FPT SOFTWARE QUY NHON AI Valley

SPRING BOOT Integration With Other Libraries

KienNCT3 - Mentor HaiNV21

Content:

1) A Quick Guide to Using Keycloak with Spring Boot

2) Guide to Mustache with Spring Boot

3) Getting Started with GraphQL and Spring Boot

4) Apache Camel with Spring Boot

5) DynamoDB in a Spring Boot Application Using Spring Data

6) Spring Boot Configuration with Jasypt

7) JHipster

What is “Integration with other libraries” in context of Spring Boot

- The concept of Integration With Other Libraries refers to combining a Spring Boot app with external libraries. This allows us to use functions and features that Spring Boot does not provide, or take advantage of existing libraries to solve specific problems. For example:

+ Hibernate: For working with databases and object management (ORM).
+ UI library: Thymeleaf, Freemarker, or React to build user interfaces.
+ Security library: Use Spring Security to authenticate and authorize users.
+ External API calling library: Use RestTemplate or Feign to call APIs from other services.

- Integration with other libraries helps us build more powerful and flexible Spring Boot applications, using the tools and features available in these libraries.

A Quick Guide to Using Keycloak with Spring Boot

Keycloak is an Identity and Access Management (IAM) open-source for modern applications and services. It is used to manage authentication and access rights. It provides both SAML and OpenID protocol solutions.

- About SAML (Security Assertion Markup Language) protocol:

+ It is a special protocol for sharing credentials and access rights between different organizations.

+ Using SAML, users can sign in to one application and access other applications without having to sign in again.

For example: A company has an internal application system, including an HR application and a project management application. Instead of requiring users to log in to each application individually, they could implement a SAML system where a single login would provide access to both applications.

A Quick Guide to Using Keycloak with Spring Boot

- About OpenID protocol:
+ OpenID is a popular protocol for securely authenticating and authorizing users.
+ Unlike SAML, OpenID is typically used to authenticate users across standalone web applications instead of sharing credentials across organizations.
+ In the OpenID model, one party called OpenID Provider (OP) provides authentication to users, while the other parties are Service Providers (SPs) that use this authentication information to determine the user's identity and provide services to them.
+ For example, an ecommerce website can allow users to log in with a Google account. When a user selects the Google sign-in option, they are redirected to the Google (OpenID Provider) sign-in page, where they enter their sign-in information. Google then provides authentication information to the sales website (Service Provider), allowing users to access the website without having to create a new account.

A Quick Guide to Using Keycloak with Spring Boot

Here are some important points about Keycloak:

- Authentication and authorization: Keycloak supports user authentication and provides access based on role and authority. We can use it to authenticate users through protocols like OpenID Connect.

- Single Sign-On (SSO): Keycloak allows you to build an SSO solution, helping users just sign in once and then be able to access multiple applications without having to sign in again.

- Manage users and roles: We can create and manage users, roles, and permissions in Keycloak. It provides a web interface to perform these operations.

- Integration with Spring Boot: Keycloak supports integration with Spring Boot through libraries and configuration. We can use Keycloak to protect our Spring Boot applications.

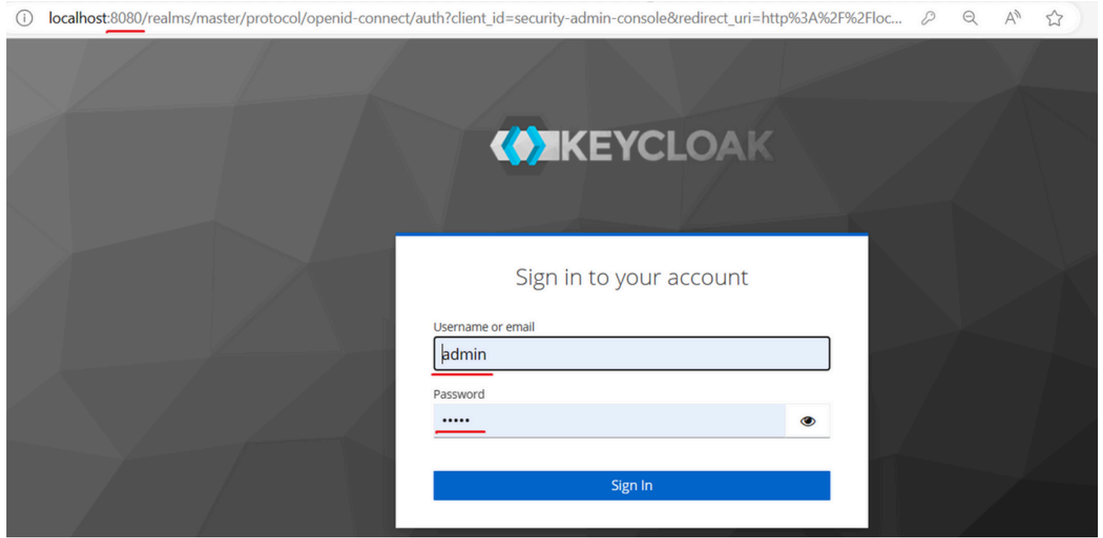
A Quick Guide to Using Keycloak with Spring Boot

1) Start Keycloak (With Docker - Optional)

```
PS C:\Users\TRUNG KIEN> docker run -p 8080:8080 -e KEYCLOAK_ADMIN=admin -e KEYCLOAK_ADMIN_PASSWORD=admin quay.io/keycloa
k/keycloak:24.0.2 start-dev
Unable to find image 'quay.io/keycloak/keycloak:24.0.2' locally
24.0.2: Pulling from keycloak/keycloak
82e56b4fb992: Pull complete
7c2aa5ccfe2e: Pull complete
b15c2c0d9829: Pull complete
be3b074cd0ad: Pull complete
Digest: sha256:b8a3f00fc433f2999bc304b73df331e2005037e8f5673f644f9c0eacd5fbe048
Status: Downloaded newer image for quay.io/keycloak/keycloak:24.0.2
Updating the configuration and installing your custom providers, if any. Please wait.
2024-04-03 04:07:53,548 INFO  [io.qua.dep.QuarkusAugmentor] (main) Quarkus augmentation completed in 10396ms
2024-04-03 04:07:57,524 WARN  [org.infinispan.CONFIG] (keycloak-cache-init) ISPN000569: Unable to persist Infinispan int
ernal caches as no global state enabled
2024-04-03 04:07:57,728 INFO  [org.infinispan.CONTAINER] (keycloak-cache-init) ISPN000556: Starting user marshaller 'org
.infinispan.jboss.marshalling.core.JBossUserMarshaller'
2024-04-03 04:07:58,022 INFO  [org.keycloak.quarkus.runtime.hostname.DefaultHostnameProvider] (main) Hostname settings:
Base URL: <unset>, Hostname: <request>, Strict HTTPS: false, Path: <request>, Strict BackChannel: false, Admin URL: <uns
et>, Admin: <request>, Port: -1, Proxied: false
2024-04-03 04:08:02,141 WARN  [io.quarkus.agroal.runtime.DataSources] (JPA Startup Thread) Datasource <default> enables
XA but transaction recovery is not enabled. Please enable transaction recovery by setting quarkus.transaction-manager.en
able-recovery=true, otherwise data may be lost if the application is terminated abruptly
2024-04-03 04:08:04,239 INFO  [org.keycloak.broker.provider.AbstractIdentityProviderMapper] (main) Registering class org
.keycloak.broker.provider.mappersync.ConfigSyncEventListener
2024-04-03 04:08:06,230 INFO  [org.keycloak.quarkus.runtime.storage.legacy.liquibase.QuarkusJpaUpdaterProvider] (main) I
nitializing database schema. Using changelog META-INF/jpa-changelog-master.xml

UPDATE SUMMARY
Run:                        124
Previously run:               0
```

"docker run -p 8080:8080 -e KEYCLOAK_ADMIN=admin -e KEYCLOAK_ADMIN_PASSWORD=admin quay.io/keycloak/keycloak:24.0.2 start-dev"

- This command starts Keycloak exposed on the local port 8080 and creates an initial admin user with the username admin and password admin.

