

# Tema 4

## Autenticación y autorización

J. Gutiérrez

Departament d'Informàtica  
Universitat de València

DAW-TS (ISAW).  
Curso 14-15

## Índice

- 1 *Introducción*
- 2 *Configuración de un JDBC Realm*
- 3 *Uso del realm en la aplicación web*

En las aplicaciones que se han visto hasta ahora no se ha pedido autenticación para el acceso a las mismas.

La idea es que para el acceso a determinados recursos (o a todos) el usuario deba proporcionar información de usuario y contraseña.

Esta es enviada al servidor y éste debe comprobar si coincide con la que tiene almacenada.

Hay varias formas de obtener la información desde el cliente:

- Delegando en el navegador para que muestre una ventana para recoger el usuario y contraseña.

Esta información se puede enviar al servidor de dos formas:

- ▶ Codificada en Base64 (no está encriptada)
  - ▶ Realizando un compendio del mensaje (por ejemplo MD5) y enviándolo al servidor.
- Mediante un formulario que envía el servidor al navegador.

## Security Realm

*Security Realm*: es el mecanismo mediante el cual el servidor almacena la información de usuarios y de grupos. Ejemplos:

- Fichero (File realm).
- Base de datos relacional (JDBC realm).
- LDAP
- Repositorio de certificados

Vamos a ver JDBC.

<http://docs.oracle.com/cd/E19226-01/820-7627/bnbxj/index.html>

## Índice

- 1 *Introducción*
- 2 *Configuración de un JDBC Realm*
- 3 *Uso del realm en la aplicación web*

*JDBC Realm*

En este caso la información se almacena en una base de datos.

```
CREATE DATABASE 'webappusers';  
use 'webappusers';
```

Hay que crear tres tablas: una que almacena usuarios, otra grupos y otra que las relaciona (en qué grupos está cada usuario).

La información mínima que debe contener la tabla de usuarios es:

```
CREATE TABLE 'users'(  
  'user_id' int(10) NOT NULL AUTO_INCREMENT,  
  'username' varchar(10) NOT NULL,  
  'password' char(32) NOT NULL,  
  PRIMARY KEY ('user_id')  
);
```

*JDBC Realm*

La tabla con la información de grupos:

```
CREATE TABLE 'groups'(  
  'group_id' int(10) NOT NULL AUTO_INCREMENT,  
  'group_name' varchar(20) NOT NULL,  
  PRIMARY KEY ('group_id')  
);
```

Y la tabla que los relaciona:

```
CREATE TABLE 'user_groups' (  
  'user_id' int(10) NOT NULL,  
  'group_id' int(10) NOT NULL,  
  PRIMARY KEY ('user_id','group_id')  
);
```



*JDBC Realm*

A continuación se crea una vista (que permite unir datos de las diferentes tablas) de la cual obtendrá la información Glassfish:

```
CREATE VIEW 'v_user_role' AS
  SELECT u.username, u.password, g.group_name
  FROM 'user_groups' ug
    INNER JOIN 'users' u on u.user_id = ug.user_id
    INNER JOIN 'groups' g ON g.group_id = ug.group_id;
```

*JDBC Realm*

```
INSERT INTO 'groups' ('group_name') VALUES  
('user'), ('admin');  
  
INSERT INTO 'users' ('username','password') VALUES  
('liskova','TGHUWF1sNbG4QrWV3+Ip0bpatyyZFgd19Jksw/0w9kk='),  
('ullmanova','bpVpH2WQ0piJSnEnL2DKqhEdUJYoxeEqU0bX95oZ1Yw=');  
  
INSERT INTO 'user_groups' ('user_id','group_id') VALUES  
(1,1),(2,1),(2,2);
```

Creamos un usuario y le damos permisos para acceder a las tablas de esta base de datos.

```
GRANT ALL ON webappusers.* TO 'webapp'@'localhost' identified by 'webapppasswd';
```

Este será el usuario con el que configuraremos GlassFish.

