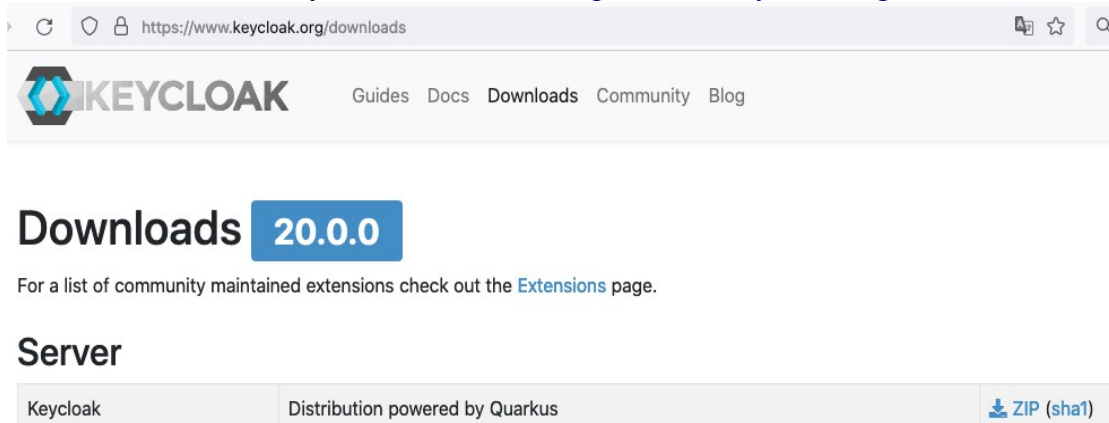


Inhaltsverzeichnis

| | |
|---------------------------------|----|
| 1Setup Keycloak Server..... | 2 |
| 2Setup Spring boot project..... | 12 |
| 3Test oauth2 login..... | 13 |
| 4Test JWT access..... | 15 |
| 5Springdoc/Openapi..... | 17 |
| 6Vaadin front end..... | 20 |

1 Setup Keycloak Server

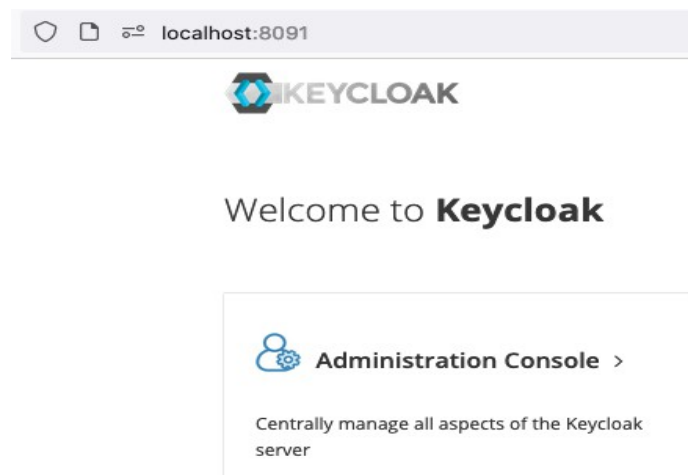
1. Download the latest keycloak server from <https://www.keycloak.org/downloads>.



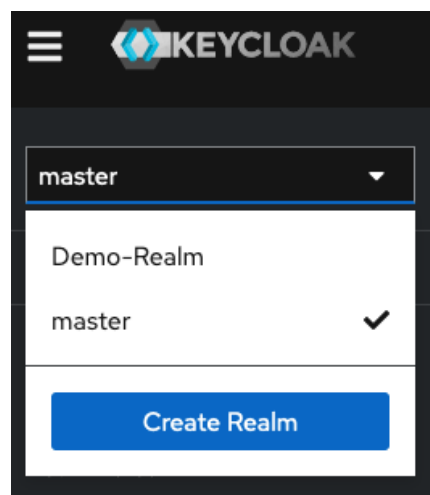
2. Unzip, open a terminal and change to directory keycloak-20.0.0 and run: `bin/kc.sh start-dev --http-port 8091`

```
keycloak-20.0.0 % bin/kc.sh start-dev --http-port 8091
```

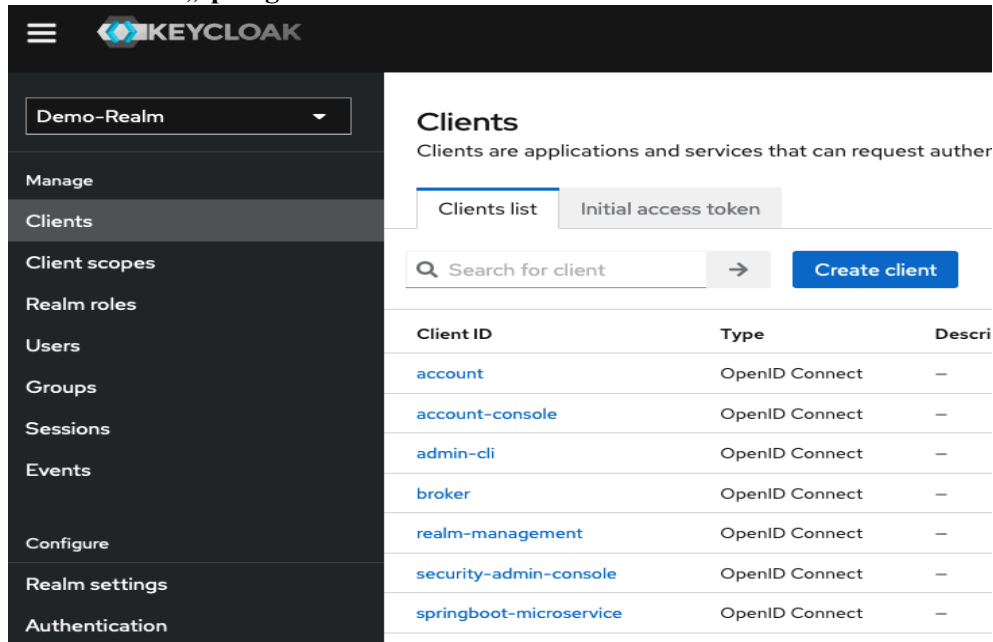
3. Open keycloak admin console



4. Have a look at <https://www.djamware.com/post/6225b66ba88c55c95abca0b6/spring-boot-security-postgresql-and-keycloak-rest-api-oauth2>
5. Create an admin account
6. Create a new Realm **Demo-Realm** and select it