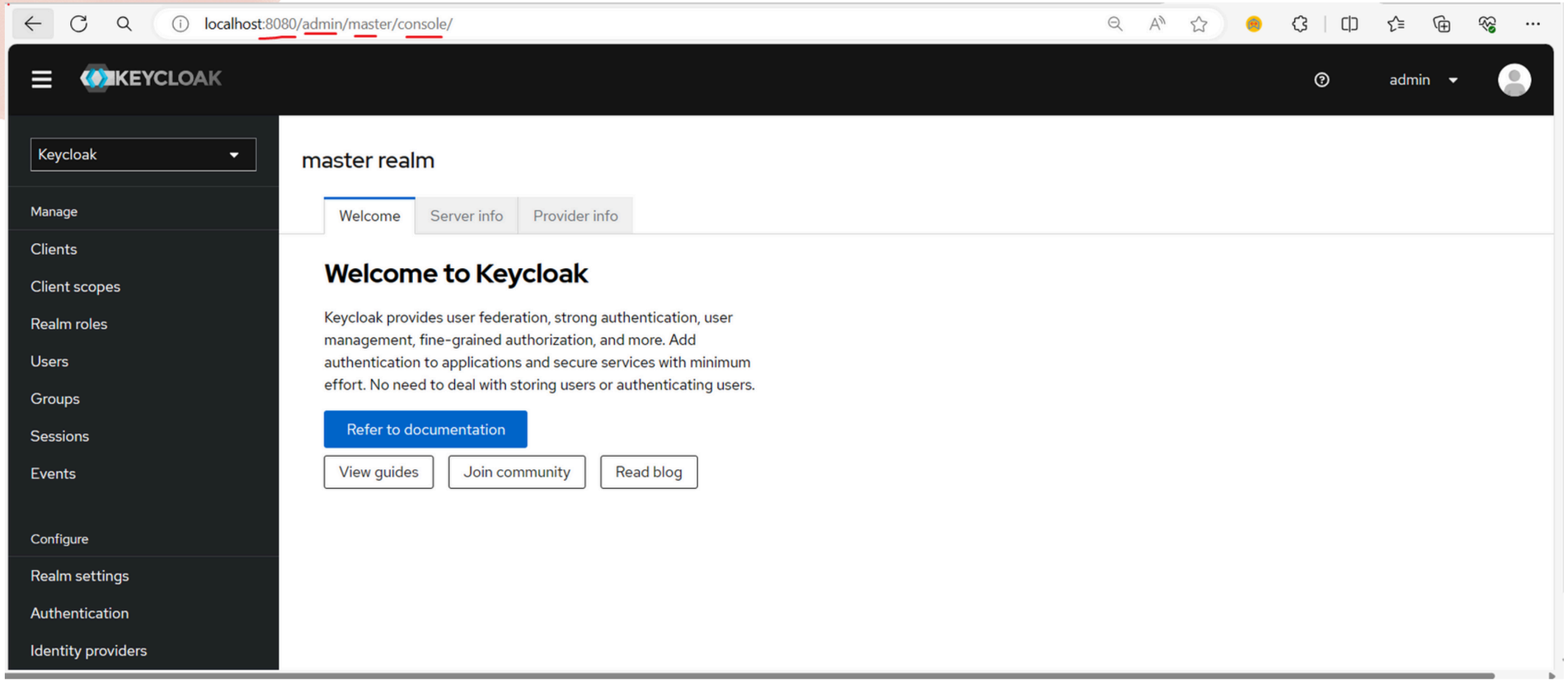


Access port 8080 on localhost to login admin console

A Quick Guide to Using Keycloak with Spring Boot

1) Start Keycloak

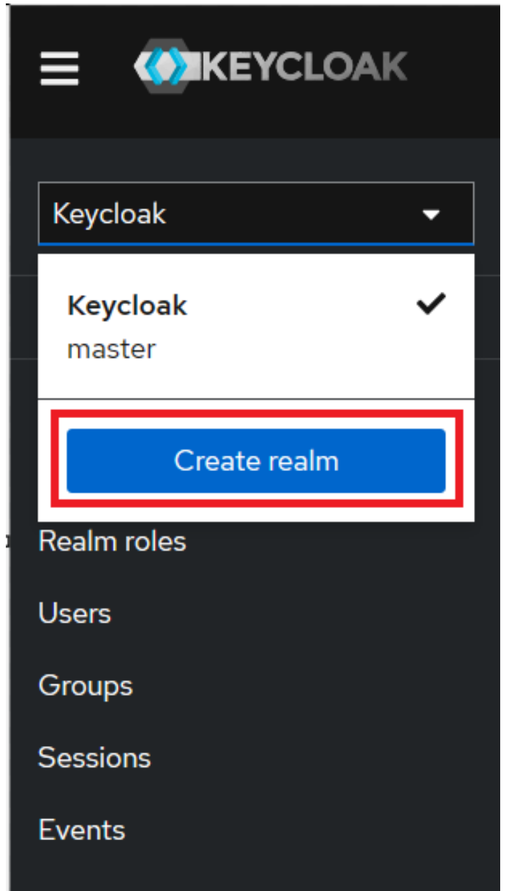
After login successfully, here is the UI of admin console:

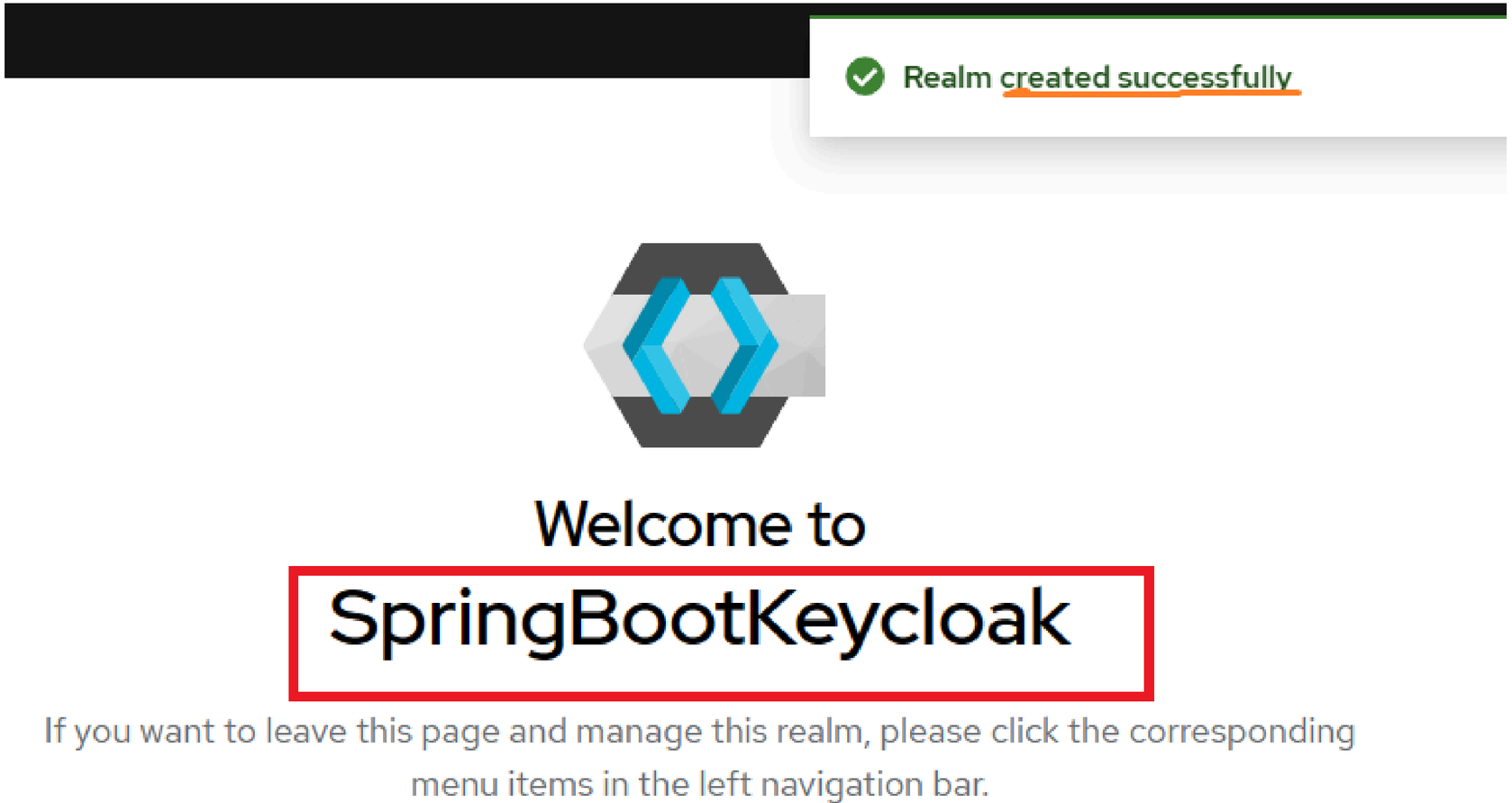


A Quick Guide to Using Keycloak with Spring Boot

2) Create Realm

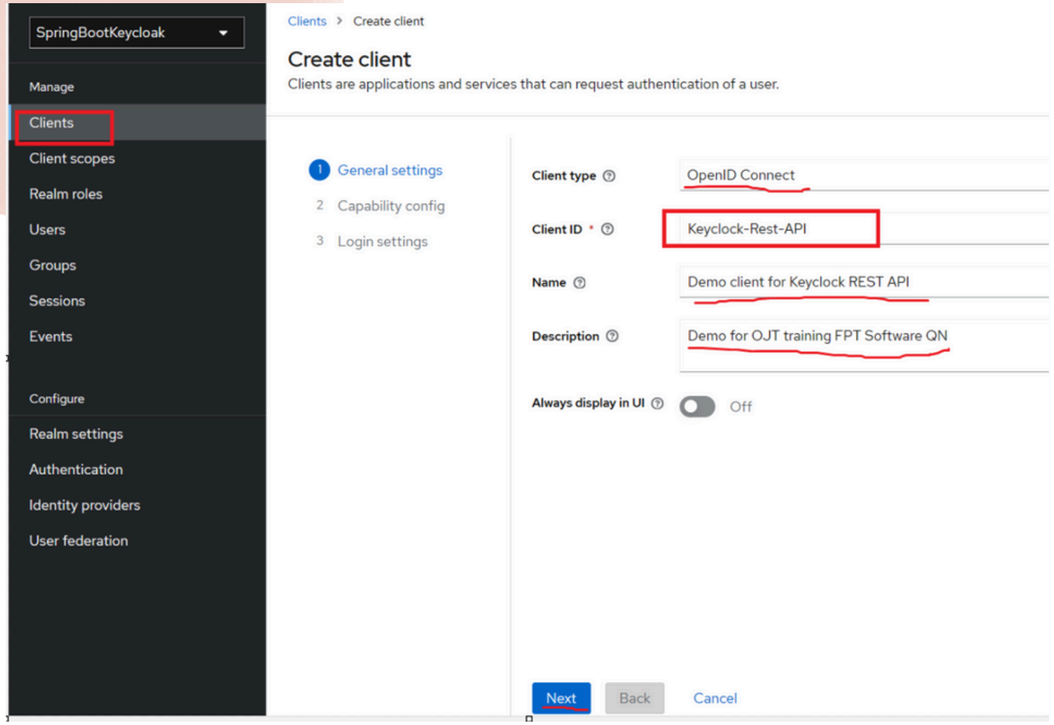
Realm in Keycloak acts as a security scope. It allows us to manage users, roles, and other resources in a specific space. In this realm, we will add our Spring Boot application as a client