*JDBC Realm*

La valor que aparece en el campo password de la tabla usuarios es el resultado de aplicar un SHA-256 sobre la contraseña y a continuación codificar el resultado en Base64. El código que se ha utilizado para conseguir esto es:

```
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import org.apache.commons.codec.binary.Base64;

public class MesDig {
    public static void main(String[] args){
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            String text = args[0];
            md.update(text.getBytes("UTF-8"));
            byte[] digest = md.digest();
            System.out.println(Base64.encodeBase64String(digest));
        } catch (Exception ex) {
            System.out.println(ex.toString());
        }
    }
}
```

donde se ha usado la clase Base64 de Common Codecs de Apache:

*JDBC Realm*

Tras insertar estos valores vemos qué información contiene la vista que se ha creado anteriormente:

```
select * from v_user_role;
```

username	password	group_name
liskova	TGHUWFisNbG4QrWV3+Ip0bpatyyZFgd19Jksw/0w9kk=	user
ullmanova	bpVpH2WQ0piJSnEnL2DKqhEdUJYoxeEqU0bX95oZ1Yw=	user
ullmanova	bpVpH2WQ0piJSnEnL2DKqhEdUJYoxeEqU0bX95oZ1Yw=	admin

*JDBC Realm*

## Pasos para la configuración en Glassfish:

- 1 Creación de un pool de conexiones con el nombre `userspool` a la base de datos.
- 2 Creación de un nuevo recurso JDBC accesible vía JNDI con el nombre `jdbc/userspool`.
- 3 Creación de un nuevo dominio JDBC de seguridad llamado `jdbcRealm` que usa el recurso JDBC.

## JDBC Realm: Creación del pool de conexiones

- Módulos de Ciclo de Vida
- Datos de Supervisión
- Recursos
  - JDBC
    - Recursos JDBC
    - Pools de Conexiones JDBC**
  - Conectores
  - Configuraciones de Adaptador
  - Recursos de JMS
  - Sesiones JavaMail
  - JNDI
- Ajuste de Rendimiento
- Configuraciones
  - default-config
  - server-config**
    - Configuración de JVM
    - Configuración de Registrado
    - Contenedor Web
    - Contenedor EJB
    - Java Message Service

### Nuevo Pool de Conexiones JDBC (Paso 1 de 2)

Identifique la configuración general del pool de conexiones.

#### Configuración General

**Nombre de Pool:** \*

**Tipo de Recurso:**

**Proveedor de Controladores de la Base de Datos:**

**Introspección:**

☐ **Activada**

Se debe indicar si la clase de origen de

Seleccionar o introducir un proveedor de

Si está activado, los nombres de clase i introspección.

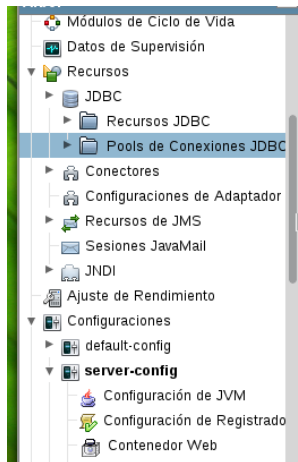
## JDBC Realm: Creación del pool de conexiones

Número total de actualizaciones disponibles: 1

- Módulos de Ciclo de Vida
- Datos de Supervisión
- Recursos
  - JDBC
    - Recursos JDBC
    - Pools de Conexiones JDBC**
    - Conectores
    - Configuraciones de Adaptador
    - Recursos de JMS
    - Sesiones JavaMail
    - JNDI
  - Ajuste de Rendimiento
  - Configuraciones
    - default-config
    - server-config**
  - Configuración de JVM