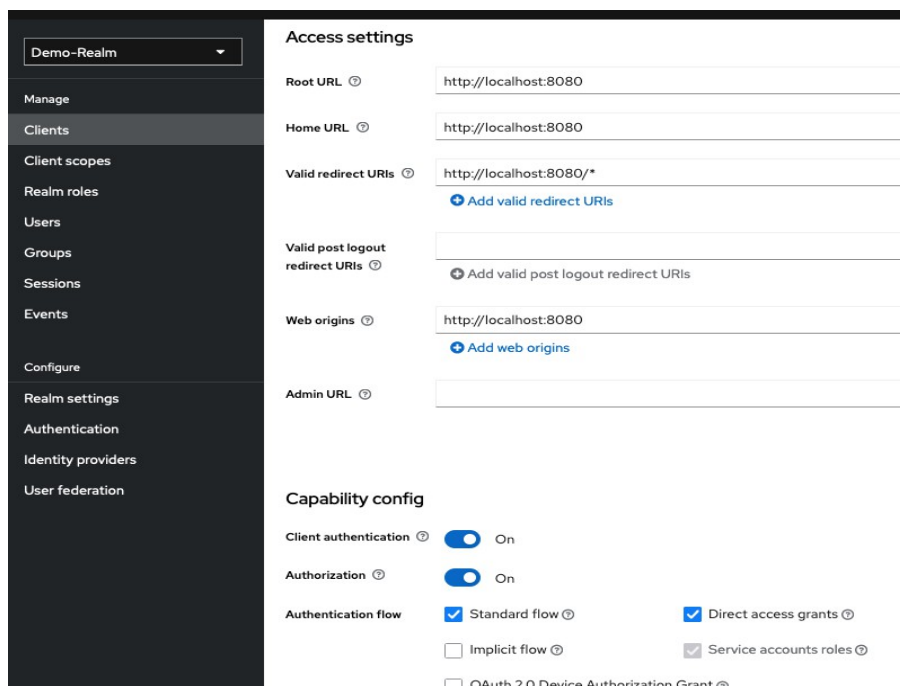
7. Create a new client „springboot-microservice“



The screenshot shows the Keycloak 'Clients' page for the 'Demo-Realm'. The left sidebar contains navigation links: Manage, Clients (selected), Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, and Authentication. The main content area is titled 'Clients' and includes a description: 'Clients are applications and services that can request authentication'. There are two tabs: 'Clients list' (active) and 'Initial access token'. A search bar with the placeholder 'Search for client' and a 'Create client' button are present. Below is a table of existing clients.

| Client ID | Type | Description |
|-------------------------|----------------|-------------|
| account | OpenID Connect | — |
| account-console | OpenID Connect | — |
| admin-cli | OpenID Connect | — |
| broker | OpenID Connect | — |
| realm-management | OpenID Connect | — |
| security-admin-console | OpenID Connect | — |
| springboot-microservice | OpenID Connect | — |

8. Fill in Root-Url, Home-Url, Valid-Redirect-Urls and Web-Origins and switch on „Client authentication“ and „Authorization“



The screenshot shows the 'Access settings' page for the 'Demo-Realm'. The left sidebar is the same as in the previous image. The main content area is titled 'Access settings' and contains several input fields for configuration. Below these is the 'Capability config' section with toggle switches and checkboxes.

Access settings

- Root URL:
- Home URL:
- Valid redirect URIs: [Add valid redirect URIs](#)
- Valid post logout redirect URIs: [Add valid post logout redirect URIs](#)
- Web origins: [Add web origins](#)
- Admin URL:

Capability config

- Client authentication: ☒ On
- Authorization: ☒ On
- Authentication flow: ☒ Standard flow ☒ Direct access grants ☐ Implicit flow ☒ Service accounts roles ☐ OAuth 2.0 Device Authorization Grant

9. Create two Realm-Roles **app-admin** and **app-user**

The screenshot shows the 'Realm roles' page in Keycloak. On the left is a sidebar with navigation links: Demo-Realm, Manage, Clients, Client scopes, Realm roles (selected), Users, Groups, Sessions, and Events. The main content area is titled 'Realm roles' with a subtitle 'Realm roles are the roles that you define for use in the current realm.' and a 'Learn more' link. Below this is a search bar 'Search role by name' with a right arrow and a 'Create role' button. A table lists the existing roles:

| Role name | Composite |
|----------------------------|-----------|
| app-admin | True |
| app-user | True |
| default-roles-demo-realm ⓘ | True |
| offline_access | False |
| uma_authorization | False |

10. Switch to client **springboot-microservice** and create two roles **admin** and **user**

The screenshot shows the 'Client details' page for the 'springboot-microservice' client. The sidebar is the same as in the previous screenshot, with 'Clients' selected. The main content area is titled 'springboot-microservice' with an 'OpenID Connect' button. Below this is a subtitle 'Clients are applications and services that can request authentication of a user.' and a set of tabs: Settings, Keys, Credentials, Roles (selected), Client scopes, Authorization, and Service accounts. Below the tabs is a search bar 'Search role by name' with a right arrow and a 'Create role' button. A table lists the roles for this client:

| Role name | Composite |
|----------------|-----------|
| admin | False |
| uma_protection | False |
| user | False |

11. Switch to Realm-Roles and select Role **app-user**, click on Action and select „Add associated roles“

The screenshot shows the 'Role details' page for the 'app-user' role. The breadcrumb is 'Realm roles > Role details'. The role name 'app-user' is shown with a 'Composite' tag. Below this are tabs: Details (selected), Associated roles, Attributes, Users in role, and Permissions. On the right, an 'Action' dropdown menu is open, showing 'Add associated roles' and 'Delete this role'. The 'Role name' field is filled with 'app-user'. The 'Description' field is empty. At the bottom are 'Save' and 'Revert' buttons.

12. Select „filter by clients“ and enter „springboot-microservices“ and select role **user**

Assign roles to app-user account

Filter by clients

Q spring

×

→

1-3

<

>

| <input type="checkbox"/> | Name | Description |
|-------------------------------------|--|-------------|
| <input type="checkbox"/> | springboot-microservice admin | |
| <input type="checkbox"/> | springboot-microservice uma_protection | |
| <input checked="" type="checkbox"/> | springboot-microservice user | |

1-3

<

>

Assign Cancel

13. Realm-Role app-user is now a composite role

[Realm roles](#) > Role details

app-user Composite

Details

Associated roles

Attributes

Users in role

Permissions

Q Search by name

→

☒ Hide inherited roles

Assign role

Unassign

| <input type="checkbox"/> | Name | Inherited | Description |
|--------------------------|------------------------------|-----------|-------------|
| <input type="checkbox"/> | springboot-microservice user | False | — |