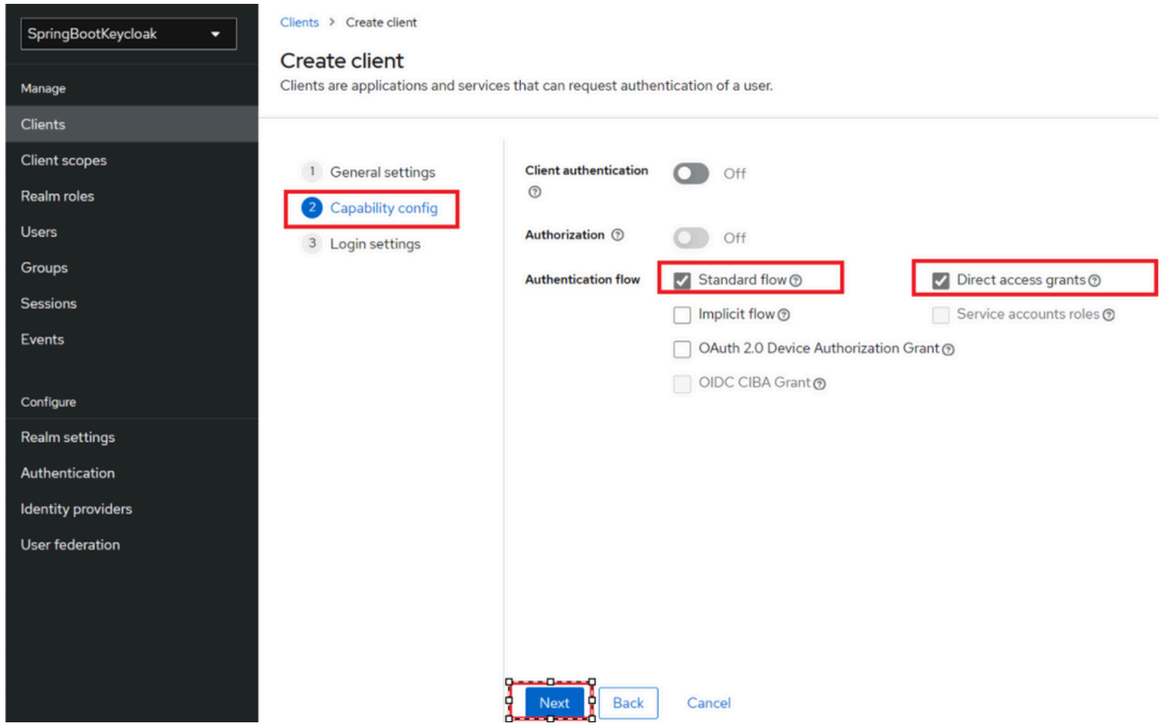




A Quick Guide to Using Keycloak with Spring Boot

3) Create Client

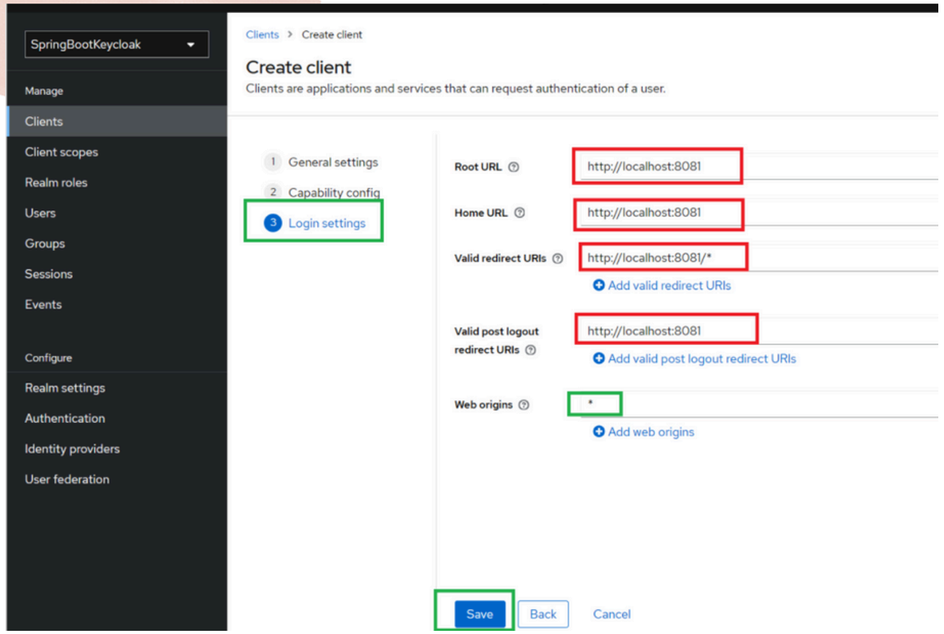


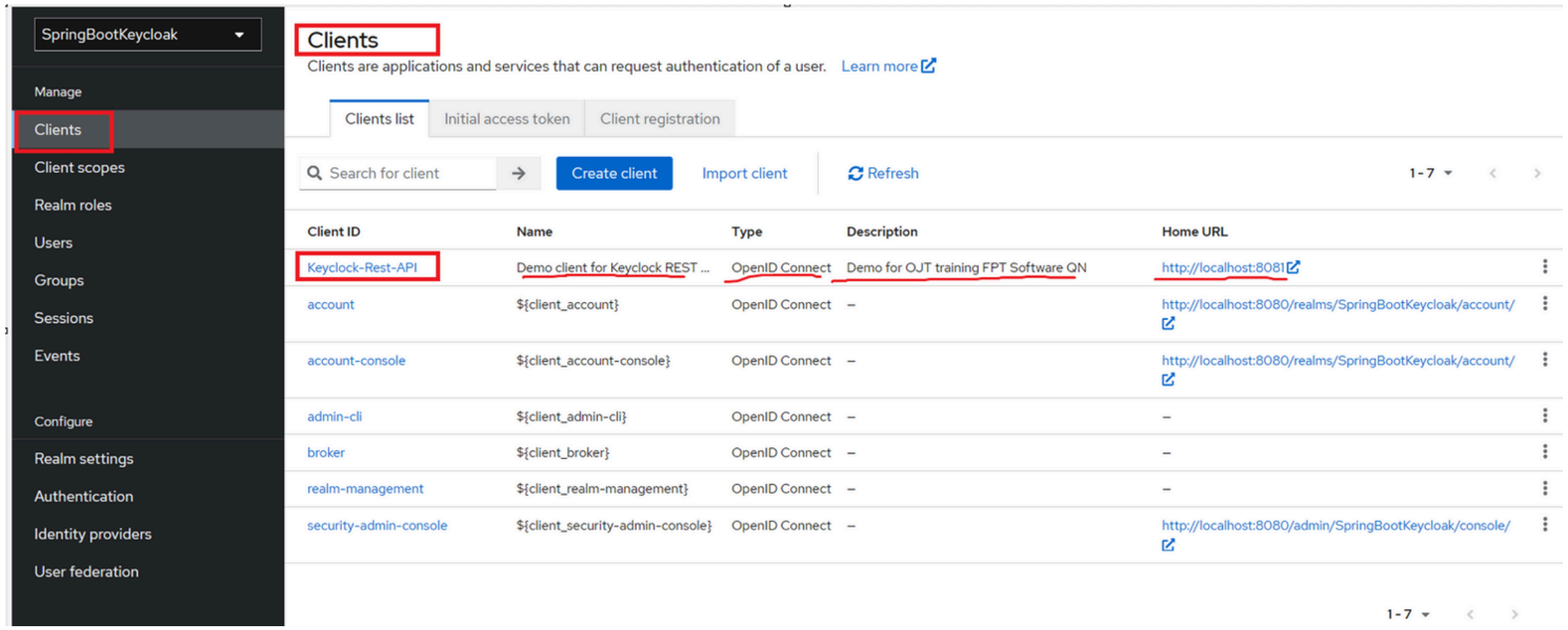


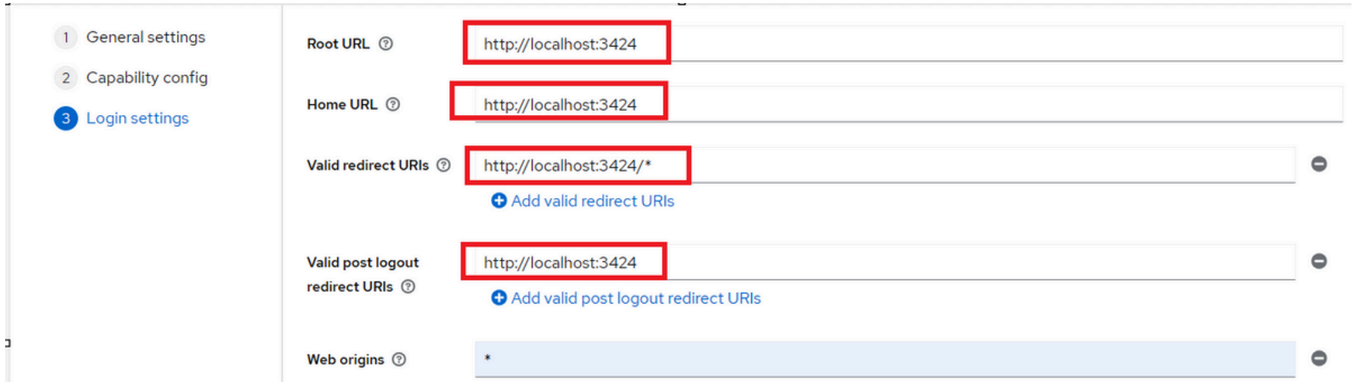
A Quick Guide to Using Keycloak with Spring Boot

