

Reino de Base de datos (SQL)

{scrollbar}

En esta sección nos enfocaremos en el uso de una base de datos para la verificación y recuperación de nombre de usuario y contraseñas.

Para este ejemplo hemos creado una nueva base de datos llamada **BaseDeDatosDeSeguridad**, dentro de la infraestructura Derby incluida. Los siguientes pasos resumen al procedimiento empleado: crear la base de datos y sus tablas, cargar datos ejemplo y crear al pool de conexiones. Instrucciones detalladas en como definir pools de conexiones a bases de datos se describen en la sección [Configurando pools de bases de datos](#).

Crear base de datos y cargar datos ejemplo

- En el menú **Console Navigation** (*Navegación de Consola*) a mano izquierda, haz clic en **Database Manager** (*Administrador de Bases de datos*).
- Ingresa **BaseDeDatosDeSeguridad** en el campo **Create DB:** (*Crear BD:*) y haz clic en **Create** (*Crear*).
- Selecciona **BaseDeDatosDeSeguridad** en el menú desplegable **Use DB:** (*Usa BD:*) e ingresa los siguientes comandos para después dar clic en **Run SQL** (*Ejecutar SQL*).

```
create table usuarios
(nombre_usuario varchar(15),
password varchar(15));
create table grupos
(nombre_usuario varchar(15),
nombre_grupo varchar(15));
insert into usuarios values('usuario_uno', 'p1');
insert into usuarios values('usuario_dos', 'p2');
insert into usuarios values('usuario_tres', 'p3');
insert into grupos values('usuario_uno', 'admin');
insert into grupos values('usuario_dos', 'admin');
insert into grupos values('usuario_tres', 'usuario');
```

Crear pool de conexiones

- En el menú **Console Navigation** (*Navegación de Consola*) a mano izquierda, haz clic en **Database Pools** (*Pools de Base de Datos*).
- Haz clic en **Using the Geronimo database pool wizard** (*Usando ayudante de Geronimo en pool de base de datos*).
- Ingresa **BaseDeDatosDeSeguridad** como el nombre del pool de base de datos.
- Selecciona **Derby embedded** (*Derby embebido*) del menú desplegable de tipo de pool de base de datos y haz clic en **Next** (*Siguiente*).
- Verifica que la clase controladora JDBC sea **org.apache.derby.jdbc.EmbeddedDriver**.
- Del menú desplegable **Driver JAR** (*JAR Controlador*), selecciona **org.apache.derby/derby/10.1.1.0/jar**.
- Deja **vacíos** al nombre de usuario y password de conexión.
- Ingresa **BaseDeDatosDeSeguridad** como el nombre de la base de datos y haz clic en **Next** (*Siguiente*).
- Haz clic en **Test Connection** (*Probar Conexión*).
- Haz clic en **Deploy** (*Activar*).

Agregar un nuevo reino de seguridad

Para crear un nuevo reino de seguridad, haz clic en **Add new security realm** (*Agregar nuevo reino de seguridad*) desde el portlet **Security Realms** (*Reinos de Seguridad*).

The screenshot shows a web-based console window titled "Security Realms" with a "[view]" link in the top right. The main content area is titled "Create Security Realm -- Step 1: Select Name and Type". It contains two main input sections: "Name of Security Realm:" with a text input field containing "reino_de_seguridad_derby" and a descriptive note below it stating "A name that is different than the name for any other security realms in the server (no spaces in the name please). Other components will use this name to refer to the security realm."; and "Realm Type:" with a dropdown menu currently showing "Database (SQL) Realm" and another descriptive note below it stating "The type of login module used as the master for this security realm. Select 'Other' for manual configuration options including custom login modules and realms that use multiple login modules to populate user principals.". At the bottom left of the form area is a "Cancel" link, and at the bottom center is a "Next" button.

Ingresa **reino_de_seguridad_derby** en el campo **Name of Security Realm:** (*Nombre del Reino de Seguridad:*) y selecciona **Database (SQL) Realm** (*Reino de Base de datos (SQL)*) del menú desplegable **Realm type:** (*Tipo de Reino:*), para después hacer clic en **Next** (*Siguiente*).

La siguiente pantalla configura al modulo de login. Los primeros dos campos que necesites llenar variarán potencialmente, dependiendo del tipo de base de datos. En este caso, estamos empleando una base de datos Derby embebida, por lo que los User SELECT SQL (*select SQL de Usuario*) y Group SELECT SQL (*select SQL de Grupo*) deberían leerse como sigue:

User SELECT SQL: `select nombre_usuario, password from APP.usuarios where nombre_usuario=?`

Group SELECT SQL: `select nombre_usuario, nombre_grupo from APP.grupos where nombre_usuario=?`

Nota que **APP** es el esquema predeterminado para la base de datos Derby embebida, y requiere preceder la tabla en el enunciado SQL. Estos enunciados suelen ser distintos de una base de datos a otra, por ejemplo, este procedimiento también se probó con DB2, donde los enunciados SQL fueron:

User SELECT SQL: `select nombre_usuario, password from usuarios where nombre_usuario=?`

Group SELECT SQL: `select nombre_usuario, nombre_grupo from grupos where nombre_usuario=?`

Al haber definido los enunciados SQL para la recuperación usuarios y grupos, necesitas elegir del menú desplegable **Database Pool** (*Pool de Base de datos*) al pool de conexiones a la base de datos que creaste en el paso previo. Agrega los valores requeridos como se muestra a continuación y haz clic en **Next** (*Siguiente*).

Database Pool: BaseDeDatosDeSeguridad

JDBC Driver Class: `org.apache.derby.jdbc.EmbeddedDriver`

Driver JAR: `org.apache.derby/derby/10.1.1.0/jar`

JDBC URL: `jdbc:derby:BaseDeDatosDeSeguridad`

[\[view\]](#)

Security Realms

Create Security Realm -- Step 2: Configure Login Module

User SELECT SQL:

A SQL statement to load user/password information. It should return 2 columns, the first holding a username and the second holding a password. The statement may use the PreparedStatement syntax of ? for a parameter, in which case the username will be set for every parameter. A typical setting would be `SELECT username, password FROM app_users WHERE username=?`

Group SELECT SQL:

A SQL statement to load group information for a user. It should return 2 columns, the first holding a username and the second holding a group name. The statement may use the PreparedStatement syntax of ? for a parameter, in which case the username will be set for every parameter. A typical setting would be `SELECT username, group_name FROM user_groups WHERE username=?` or for a more normalized schema, `SELECT u.username, g.name FROM app_users u, groups g, user_groups ug WHERE ug.user_id=users.id AND ug.group_id=g.id AND u.username=?`

A SQL security realm must either have a database pool or JDBC connectivity settings to connect to the database. Please select EITHER the database pool, OR the rest of the JDBC settings.

Database Pool:

A database pool that the login module will use to connect to the database. If this is specified, none of the rest of the settings after this are necessary.

JDBC Driver Class:

The fully-qualified JDBC driver class name. This driver must be located in the JAR specified in the next field.

Driver JAR:

The JAR holding the selected JDBC driver. Should be installed under `GERONIMO/repository/` to appear in this list.

JDBC URL:

The JDBC URL that specifies the details of the database to connect to. This has a different form for each JDBC driver.

JDBC Username:

The username used to connect to the database

JDBC Password:

The password used to connect to the database

[Cancel](#)

El siguiente paso te dejará permitir la auditoría para el monitor de intentos de login mediante este reino. En este paso también puedes configurar al bloqueo de cuenta, basado en la cantidad de intentos fallidos en acceso, dentro de un periodo de tiempo. La última opción en este paso, **Store Password** (*Guardar Password*), al estar habilitada permitirá al reino el guardar la password del usuario en una credencial privada en el tema.

Security Realms
[view]

Create Security Realm -- Step 3: Advanced Configuration

Enable Auditing: Log File:
 If enabled, every login attempt will be recorded to the specified file. The path should be relative to the Geronimo home directory (a typical value would be var/log/login-attempts.log).

Enable Lockout: Lock a user after failures within seconds and keep the account locked for seconds.
 If enabled, a certain number of failed logins in a particular time frame will cause a user's account to be locked for a certain period of time. This is a defense against brute force account cracking attacks.

Store Password:
 If enabled, the realm will store each user's password in a private credential in the Subject. This will allow access to the password later after the login process has completed. This is not normally required.

[Cancel](#)

En este punto, tienes configurado a tu nuevo reino de seguridad, el siguiente paso es probarlo y después activarlo. Haz clic en **Test a Login** (*Probar un Login*).