14. Do the same with Realm-Role app-admin and associate role admin

[Realm roles](#) > Role details

app-admin Composite

Details

Associated roles

Attributes

Users in role

Permissions

Q Search by name

→

☒ Hide inherited roles

Assign role

Unassign

| <input type="checkbox"/> | Name | Inherited |
|--------------------------|-------------------------------|-----------|
| <input type="checkbox"/> | springboot-microservice admin | False |

15. Create 3 Users named employee1 – employee3, set „Email verified“ to „On“

[Users](#) > Create user

Create user


| | |
|--|--|
| Username * | employee1 |
| Email | m0.m0@company.com |
| Email verified ? | <input checked="" type="checkbox"/> On |
| First name | Max0 |
| Last name | Mustermann0 |
| Required user actions ? | Select action |
| Groups ? | Join Groups |
| <div>Create Cancel</div> | |

16. Set user credentials

[Users](#) > User details

employee1

| | | | | |
|---------|------------|-------------|--------------|--------|
| Details | Attributes | Credentials | Role mapping | Groups |
|---------|------------|-------------|--------------|--------|

| | | |
|---|----------|--|
| ? | Type | User label |
| ⋮ | Password | My password  |

17. Assign Realm-Role **app-user** to employee1

employee1

| | | | | | | |
|---------|------------|-------------|--------------|--------|----------|------------------|
| Details | Attributes | Credentials | Role mapping | Groups | Consents | Identity provide |
|---------|------------|-------------|--------------|--------|----------|------------------|

→ ☒ Hide inherited roles Assign role Unassign

| <input type="checkbox"/> Name | Inherited | Description |
|---|-----------|------------------------|
| <input type="checkbox"/> app-user | False | – |
| <input type="checkbox"/> default-roles-demo-realm | False | \${role_default-roles} |

18. Remove all actions in field „Required user action“ and save
19. Repeat everything from point 14-17 with employee2 and assign Realm-Role app-admin

employee2

| | | | | | |
|---------|------------|-------------|--------------|--------|-----|
| Details | Attributes | Credentials | Role mapping | Groups | Cor |
|---------|------------|-------------|--------------|--------|-----|

→ ☒ Hide inherited roles Assign role

| <input type="checkbox"/> Name | Inherited |
|---|-----------|
| <input type="checkbox"/> default-roles-demo-realm | False |
| <input type="checkbox"/> app-admin | False |

20. Repeat everything from point 14-17 with employee3 and assign Realm-Role **app-admin** and **app-user**

employee3

| | | | | | |
|---------|------------|-------------|--------------|--------|-------|
| Details | Attributes | Credentials | Role mapping | Groups | Conse |
|---------|------------|-------------|--------------|--------|-------|

→ ☒ Hide inherited roles Assign role

| <input type="checkbox"/> Name | Inherited |
|---|-----------|
| <input type="checkbox"/> app-user | False |
| <input type="checkbox"/> default-roles-demo-realm | False |
| <input type="checkbox"/> app-admin | False |

21. Switch to client scopes and add new scope read

Client scopes

Client scopes are a common set of protocol mappers and roles that are shared between i

🔍 Name

✕

➔

Create client scope

| <input type="checkbox"/> | Name | Assigned type | Protocol |
|--------------------------|------|---------------|-------------|
| <input type="checkbox"/> | read | None | OpenID Conn |

22. Assign Realm-Roles app-admin, app-user and Roles user and admin to scope read

[Client scopes](#) > Client scope details

read openid-connect

Settings

Mappers

Scope

🔍 Search by name

➔

☒ Hide inherited roles

Assign role

| <input type="checkbox"/> | Name | Inherited |
|--------------------------|--|-----------|
| <input type="checkbox"/> | app-user | False |
| <input type="checkbox"/> | app-admin | False |
| <input type="checkbox"/> | springboot-microservice admin | False |
| <input type="checkbox"/> | springboot-microservice user | False |

23. Switch to client „springboot-microservices“ and add scope read as „Optional“

[Clients](#) > Client details

springboot-microservice OpenID Connect

Clients are applications and services that can request authentication of a user.

Settings

Keys

Credentials

Roles

Client scopes

Authorization

Service

Setup

Evaluate

🔍 Name

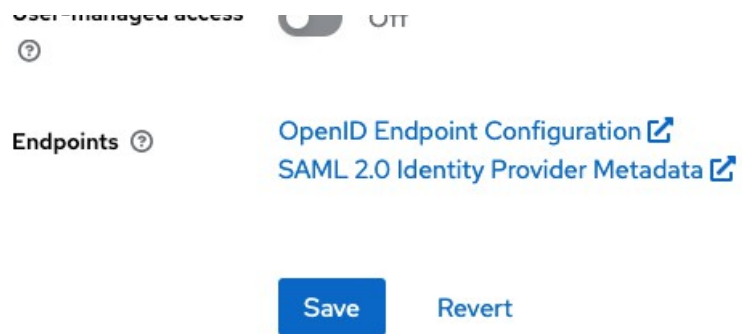
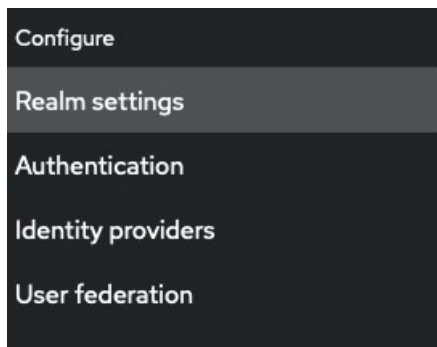
✕

