UT-05. Autenticación por formularios

1. Autenticación por formularios

Consta de los siguientes pasos:

1. Configurar el Security Realm
2. Configurar el repositorio de seguridad (JDBC en este ejemplo)
3. Establecer la configuración de seguridad de la aplicación en web.xml
4. Establecer grupos y roles en web.xml
5. Habilitar protocolo de comunicaciones seguro HTTPS en web.xml

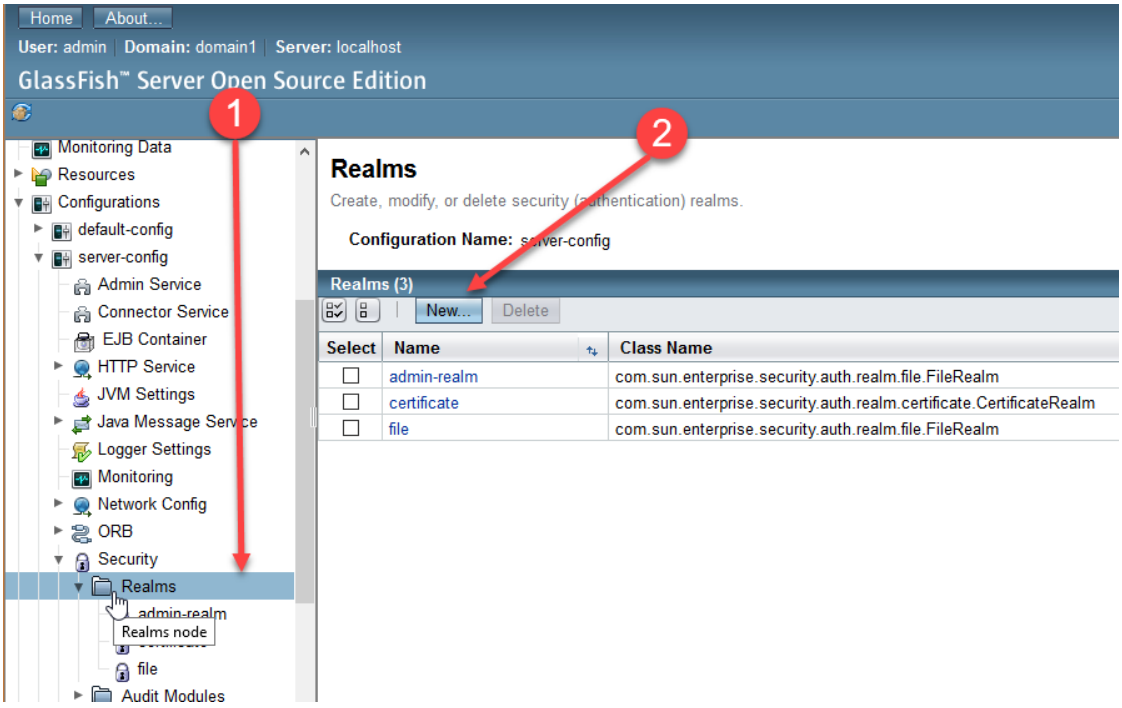
2. Crear Realm JDBC en Glassfish

2.1 Prerrequisito

Se ha creado un recurso JDBC de nombre JNDI jdbc/sakila.

2.2 Crear Realm

1. **Configurations > server-config > Security > Realms**
2. **Pulsar "New..."**



2.2.1 Definir las propiedades del jdbcRealm

Properties specific to this Class

JAAS Context: *	jdbcRealm Identifier for the login module to use for this realm
JNDI: *	jdbc/sakila JNDI name of the JDBC resource used by this realm
User Table: *	users Name of the database table that contains the list of authorized users for this realm
User Name Column: *	user_name Name of the column in the user table that contains the list of user names
Password Column: *	password Name of the column in the user table that contains the user passwords
Group Table: *	membership Name of the database table that contains the list of groups for this realm
Group Table User Name Column:	user_name Name of the column in the user group table that contains the list of groups for this realm
Group Name Column: *	group_name Name of the column in the group table that contains the list of group names
Password Encryption Algorithm: *	none This denotes the algorithm for encrypting the passwords in the database. It is a security risk to leave this field empty.
Assign Groups:	 Comma-separated list of group names
Database User:	2dawa Specify the database user name in the realm instead of the JDBC connection pool
Database Password:	2dawA2!06 Specify the database password in the realm instead of the JDBC connection pool
Digest Algorithm:	 Digest algorithm (default is SHA-256); note that the default was MD5 in GlassFish versions prior to 3.1
Encoding:	 Encoding (allowed values are Hex and Base64)
Charset:	 Character set for the digest algorithm

3. Establecer configuración de seguridad de la aplicación

3.1 Definir mecanismo de login (web.xml)

Definimos autenticación por formularios (FORM), asociada al real de Glassfish (sakila-realm) y también definimos las páginas JSF que servirán para el login y error de login.

```
<login-config>
    <auth-method>FORM</auth-method>
    <realm-name>sakila-realm</realm-name>
    <form-login-config>
        <form-login-page>/login.xhtml</form-login-page>
        <form-error-page>/error.xhtml</form-error-page>
    </form-login-config>
</login-config>
```

3.2 Definir los roles que tendrá la aplicación

El rol es una agrupación de permisos sobre la aplicación (gestionar elementos de la base de datos), operaciones sobre registros de alquiler-ventas, etc.

Definimos los roles: **ADMIN, MANAGER y PUBLIC**

```
<security-role>
    <description/>
    <role-name>ADMIN</role-name>
</security-role>
<security-role>
    <description/>
    <role-name>MANAGER</role-name>
</security-role>
<security-role>
    <description/>
    <role-name>PUBLIC</role-name>
</security-role>
```

3.3 Definir control de acceso sobre zonas de la aplicación (protegemos recursos según patrón de URL)

```
<security-constraint>
    <display-name>PrivateAreaConstraint</display-name>
    <web-resource-collection>
        <web-resource-name>privatecoll</web-resource-name>
        <description>Área privada</description>
        <url-pattern>/faces/staff/*</url-pattern>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
    </web-resource-collection>
    <auth-constraint>
        <description/>
        <role-name>MANAGER</role-name>
        <role-name>ADMIN</role-name>
    </auth-constraint>
    <user-data-constraint>
        <description/>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

Definimos staff como zona protegida, a la cual tendrán acceso los roles **MANAGER** y **ADMIN.**

Además definimos el transporte HTTP necesario como CONFIDENTIAL (SSL).

3.4 Asociamos roles con grupos de seguridad del Realm de Glassfish (glassfish-web.xml)

Los grupos de usuarios definidos en las tablas de base de datos, los vamos a vincular con roles de la aplicación, utilizando el descriptor específico de Glassfish **glassfish-web.xml**

Grupo		**Role**
sysops	-->	ADMIN
staff	-->	MANAGER
customers	-->	PUBLIC

```
<security-role-mapping>
  <role-name>ADMIN</role-name>
  <group-name>sysops</group-name>
</security-role-mapping>
<security-role-mapping>
  <role-name>MANAGER</role-name>
  <group-name>staff</group-name>
</security-role-mapping>
<security-role-mapping>
  <role-name>PUBLIC</role-name>
  <group-name>customers</group-name>
</security-role-mapping>
```

3.5 Creación de la página de autenticación **login.xhtml**

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE     html     PUBLIC     "-//W3C//DTD     XHTML     1.0     Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Login Form</title>
    </h:head>
    <h:body>
        <h2>Hello, please log in:</h2>
        <form name="loginForm" method="POST" action="j_security_check">
            <p><strong>Please type your user name: </strong>
                <input type="text" name="j_username" size="25"/></p>
            <p><strong>Please type your password: </strong>
                <input type="password" size="15" name="j_password"/></p>
            <p>
                <input type="submit" value="Submit"/>
                <input type="reset" value="Reset"/></p>
        </form>
    </h:body>
</html>
```

3.6 Creación de la página error.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE     html     PUBLIC     "-//W3C//DTD     XHTML     1.0     Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:p="http://primefaces.org/ui">
    <h:head>
        <title>Login Error</title>
    </h:head>
    <h:body>
        <h2>Invalid user name or password.</h2>
        <p:messages/>

        <p>Please enter a user name or password that is authorized to access this
            application. For this application, this means a user that has been
            created in the <code>file</code> realm and has been assigned to the
            <em>group</em> of <code>TutorialUser</code>.</p>
        <h:link outcome="login">Return to login page</h:link>

    </h:body>
</html>
```

4. Anexo I. Estructura de base de datos

```
use sakila;

SET foreign_key_checks = 0;

drop table IF exists users_groups;
drop table IF exists users;
drop table IF exists groups;

create table users (
      user_id smallint primary key
    , user_name varchar(50) not null
    , password varchar(256) not null
);
alter table users add constraint AK_users_username unique(user_name);

create table groups
(
   group_id int primary key auto_increment
   , group_name varchar(50) not null
   , description varchar(300)
);
alter table groups add constraint AK_groups_groupname unique(group_name);

create table users_groups
(
  group_id int not null,
  user_name varchar(50) not null
);
alter table users_groups add primary key (group_id, user_name);

alter table users_groups add constraint FK_usersgroups_users foreign key (user_name)
references users(user_name);

alter table users_groups add constraint FK_usersgroups_groups foreign key (group_id)
references groups(group_id);

insert into users (user_id, user_name, password)
select customer_id, substring_index(email, '@', 1), sha2('temporal', 256) from
customer;

insert into groups (group_name, description) values ('sysops', 'Grupo de
administradores');
insert into groups (group_name, description) values ('staff', 'Personal de la
tienda');
insert into groups (group_name, description) values ('customers', 'Grupo de
clientes');

insert into users_groups (group_id, user_name) values (1, 'MARY.SMITH');
insert into users_groups (group_id, user_name) values (1, 'LUIS.YANEZ');
insert into users_groups (group_id, user_name) values (2, 'LUIS.YANEZ');
insert into users_groups (group_id, user_name) values (2, 'MIKE.WAY');
```

```
insert into users_groups (group_id, user_name) values (3, 'TONY.CARRANZA');
insert into users_groups (group_id, user_name) values (3, 'SARA.PERRY');

create view membership as
select ug.user_name, ug.group_id, g.group_name from
groups g inner join users_groups ug on g.group_id=ug.group_id
inner join users u on ug.user_name=u.user_name
```