Ingresa un nombre de usuario y contraseña válidos para ser recuperado de la base de datos, después haz clic en **Next** (*Siguiente*).

Security Realms
[view]

Create Security Realm -- Step 4: Test Login

From here you can enter a username and password for the main login module in the realm, and see if the login is successful and which Principals are generated for the user. This is meant to be an indication of whether the settings for the main login module are correct. It does not invoke advanced features such as auditing or lockout.

Username:
 The username to use to log in to the realm.

Password:
 The password to use to log in to the realm.

[Cancel](#)

Deberías recibir un mensaje de confirmación indicando que el login fue exitoso; haz clic en **Deploy Realm** (*Activar Reino*) para cargar esta configuración al servidor.

Security Realms
[view]

Create Security Realm -- Step 5: Login Results

Test Results: Login succeeded with 2 principals

Principals: usuario_uno org.apache.geronimo.security.realm.providers.GeronimoUserPrincipal
 admin org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal

[Cancel](#)

Ya cuentas con un nuevo, completamente configurado, reino de seguridad que reconoce nombres de usuario y passwords de la base de datos Derby que creaste.

El siguiente ejemplo te muestra al plan de activación para este reino de seguridad. Como alternativa a la Consola de Administración de Geronimo, puedes guardar este ejemplo en un archivo (por ej. reino_de_seguridad_derby.xml) y activarlo con la [Herramienta de activación](#) mediante la ejecución del comando siguiente:

```
<geronimo_home>\bin\deploy --user system --password manager deploy <realm_path>\reino_de_seguridad_derby.xml
```

```
xmlsolidreino_de_seguridad_derby <module xmlns="http://geronimo.apache.org/xml/ns/deployment-1.1"> <environment> <moduleId> <groupId>console</groupId> <artifactId>realm-reino_de_seguridad_derby</artifactId> <version>1.0</version> <type>car</type> </moduleId> <dependencies> <dependency> <groupId>geronimo</groupId> <artifactId>j2ee-security</artifactId> <type>car</type> </dependency> </dependencies> </environment> <gbean name="reino_de_seguridad_derby" class="org.apache.geronimo.security.realm.GenericSecurityRealm"> <attribute name="realmName">reino_de_seguridad_derby</attribute> <reference name="ServerInfo"> <name>ServerInfo</name> </reference> <reference name="LoginService"> <name>JaasLoginService</name> </reference> <xml-reference name="LoginModuleConfiguration"> <log:login-config xmlns:log="http://geronimo.apache.org/xml/ns/login-module-configuration-1.1">
```

```
org/xml/ns/loginconfig-1.1"> <log:login-module control-flag="REQUIRED" server-side="true" wrap-principals="false"> <log:login-domain-
name>reino_de_seguridad_derby</log:login-domain-name> <log:login-module-class>org.apache.geronimo.security.realm.providers.SQLLoginModule<
/og:login-module-class> <log:option name="jdbcDriver">org.apache.derby.jdbc.EmbeddedDriver</log:option> <log:option name="userSelect">select
nombre_usuario, password from APP.usuarios where nombre_usuario=?</log:option> <log:option name="groupSelect">select nombre_usuario,
nombre_grupo from APP.grupos where nombre_usuario=?</log:option> <log:option name="jdbcURL">jdbc:derby:BaseDeDatosDeSeguridad</log:option>
</log:login-module> <log:login-module control-flag="OPTIONAL" server-side="true" wrap-principals="false"> <log:login-domain-
name>reino_de_seguridad_derby-Audit</log:login-domain-name> <log:login-module-class>org.apache.geronimo.security.realm.providers.
FileAuditLoginModule</log:login-module-class> <log:option name="file">var/log/reino_de_seguridad.log</log:option> </log:login-module> <log:login-
module control-flag="REQUIRE" server-side="true" wrap-principals="false"> <log:login-domain-name>reino_de_seguridad_derby-Lockout</log:login-
domain-name> <log:login-module-class>org.apache.geronimo.security.realm.providers.RepeatedFailureLockoutLoginModule</log:login-module-class>
<log:option name="lockoutDurationSecs">60</log:option> <log:option name="failurePeriodSecs">10</log:option> <log:option name="failureCount">5</log:
option> </log:login-module> </log:login-config> </xml-reference> </gbean> </module>
```