➔

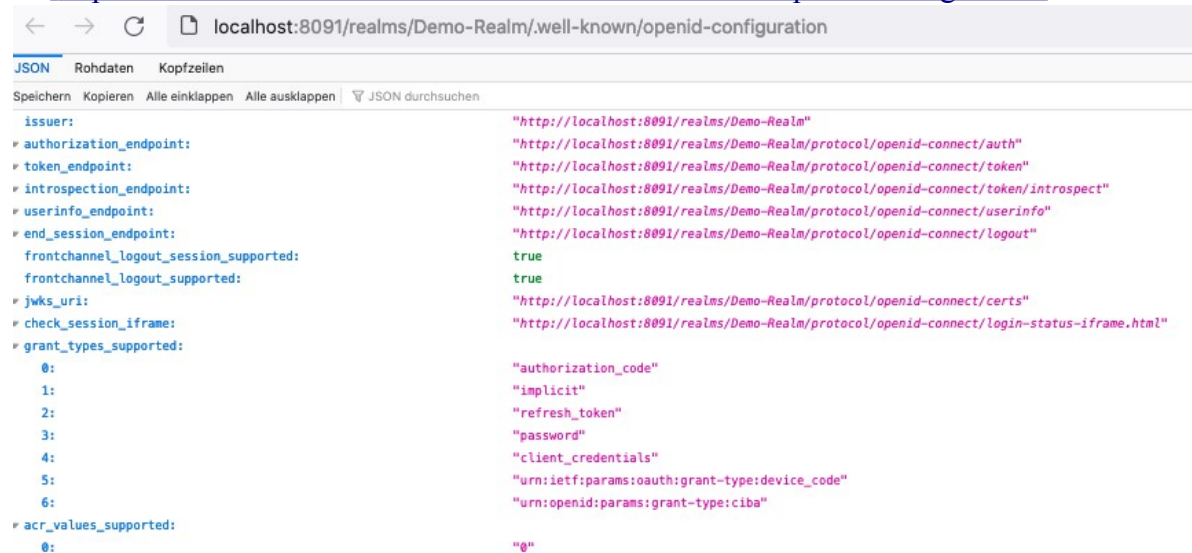
Add client scope

| <input type="checkbox"/> | Assigned client scope | Assigned type | Descri |
|--------------------------|-----------------------------------|---------------|--------|
| <input type="checkbox"/> | springboot-microservice-dedicated | none | Dedica |
| <input type="checkbox"/> | read | Optional | |

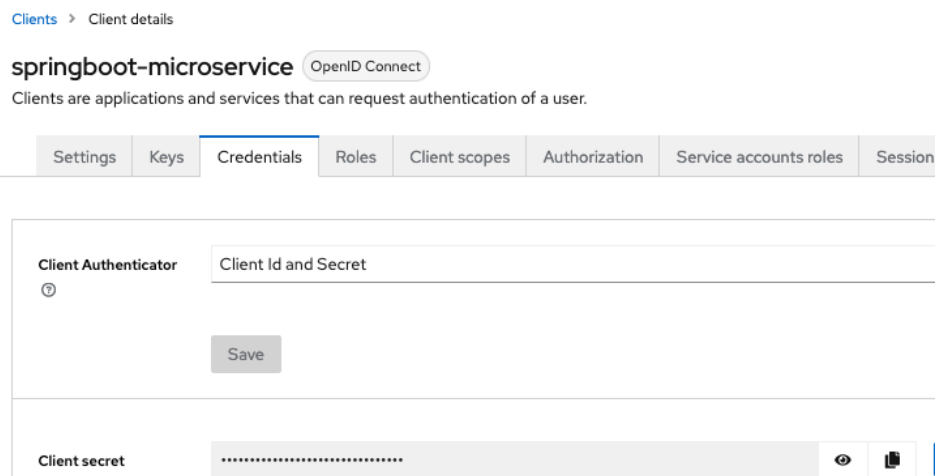
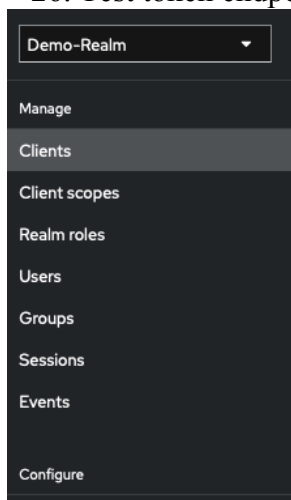
24. Switch to Realm settings an open „OpenID Endpoint Configuration“



25. <http://localhost:8091/realms/Demo-Realm/.well-known/openid-configuration>



26. Test token endpoint → Copy client secret from Client „springboot-microservices“



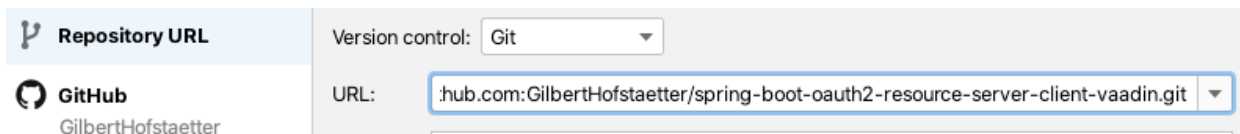
27. Use Postman or a REST client of your choice to receive an access token

29. Open <https://jwt.io/> to decode our access token
30. See section resource-access for our applied user role

```
    "realm_access": {
      "roles": [
        "offline_access",
        "default-roles-demo-realm",
        "uma_authorization",
        "app-user"
      ]
    },
    "resource_access": {
      "springboot-microservice": {
        "roles": [
          "user"
        ]
      }
    },
    "account": {
```

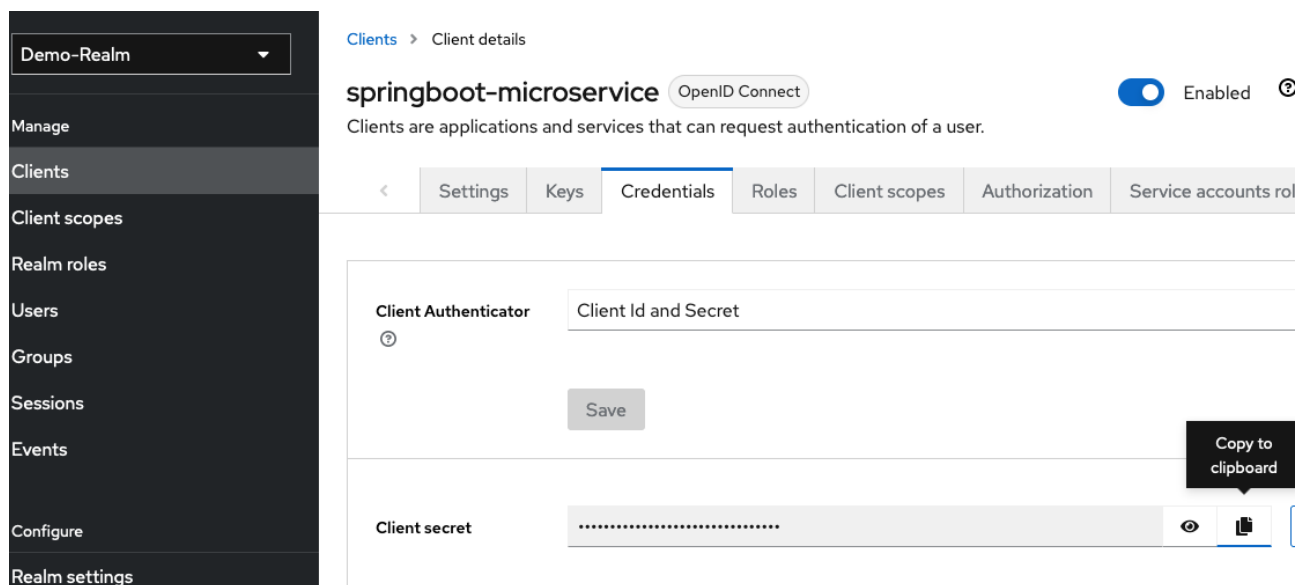
2 Setup Spring boot project

- Clone from [git@github.com:GilbertHofstaetter/spring-boot-oauth2-resource-server-client-vaadin.git](https://github.com/GilbertHofstaetter/spring-boot-oauth2-resource-server-client-vaadin.git)

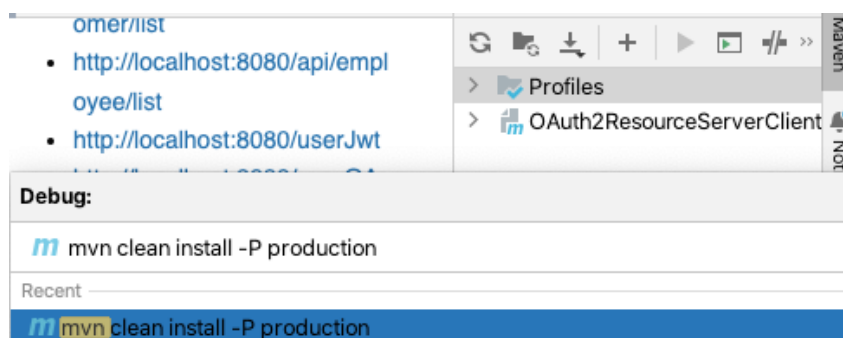


- Replace client secret in application.yml with your client secret from keycloak client springboot-microservice

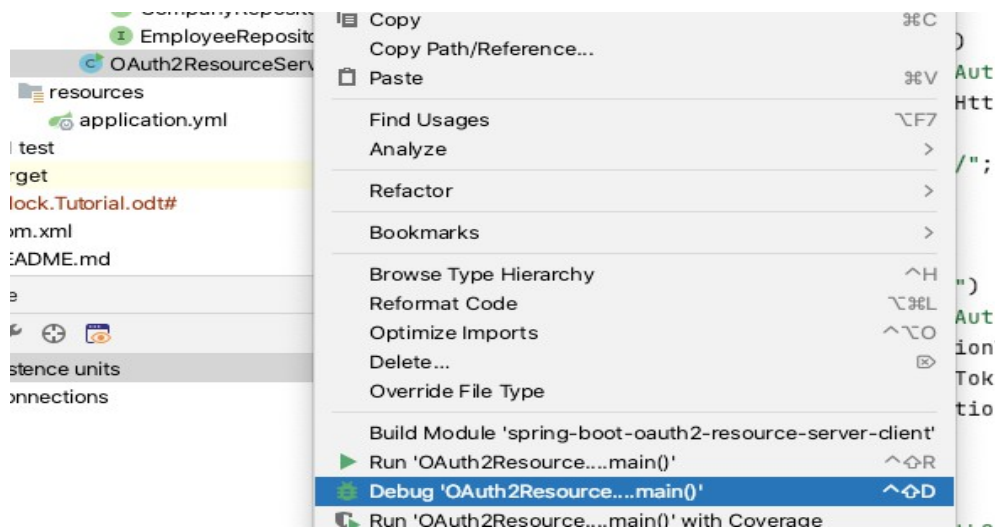
```
registration:
  keycloak:
    client-id: springboot-microservice
    client-secret: L7CYws3HULR0IAsA4aSwssvZZrS9iz10
    scope: openid, profile, read
```



- mvn clean install -P production

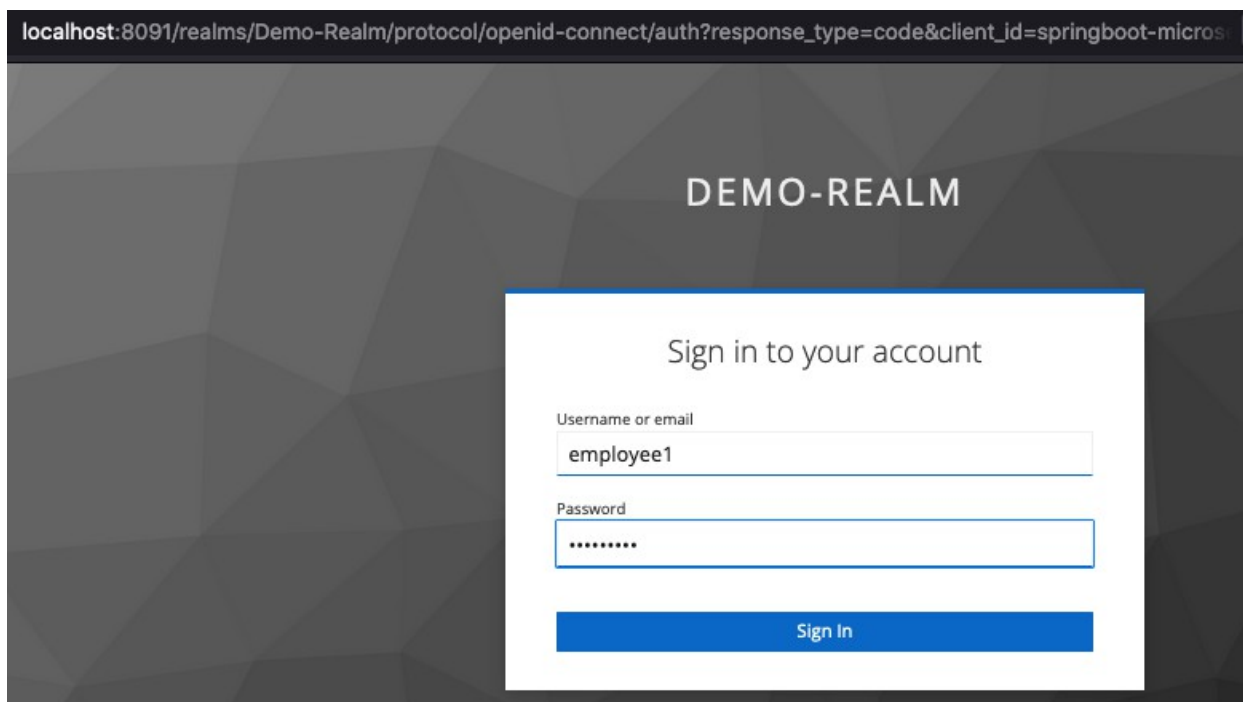


- Start project



3 Test oauth2 login

- Open <http://localhost:8080/api/employee/list> in your browser



- Enter login infos for user employee1 → Data gets listed → Role user is applied

```
localhost:8080/api/employee/list
JSON Rohdaten Kopfzeilen
Speichern Kopieren Alle einklappen Alle ausklappen JSON durchsuchen

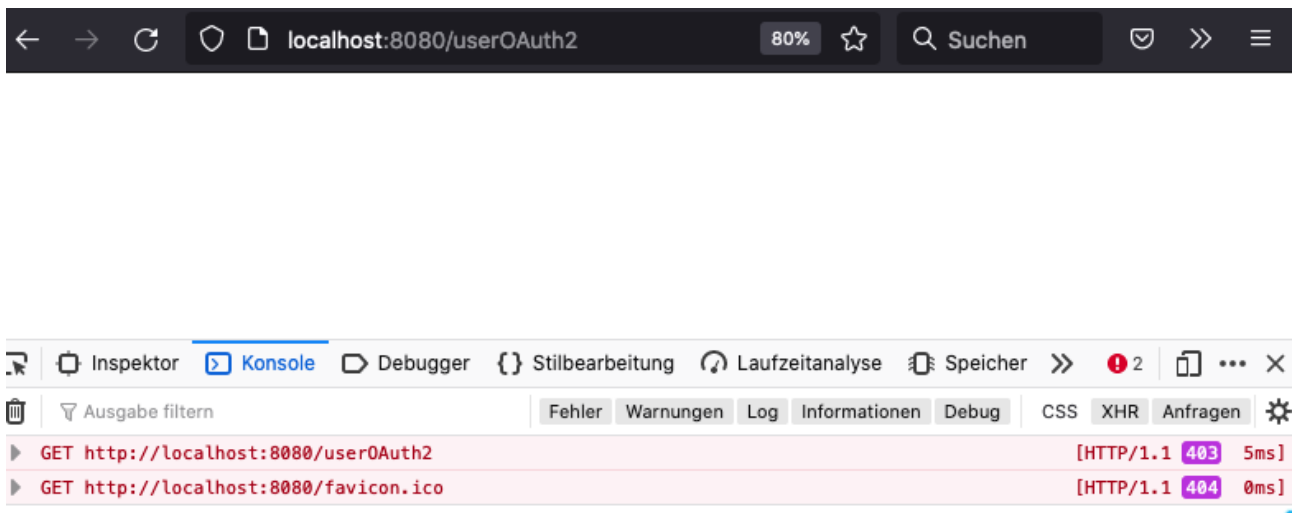
[
  {
    id: 1,
    firstName: "Max0",
    lastName: "Mustermann0",
    email: "m0.m0@company.com",
    birthDate: "2022-11-06T12:37:52.319+00:00",
    company: {
      id: 4,
      name: "Company"
    }
  },
  {
    id: 2,
    firstName: "Max1",
    lastName: "Mustermann1",
    email: "m1.m1@company.com",
    birthDate: "2022-11-06T12:37:52.425+00:00",
    company: {
      id: 4
    }
  },
  {
    id: 3,
    firstName: "Max2",
    lastName: "Mustermann2",
    email: "m2.m2@company.com",
    birthDate: "2022-11-06T12:37:52.427+00:00",
    company: {
      id: 4
    }
  }
]
```

- Open <http://localhost:8080/api/company/list> → already authenticated

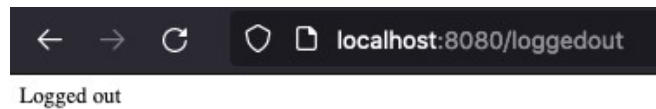
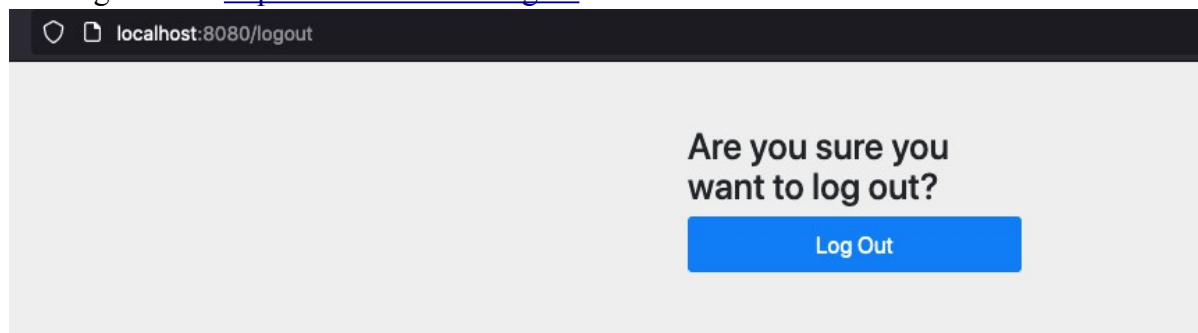
```
localhost:8080/api/company/list
JSON Rohdaten Kopfzeilen
Speichern Kopieren Alle einklappen Alle ausklappen JSON durchsuchen

{
  id: 4,
  name: "Company",
  employees: [
    {
      id: 2,
      firstName: "Max1",
      lastName: "Mustermann1",
      email: "m1.m1@company.com",
      birthDate: "2022-11-06T12:37:52.425+00:00"
    },
    {
      id: 3,
      firstName: "Max2",
      lastName: "Mustermann2",
      email: "m2.m2@company.com",
      birthDate: "2022-11-06T12:37:52.427+00:00"
    },
    {
      id: 1,
      firstName: "Max0",
      lastName: "Mustermann0",
      email: "m0.m0@company.com",
      birthDate: "2022-11-06T12:37:52.319+00:00"
    }
  ]
}
```

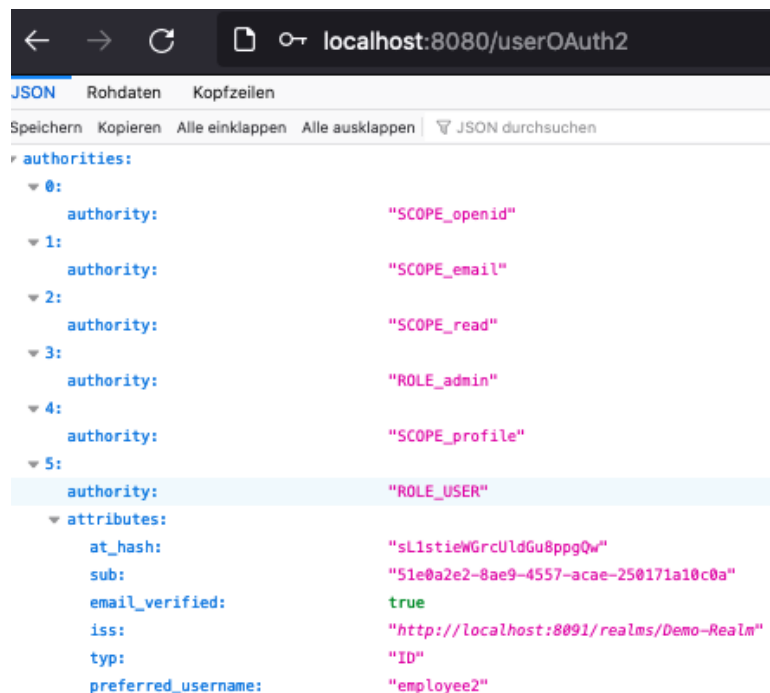
- Try to open <http://localhost:8080/userOAuth2> → a blank page appears and status 403 is reported → endpoint `./userOAuth2` needs admin role to perform → logout and sign in as user `employee2` or `employee3`