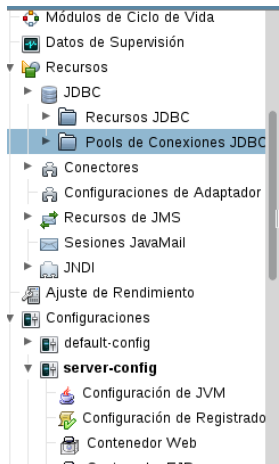
<input type="checkbox"/>	ConnectionLifecycleInterceptors		
<input type="checkbox"/>	ConnectTimeout		0
<input type="checkbox"/>	ContinueBatchOnError		true
<input type="checkbox"/>	CreateDatabaseIfNotExist		false
<input checked="" type="checkbox"/>	DatabaseName		webappusers
<input type="checkbox"/>	DefaultAuthenticationPlugin		com.mysql.jdbc
<input type="checkbox"/>	DefaultFetchSize		0
<input type="checkbox"/>	DetectCustomCollations		false
<input type="checkbox"/>	DetectServerPreparedStmts		
<input type="checkbox"/>	DisabledAuthenticationPlugins		
<input type="checkbox"/>	DisconnectOnExpiredPasswords		true
<input type="checkbox"/>	DontCheckOnDuplicateKeyUpdateInSQL		false
<input type="checkbox"/>	DontTrackOpenResources		false
<input type="checkbox"/>	DumpMetadataOnColumnNotFound		false
<input type="checkbox"/>	DumpQueriesOnException		false

## JDBC Realm: Creación del pool de conexiones



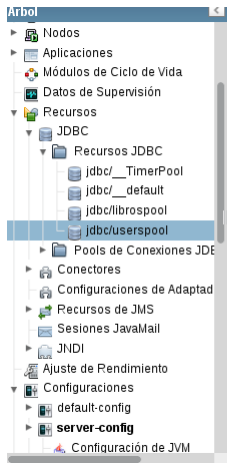
<input type="checkbox"/>	NetTimeoutForStreamingResults	600
<input type="checkbox"/>	NoAccessToProcedureBodies	false
<input type="checkbox"/>	NoDatetimeStringSync	false
<input type="checkbox"/>	NoTimezoneConversionForTimeType	false
<input type="checkbox"/>	NullCatalogMeansCurrent	true
<input type="checkbox"/>	NullNamePatternMatchesAll	true
<input type="checkbox"/>	OverrideSupportsIntegrityEnhancementFacility	false
<input type="checkbox"/>	PacketDebugBufferSize	20
<input type="checkbox"/>	PadCharsWithSpace	false
<input type="checkbox"/>	Paranoid	false
<input type="checkbox"/>	ParseInfoCacheFactory	com.mysql.jdbc.PerConnection
<input type="checkbox"/>	Password	webapppasswd
<input type="checkbox"/>	PasswordCharacterEncoding	
<input type="checkbox"/>	Pedantic	false
<input type="checkbox"/>	PinGlobalTxToPhysicalConnection	false
<input type="checkbox"/>	PopulateInsertRowWithDefaultValues	false
<input type="checkbox"/>	Port	3306
<input type="checkbox"/>	PortNumber	3306



*JDBC Realm: Creación del pool de conexiones*

<input type="checkbox"/>	UseJDBCCompliantTimezoneShift	false
<input type="checkbox"/>	UseJvmCharsetConverters	false
<input type="checkbox"/>	UseLegacyDatetimeCode	true
<input type="checkbox"/>	UseLocalSessionState	false
<input type="checkbox"/>	UseLocalTransactionState	false
<input type="checkbox"/>	UseNanosForElapsedTime	false
<input type="checkbox"/>	UseOldAliasMetadataBehavior	false
<input type="checkbox"/>	UseOldUTF8Behavior	false
<input type="checkbox"/>	UseOnlyServerErrorMessages	true
<input checked="" type="checkbox"/>	User	webapp
<input type="checkbox"/>	UseReadAheadInput	true
<input type="checkbox"/>	UseServerPreparedStmts	false
<input type="checkbox"/>	UseServerPrepStmts	false
<input type="checkbox"/>	UseSqlStateCodes	true
<input type="checkbox"/>	UseSSL	false
<input type="checkbox"/>	UseSSPSCompatibleTimezoneShift	false
<input type="checkbox"/>	UseStreamLengthsInPrepStmts	true
<input type="checkbox"/>	UseTimezone	false

## JDBC Realm: Creación del recurso JDBC



### Editar Recurso JDBC

Edite un origen de datos JDBC existente.