3) Create Client

Check Client was created successfully or not

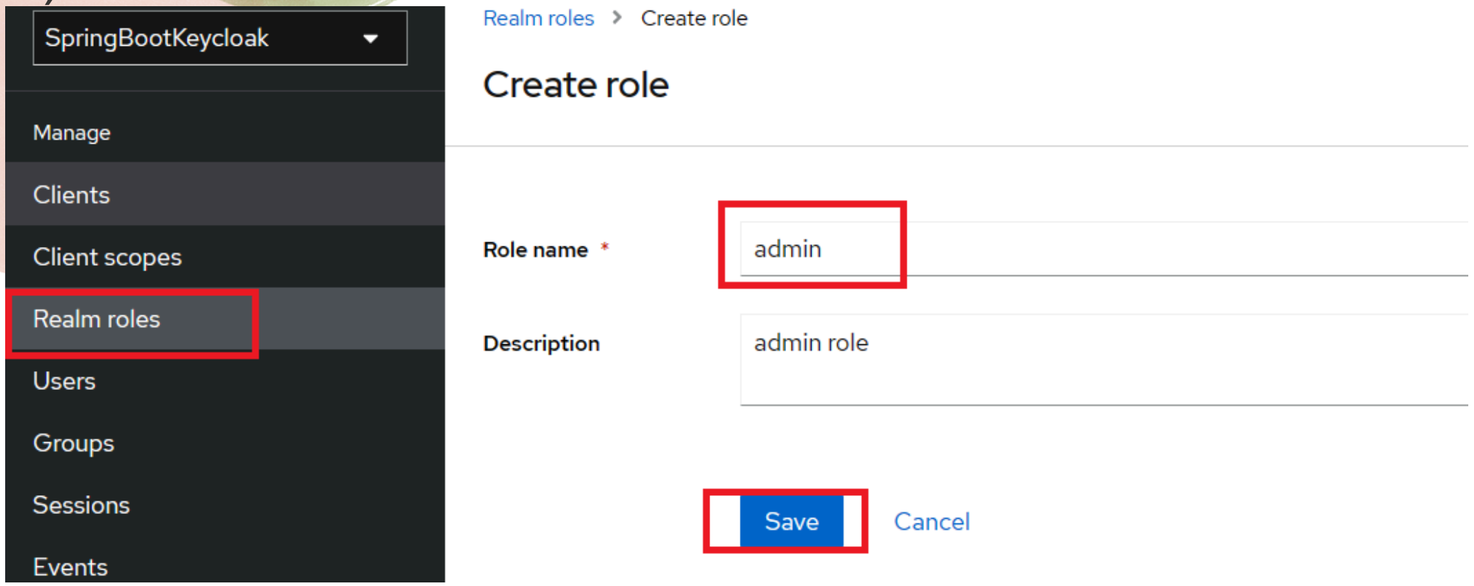


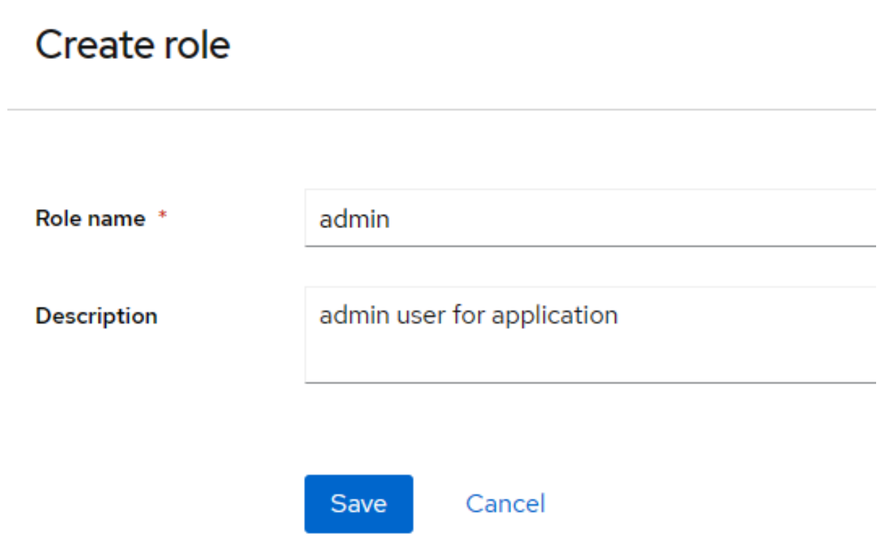




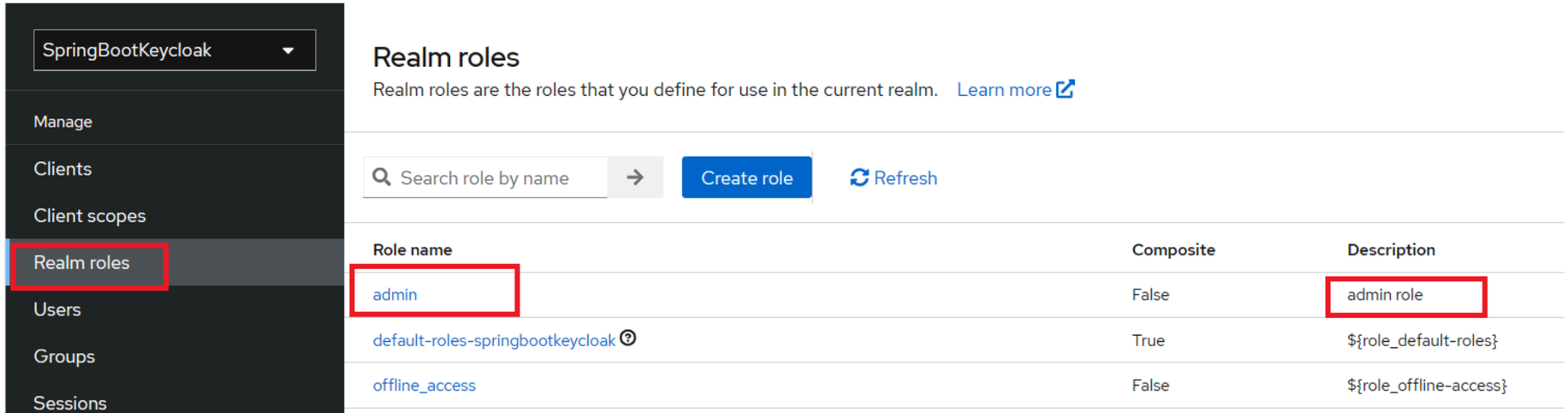
A Quick Guide to Using Keycloak with Spring Boot

4) Create a Role



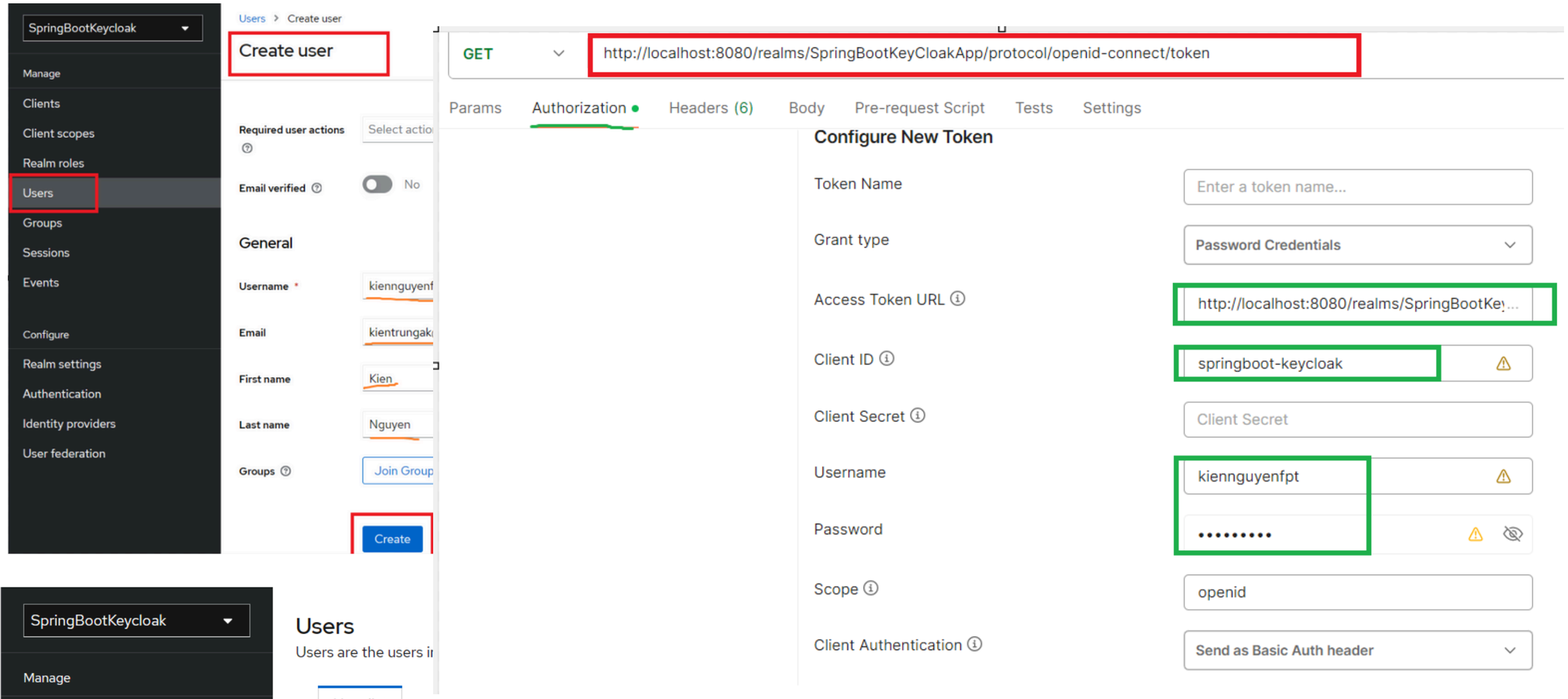


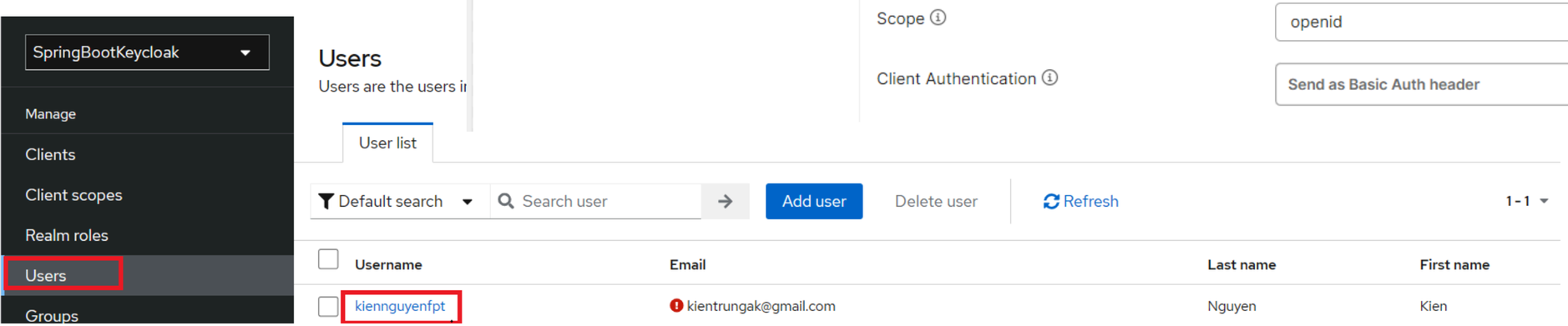
Check role was created successfully or not



4) Create a User

Now that we've set up most of our environment for our Keycloak server, it's time to establish a new Keycloak user





Check result:

4) Create a User

kiennguyenfpt

Details | Credentials | Role mapping | Groups | Consents | Identity provider links | Sessions

Search by name | Hide inherited roles | Assign role | Unassign | Refresh

Name	Inherited	Description
default-roles-springbootkeycloak	False	${role_default-roles}