- Logout with <http://localhost:8080/logout>



- Open again <http://localhost:8080/userOAuth2> → login with user employee2 or employee3



4 Test JWT access

- Access <http://localhost:8080/api/employee/list> via REST client
- Retrieve an access token as employee3 (don't forget to request optional **scope read**)

[illegible]

- Call <http://localhost:8080/api/employee/list> with Authorisation-Header Bearer ***your access token ***

GET

Headers ▾

Authorization

[+ Add header](#)

[Basic auth >](#)

Response (0.104s) - http://localhost:8080/api/employee/list

200

Headers >

```
[
  {
    "id": 1,
    "firstName": "Max0",
    "lastName": "Mustermann0",
    "email": "m0.m0@company.com",
    "birthDate": "2022-11-06T12:37:52.319+00:00",
    "company": {
      "id": 4,
      "name": "Company"
    }
  },
  {
    "id": 2,
    "firstName": "Max1",
    "lastName": "Mustermann1",
    "email": "m1.m1@company.com",
    "birthDate": "2022-11-06T12:37:52.425+00:00",
    "company": 4
  },
  {
    "id": 3,
    "firstName": "Max2",
    "lastName": "Mustermann2",
    "email": "m2.m2@company.com",
    "birthDate": "2022-11-06T12:37:52.427+00:00",
    "company": 4
  }
]
```

- Call <http://localhost:8080/api/company/list>

GET

Headers ▾

Authorization

[+ Add header](#)

[Basic auth >](#)

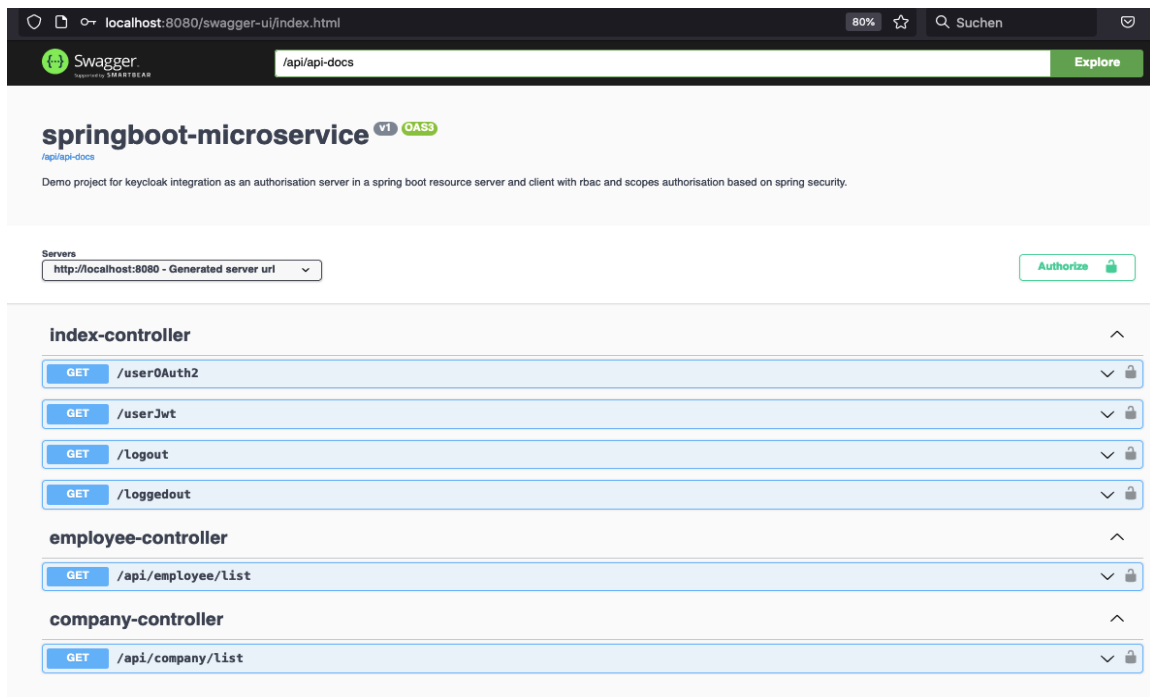
Response (0.035s) - http://localhost:8080/api/company/list

200

Headers >

```
[
  {
    "id": 4,
    "name": "Company",
    "employees": [
      {
        "id": 2,
        "firstName": "Max1",
        "lastName": "Mustermann1",
        "email": "m1.m1@company.com",
        "birthDate": "2022-11-06T12:37:52.425+00:00"
      },
      {
        "id": 3,
        "firstName": "Max2",
        "lastName": "Mustermann2",
        "email": "m2.m2@company.com",
        "birthDate": "2022-11-06T12:37:52.427+00:00"
      },
      {
        "id": 1,
        "firstName": "Max0",
        "lastName": "Mustermann0",
        "email": "m0.m0@company.com",
        "birthDate": "2022-11-06T12:37:52.319+00:00"
      }
    ]
  }
]
```

- Call <http://localhost:8080/userJwt>



- Click „Authorize“
- Fill in employee3 as user, userpassword, client-id (springboot-microservice), client secret and select scope „read“
 - Password Flow is used

Available authorizations

keycloak (OAuth2, password)

Token URL: http://localhost:8091/realms/Demo-Realm/protocol/openid-connect/token

Flow: password

username:
employee3

password:
.....