[Cargar Valores por Defecto](#)

**Nombre JNDI:** jdbc/userspool

**Nombre de Pool:**

Use la página [Pools de Conexiones JDBC](#) para crear pools nuevos

**Descripción:**

**Estado:** ☒ Activada

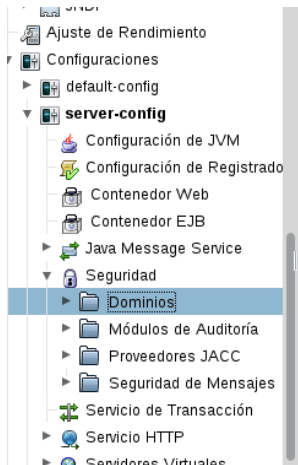
#### Propiedades Adicionales (0)

[Agregar Propiedad](#)

[Suprimir Propiedades](#)

Nombre	Valor
No se han encontrado elementos.	

## JDBC Realm: Creación del dominio JDBC de seguridad



### Dominios

Cree, modifique o suprima dominios de seguridad (autenticación).

**Nombre de Configuración:** server-config

#### Dominios (3)

<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="Nuevo..."/>	<input type="button" value="Suprimir"/>
	Nombre		Nombre de Clase
<input type="checkbox"/>	admin-realm		com.sun.enterprise.security.auth.realm.file.FileRealm
<input type="checkbox"/>	certificate		com.sun.enterprise.security.auth.realm.certificate.CertificateRealm
<input type="checkbox"/>	file		com.sun.enterprise.security.auth.realm.file.FileRealm

## JDBC Realm: Creación del dominio JDBC de seguridad



**Nombre de Configuración:** server-config

**Nombre de Dominio:** jdbcRealm

**Nombre de Clase:** com.sun.enterprise.security.auth.realm.jdbc.JDBCRealm

### Propiedades específicas de esta clase

**Contexto JAAS:** \*

Identificador del módulo de conexión que se utiliza

**JNDI:** \*

Nombre de JNDI del recurso de JDBC utilizado

**Tabla de Usuarios:** \*

Nombre de la tabla de base de datos que contiene

**Columna de Nombres de Usuario:** \*

Nombre de la columna de la tabla de usuarios o

**Columna de Contraseñas:** \*

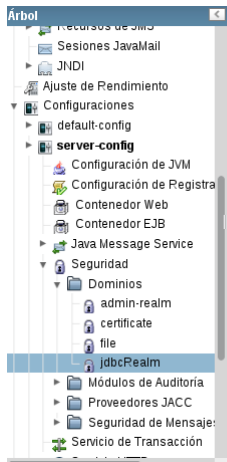
Nombre de la columna de la tabla de usuarios o

**Tabla de Grupos:** \*

Nombre de la tabla de base de datos que contiene

**Columna de Nombres de Usuario de la Tabla de Grupos:**

## JDBC Realm: Creación del dominio JDBC de seguridad



**Tabla de Grupos:** \*

Nombre de la tabla de base de datos que contiene la lista de lo

**Columna de Nombres de Usuario de la Tabla de Grupos:**

Nombre de la columna de la tabla de grupos de usuarios que c

**Columna de Nombres de Grupo:** \*

Nombre de la columna de la tabla de grupos que contiene la lis

**Asignar Grupos:**

Lista separada por comas de nombres de grupo

**Usuario de Base de Datos:**

Especifique el nombre de usuario de la base de datos del dom

**Contraseña de Base de Datos:**

Especifique la contraseña de la base de datos del dominio en l

**Algoritmo de Resumen:**

Algoritmo de resumen (el valor por defecto es SHA-256); tenga i  
GlassFish anteriores a la 3.1

**Algoritmo de Cifrado de Contraseña:**

Esto indica el algoritmo para el cifrado de las contraseñas en la  
de seguridad.

**Codificación:**

Codificación (los valores permitidos son Hex y Base64)

**Juego de Caracteres:**

Juego de caracteres del algoritmo de resumen

*Titulo*

En este caso se ha indicado que

## Índice

- 1 *Introducción*
- 2 *Configuración de un JDBC Realm*
- 3 *Uso del realm en la aplicación web*

## *Principal*

Representa un usuario

## *Role*

Sirve para definir niveles de acceso

Los roles se asocian a usuarios y a grupos usando ficheros de configuración que dependen del tipo de servidor.

El desarrollador debe especificar qué roles pueden acceder a una funcionalidad determinada de la aplicación.

Más información sobre usuarios, grupos y roles:

[http:](http://docs.oracle.com/cd/E19226-01/820-7627/bnbxj/index.html)

[//docs.oracle.com/cd/E19226-01/820-7627/bnbxj/index.html](http://docs.oracle.com/cd/E19226-01/820-7627/bnbxj/index.html)



La asociación en Glassfish se realiza en el fichero `glassfish-web.xml`:

```
<security-role-mapping>
  <role-name>registered_users</role-name>
  <group-name>user</group-name>
</security-role-mapping>

<security-role-mapping>
  <role-name>administrators</role-name>
  <group-name>admin</group-name>
</security-role-mapping>
```

## Índice

### 3 *Uso del realm en la aplicación web*

- Autenticación
- Autorización

La configuración de la autenticación de forma declarativa se realiza en el fichero `web.xml`:

```
<login-config>
  <auth-method> <!--Seleccionar uno-->
    BASIC| DIGEST|FORM|CLIENT-CERT
  </auth-method>
  <realm-name> <!-- Nombre que se ha dado al realm en Glassfish -->
    Nombre del realm configurado en Glassfish
  </realm-name>
  <form-login-config>
    <form-login-page>
      url de la pagina de login
    </form-login-page>
    <form-error-page>
      url de la pagina de error si falla el login
    </form-error-page>
  </form-login-config>
</login-config>
```

Cuando se usa FORM hay que proporcionar una página que contenga un formulario donde se solicite el usuario y contraseña al usuario

```
<!-- Pagina login.jsp -->
<html>
  ...
  <form method="POST" action="j_security_check">
    <input type="text" name="j_username"/>
    <input type="password" name="j_password"/>
    <input type="submit" value="Enviar"/>
  </form>
  ...
</body>
</html>
```

`j_security_action` es el módulo que comprueba si los valores pasados en `j_username` y en `j_password` son válidos. `j_security_check` es un componente que proporciona Java EE (no lo tenemos que programar nosotros).

Y en la configuración de la autenticación en el fichero web.xml:

```
<login-config>
  <auth-method>
    FORM
  </auth-method>
  <realm-name> <!-- Nombre que se ha dado al realm en glassfish -->
    app1-jdbc-realm
  </realm-name>
  <form-login-config>
    <form-login-page>
      login.jsp
    </form-login-page>
    <form-error-page>
      error.jsp
    </form-error-page>
  </form-login-config>
</login-config>
```

## Autenticación vía código:

La interfaz `HttpServletRequest` tiene varios métodos para permitir la autenticación:

```
// Permite realizar el login desde el código  
void login(String user, String password)
```

```
// Cuando se ejecute forzará al cliente a proporcionar  
// un usuario y password  
void authenticate()
```

## Índice

### 3 *Uso del realm en la aplicación web*

- Autenticación
- Autorización

Autorización: regula a qué funcionalidad puede acceder un usuario en función de su rol.

Se puede conseguir de varias formas:

- De forma declarativa
- Desde el código
- Combinación declarativa y en el código



## Autorización declarativa en el fichero web.xml

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Nombre</web-resource-name>
    <url-pattern>Patron URL</url-pattern>
    <http-method>GET|POST|PUT|...</http-method> <!-- Varias veces -->
    <http-method-omission>GET|POST|PUT|...</http-method-omission>
  </web-resource-collection>
  <auth-constraint>
    <role-name>Nombre del rol declarado anteriormente</role-name>
  </auth-constraint>
  <user-data-constraint>
    <!-- CONFIDENTIAL e INTEGRAL implican usar HTTPS -->
    <transport-guarantee>CONFIDENTIAL|INTEGRAL|NONE</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

Si el cliente solicita una URL que encaja con el patrón URL especificado y no está autenticado entonces se le pedirá la autenticación (según lo configurado en el elemento login-config). A continuación se le comprobará si pertenece al rol indicado. Si no pertenece entonces se le devolverá un mensaje HTTP con código 403 indicando que no está autorizado.

*Ejemplo: autorización declarativa en XML*

## Ejemplo de autorización declarativa en el fichero web.xml

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Acceso a tienda</web-resource-name>
    <url-pattern>/tienda/*</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>
  <auth-constraint>
    <role-name>registered_users</role-name>
  </auth-constraint>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

## Autorización declarativa usando anotaciones

- `@ServletSecurity` equivalente a `<security-constraint>`
- `@HTTPConstraint` para indicar todos los métodos HTTP
- `@HTTPMethodConstraint` equivalente a `<http-method>`
- `@DeclareRoles`(lista de roles) equivalente a `<security-role>`
- `@RolesAllowed` (nombre del rol) para especificar el rol que puede usar un determinado método.

## *Ejemplo de autorización declarativa con anotaciones*

```
@WebServlet("/tienda/productos")
@ServletSecurity(@HttpConstraint(rolesAllowed={"registered_users"}))
public class Productos extends HttpServlet{
    ...
}
```

```
@WebServlet("/tienda/productos")
@ServletSecurity(
    httpMethodConstraints={
        @HttpMethodConstraint("GET"),
        @HttpMethodConstraint("POST",rolesAllowed={"registered_users"})
    }
)
public class Productos extends HttpServlet{
    ...
}
```

```
@WebServlet("/tienda/productos")
@ServletSecurity(
    value=@HttpConstraint(
        transportGuarantee=ServletSecurity.TransportGuarantee.CONFIDENTIAL),
    httpMethodConstraints={
        @HttpMethodConstraint(value="TRACE", transportGuarantee=ServletSecurity.
            TransportGuarantee.NONE)
    })
public class Productos extends HttpServlet {
    ...
}
```

El protocolo debe ser HTTPS excepto para el método TRACE que no requiere HTTPS.

Más ejemplos en:

[https://blogs.oracle.com/swchan/entry/follow\\_up\\_on\\_servlet\\_3](https://blogs.oracle.com/swchan/entry/follow_up_on_servlet_3)

## Autorización desde el código

La autorización desde el código se puede realizar usando los siguientes métodos que están declarados en la clase `HttpServletRequest`:

```
// Devuelve el login del usuario que realiza la petición o null si no está autenticado  
String getRemoteUser()  
  
// Devuelve java.security.Principal que contiene el nombre del usuario autenticado  
Principal getUserPrincipal  
  
// Comprueba si el usuario autenticado pertenece a un determinado rol  
boolean isUserInRole(String rol)
```

## *Ejemplo de autorización desde el código*

```
@WebServlet("/tienda/productos")
@DeclareRoles("registered_users")
public class Productos extends HttpServlet{
    public void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ...{

        if (isUserInRole("registered_users"){
            // Acción
        }

    }
}
```