Then we assign role to User

Assign roles to kiennguyenfpt

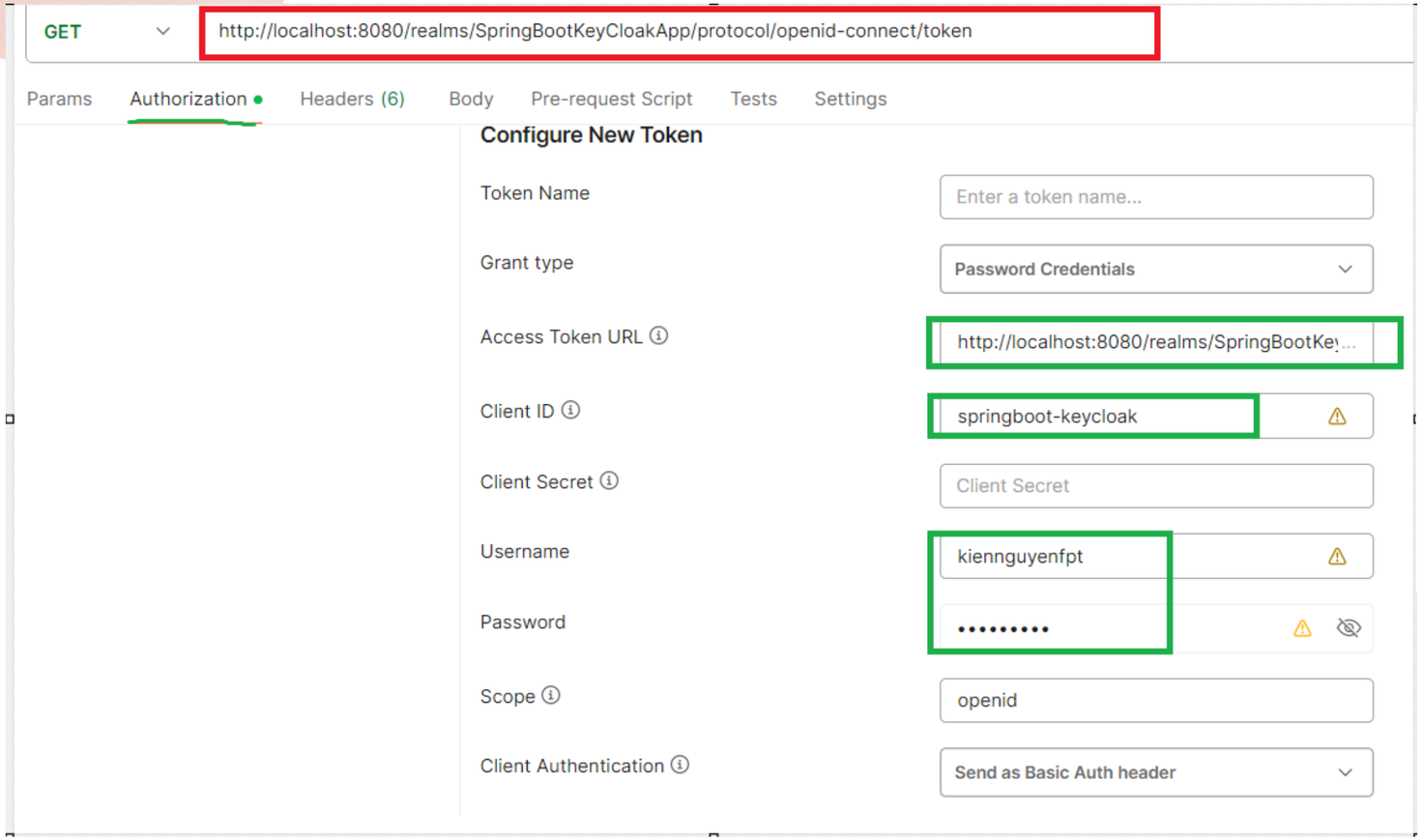
Filter by realm roles | Search by role name | Refresh | 1 - 3

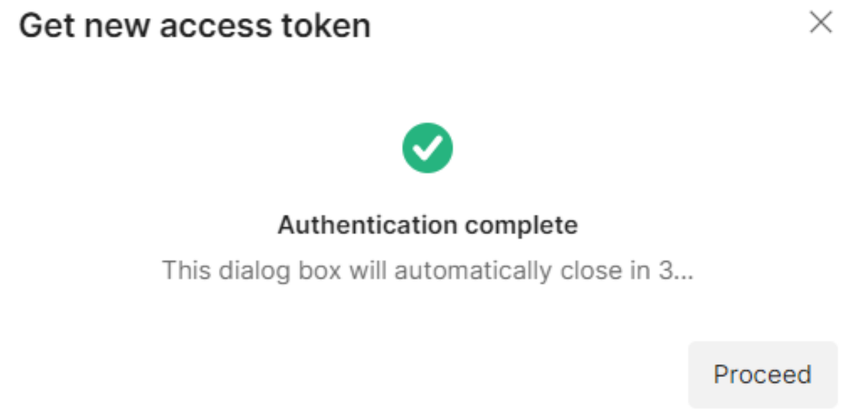
Name	Description
admin	admin role
offline_access	${role_offline-access}
uma_authorization	${role_uma_authorization}

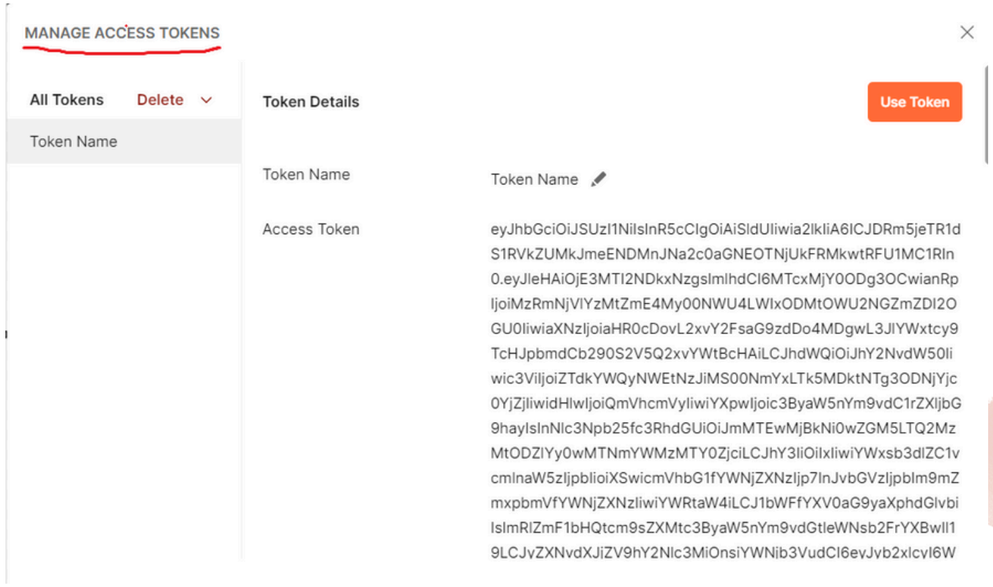
1 - 3

Assign | Cancel

5) Generate JWT



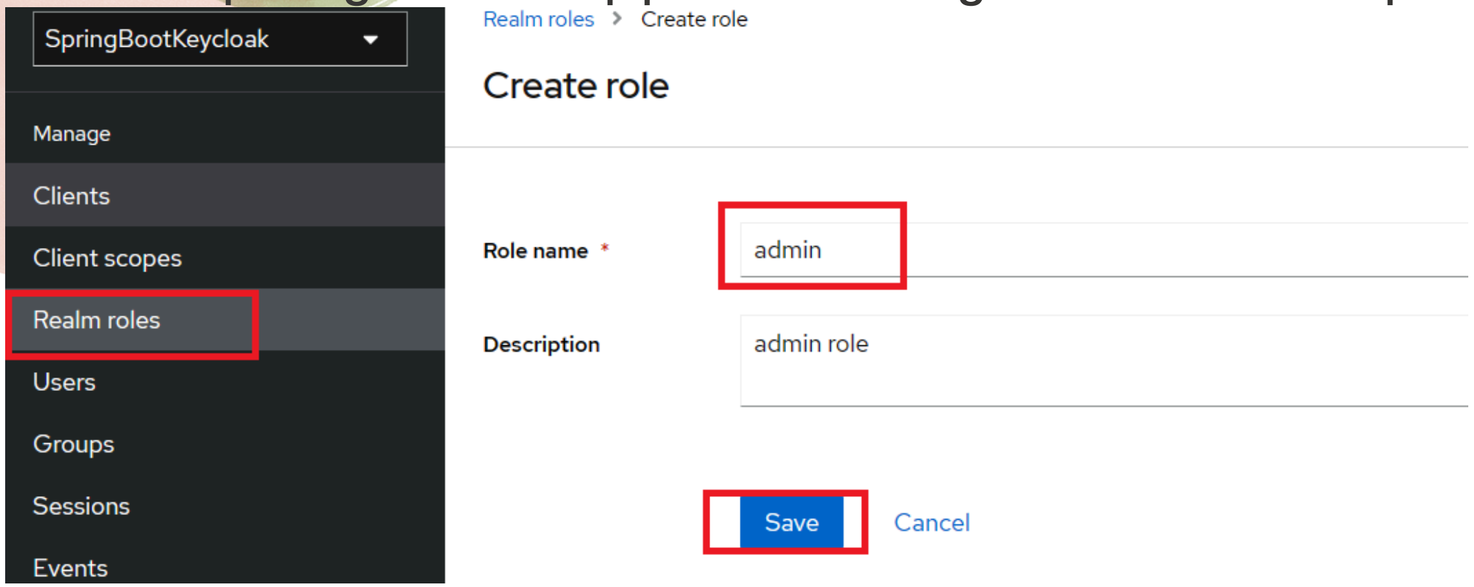




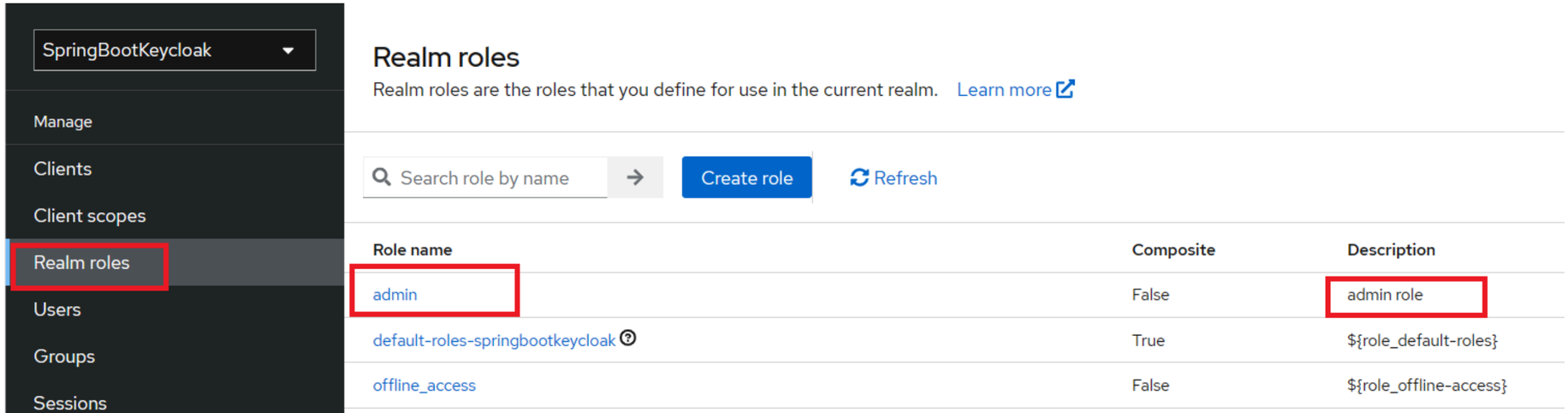
Then we assign role to User

A Quick Guide to Using Keycloak with Spring Boot

Build Spring Boot App that integration with Spring Boot



Check role was created successfully or not



Guide to Mustache with Spring Boot

- Mustache is a simple yet powerful website template library, used to design interfaces for web applications.
- Mustache provides devs with an easy way to design website interfaces without having to learn a complicated template language.

