Master 2 MIASHS
2019-2020
Jérôme DAVID

TP3 Java EE

Authentication

In order to secure the access to our application, we will see how to hash passwords and how to manage authentication in TomEE.

Password hashing

For information for password hashing refer to the other document or search on the web.

To be able to handle password hashing we use the following injection:

```
@Inject
private transient Pbkdf2PasswordHash hashAlgo;
```

It can hash password:

```
hashAlgo.generate(“myPassord”.toCharArray())
```

It can verify hashes:

```
hashAlgo.verify(“passwordToVerify”.toCharArray(),
“SOME_HASH_AND_SALTED_PASSWORD”))
```

This has been integrated into the `SempicUserFacade` class of the project.

To be able to use the injection…, the following dependency has to be added in the pom.xml:

```
<!-- for HashAlgorithm -->
<dependency>
   <groupId>org.glassfish.soteria</groupId>
   <artifactId>jakarta.security.enterprise</artifactId>
   <version>1.0.1</version>
</dependency>
```

Authentification

The authentication works differently depending on the application server you use. Here, you will find how to implement one solution in TomEE. A standard solution has been introduced in JavaEE8 but it is still not supported in TomEE.

Steps to add authentication:

1- Create a login page with a form with the post action "j_security_check" and two inputs with the name "j_username" and "j_password"

Add in web.xml the declaration this form. For instance:

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

Create a file context.xml (with Web Pages/META-INF) that looks like:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/SempicJPA" preemptiveAuthentication="true">
  <!-- AUthentification config for TomEE, to be commented with
Gassfish -->
  <Valve
className="org.apache.catalina.authenticator.FormAuthenticator"/>
  <Realm cdi="true"
className="org.apache.tomee.catalina.realm.LazyRealm"
realmClass="fr.uga.miashs.sempic.backingbeans.SessionTools"/>
</Context>
```

This file declare that our application will use a realm implmented in the class `fr.uga.miashs.sempic.backingbeans.SessionTools`

This class should at least contain the two following methods:

- `public Principal authenticate(final String username, String password)`
- `public boolean hasRole(final Principal principal, final String role)`

The first method should return `null` if the authentication fails.
The second method should return `true` for:

- admin with role ADMIN
- admin with the role USER
- user with role USER

Add in the web.xml the constraints:

```
  <security-constraint>
        <display-name>Admin</display-name>
        <web-resource-collection>
            <web-resource-name>adminpages</web-resource-name>
            <description/>
            <url-pattern>/faces/admin/*</url-pattern>
            <url-pattern>/admin/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <description/>
            <role-name>ADMIN</role-name>
        </auth-constraint>
    </security-constraint>
    <security-constraint>
        <display-name>User</display-name>
        <web-resource-collection>
            <web-resource-name>userpages</web-resource-name>
            <description/>
            <url-pattern>/faces/user/*</url-pattern>
            <url-pattern>/user/*</url-pattern>
            <url-pattern>/faces/home.xhtml</url-pattern>
        </web-resource-collection>
```

```
        <auth-constraint>
            <description/>
            <role-name>USER</role-name>
        </auth-constraint>
    </security-constraint>
```

The first constraint states that directories /faces/admin and admin are restricted to user with role ADMIN only. The second tells that the file /faces/home.xhtml and the directories /faces/user/ and /user are only allowed to user with the role USER.