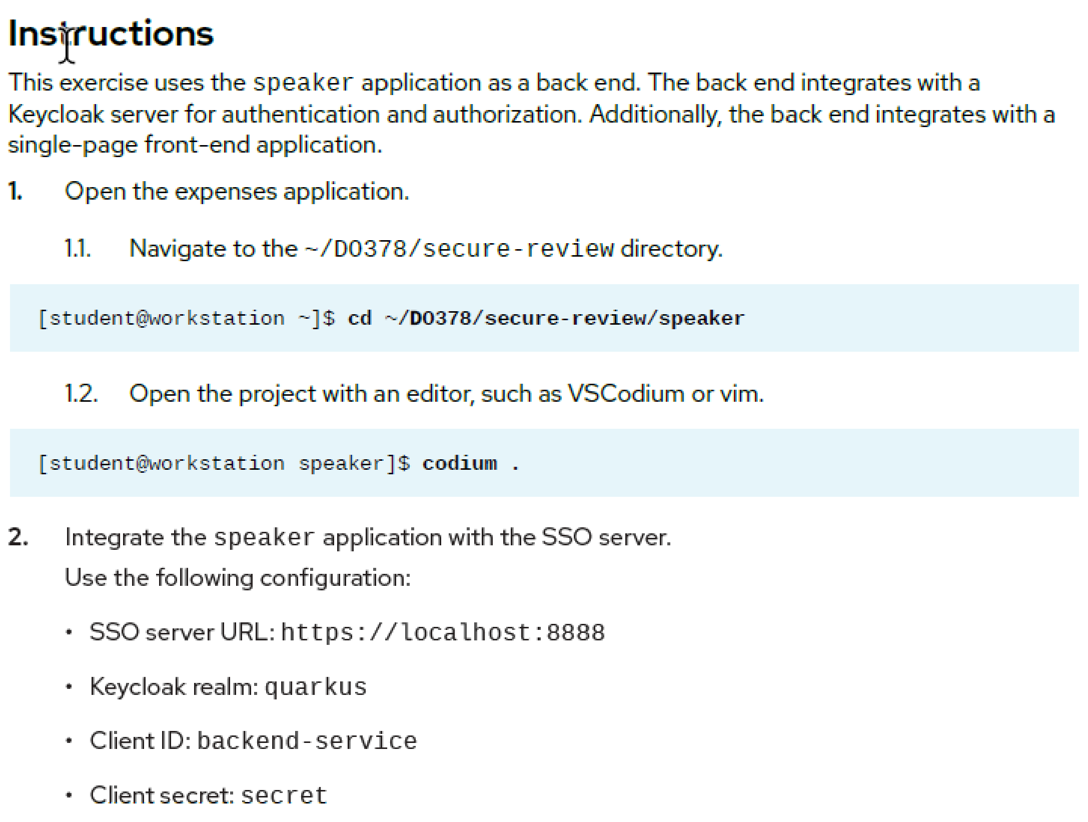
LAB 15: QUARKUS SECURE REVIEW

Autor: José Díaz

Github Repo: https://github.com/joedayz/quarkus-bcp-2025.git

Abre el proyecto **secure-review-start**.



Instructions

This exercise uses the `speaker` application as a back end. The back end integrates with a Keycloak server for authentication and authorization. Additionally, the back end integrates with a single-page front-end application.

1. Open the expenses application.

   1.1. Navigate to the `~/DO378/secure-review` directory.

```
[student@workstation ~]$ cd ~/DO378/secure-review/speaker
```

   1.2. Open the project with an editor, such as VSCodium or vim.

```
[student@workstation speaker]$ codium .
```

2. Integrate the `speaker` application with the SSO server.
   Use the following configuration:

   - SSO server URL: `https://localhost:8888`
   - Keycloak realm: `quarkus`
   - Client ID: `backend-service`
   - Client secret: `secret`

3. Configure CORS for the `speaker` application. The application should allow only requests from the `localhost` origin on port `8080`. Deny browser requests from other origins.

4. Configure the application endpoint authorization.

   Use the following configuration:

   - `GET /speakers`: requires the `read` role.
   - `GET /speakers/{uuid}`: requires the `read` role.
   - `POST /speakers`: requires the `modify` role.
   - `PUT /speakers/{uuid}`: requires the `modify` role.

5. Optionally, use the `speaker-dashboard` front end to test the `speaker` application.

   5.1. Start the `speaker` service.

```
[student@workstation speaker]$ mvn quarkus:dev
```

   5.2. In a web browser, for example Firefox, open `https://localhost:8888` and accept the self-signed certificate. This is necessary for the front-end application to redirect users to the Keycloak login page.

   5.3. Open `http://localhost:8080`. Use the `user` name and `redhat` password. You see a dashboard with four speakers.

   5.4. Click Add a speaker. Enter `Example` as the first name, and `Speaker` as the last name. Then, click Confirm.

   You are presented with an error, because `user` is not authorized to create speakers. Close all browser windows to log out.