- In Spring Boot, Mustache is used to design the interface for websites. When integrated with Spring Boot, Mustache works using template files stored in the resources/templates directory.
- These template files can be used to create dynamic web pages, allowing users to interact with the application.
- Mustache also provides programmers with a way to integrate data into the web page, helping with display data becomes easier.

Guide to Mustache with Spring Boot

Dependency:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mustache</artifactId>
</dependency>
```

Guide to Mustache with Spring Boot

Dependency:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mustache</artifactId>
</dependency>
```

Getting Started with GraphQL and Spring Boot

- GraphQL is a query language and runtime environment developed by Facebook, aimed at optimizing data querying in web and mobile applications.
- GraphQL allows us to describe how data will be returned from server to client more flexibly and precisely than traditional RESTful APIs.

For example of RESTful API:

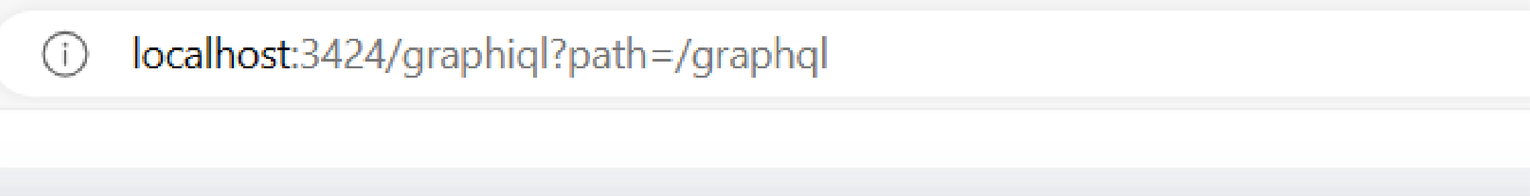
+ GET /books: Get a list of all books.
+ GET /books/{id}: Get information about a specific book based on ID.
+ POST /books: Create a new book.
+ PUT /books/{id}: Update information about an existing book.
+ DELETE /books/{id}: Delete a book based on ID.

Getting Started with GraphQL and Spring Boot

- GraphQL is a query language and runtime environment developed by Facebook, aimed at optimizing data querying in web and mobile applications.
- GraphQL allows us to describe how data will be returned from server to client more flexibly and precisely than traditional RESTful APIs.

For example of RESTful API:

+ GET /books: Get a list of all books.
+ GET /books/{id}: Get information about a specific book based on ID.
+ POST /books: Create a new book.
+ PUT /books/{id}: Update information about an existing book.
+ DELETE /books/{id}: Delete a book based on ID.

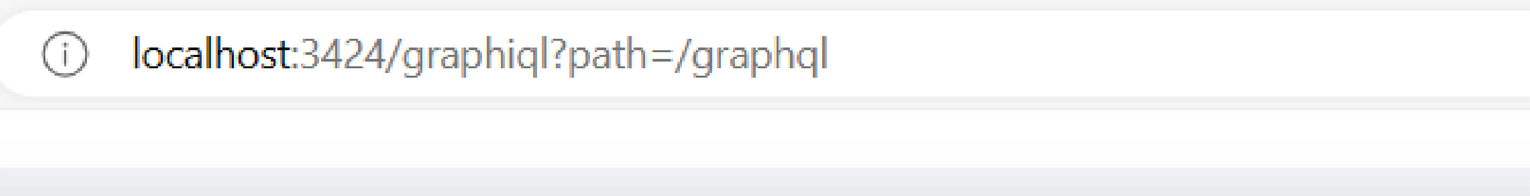


Apache Camel with Spring Boot

- GraphQL is a query language and runtime environment developed by Facebook, aimed at optimizing data querying in web and mobile applications.
- GraphQL allows us to describe how data will be returned from server to client more flexibly and precisely than traditional RESTful APIs.

For example of RESTful API:

+ GET /books: Get a list of all books.
+ GET /books/{id}: Get information about a specific book based on ID.
+ POST /books: Create a new book.
+ PUT /books/{id}: Update information about an existing book.
+ DELETE /books/{id}: Delete a book based on ID.



DynamoDB in a Spring Boot Application Using Spring Data

- DynamoDB is a managed NoSQL database service provided by Amazon Web Services (AWS), meaning it does not follow the relational model of traditional SQL databases.
- Instead, it uses a flexible data model, has no fixed schema, and supports structured, semi-structured, and unstructured data.
- It is designed to provide high performance, availability, and flexible scalability for web and mobile applications with large data requirements.

Spring Boot Configuration with Jasypt

- DynamoDB is a managed NoSQL database service provided by Amazon Web Services (AWS), meaning it does not follow the relational model of traditional SQL databases.
- Instead, it uses a flexible data model, has no fixed schema, and supports structured, semi-structured, and unstructured data.
- It is designed to provide high performance, availability, and flexible scalability for web and mobile applications with large data requirements.

Introduce to JHipster (combine with REST SQL)

- Definition of JHipster

Introduce to JHipster

1) Install JHipster: use this command npm install -g generator-jhipster

```
C:\Users\TRUNG KIEN>npm install -g generator-jhipster
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated uuid@3.4.0: Please upgrade  to version 7 or higher.  Older versions may use Math.random() in certain
 circumstances, which is known to be problematic.  See https://v8.dev/blog/math-random for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142

added 846 packages in 1m

230 packages are looking for funding
  run `npm fund` for details
npm notice
npm notice New minor version of npm available! 10.2.4 -> 10.5.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.5.1
npm notice Run npm install -g npm@10.5.1 to update!
npm notice
```

Introduce to JHipster

1) Install JHipster: use this command npm install -g generator-jhipster

```
C:\Users\TRUNG KIEN\Jhipster_demo>jhipster

                    https://www.jhipster.tech
Welcome to JHipster v8.2.1

 ------------------------------------------------------------------------------------------------------

  Documentation for creating an application is at https://www.jhipster.tech/creating-an-app/

  Application files will be generated in folder: C:\Users\TRUNG KIEN\Jhipster_demo
 ------------------------------------------------------------------------------------------------------

? What is the base name of your application? mySpringBootApp
? Which *type* of application would you like to create? Monolithic application (recommended for simple projects)
? Besides Junit, which testing frameworks would you like to use?
? Do you want to make it reactive with Spring WebFlux? No
? What is your default Java package name? com.knfpt.jhipster
? Which *type* of authentication would you like to use? JWT authentication (stateless, with a token)
? Which *type* of database would you like to use? SQL (H2, PostgreSQL, MySQL, MariaDB, Oracle, MSSQL)
? Which *production* database would you like to use? MySQL
? Which *development* database would you like to use? H2 with in-memory persistence
? Which cache do you want to use? (Spring cache abstraction) No cache - Warning, when using an SQL database, this will
disable the Hibernate 2nd level cache!
? Would you like to use Maven or Gradle for building the backend? Maven
? Which other technologies would you like to use?
? Which *Framework* would you like to use for the client? Angular
? Besides Jest/Vitest, which testing frameworks would you like to use?
? Do you want to generate the admin UI? Yes
? Would you like to use a Bootswatch theme (https://bootswatch.com/)? Default JHipster
INFO! Disabling hibernate cache for cache provider no
? Would you like to enable internationalization support? No
? Please choose the native language of the application English
? Please choose additional languages to install
```

Introduction of JHipster

```
C:\Users\TRUNG KIEN\Jhipster_demo>mvnw
[INFO] Scanning for projects...
[INFO]
[INFO] ---------------< com.knfpt.jhipster:my-spring-boot-app >----------------
[INFO] Building My Spring Boot App 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] >>> spring-boot:3.2.3:run (default-cli) > test-compile @ my-spring-boot-app >>>
[INFO]
[INFO] --- enforcer:3.4.1:enforce (enforce-versions) @ my-spring-boot-app ---
[INFO] Rule 0: org.apache.maven.enforcer.rules.version.RequireMavenVersion passed
[INFO] Rule 1: org.apache.maven.enforcer.rules.version.RequireJavaVersion passed
[INFO]
[INFO] --- enforcer:3.4.1:enforce (enforce-dependencyConvergence) @ my-spring-boot-app ---
[INFO] Rule 0: org.apache.maven.enforcer.rules.dependency.DependencyConvergence passed
[INFO]
[INFO] --- resources:3.3.1:copy-resources (config-resources) @ my-spring-boot-app ---
[INFO] Copying 4 resources from src\main\resources to target\classes
[INFO]
```

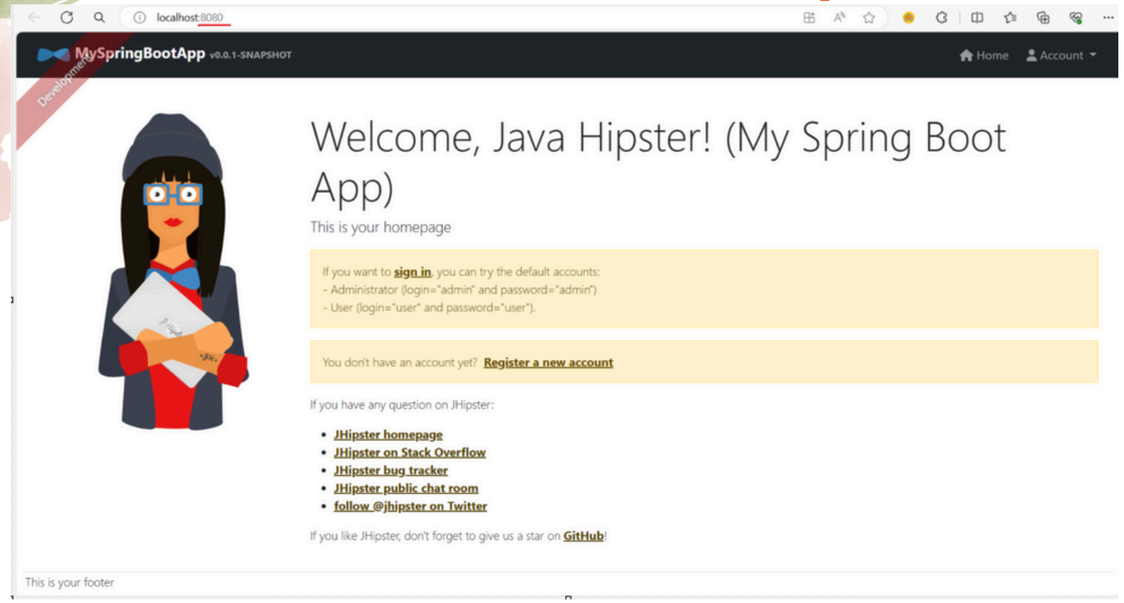
Use command mvnw to build and run anything

