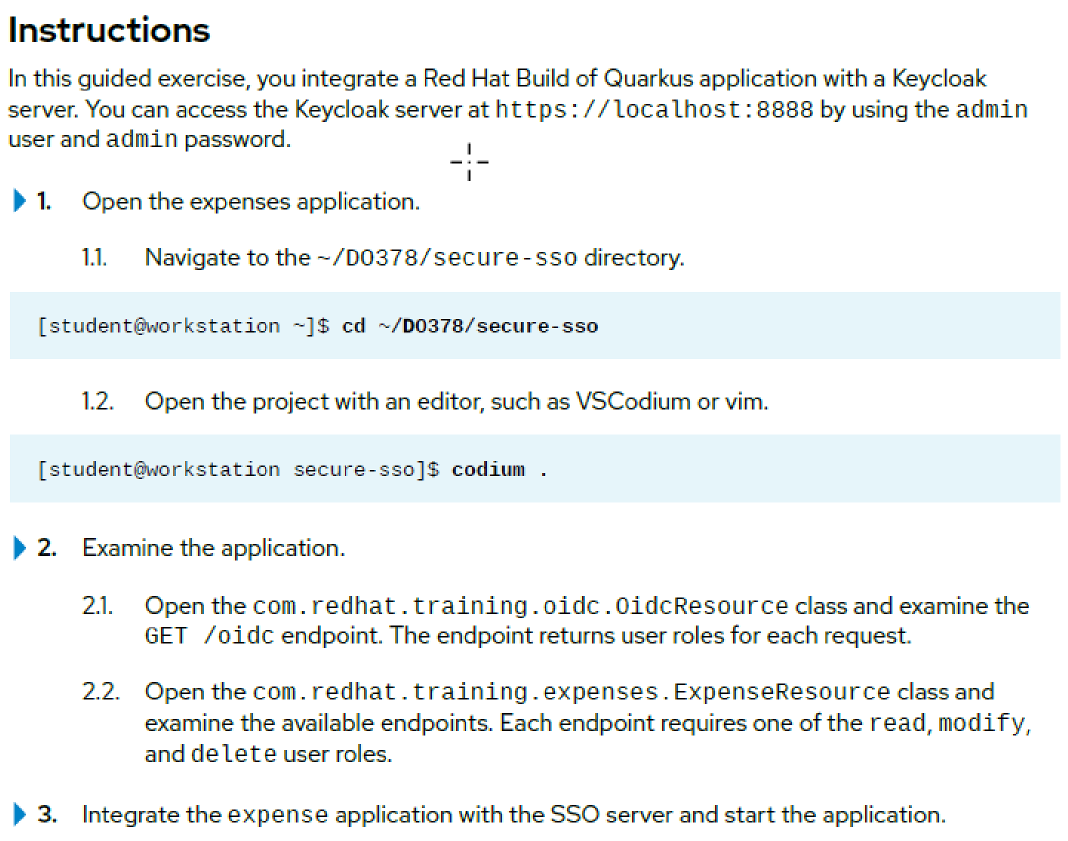
LAB 15: QUARKUS SECURE SSO

Autor: José Díaz

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

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



Instructions

In this guided exercise, you integrate a Red Hat Build of Quarkus application with a Keycloak server. You can access the Keycloak server at `https://localhost:8888` by using the `admin` user and `admin` password.

1. Open the expenses application.

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

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

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

```
[student@workstation secure-sso]$ codium .
```

2. Examine the application.

2.1. Open the `com.redhat.training.oidc.OidcResource` class and examine the `GET /oidc` endpoint. The endpoint returns user roles for each request.

2.2. Open the `com.redhat.training.expenses.ExpenseResource` class and examine the available endpoints. Each endpoint requires one of the `read`, `modify`, and `delete` user roles.

3. Integrate the `expense` application with the SSO server and start the application.

Use the following configuration:

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

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

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

3.2. 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
```

3.3. Start the application.

```
[student@workstation secure-sso]$ mvn quarkus:dev
```

4. Verify that the `user` account can execute the `GET /expense` endpoint, but does not have permissions to execute the `DELETE /expense/{UUID}` endpoint.

4.1. In a new terminal window, navigate to the `~/DO378/secure-sso` directory.

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

4.2. Inspect the `get_token.sh` script. The script sends a request to the SSO server with credentials and exports a bearer token.

4.3. Use the `get_token.sh` script to get a bearer token from the OIDC server for the `user` account. Use the `redhat` password. The `get_token.sh` script exports the bearer token as the `TOKEN` shell variable.

```
[student@workstation secure-sso]$ source get_token.sh user redhat
Token succesfuly retrieved.
```

4.4. Optionally verify that the `TOKEN` variable contains the bearer token.

```
[student@workstation secure-sso]$ echo $TOKEN
eyJh....gDlXrGA
```

4.5. Verify that the `user` account uses the `read` role by using the `GET /oidc` endpoint. Authenticate your request by using the `Authorization` header.

```
[student@workstation secure-sso]$ curl -s http://localhost:8080/oidc \
-H "Authorization: Bearer $TOKEN" | jq
{
  "roles": [
    "read",
    "offline_access",
    "default-roles-quarkus",
    "uma_authorization"
  ]
}
```

4.6. Use the `user` bearer token to call the `GET /expense` endpoint.

```
[student@workstation secure-sso]$ curl -s http://localhost:8080/expense \
-H "Authorization: Bearer $TOKEN" | jq
[
  {
    "uuid": "3f1817f2-3dcf-472f-a8b2-77bfe25e79d1",
    "name": "Kubernetes Patterns",
    "paymentMethod": "CASH",
    "amount": 10.00
  },
...output omitted...
]
```

4.7. Attempt to remove one of the items by calling the `DELETE /expense/{UUID}` endpoint with the `user` token. Use a UUID from the output of the previous request.

```
[student@workstation secure-sso]$ UUID=3f1817f2-3dcf-472f-a8b2-77bfe25e79d1
[student@workstation secure-sso]$ curl -vX DELETE \
  -H "Authorization: Bearer $TOKEN" \
http://localhost:8080/expense/$UUID
...output omitted...
< HTTP/1.1 403 Forbidden
< www-authenticate: Bearer
< content-length: 0
<
* Connection #0 to host localhost left intact
```

5. Verify that the `superuser` account can execute the `DELETE /expenses/{UUID}` endpoint.

5.1. Use the `get_token.sh` script to get a bearer token from the OIDC server for the `superuser` account. Use the `redhat` password.

```
[student@workstation secure-sso]$ source get_token.sh superuser redhat
Token succesfuly retrieved.
```

5.2. Verify that the `superuser` account uses the `read`, `modify`, and `delete` roles by using the `GET /oidc` endpoint.

```
[student@workstation secure-sso]$ curl -s http://localhost:8080/oidc \
-H "Authorization: Bearer $TOKEN" | jq
{
  "roles": [
    "modify",
    "read",
    "offline_access",
    "default-roles-quarkus",
    "uma_authorization",
    "delete"
  ]
}
```

5.3. Re-execute calling the `DELETE /expense/{UUID}` endpoint with the `superuser` token.

```
[student@workstation secure-sso]$ curl -X DELETE \
 -H "Authorization: Bearer $TOKEN" \
 http://localhost:8080/expense/$UUID | jq
[
  {
    "uuid": "4fe78f4f-3335-4585-8677-5d9c8bbb539e",
    "name": "Red Hat OpenShift for Developers",
    "paymentMethod": "CASH",
    "amount": 15.00
  },
...output omitted...
]
```

The call succeeds and deletes the item.

6. In the terminal with the active application, press **q** to stop the `expense` application.

Finish

On the `workstation` machine, use the `lab` 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-sso
```

This concludes the section.

enjoy!

Jose