   5.5. In a new window, open `http://localhost:8080`. Use the `superuser` name and `redhat` password.

   5.6. Click Add a speaker. Enter `Example` as the first name, and `Speaker` as the last name. Then, click Confirm.

   The call succeeds, because the `superuser` user can create speakers. Close the browser window.

   5.7. Return to the terminal window that runs the `speaker` service, and then press q to stop the application.

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation secure-review]$ lab grade secure-review
```

Finish

Run the `lab finish` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

Solución:

Instructions

This exercise uses the `speaker` application as a back end. The back end integrates with a Keycloak server for authentication and authorization. Additionally, the back end integrates with a single-page front-end application.

1. Open the expenses application.

   1.1. Navigate to the `~/DO378/secure-review` directory.

```
[student@workstation ~]$ cd ~/DO378/secure-review/speaker
```

   1.2. Open the project with an editor, such as VSCodium or vim.

```
[student@workstation speaker]$ codium .
```

2. Integrate the `speaker` application with the SSO server.
   Use the following configuration:

   - SSO server URL: `https://localhost:8888`
   - Keycloak realm: `quarkus`
   - Client ID: `backend-service`
   - Client secret: `secret`

2.1. Add the `quarkus-oidc` extension to the project.

```
[student@workstation speaker]$ mvn quarkus:add-extension -Dextensions=oidc
...output omitted...
[INFO] [SUCCESS] ... Extension io.quarkus:quarkus-oidc has been installed
...output omitted...
```

2.2. Configure the OIDC integration by adding the following properties in the `src/main/resources/application.properties` file.

```
# RHSSO settings
quarkus.oidc.auth-server-url=https://localhost:8888/realms/quarkus
quarkus.oidc.client-id=backend-service
quarkus.oidc.credentials.secret=secret
quarkus.oidc.tls.verification=none
```

2.3. Verify that the `ConfigTest` test suite is passing.

```
[student@workstation speaker]$ mvn clean test -Dtest=ConfigTest
...output omitted...
... Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
...output omitted...
```

3. Configure CORS for the `speaker` application. The application should allow only requests from the `localhost` origin on port `8080`. Deny browser requests from other origins.

3.1. Add the following properties in the `src/main/resources/application.properties` file.

```
# CORS settings
quarkus.http.cors=true
quarkus.http.cors.origins=http://localhost:8080
```

3.2. Verify that the `CorsTest` test suite is passing.

```
[student@workstation speaker]$ mvn clean test -Dtest=CorsTest
...output omitted...
... Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
...output omitted...
```

4. Configure the application endpoint authorization.

Use the following configuration:

- `GET /speakers`: requires the `read` role.
- `GET /speakers/{uuid}`: requires the `read` role.
- `POST /speakers`: requires the `modify` role.
- `PUT /speakers/{uuid}`: requires the `modify` role.

4.1. Open the `com.redhat.training.SpeakerResource` class, and use the `@RolesAllowed` annotation to secure the endpoints.

```
...code omitted...

@GET
@RolesAllowed("read")
public List<Speaker> getSpeakers() {

...code omitted...


@GET
@Path("/{uuid}")
@RolesAllowed("read")
public Optional<Speaker> findByUuid(@PathParam("uuid") String uuid) {

...code omitted...

@Transactional
@POST
@RolesAllowed("modify")
public Speaker insert(Speaker speaker) {

...code omitted...

@Transactional
@PUT
@Path("/{uuid}")
@RolesAllowed("modify")
public Speaker update(@PathParam("uuid") String uuid, Speaker speaker) {

...code omitted...
```

4.2. Verify that the `SpeakerResourceTest` test suite is passing.

```
[student@workstation speaker]$ mvn clean test -Dtest=SpeakerResourceTest
...output omitted...
... Tests run: 9, Failures: 0, Errors: 0, Skipped: 0
...output omitted...
```

5. Optionally, use the `speaker-dashboard` front end to test the `speaker` application.

5.1. Start the `speaker` service.

```
[student@workstation speaker]$ mvn quarkus:dev
```

5.2. In a web browser, for example Firefox, open `https://localhost:8888` and accept the self-signed certificate. This is necessary for the front-end application to redirect users to the Keycloak login page.

5.3. Open `http://localhost:8080`. Use the `user` name and `redhat` password. You see a dashboard with four speakers.

5.4. Click Add a speaker. Enter `Example` as the first name, and `Speaker` as the last name. Then, click Confirm.

You are presented with an error, because `user` is not authorized to create speakers. Close all browser windows to log out.

5.5. In a new window, open `http://localhost:8080`. Use the `superuser` name and `redhat` password.

5.6. Click Add a speaker. Enter `Example` as the first name, and `Speaker` as the last name. Then, click Confirm.

The call succeeds, because the `superuser` user can create speakers. Close the browser window.

5.7. Return to the terminal window that runs the `speaker` service, and then press `q` to stop the application.

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation secure-review]$ lab grade secure-review
```

Finish

Run the `lab finish` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish secure-review
```

This concludes the section.

enjoy!

Jose