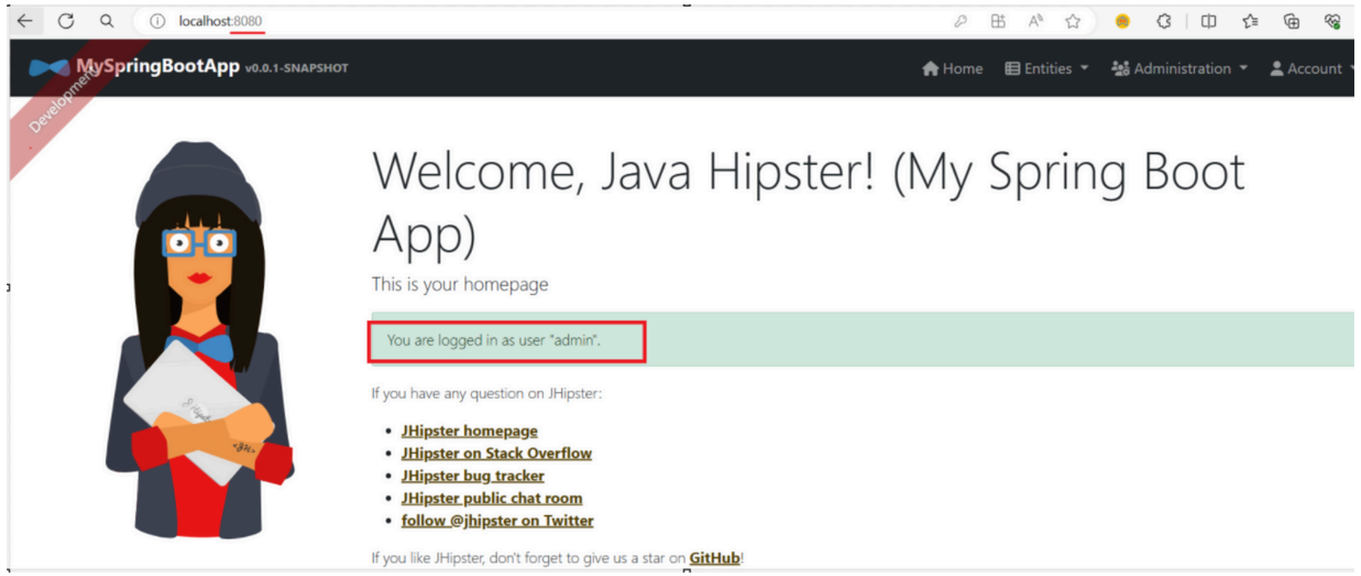
Result:

```
----------------------------------------------------------
        Application 'mySpringBootApp' is running! Access URLs:
        Local:          http://localhost:8080/
        External:       http://192.168.112.180:8080/
        Profile(s):     [dev, api-docs]
----------------------------------------------------------
```

Introduction of JHipster

Result:





Use command mvnw to build and run anything

Introduction of JHipster

Add new entity to Jhipster

```
C:\Users\TRUNG KIEN\Jhipster_demo>jhipster entity author

                    https://www.jhipster.tech
Welcome to JHipster v8.2.1

 ------------------------------------------------------------------------------

  Documentation for creating an application is at https://www.jhipster.tech/creating-an-app/

  Application files will be generated in folder: C:\Users\TRUNG KIEN\Jhipster_demo
 ------------------------------------------------------------------------------

INFO!
The entity Author is being created.


Generating field #1

? Do you want to add a field to your entity? Yes
? What is the name of your field? name
? What is the type of your field? String
? Do you want to add validation rules to your field? No

================= Author =================
Fields
name (String)
```

```
Generating field #3

? Do you want to add a field to your entity? Yes
? What is the name of your field? address
? What is the type of your field? String
? Do you want to add validation rules to your field? No

================= Author =================
Fields
name (String)
age (Integer)
address (String)


Generating field #4

? Do you want to add a field to your entity? Yes
? What is the name of your field? phone
? What is the type of your field? String
? Do you want to add validation rules to your field? No

================= Author =================
Fields
name (String)
age (Integer)
address (String)
phone (String)


Generating field #5

? Do you want to add a field to your entity? No

================= Author =================
Fields
name (String)
age (Integer)
address (String)
phone (String)


Generating relationships to other entities

? Do you want to add a relationship to another entity? No

================= Author =================
Fields
name (String)
age (Integer)
address (String)
phone (String)
```

Use command mvnw to build and run anything

Introduction of JHipster

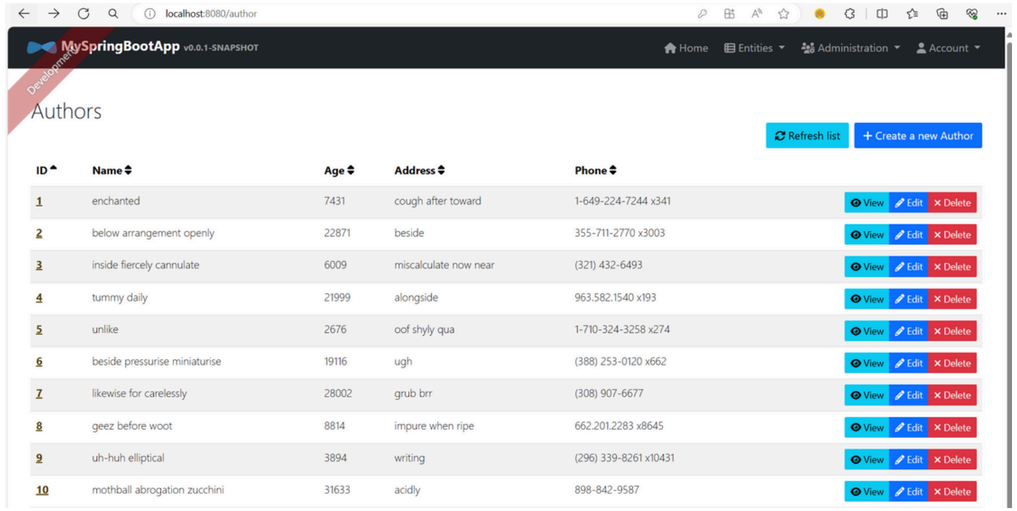
Add new entity to Jhipster

```
No change to package.json was detected. No package manager install will be executed.
Entity undefined generated successfully.
√ Application successfully committed to Git from C:/Users/TRUNG KIEN.
√ Spring Boot application generated successfully.
  Run your Spring Boot application:
  ./mvnw (mvnw if using Windows Command Prompt)
√ Angular application generated successfully.
  Start your Webpack development server with:
  npm start


Congratulations, JHipster execution is complete!
If you find JHipster useful consider sponsoring the project https://www.jhipster.tech/sponsors/

Thanks for using JHipster!
```

Use command mvnw to build and run anything

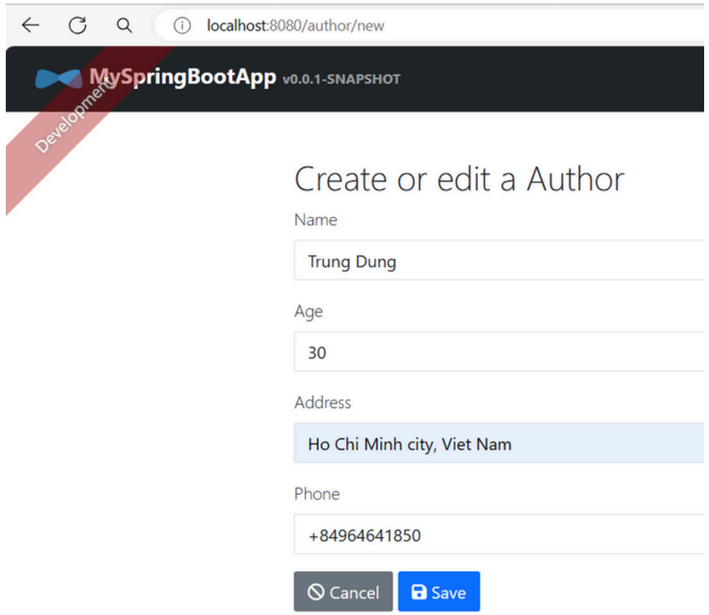
Introduction of JHipster

Here is the UI that was generated by Jhipster, and now we will perform CRUD operations:



ID	Name	Age	Address	Phone	
1	enchanted	7431	cough after toward	1-649-224-7244 x341	View Edit Delete
2	below arrangement openly	22871	beside	355-711-2770 x3003	View Edit Delete
3	inside fiercely cannulate	6009	miscalculate now near	(321) 432-6493	View Edit Delete
4	tummy daily	21999	alongside	963.582.1540 x193	View Edit Delete
5	unlike	2676	oof shyly qua	1-710-324-3258 x274	View Edit Delete
6	beside pressurise miniaturise	19116	ugh	(388) 253-0120 x662	View Edit Delete
7	likewise for carelessly	28002	grub brr	(308) 907-6677	View Edit Delete
8	geez before woot	8814	impure when ripe	662.201.2283 x8645	View Edit Delete
9	uh-huh elliptical	3894	writing	(296) 339-8261 x10431	View Edit Delete
10	mothball abrogation zucchini	31633	acidly	898-842-9587	View Edit Delete

Add new author:

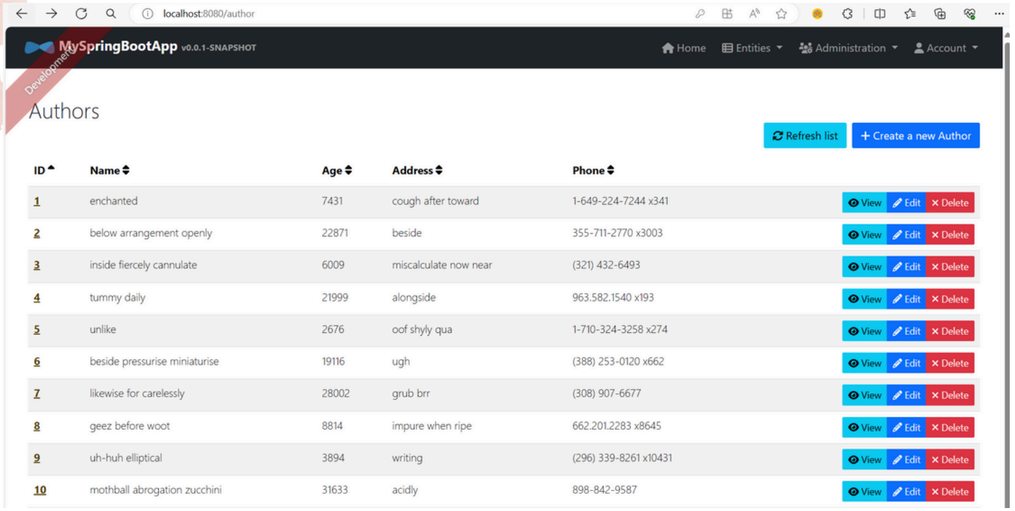


Result after add successfully:

ID	Name	Age	Address	Phone	
1	enchanted	7431	cough after toward	1-649-224-7244 x341	View Edit Delete
2	below arrangement openly	22871	beside	355-711-2770 x3003	View Edit Delete
3	inside fiercely cannulate	6009	miscalculate now near	(321) 432-6493	View Edit Delete
4	tummy daily	21999	alongside	963.582.1540 x193	View Edit Delete
5	unlike	2676	oof shyly qua	1-710-324-3258 x274	View Edit Delete
6	beside pressurise miniaturise	19116	ugh	(388) 253-0120 x662	View Edit Delete
7	likewise for carelessly	28002	grub brr	(308) 907-6677	View Edit Delete
8	geez before woot	8814	impure when ripe	662.201.2283 x8645	View Edit Delete
9	uh-huh elliptical	3894	writing	(296) 339-8261 x10431	View Edit Delete
10	mothball abrogation zucchini	31633	acidly	898-842-9587	View Edit Delete
1500	Trung Kien	23	Ho Chi Minh city, Viet Nam	+84964641839	View Edit Delete
1501	Trung Dung	30	Ho Chi Minh city, Viet Nam	+84964641850	View Edit Delete

Introduction of JHipster

Before:

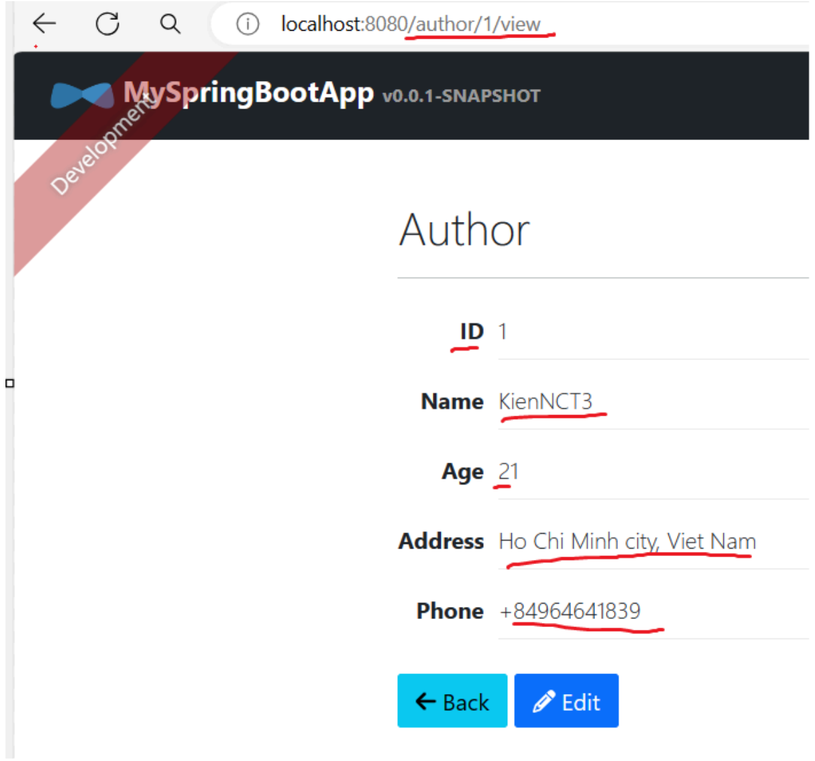


ID	Name	Age	Address	Phone	
1	enchanted	7431	cough after toward	1-649-224-7244 x341	View Edit Delete
2	below arrangement openly	22871	beside	355-711-2770 x3003	View Edit Delete
3	inside fiercely cannulate	6009	miscalculate now near	(321) 432-6493	View Edit Delete
4	tummy daily	21999	alongside	963.582.1540 x193	View Edit Delete
5	unlike	2676	oof shyly qua	1-710-324-3258 x274	View Edit Delete
6	beside pressurise miniaturise	19116	ugh	(388) 253-0120 x662	View Edit Delete
7	likewise for carelessly	28002	grub brr	(308) 907-6677	View Edit Delete
8	geez before woot	8814	impure when ripe	662.201.2283 x8645	View Edit Delete
9	uh-huh elliptical	3894	writing	(296) 339-8261 x10431	View Edit Delete
10	mothball abrogation zucchini	31633	acidly	898-842-9587	View Edit Delete

Update author with id 1:

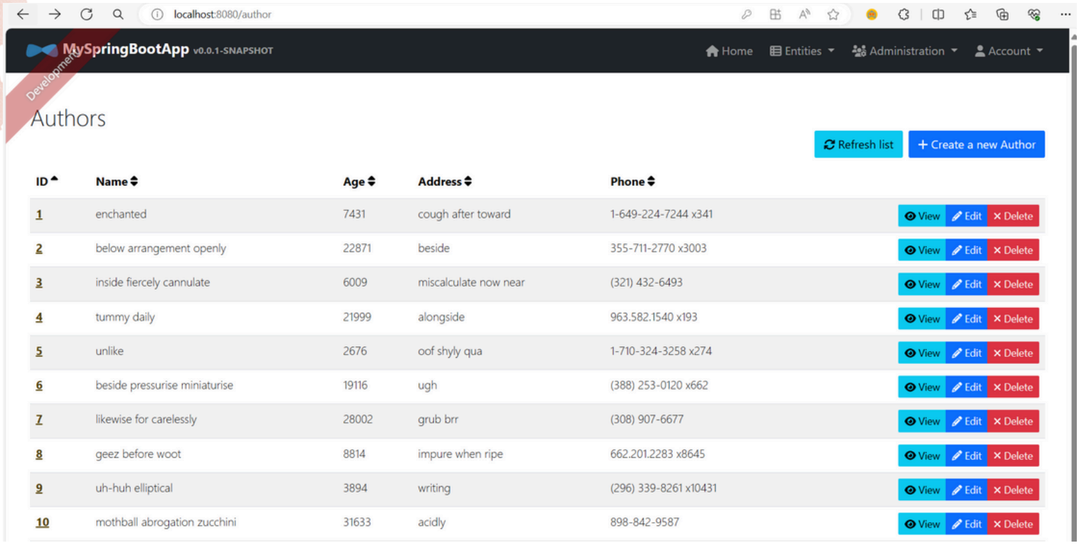
ID	Name	Age	Address	Phone	
1	KienNCT3	21	Ho Chi Minh city, Viet Nam	+84964641839	View Edit Delete
2	below arrangement openly	22871	beside	355-711-2770 x3003	View Edit Delete
3	inside fiercely cannulate	6009	miscalculate now near	(321) 432-6493	View Edit Delete
4	tummy daily	21999	alongside	963.582.1540 x193	View Edit Delete
5	unlike	2676	oof shyly qua	1-710-324-3258 x274	View Edit Delete
6	beside pressurise miniaturise	19116	ugh	(388) 253-0120 x662	View Edit Delete
7	likewise for carelessly	28002	grub brr	(308) 907-6677	View Edit Delete
8	geez before woot	8814	impure when ripe	662.201.2283 x8645	View Edit Delete
9	uh-huh elliptical	3894	writing	(296) 339-8261 x10431	View Edit Delete
1500	Trung Kien	23	Ho Chi Minh city, Viet Nam	+84964641839	View Edit Delete
1501	Trung Dung	30	Ho Chi Minh city, Viet Nam	+84964641850	View Edit Delete

Information of author with id 1:

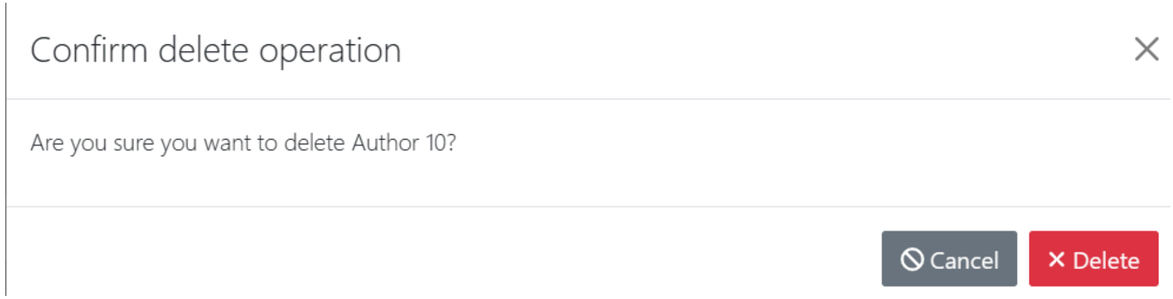
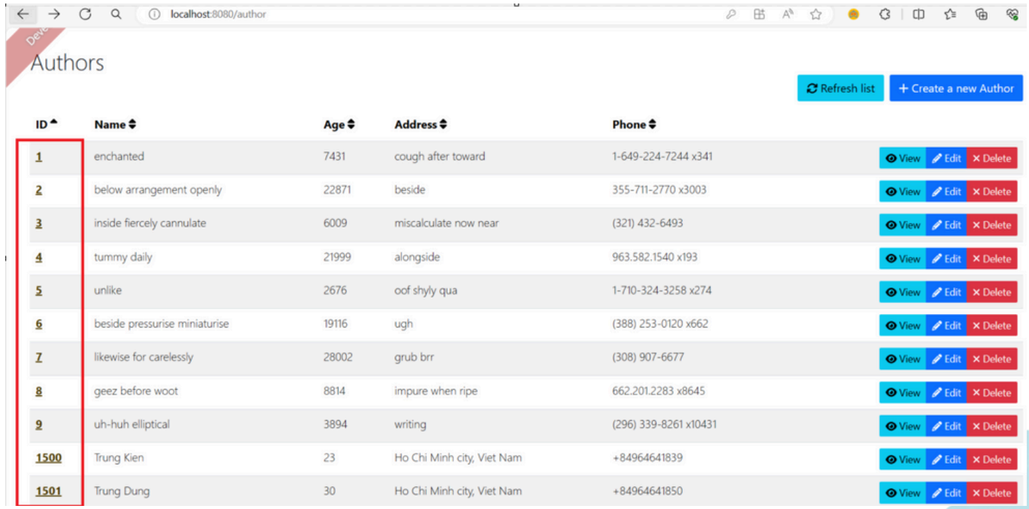


Introduction of JHipster

Before:



Delete author with id 10



Result after delete successfully: