# Uso del JAASRealm en Apache Tomcat

**Introducción**

JAASRealm es una implementación de la interfaz Tomcat Realm y sirve para autenticar a los usuarios utilizando el framework de autenticación y autorización JAAS.

Tomcat integra por defecto el uso de JAAS, por lo tanto no hay que incluirle librerías externas o similares.

JAASRealm permite al programador desarrollar un mecanismo de seguridad propio basado en la implementación personalizada de las interfaces JAAS LoginModule y JAAS Principal.

**Quick Start (obtenido del manual de Tomcat)**

Para configurar Tomcat para que use JAASRealm con nuestro propio JAAS LoginModule se deben seguir los pasos siguientes:

1. Programar nuestras propias clases para LoginModule, User y Role basadas en las interfaces proporcionadas por JAAS (deben implementar las interfaces LoginModule y Principal respectivamente).
2. Compilar las clases y situarlas en el classpath de Tomcat.
3. Crear un fichero de configuración (jaas.config) para indicar a Tomcat qué LoginModule utilizaremos (se podrían tener varios). Situarlo en la carpeta \conf.
4. Utilizar la environment variable JAVA_OPTS para indicar a Tomcat donde está el fichero de configuración anterior y lo pueda tener en cuenta al arrancar. Para ello generalmente se crea el script setenv.sh (linux) o setenv.bat (windows) que se sitúa en la carpeta /bin del Tomcat y el propio Tomcat lanza al arrancar el servidor mediante el script startup.sh/startup.bat (situados en \bin). La línea que habría que introducir en el script setenv sería:

   Linux:
   ```
   export JAVA_OPTS=-Djava.security.auth.login.config==$CATALINA_BASE/conf/jaas.config
   ```

   Windows:
   ```
   set JAVA_OPTS=-Djava.security.auth.login.config==%CATALINA_HOME%\conf\jaas.config
   ```

5. Configurar los security-constraints del web.xml correspondiente a la aplicación web desarrollada (*no* se recomienda utilizar el web.xml general que se encuentra en la carpeta \conf).
6. Configurar el módulo JAASRealm en el server.xml (situado en \conf). Para ello se crea un elemento <Realm> dentro de un nodo <Engine>.
7. Arrancar Tomcat mediante el script startup.sh/startup.bat situado en \bin.

**Crear un ejemplo sencillo paso por paso**

1. Descargamos el **tar.gz/zip** correspondiente a la **core distribution** de apache tomcat 8 (versión actual: 8.5.4). Se puede encontrar en: https://tomcat.apache.org/download-80.cgi
2. Descomprimimos y copiamos el directorio completo en /opt.
3. Obtenemos de Soporte_ManualJAASTomcat.zip (ZIP adjunto a este documento) los ficheros PlainLoginModule.java, PlainRolePrincipal.java y PlainUserPrincipal.java. El primer código implementa la interfaz JAAS LoginModule (aquí es donde ocurre todo el proceso de autenticación y asignación de roles); mientras que los otros dos códigos implementan la interfaz JAAS Principal.
   ***NOTA*:** La autenticación que realiza el LoginModule proporcionado es muy básica, simplemente acepta el acceso cuando login==password, y asigna el rol "tomcat" al usuario que haya realizado dicho acceso.
4. Situamos los ficheros .java en la carpeta \lib del tomcat y los compilamos: *javac *.java*
5. Creamos el fichero jaas.config y lo situamos en la carpeta \conf. El contenido será el siguiente (llamamos a nuestra configuración "prova"):

```
prova {
        PlainLoginModule required debug=true;
};
```

6. Creamos el script setenv.sh tal y como se comentaba en el Quick Start y lo situamos en \bin.
   ***NOTA*:** Tomcat ejecutará automáticamente este script al iniciarse siempre y cuando utilicemos los scripts de arranque estándar que hay en \bin para arrancar el servidor (startup.sh).
7. Configuramos el server.xml (en \conf) para que utilice el JAASRealm definido. Esto es crear el elemento Realm siguiente dentro del nodo Engine:

```
<Realm className="org.apache.catalina.realm.JAASRealm"
appName="prova"
userClassNames="PlainUserPrincipal"
roleClassNames="PlainRolePrincipal"
/>
```

   ***NOTA1*:** Tomcat por defecto usa el "LockOutRealm"/"UserDatabaseRealm" (podéis verlo especificado en el server.xml justo debajo del nodo Engine). Este Realm está definido para usar el fichero tomcat-users.xml (situado en \conf) como base de datos de *usuarios; passwords; roles*. Queremos substituir este Realm por el nuestro (el JAASRealm), para ello **debemos comentar dicho Realm del server.xml** y dejar solo el JAASRealm que acabamos de configurar.
   ***NOTA2*:** Para ser más limpios, la configuración del JAASRealm se podría añadir en un fichero context.xml a situar en la carpeta \META-INF de la aplicación web, de esta forma, el Realm que utilizamos estaría ligado a la aplicación web y no afectaría a todo el servidor. De todas maneras, esto se comenta solo como mejora.
8. Para el ejemplo utilizaremos la aplicación web que viene por defecto con tomcat y que está situada en \webapps\examples\jsp\security\protected. Esta aplicación utiliza el

web.xml situado en \webapps\examples\WEB-INF como fichero de configuración. Dentro encontramos definido el siguiente security-constraint:

```
<security-constraint>
    <display-name>Example Security Constraint</display-name>
    <web-resource-collection>
      <web-resource-name>Protected Area</web-resource-name>
      <!-- Define the context-relative URL(s) to be protected -->
      <url-pattern>/jsp/security/protected/*</url-pattern>
      <!-- If you list http methods, only those methods are protected -->
      <http-method>DELETE</http-method>
      <http-method>GET</http-method>
      <http-method>POST</http-method>
      <http-method>PUT</http-method>
    </web-resource-collection>
    <auth-constraint>
      <!-- Anyone with one of the listed roles may access this area -->
      <role-name>tomcat</role-name>
      <role-name>role1</role-name>
    </auth-constraint>
  </security-constraint>

  <!-- Default login configuration uses form-based authentication -->
  <login-config>
    <auth-method>FORM</auth-method>
    <realm-name>Example Form-Based Authentication Area</realm-name>
    <form-login-config>
      <form-login-page>/jsp/security/protected/login.jsp</form-login-page>
      <form-error-page>/jsp/security/protected/error.jsp</form-error-page>
    </form-login-config>
  </login-config>

  <!-- Security roles referenced by this web application -->
  <security-role>
    <role-name>role1</role-name>
  </security-role>
  <security-role>
    <role-name>tomcat</role-name>
  </security-role>
```

En este code snippet se definen elementos importantes como: i) el security-constraint en sí que define el recurso protegido (etiqueta <url-pattern>), los diferentes métodos http a los que hay que aplicar seguridad y los roles que pueden acceder a dicho recurso; ii) el método de login que utilizaremos. En este caso se define el método FORM y se proporciona las webs para hacer el login y mostrar error; iii) los diferentes roles que utiliza la aplicación web.

Podemos usar directamente esta aplicación web de ejemplo con este security-constraint ya definido en su web.xml. No obstante, dado que hemos substituido el Realm por defecto por nuestro JAASRealm, cuando el security-constraint salte por acceso a recurso protegido, se utilizará nuestro sistema de autenticación (el PlainLoginModule que hemos implementado) para autorizar o denegar dicho acceso.

9. Arrancamos el servidor Tomcat utilizando el script startup.sh (situado en \bin).
10. Accedemos a la aplicación web poniendo en el browser: http://localhost:8080/examples/jsp/security/protected/

    Debería salir una pantalla donde podremos poner login/pass. Si ponemos login==pass accederemos a una página donde podremos comprobar que rol tenemos asignado. Como ya hemos dicho, el PlainLoginModule asigna el rol "tomcat" a cualquiera que quede autenticado, por lo tanto, si probamos *tomcat* nos dirá que tenemos asignado este rol, si probamos cualquier otro nombre de rol dirá que no lo tenemos asignado.