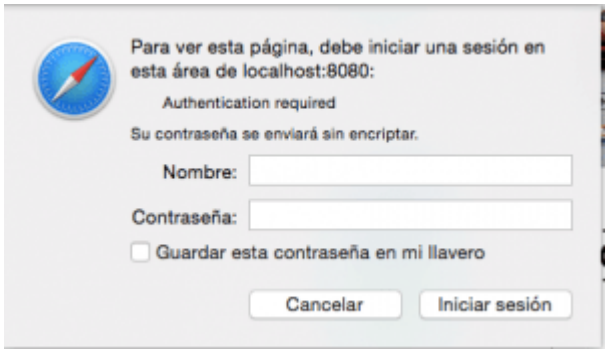
```
<security-constraint>
 <web-resource-collection>

 <web-resource-name>ServletHola</web-resource-name>
 <url-pattern>/ServletHola</url-pattern>

 </web-resource-collection>
 <auth-constraint>

 <role-name>administrador</role-name>
 </auth-constraint>

 </security-constraint>
```

Vamos a construir el Servlet más sencillo:

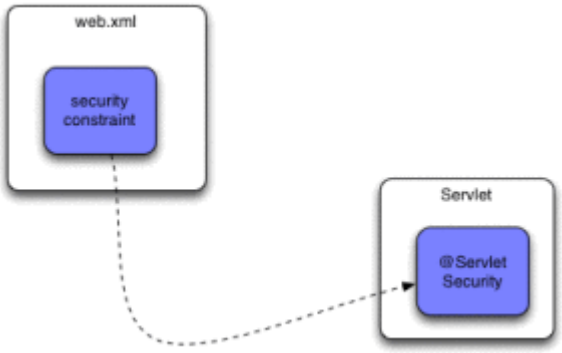
```
package com.arquitecturajava;

import java.io.IOException;
import java.io.PrintWriter;
```

```
import javax.servlet.ServletException;
import javax.servlet.annotation.HttpConstraint;
import javax.servlet.annotation.ServletSecurity;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/ServletHola")
public class ServletHola extends HttpServlet {
 private static final long serialVersionUID = 1L;

 protected void doGet(HttpServletRequest request,
 HttpServletResponse response) throws ServletException, IOException {

 PrintWriter pw = response.getWriter();
 pw.println("<html><body>has accedido</body></html>");
 pw.close();

 }
}
```

Utilizando el fichero web.xml podemos proteger el Servlet de acceso de tal forma que cuando invoquemos la URL el navegador nos solicite las credenciales de acceso.



Podemos modificar ahora nuestro Servlet para que utilice anotaciones de ServletSecurity en vez del web.xml de esta forma el web.xml quedará simplificado y el comportamiento será el mismo.



Para ello vamos a utilizar la anotación de @ServletSecurity que permite realizar practicamente las mismas operativas de seguridad que el web.xml para un recurso predeterminado.

```
package com.arquitecturajava;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.HttpConstraint;
import javax.servlet.annotation.ServletSecurity;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;

@WebServlet("/ServletHola")
@ServletSecurity(@HttpConstraint(rolesAllowed = { "administrador" }))
public class ServletHola extends HttpServlet {
 private static final long serialVersionUID = 1L;

 protected void doGet(HttpServletRequest request,
 HttpServletResponse response) throws ServletException, IOException {

 PrintWriter pw = response.getWriter();
 pw.println("<html><body>has accedido</body></html>");
 pw.close();

 }
```

De esta forma podremos eliminar del web.xml la configuración que hace referencia al web-resource-collection así como todo lo que teníamos ligado a un Servlet.

Otros artículos relacionados:JAAS y Seguridad , JAAS y Seguridad II , Provedores de Seguridad