Client credentials location:
Authorization header

client_id:
springboot-microservice

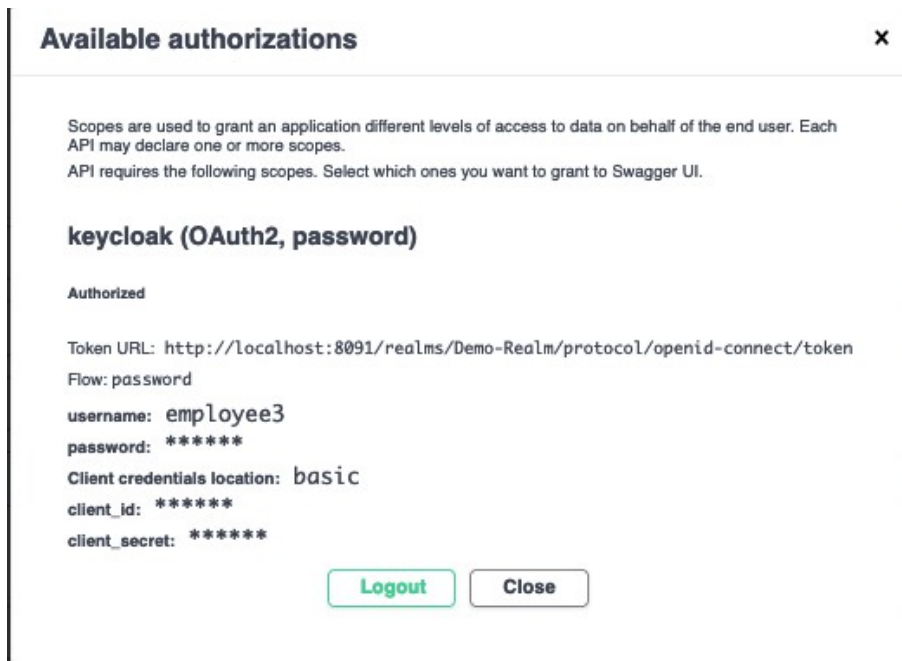
client_secret:
.....

Scopes: [select all](#) [select none](#)

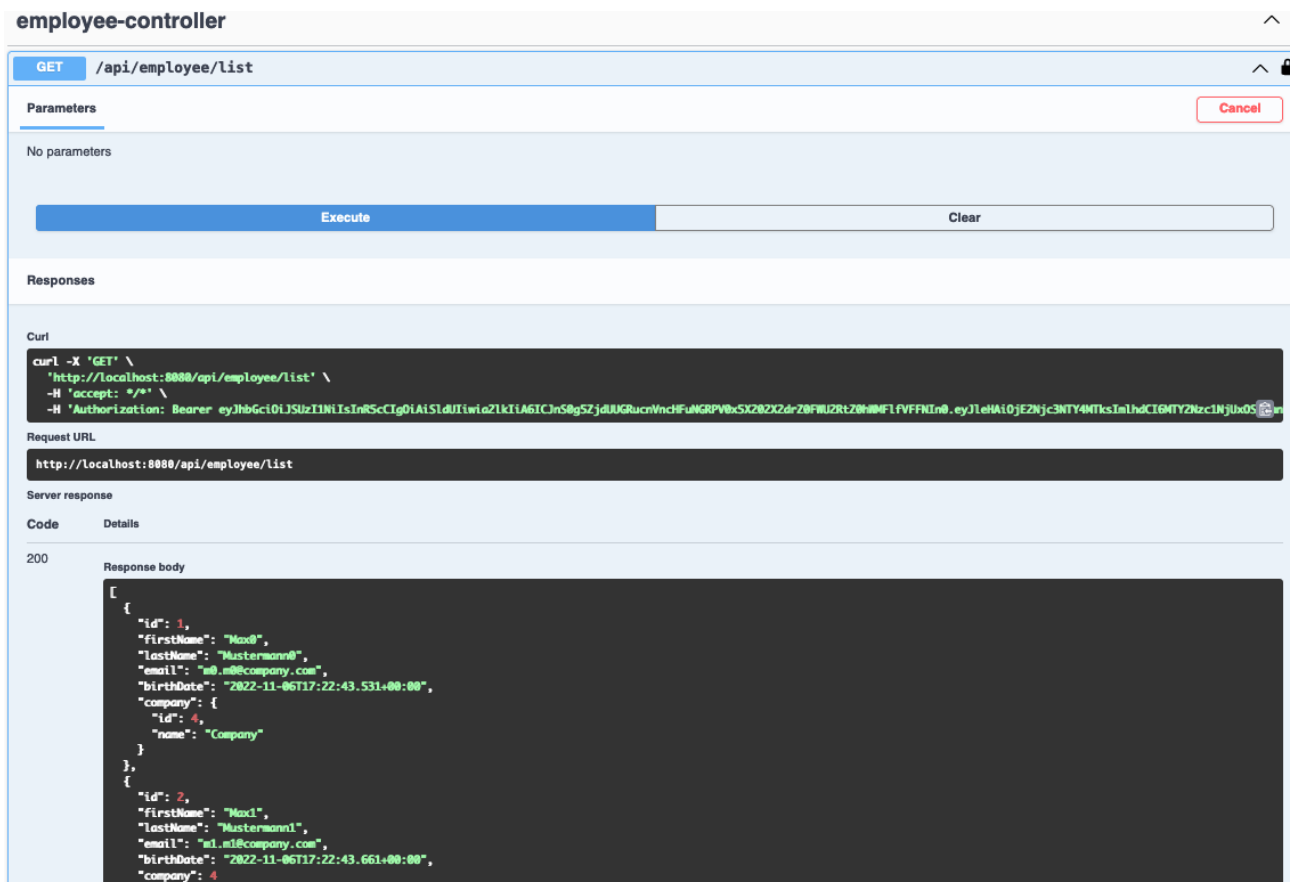
☒ read
read

Authorize

Close



- Open Endpoint `/api/employee/list`
 - Bearer token has been added to Authorization Header
 - → Execute → Call successful
 - Bearer token from Available authorizations is used instead of previous user authorization to swagger-ui



6 Vaadin front end

- Call <http://localhost:8080/ui>
- Keycloak login opens

localhost:8091/realms/Demo-Realm/protocol/openid-connect/auth?response_type=code&client_id=springboot

DEMO-REALM

Sign in to your account

Username or email

Password

Sign In

- Enter User and password
- Redirect to <http://localhost:8080/ui>

localhost:8080/ui/

Home

Logout

- Test
- Swagger-ui
- Employee
- Company

- Open Employee

localhost:8080/ui/employee

Employee

Logout

Filter by last name

| Id | First Name | Last Name | Email | Birth Date |
|----|------------|-------------|-------------------|-------------------------|
| 1 | Max0 | Mustermann0 | m0.m0@company.com | 2022-11-08 09:38:14.345 |
| 2 | Max1 | Mustermann1 | m1.m1@company.com | 2022-11-08 09:38:14.522 |
| 3 | Max2 | Mustermann2 | m2.m2@company.com | 2022-11-08 09:38:14.526 |

- Open Company

