August 2, 2018 | Techies Corner (https://www.comakeit.com/techies-corner/)

# A Quick Guide to Using Keycloak for Identity and Access Management

## IAM(Identity Access Management)

IAM or IdM(Identity Management) is a framework used to authenticate the user identity and privileges. It checks whether the users have access to necessary files, networks and other resources that the user has requested. It also checks how and by whom the information can be accessed and modified by the management of descriptive information of users. IAM systems provide tools and  some technologies to the administrators to change a user’s role, keeping track on user activities etc.

**Identity Management has four main basic functions:**

1. The pure identity function: Without regard to access or entitlements for identity creation, management, and deletion.
2. The user access(log-on) function: For example, to log-on to a service or services(a traditional view) the customer uses a smart card and its associated data.
3. The service function: For user and their devices a system delivers personalized, role-based, online, on-demand, multimedia(content) and presence-based services.
4. Identity Federation: To authenticate a user without knowing his/her password can be done by a system using federated identity.

## Single Sign-On

Single sign-on (SSO) is a property of access control for multiple related and independent software systems where user login with single ID and password to gain access to a connected system/s without different usernames or passwords. SSO is typically accomplished on Lightweight Directory Access Protocol(LDAP) and stored LDAP databases on (directory) servers, an SSO can be achieved over IP networks using cookies but only if the sites share a common DNS parent domain. Shared authentication schemes include OAuth, OpenID, OpenID Connect and Facebook Connect. All the authentication schemes which we are using need user to log in their credentials every time they access a site or application, but we no need to get confused with SSO, in this single sign-on is enough to sign into different applications. By using SSO users can enter their credentials once but always when they are signed.

## Benefits of using SSO

1. Reduce risk for 3rd party sites to access.
2. Reduce password debility from the different username and password combinations.
3. Reduce time spent for re-entering passwords for the same identity.
4. Reduce IT help desk calls for passwords, therefore, IT costs are also reduced.

## Keycloak

Keycloak is an open source identity and access management solution which mainly aims at applications and services. Users can authenticate with Keycloak rather than individual applications. So, the applications don’t have to deal with login forms, authenticating users and storing users. Once logged-in to Keycloak, users don’t have to login again to access different applications. Same thing is applicable to sign-out. Keycloak offers everything a sophisticated user management tool needs – without having to log on repeatedly with every login and into every system-as well as system security, social logins, support for mobile apps and integration into other solutions. Keycloak have implementations to LDAP and Active Directory as well.

## Password Policies

Each new realm created has no password policies associated with it while users can create as short, as long, as complex, as insecure a password, as they want. Simple settings are fine for development or learning Keycloak, but unacceptable in production environments.

## OTP

Keycloak has a number of policies you can set up for your FreeOTP or Google Authenticator One-Time Password generator. There are two types:

1. TOTP(Time based OTP)
2. HOTP(Counter based OTP)

## Client Certificate

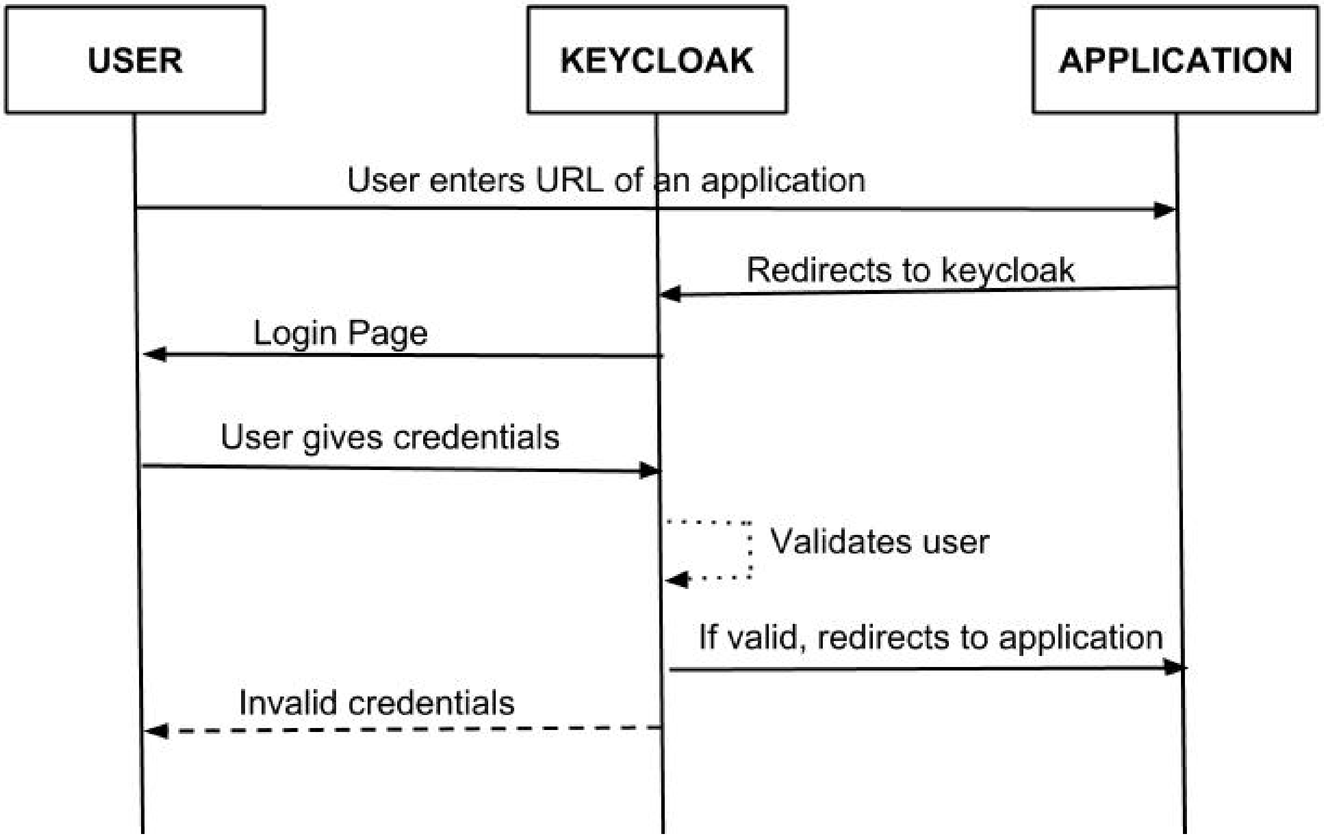
- A client authentication certificate is a certificate used to authenticate clients during an SSL handshake, users who access a server by exchanging the client authentication certificate. A client certificate would typically contain pertinent information like a digital signature, expiration date, name of client, name of CA (Certificate Authority), revocation status, SSL/TLS version number, serial number, and possibly more, all structured using the X.509 standard. Very popular web browsers like Firefox, Chrome, Safari, and Internet Explorer can readily support client certificates.
- If an application is enabled with client certificate authentication, only users who attempt to connect from clients loaded with the right client certificates will succeed. Even if a legitimate user attempts to connect with the right username and password, if that user isn’t on a client application loaded with the right client certificate, that user will not be granted access. In fact, if that user’s connecting from a Web browser, the login page (where he’s supposed to enter his username and password) might not even load at all like the one shown below.
- A server certificate is sent from the server to the client at the start of a session and is used by the client to authenticate the server. A client certificate, on the other hand, is sent from the client to the server at the start of a session and is used by the server to authenticate the client.

## Keycloak Working Procedure

**On a complete system secured with keycloak:**
A user clicks from a public page to navigate to protected area within the application. The link to this protected area is in the application settings in keycloak admin console.

- The user will be redirected indeed to the keycloak authentication page. After providing username and password, keycloak redirects the user back to the application again with a code that is valid to a very short span of time.

- The application communicates this code to keycloak along with the application ID and the application secret, then keycloak replies with the Access token, ID token, and a Refresh token. Your application will need only one of these tokens to see which claims the user has, and according to the claims, the user will be granted or denied access to the requested protected URL(s).

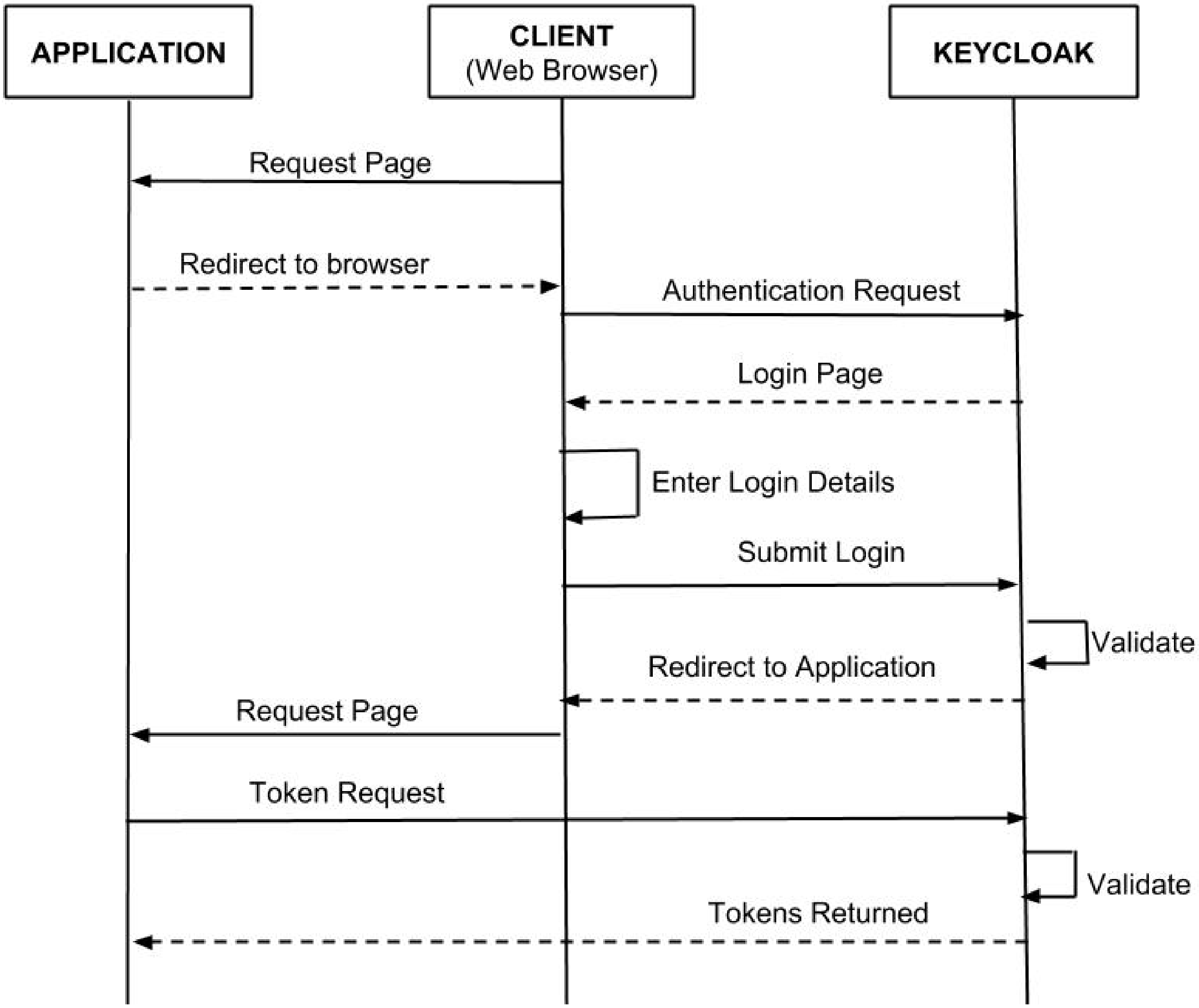


## Keycloak With OpenID Connect(OIDC)

OIDC is an authentication protocol that is an extension of OAuth 2.0. OAuth 3.0 is only a framework for building authorisation protocols, but OIDC is a full-fledged authentication and authorisation protocol. OIDC authentication flow when integrated with keycloak:

- Browser visits application. The application notices the user is not logged in, so it redirects the browser to keycloak to be authenticated. The application passes along a call-back URL(a redirect URL) as a query parameter in this browser redirect that keycloak will use when it finishes authentication.
- Keycloak authenticates the user and creates a one-time, very short lived, temporary code. Keycloak redirects back to the application using the call-back URL provided earlier and additionally adds the temporary code as a query parameter in the call-back URL.

The application extracts the temporary code and makes a background out of band REST invocation to keycloak to exchange the code for an identity, access and refresh token. Once this temporary code has been used to obtain the tokens, it can never be used again. This prevents potential replay attacks.



**Requirements**

The tools that we require are mentioned below:

**Operating systems:** Windows, Docker

**Tools:** Maven

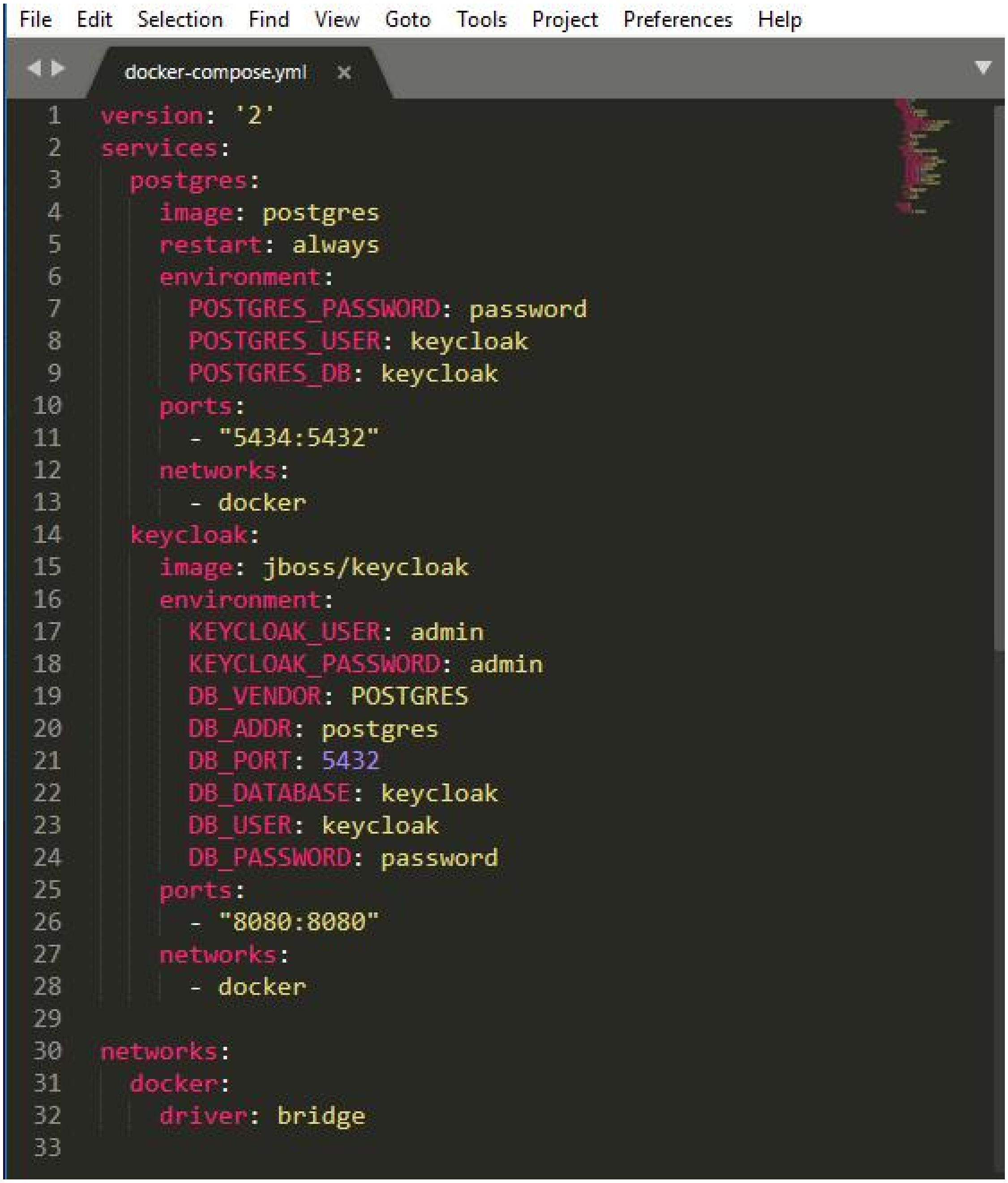
**Single-Sign-On:** Keycloak

**Development environments:** Eclipse

**Programming Language :** Java

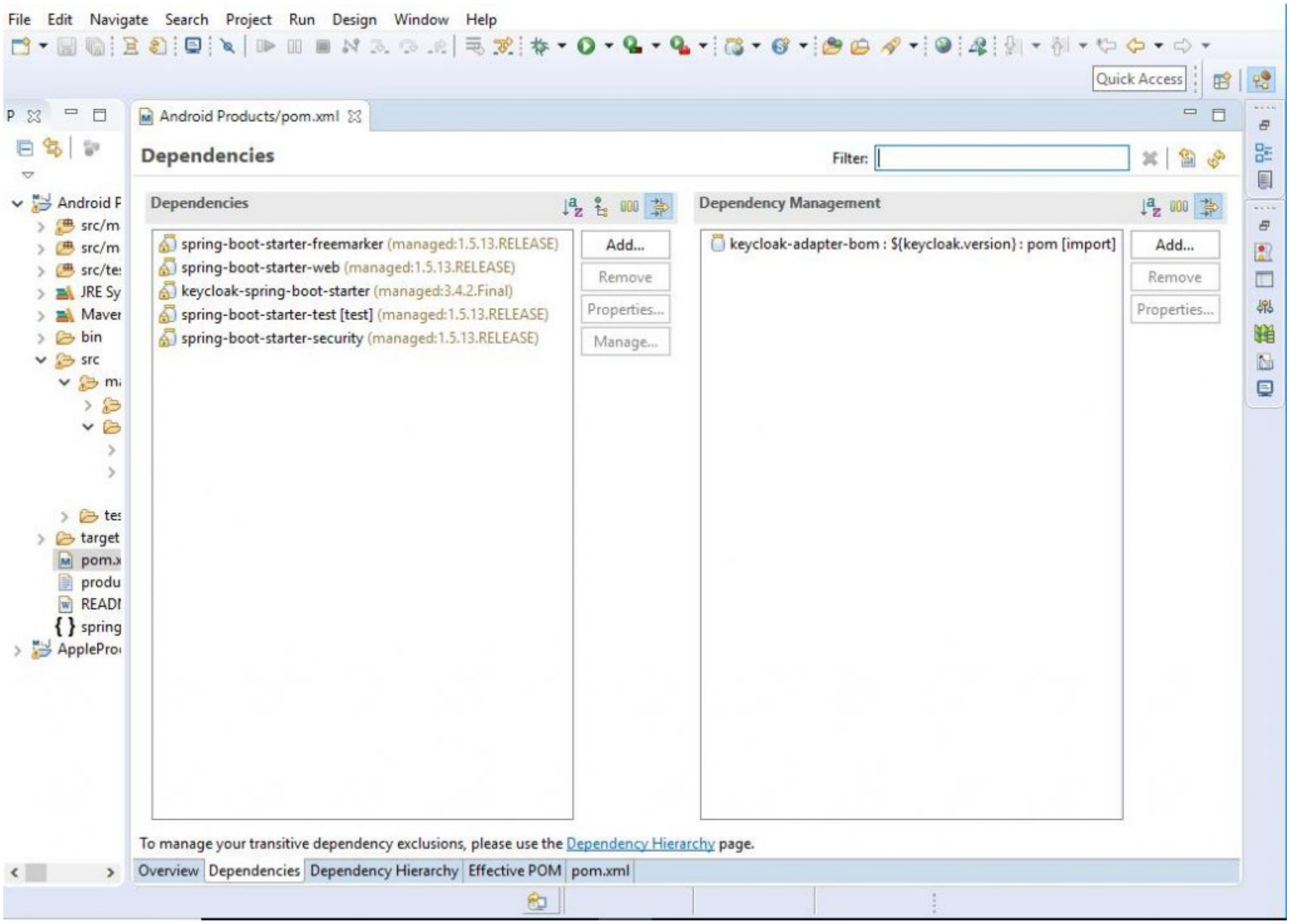
## Integrating multiple applications with keycloak

1. Add docker-compose.yml file and save it in a folder.



```yaml
version: '2'
services:
  postgres:
    image: postgres
    restart: always
    environment:
      POSTGRES_PASSWORD: password
      POSTGRES_USER: keycloak
      POSTGRES_DB: keycloak
    ports:
      - "5434:5432"
    networks:
      - docker
  keycloak:
    image: jboss/keycloak
    environment:
      KEYCLOAK_USER: admin
      KEYCLOAK_PASSWORD: admin
      DB_VENDOR: POSTGRES
      DB_ADDR: postgres
      DB_PORT: 5432
      DB_DATABASE: keycloak
      DB_USER: keycloak
      DB_PASSWORD: password
    ports:
      - "8080:8080"
    networks:
      - docker

networks:
  docker:
    driver: bridge

```

Add keycloak and spring-boot dependencies.



## Dependencies

Dependencies

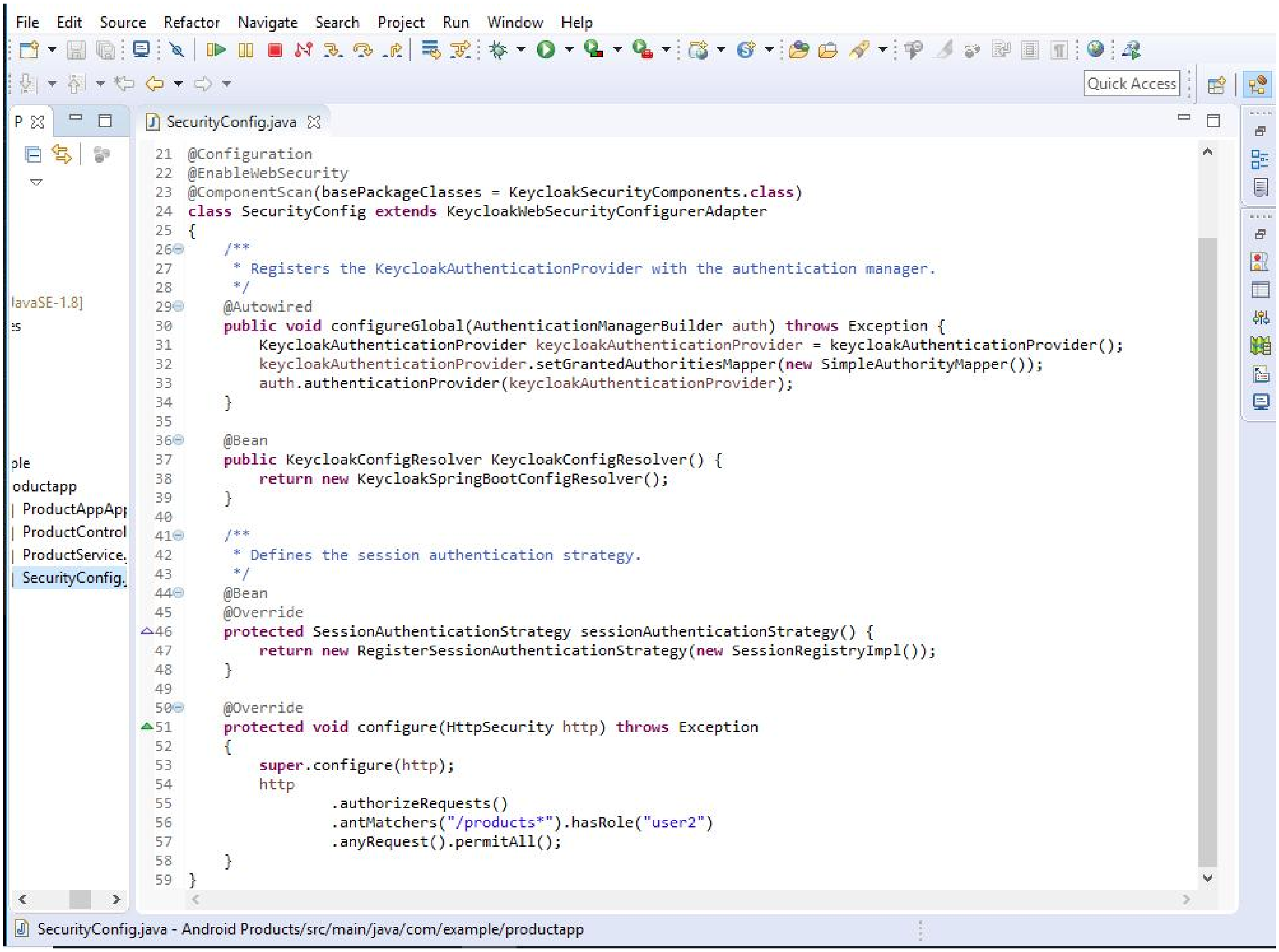
- spring-boot-starter-freemarker (managed:1.5.13.RELEASE)
- spring-boot-starter-web (managed:1.5.13.RELEASE)
- keycloak-spring-boot-starter (managed:3.4.2.Final)
- spring-boot-starter-test [test] (managed:1.5.13.RELEASE)
- spring-boot-starter-security (managed:1.5.13.RELEASE)

Dependency Management

- keycloak-adapter-bom : ${keycloak.version} : pom [import]

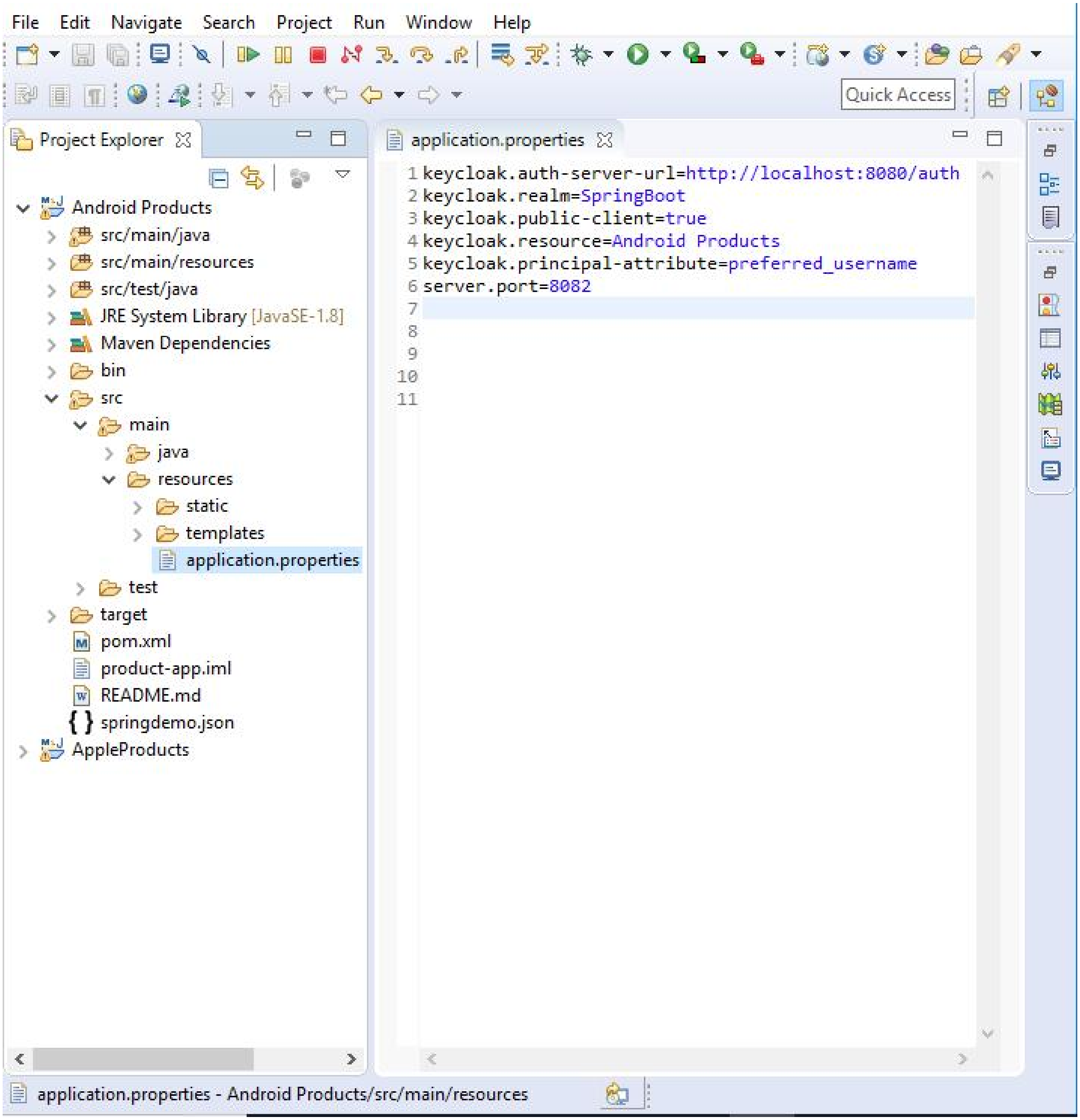
To manage your transitive dependency exclusions, please use the Dependency Hierarchy page.

CreateSecurityConfig.java file to your project.


File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
SecurityConfig.java
JavaSE-1.8]
ple
oductapp
ProductAppApp
ProductControl
ProductService.
SecurityConfig.
SecurityConfig.java - Android Products/src/main/java/com/example/productapp

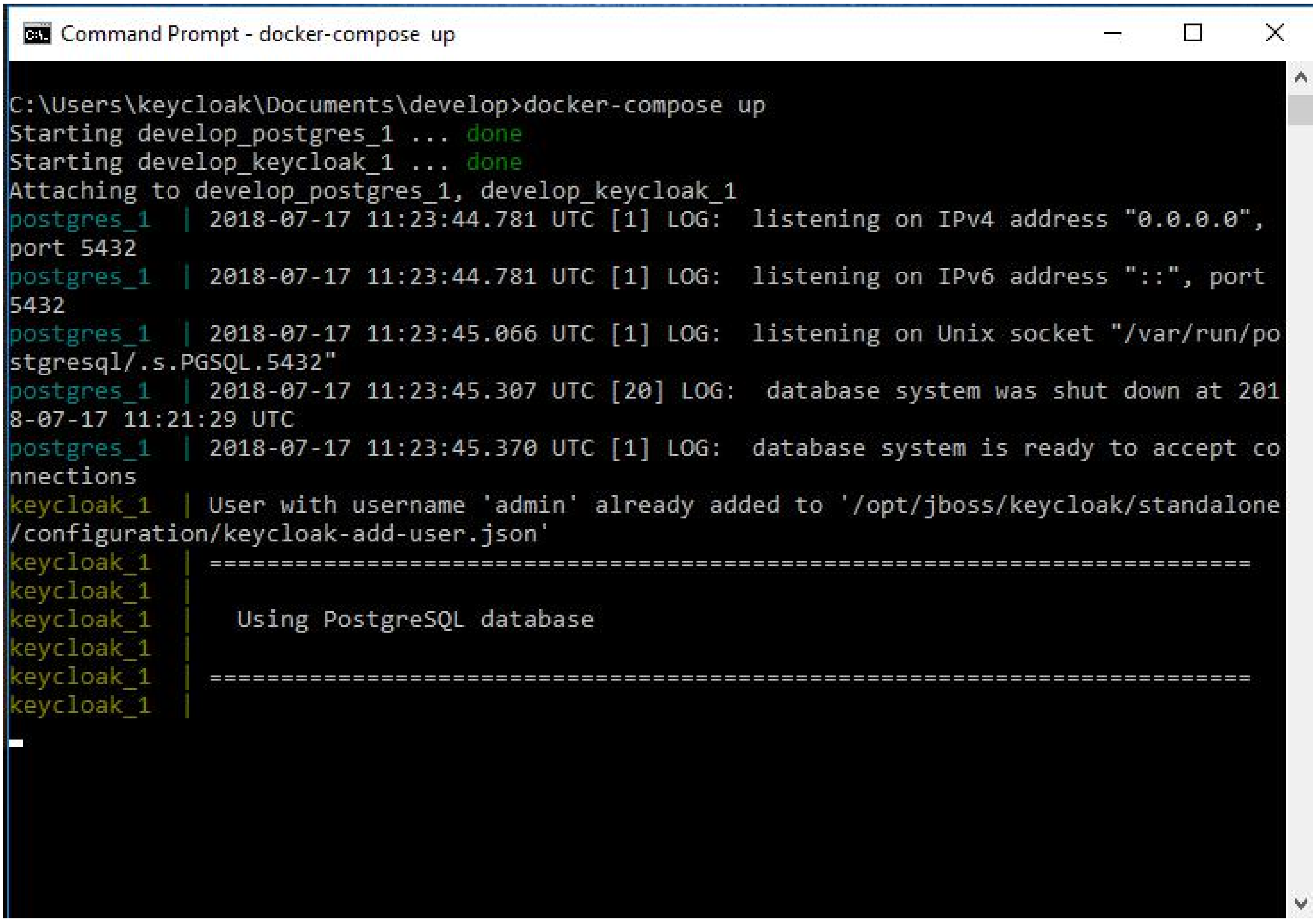
```
@Configuration
@EnableWebSecurity
@ComponentScan(basePackageClasses = KeycloakSecurityComponents.class)
class SecurityConfig extends KeycloakWebSecurityConfigurerAdapter
{
    /**
     * Registers the KeycloakAuthenticationProvider with the authentication manager.
     */
    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        KeycloakAuthenticationProvider keycloakAuthenticationProvider = keycloakAuthenticationProvider();
        keycloakAuthenticationProvider.setGrantedAuthoritiesMapper(new SimpleAuthorityMapper());
        auth.authenticationProvider(keycloakAuthenticationProvider);
    }

    @Bean
    public KeycloakConfigResolver KeycloakConfigResolver() {
        return new KeycloakSpringBootConfigResolver();
    }

    /**
     * Defines the session authentication strategy.
     */
    @Bean
    @Override
    protected SessionAuthenticationStrategy sessionAuthenticationStrategy() {
        return new RegisterSessionAuthenticationStrategy(new SessionRegistryImpl());
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception
    {
        super.configure(http);
        http
                .authorizeRequests()
                .antMatchers("/products*").hasRole("user2")
                .anyRequest().permitAll();
    }
}
```

Create applications.properties file.



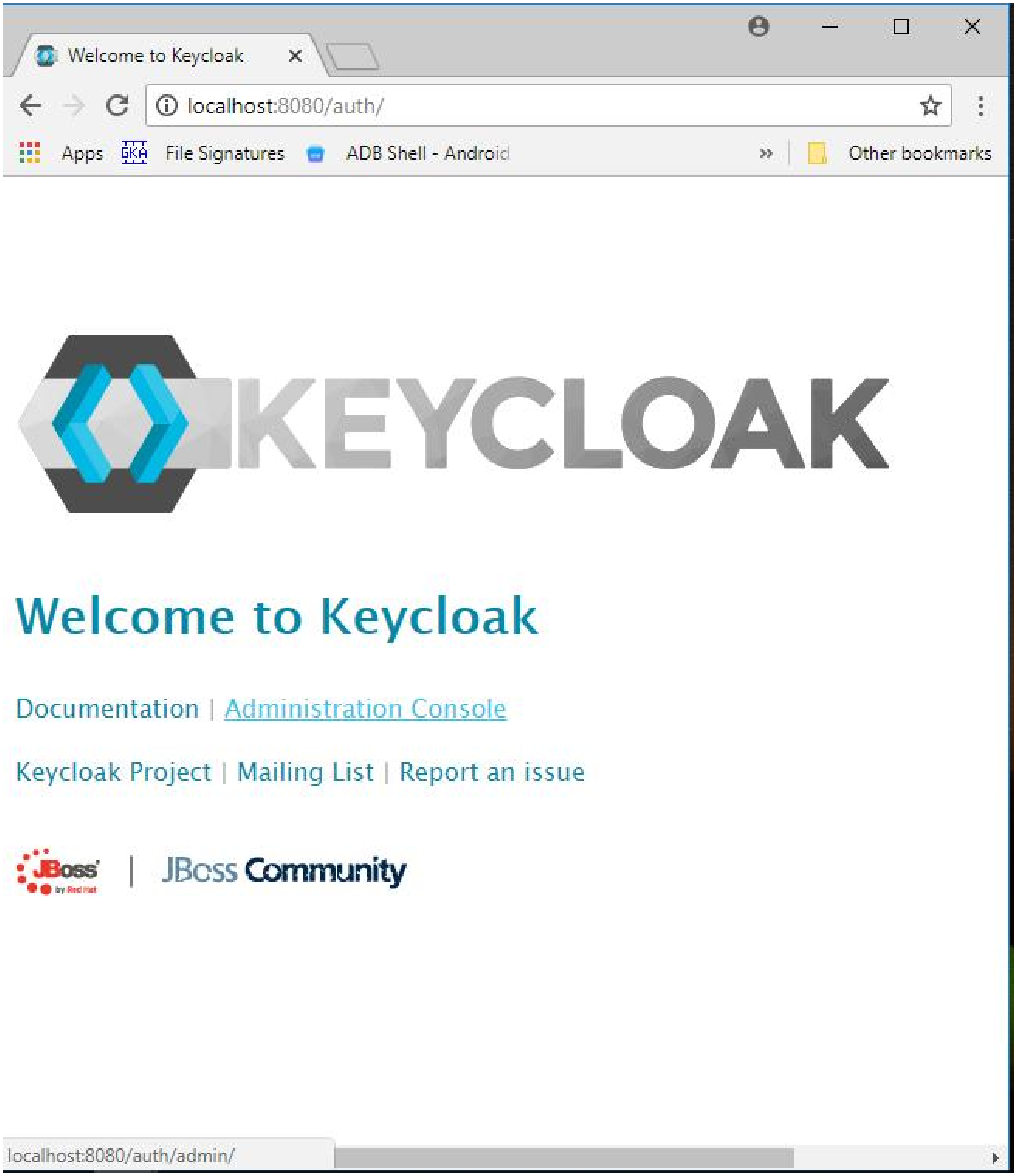
```
keycloak.auth-server-url=http://localhost:8080/auth
keycloak.realm=SpringBoot
keycloak.public-client=true
keycloak.resource=Android Products
keycloak.principal-attribute=preferred_username
server.port=8082
```

Open command prompt and run docker-compose up to run keycloak.

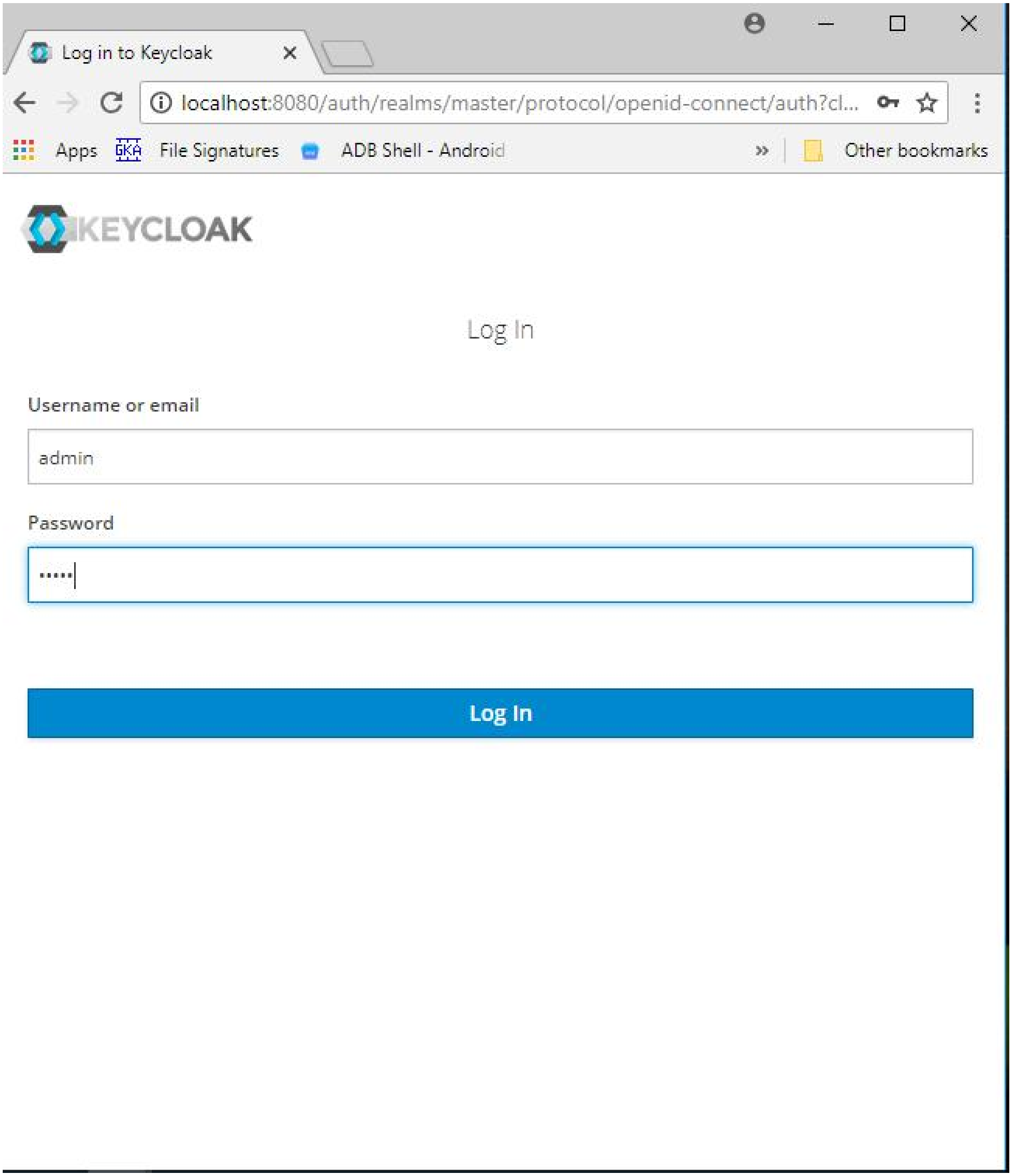


```
C:\Users\keycloak\Documents\develop>docker-compose up
Starting develop_postgres_1 ... done
Starting develop_keycloak_1 ... done
Attaching to develop_postgres_1, develop_keycloak_1
postgres_1  | 2018-07-17 11:23:44.781 UTC [1] LOG:  listening on IPv4 address "0.0.0.0",
port 5432
postgres_1  | 2018-07-17 11:23:44.781 UTC [1] LOG:  listening on IPv6 address "::", port
5432
postgres_1  | 2018-07-17 11:23:45.066 UTC [1] LOG:  listening on Unix socket "/var/run/po
stgresql/.s.PGSQL.5432"
postgres_1  | 2018-07-17 11:23:45.307 UTC [20] LOG:  database system was shut down at 201
8-07-17 11:21:29 UTC
postgres_1  | 2018-07-17 11:23:45.370 UTC [1] LOG:  database system is ready to accept co
nnections
keycloak_1  | User with username 'admin' already added to '/opt/jboss/keycloak/standalone
/configuration/keycloak-add-user.json'
keycloak_1  | =========================================================================
keycloak_1  |
keycloak_1  |   Using PostgreSQL database
keycloak_1  |
keycloak_1  | =========================================================================
keycloak_1  |
```

Open browser and enter the keycloak url. Click on Administration Console.


Welcome to Keycloak
localhost:8080/auth/
Apps
File Signatures
ADB Shell - Android
Other bookmarks
KEYCLOAK
Welcome to Keycloak
Documentation | Administration Console
Keycloak Project | Mailing List | Report an issue
JBoss by Red Hat | JBoss Community
localhost:8080/auth/admin/

Default username and password is ‘admin’ , ‘admin’.



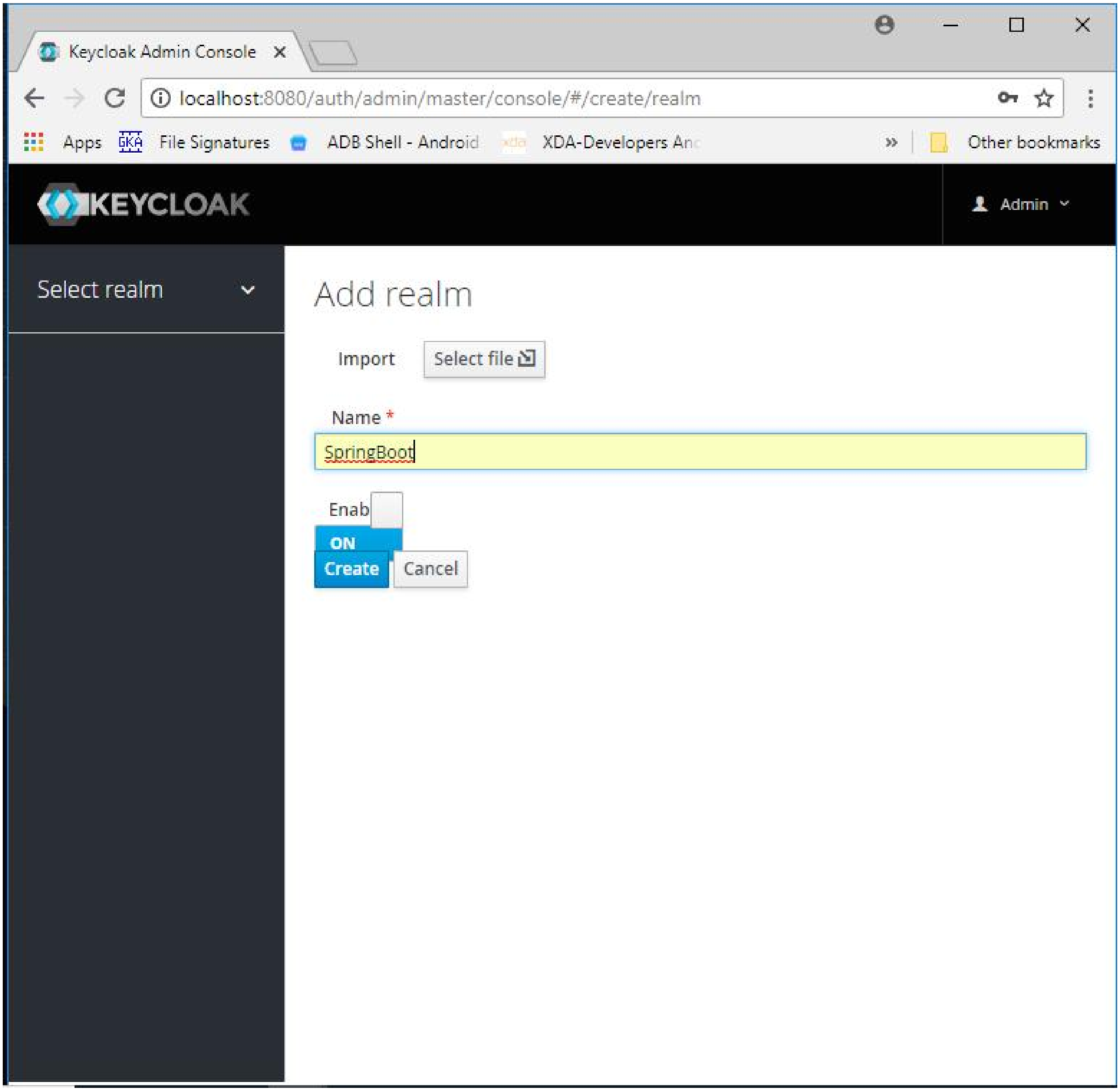
Username or email

admin

Password

•••••

Click on Add Realm. Give a name to the realm.

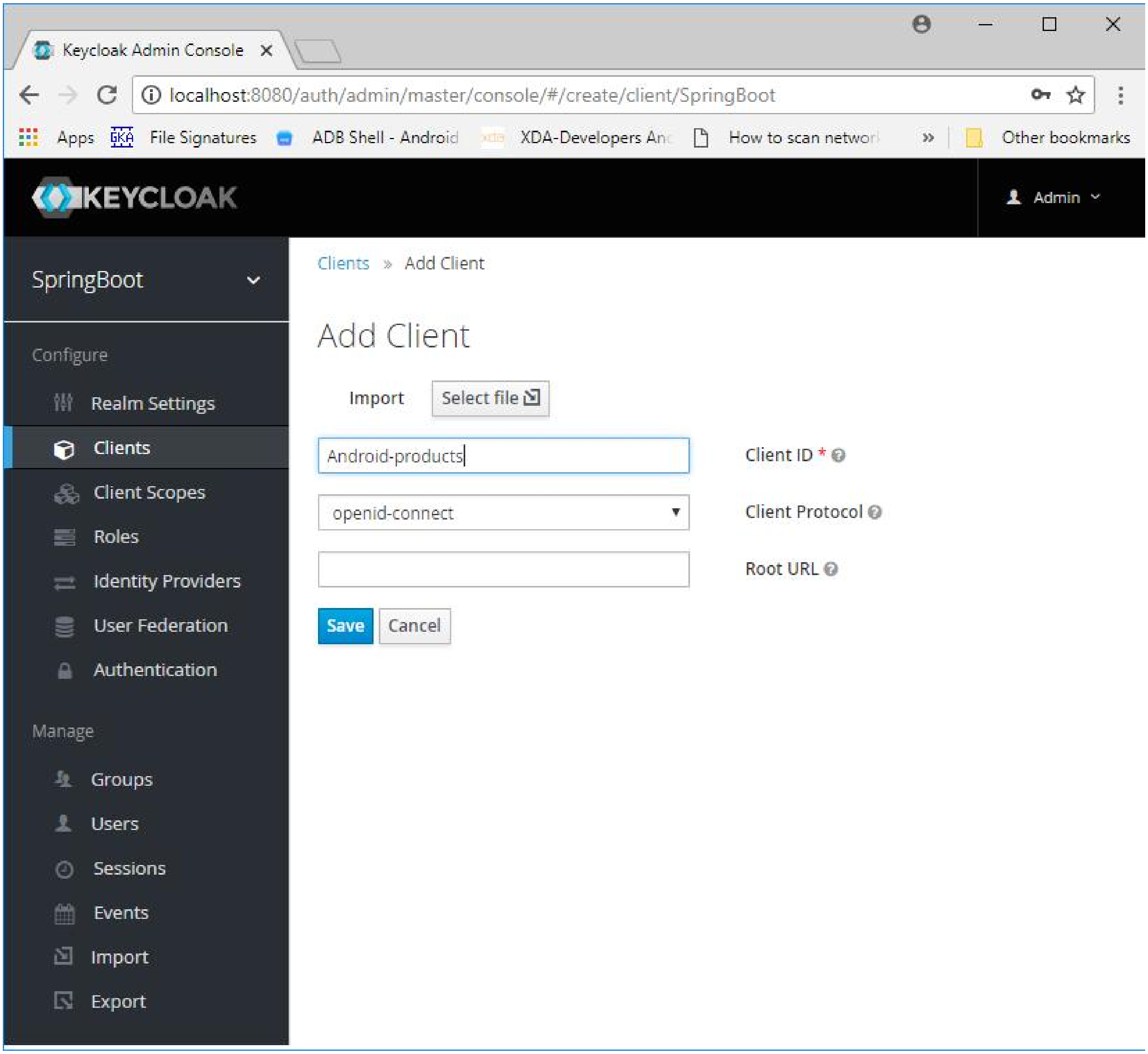


Add realm

Name * SpringBoot

Enab ON

Add a client.



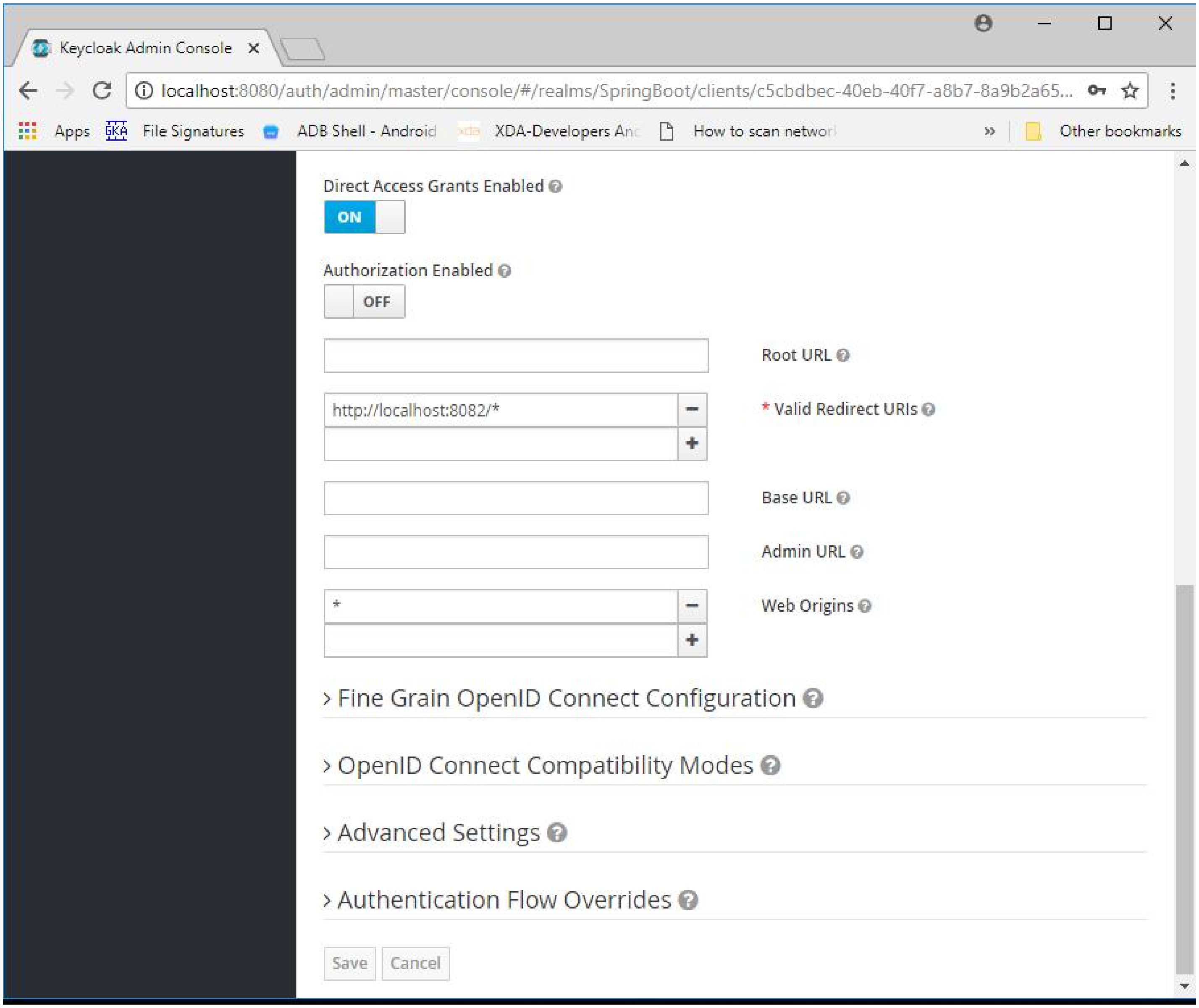
Add Client

Client ID * : Android-products

Client Protocol: openid-connect

Root URL:

Give the URL path of your application in valid redirect URL .



Keycloak Admin Console
localhost:8080/auth/admin/master/console/#/realms/SpringBoot/clients/c5cbdbec-40eb-40f7-a8b7-8a9b2a65...
Apps
File Signatures
ADB Shell - Android
XDA-Developers An
How to scan networ
Other bookmarks

Direct Access Grants Enabled: ON

Authorization Enabled: OFF

Root URL:

* Valid Redirect URIs: http://localhost:8082/*

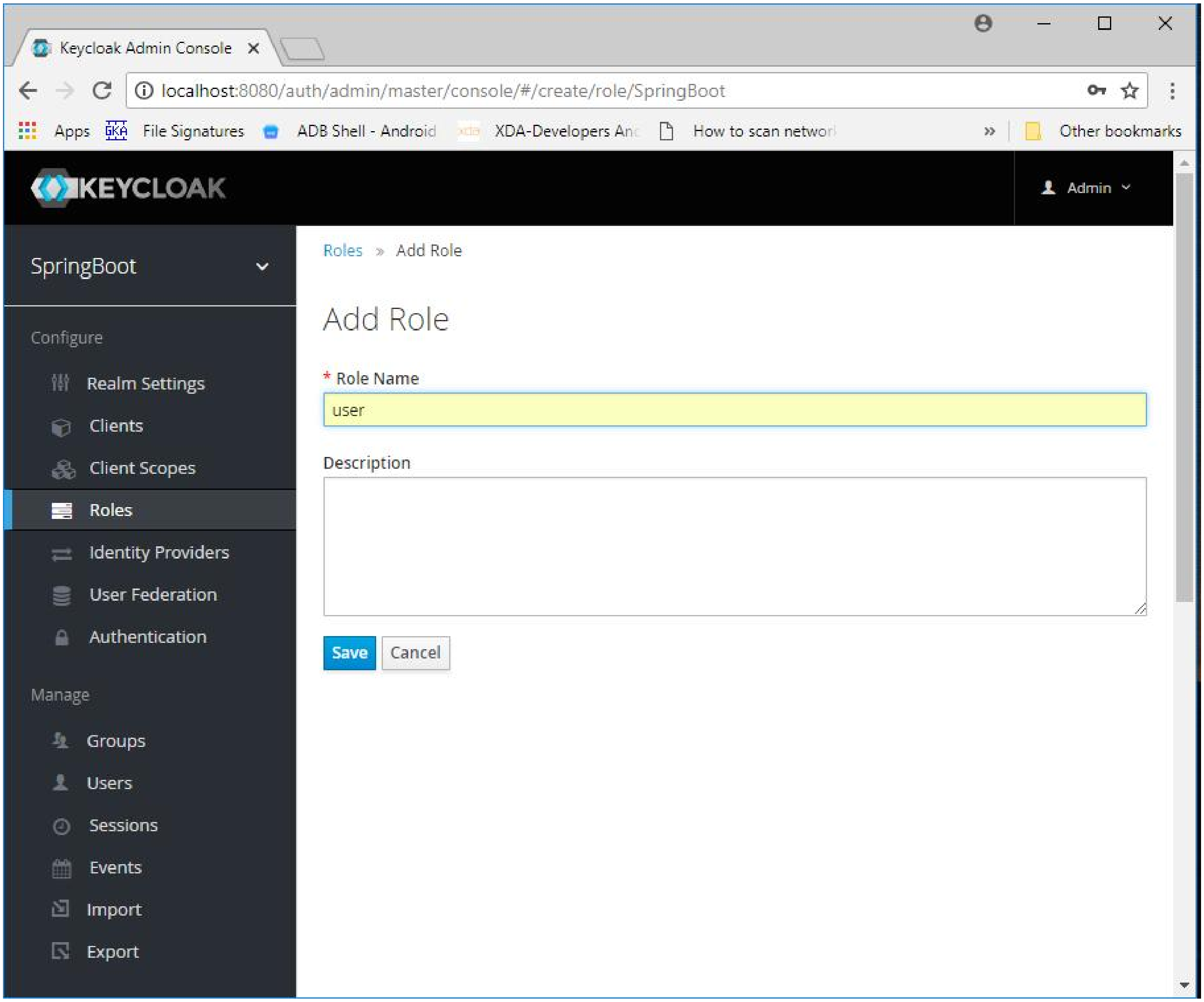
Base URL:

Admin URL:

Web Origins: *

› Fine Grain OpenID Connect Configuration
› OpenID Connect Compatibility Modes
› Advanced Settings
› Authentication Flow Overrides
Save
Cancel

Add role.

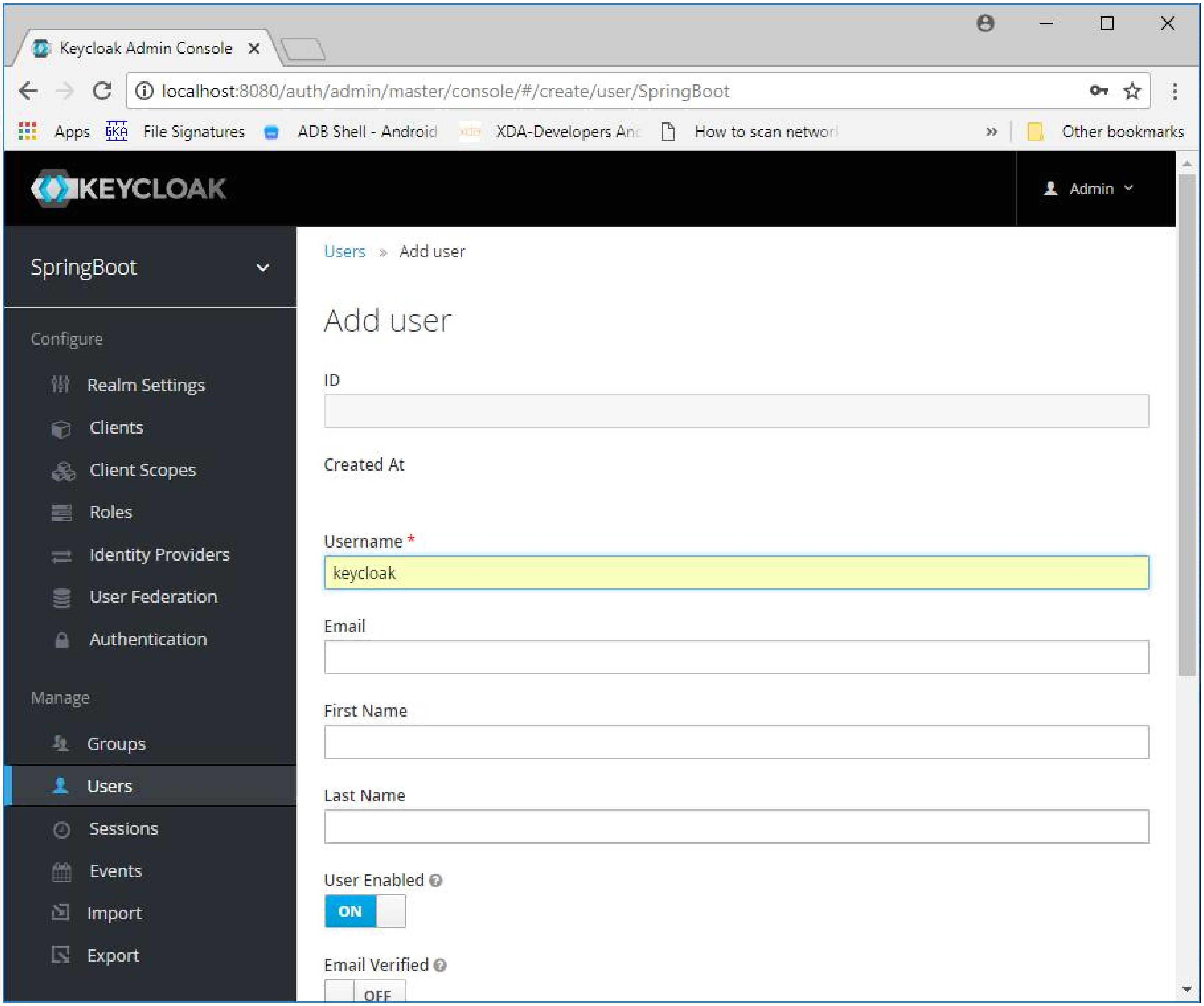


Add Role

* Role Name: user

Description:

Add user.



Add user

ID

Created At

Username * keycloak

Email

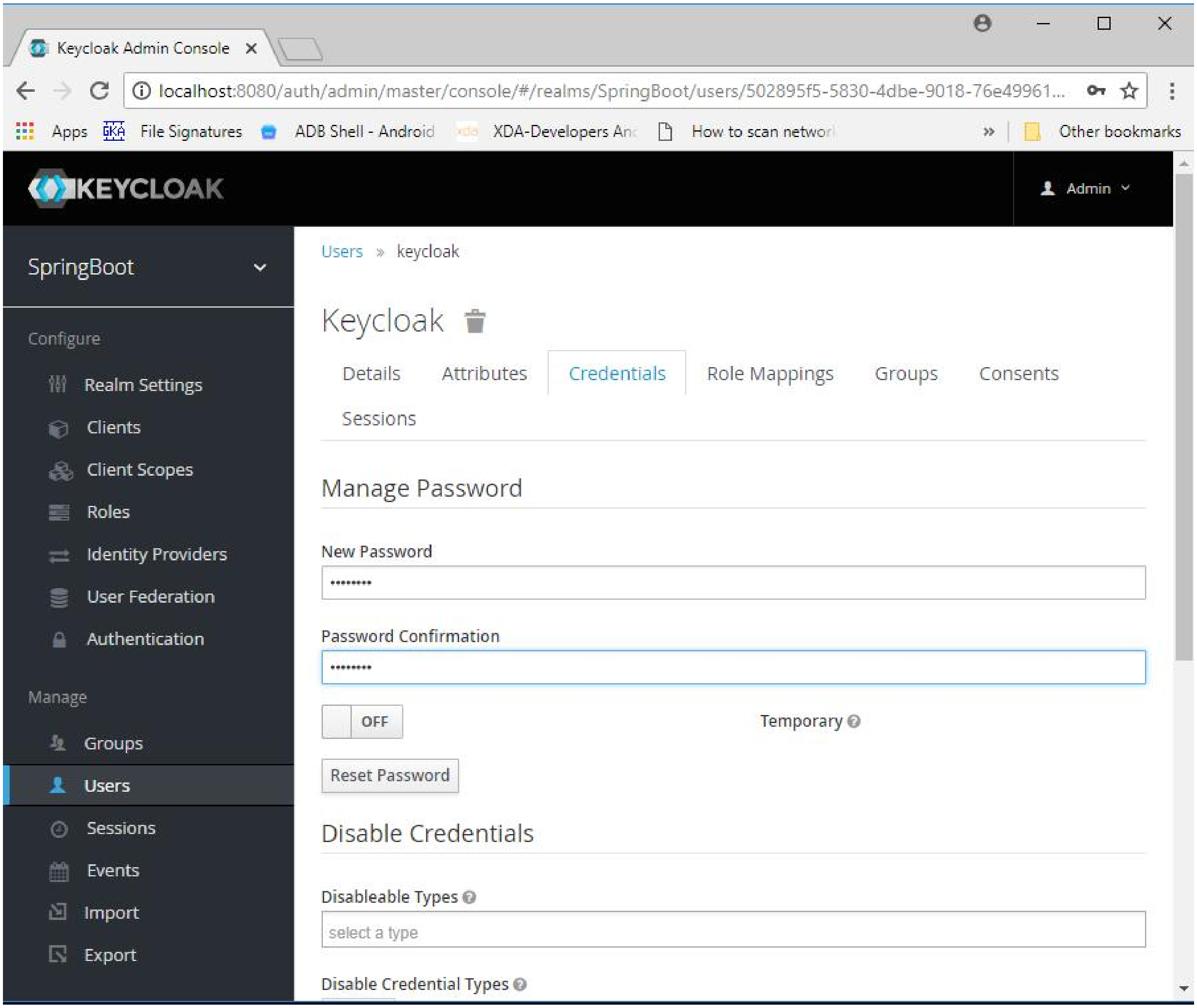
First Name

Last Name

User Enabled: ON

Email Verified: OFF

Go to credentials tab and reset password by giving new password.



Manage Password

New Password: ••••••••

Password Confirmation: ••••••••

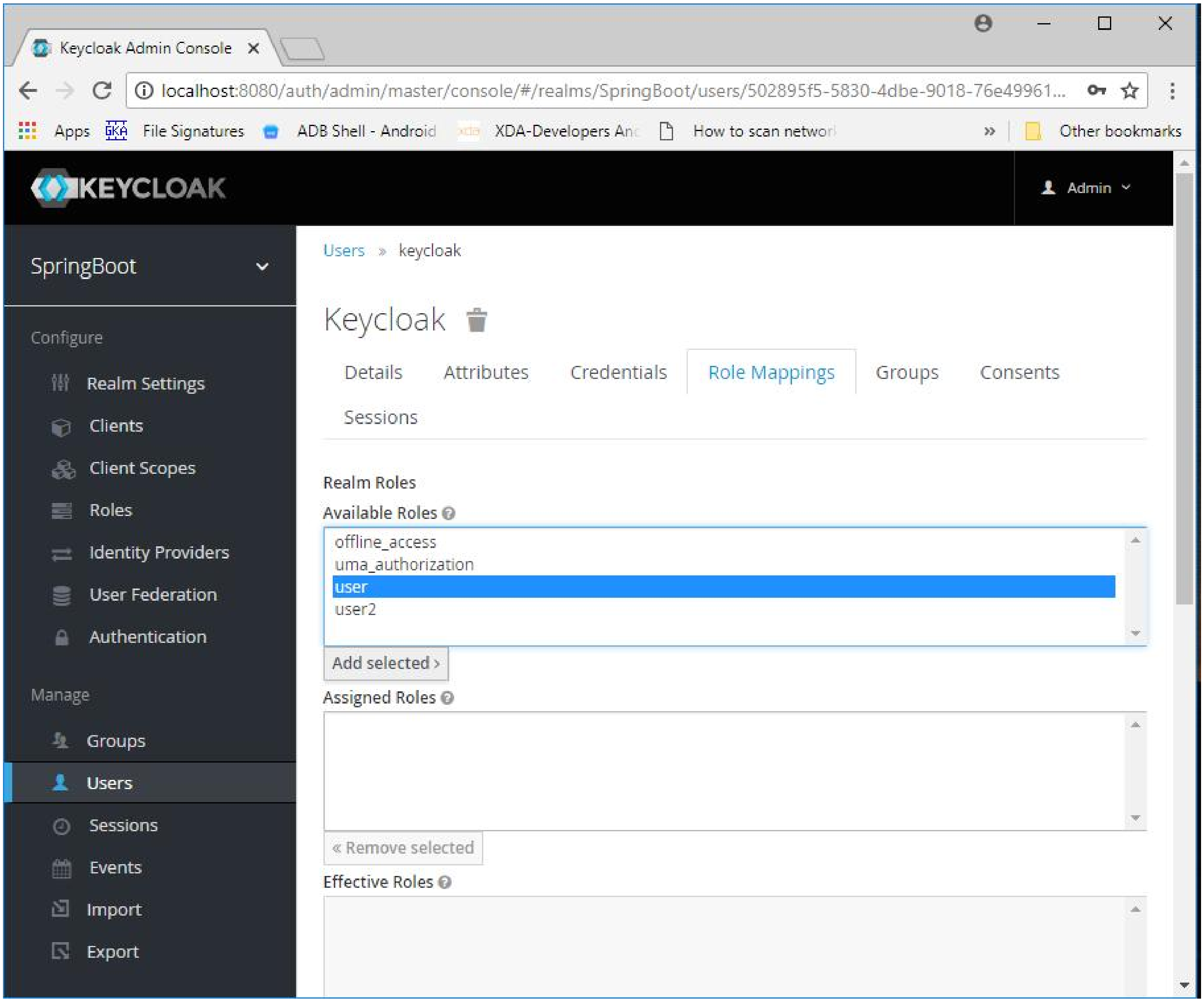
Temporary: OFF

Disable Credentials

Disableable Types: select a type

Disable Credential Types

Go to role mappings tab and add a role to the user.



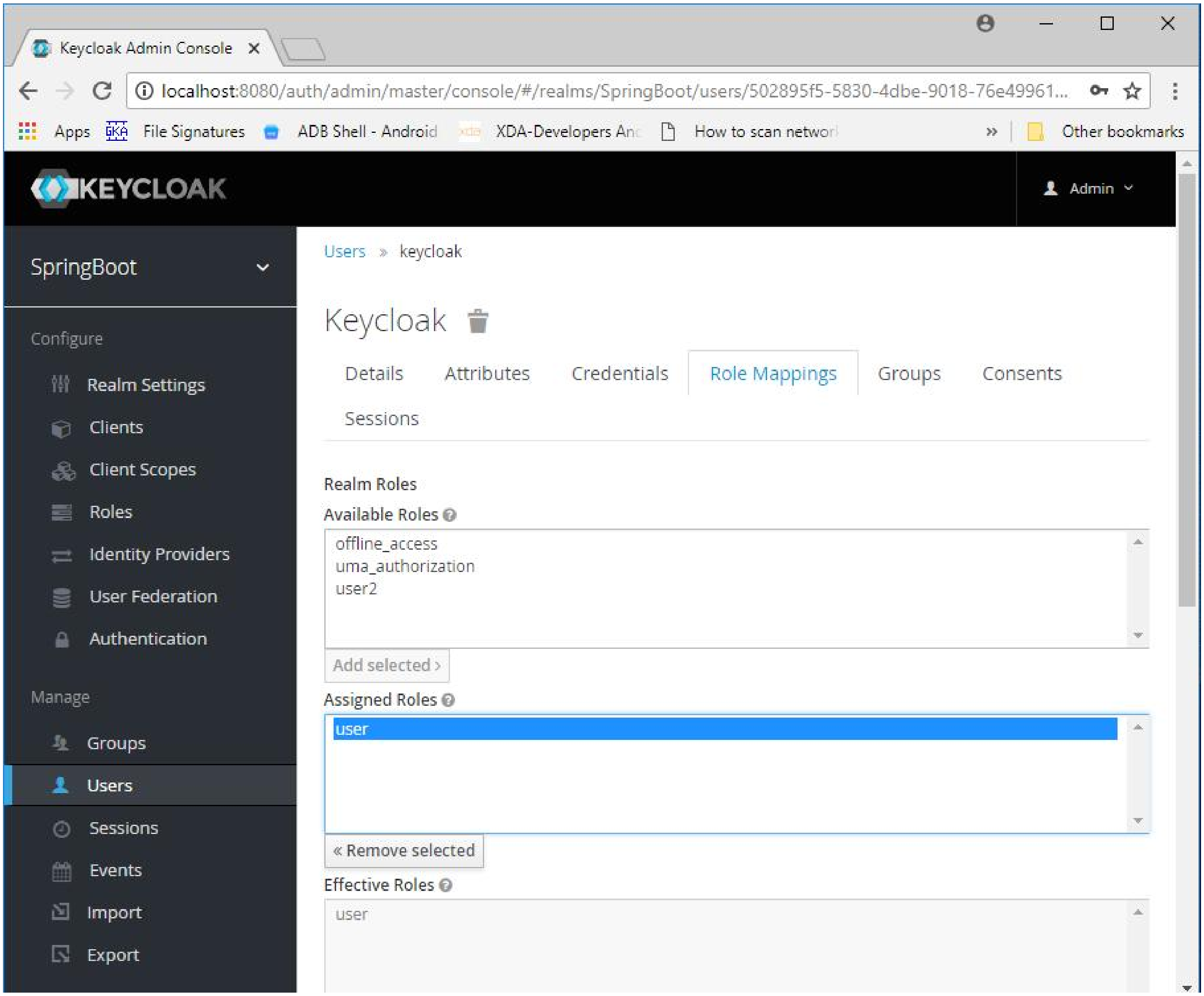
Realm Roles

Available Roles

offline_access
uma_authorization
user
user2

Assigned Roles

Effective Roles



localhost:8080/auth/admin/master/console/#/realms/SpringBoot/users/502895f5-5830-4dbe-9018-76e49961...

Realm Roles

Available Roles

offline_access
uma_authorization
user2

Assigned Roles

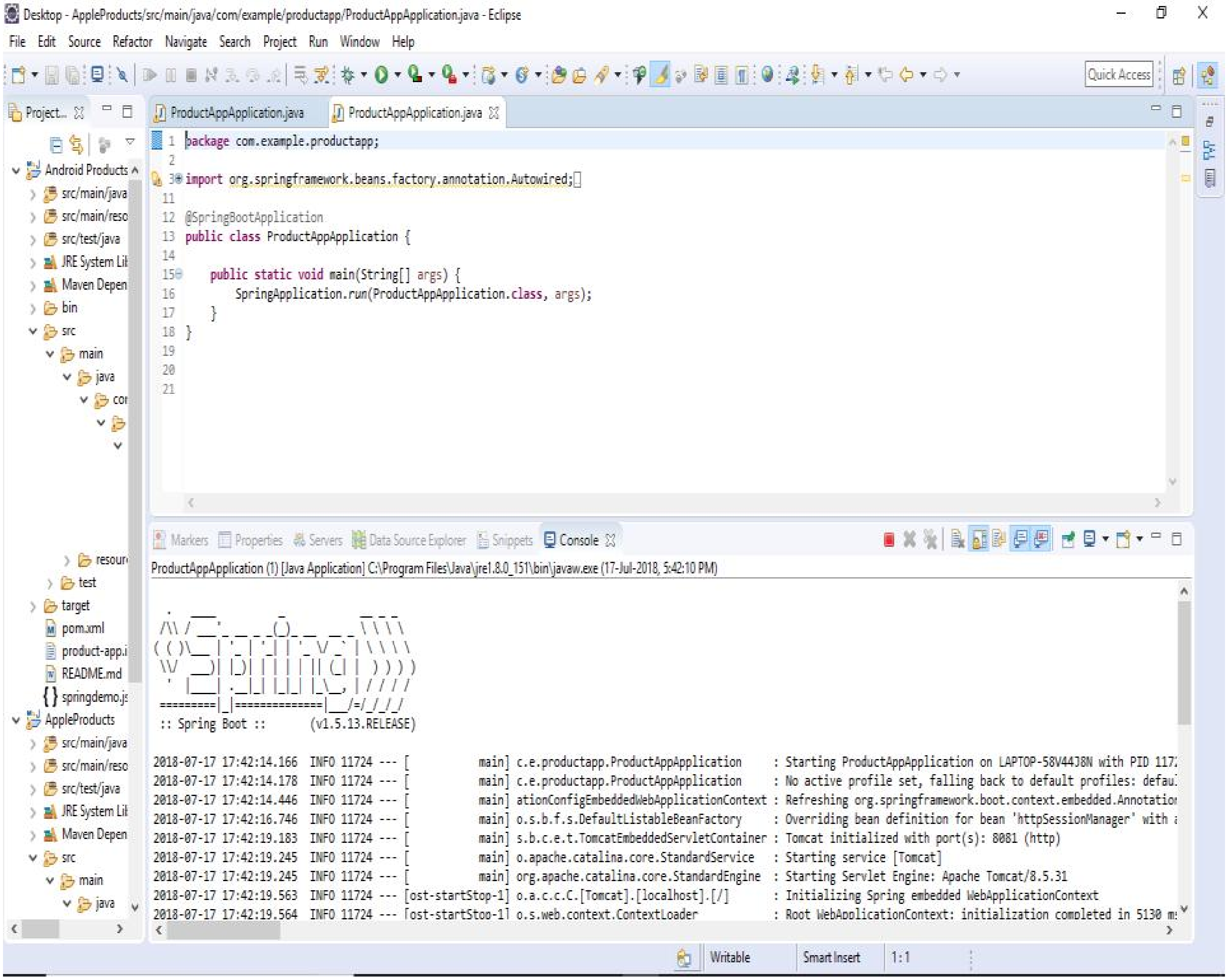
user

Effective Roles

user

Similarly create another application and add all the necessary files. In keycloak we will use the same realm but we need to add new client, role and user.

Now run two applications

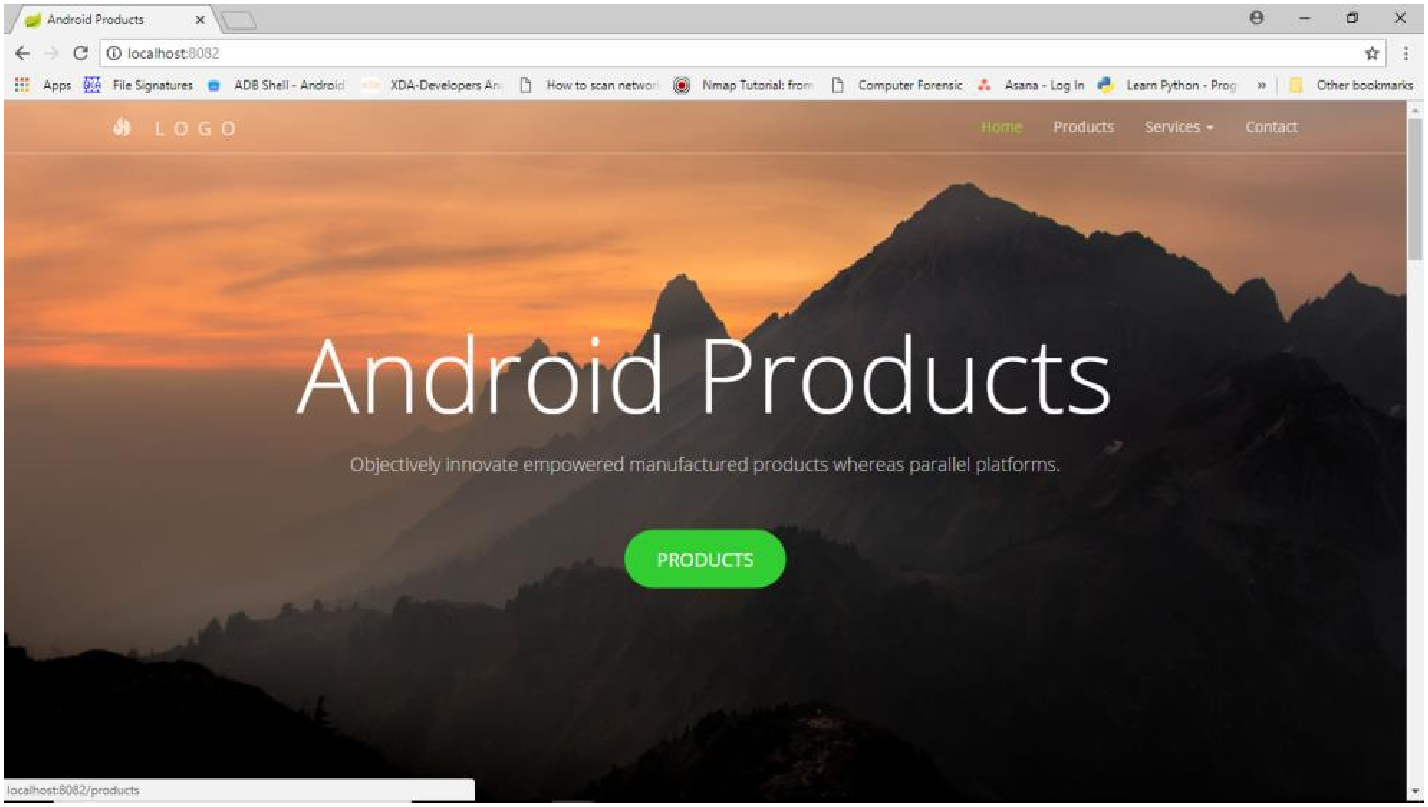


```java
package com.example.productapp;

import org.springframework.beans.factory.annotation.Autowired;

@SpringBootApplication
public class ProductAppApplication {

    public static void main(String[] args) {
        SpringApplication.run(ProductAppApplication.class, args);
    }
}
```

ProductAppApplication (1) [Java Application] C:\Program Files\Java\jre1.8.0_151\bin\javaw.exe (17-Jul-2018, 5:42:10 PM)

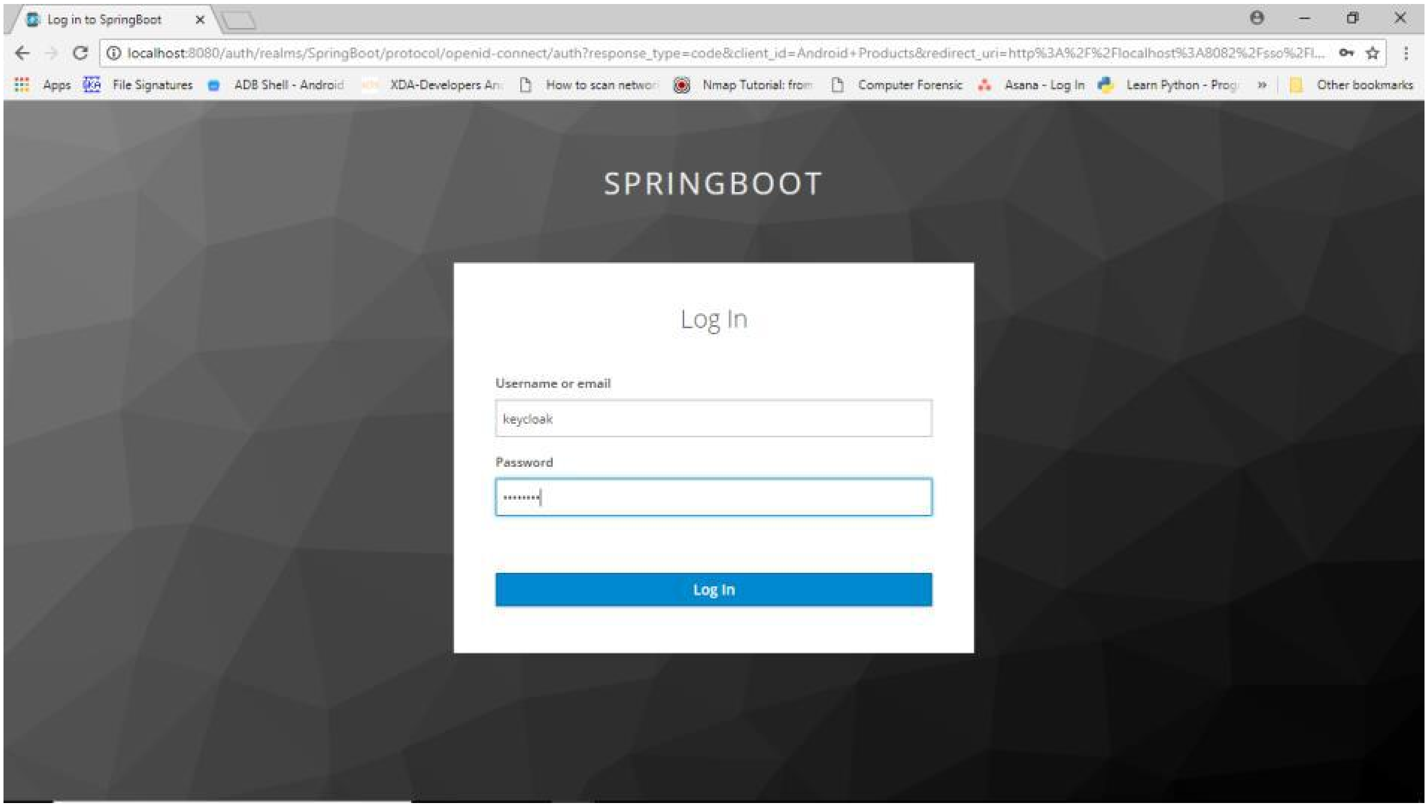
```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::       (v1.5.13.RELEASE)

2018-07-17 17:42:14.166  INFO 11724 --- [           main] c.e.productapp.ProductAppApplication     : Starting ProductAppApplication on LAPTOP-58V44J8N with PID 117
2018-07-17 17:42:14.178  INFO 11724 --- [           main] c.e.productapp.ProductAppApplication     : No active profile set, falling back to default profiles: defau
2018-07-17 17:42:14.446  INFO 11724 --- [           main] ationConfigEmbeddedWebApplicationContext : Refreshing org.springframework.boot.context.embedded.Annotatio
2018-07-17 17:42:16.746  INFO 11724 --- [           main] o.s.b.f.s.DefaultListableBeanFactory     : Overriding bean definition for bean 'httpSessionManager' with a
2018-07-17 17:42:19.183  INFO 11724 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat initialized with port(s): 8081 (http)
2018-07-17 17:42:19.245  INFO 11724 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2018-07-17 17:42:19.245  INFO 11724 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet Engine: Apache Tomcat/8.5.31
2018-07-17 17:42:19.563  INFO 11724 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2018-07-17 17:42:19.564  INFO 11724 --- [ost-startStop-1] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 5130 m
```

Open browser enter the url for first application.



Objectively innovate empowered manufactured products whereas parallel platforms.

When I click on products button, it will redirect to keycloak. Give username and password.

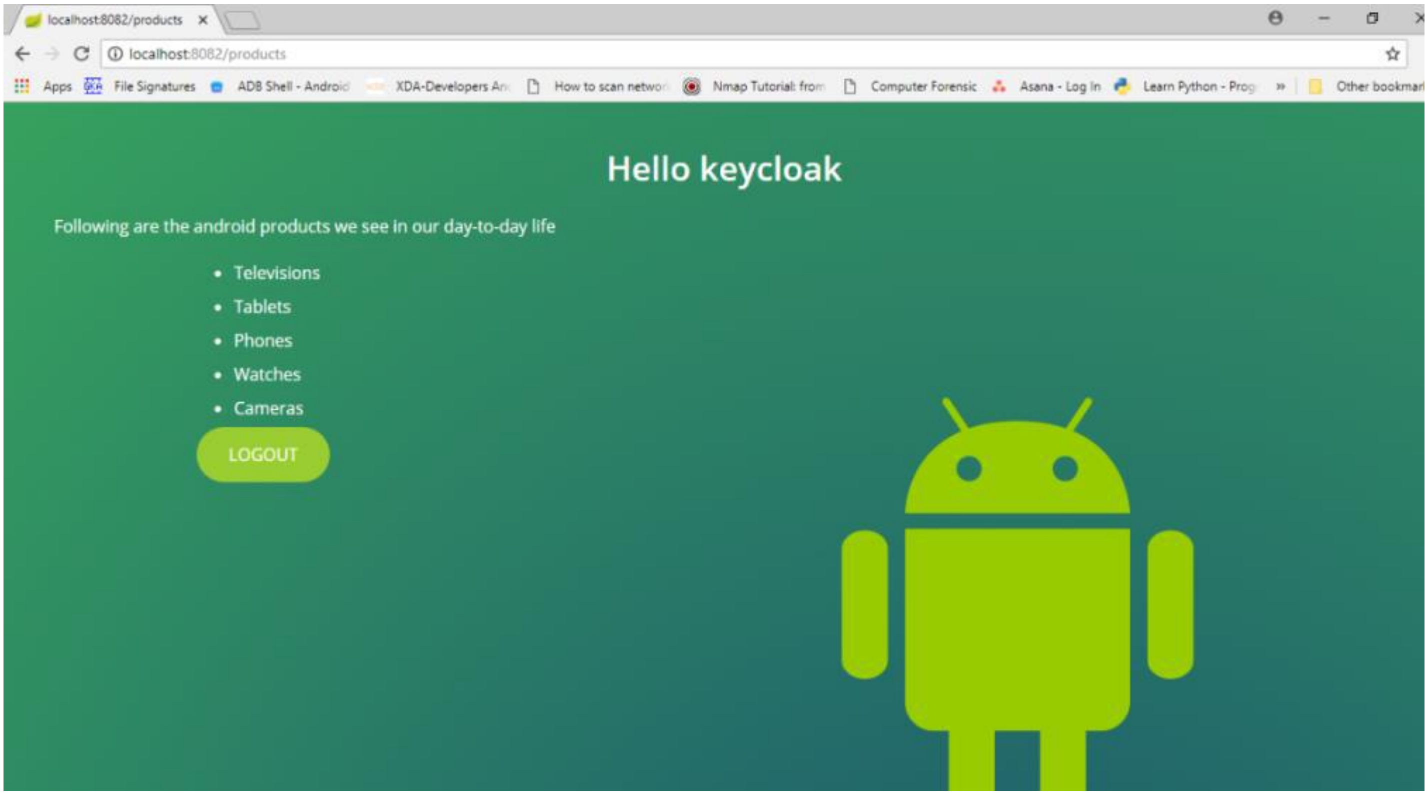


localhost:8080/auth/realms/SpringBoot/protocol/openid-connect/auth?response_type=code&client_id=Android+Products&redirect_uri=http%3A%2F%2Flocalhost%3A8082%2Fsso%2Fl...

Username or email: keycloak

Password: ••••••••

Now you will be redirected to the application.

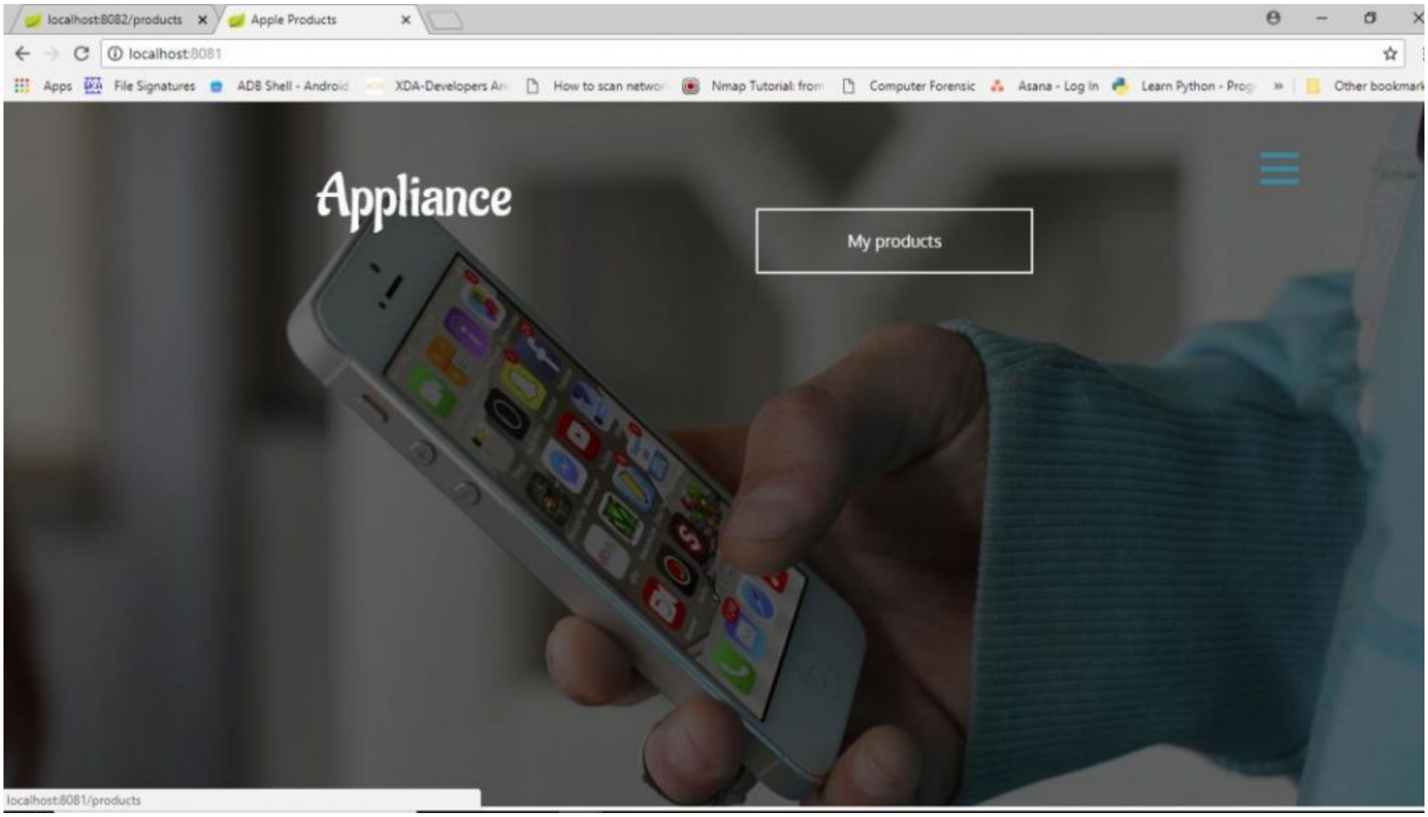


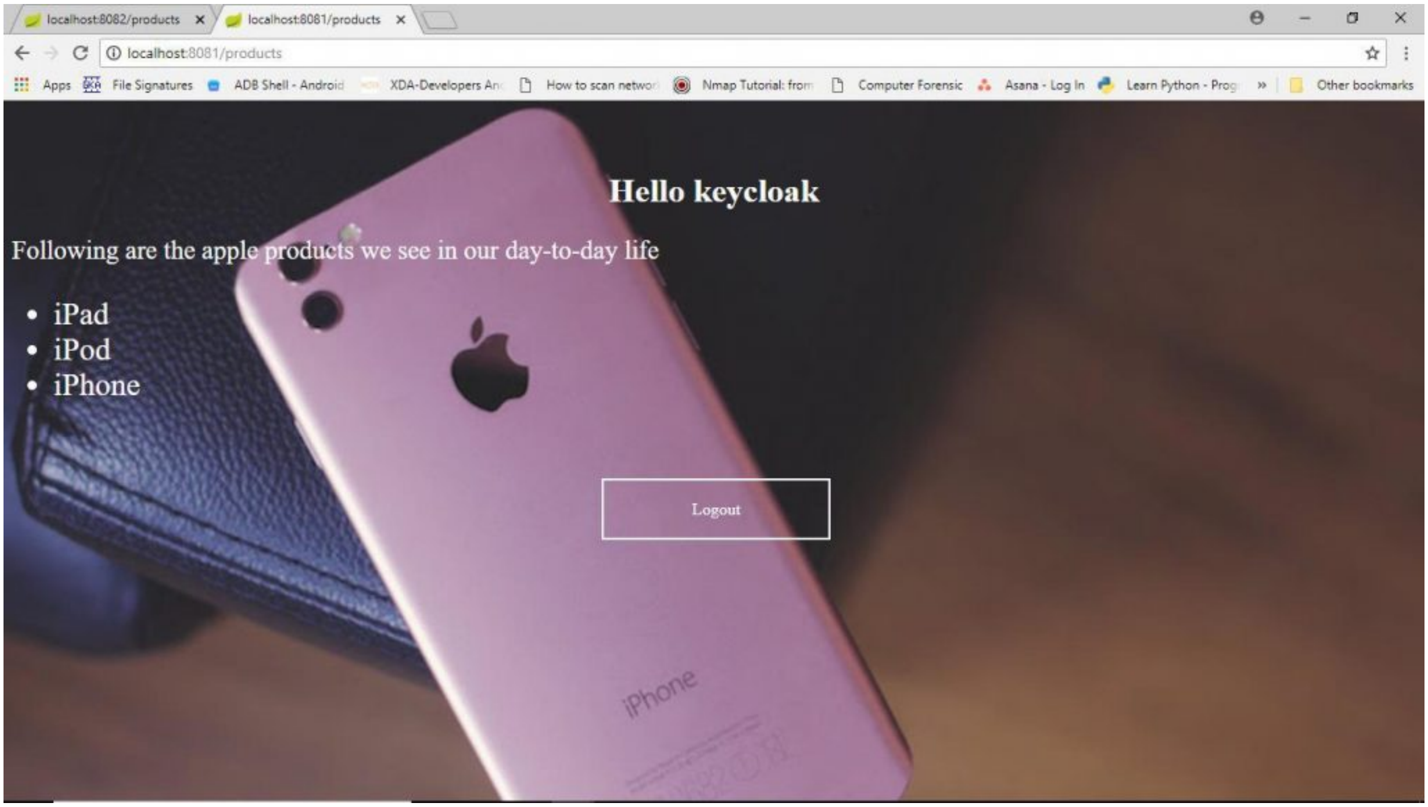
Hello keycloak

Following are the android products we see in our day-to-day life

- Televisions
- Tablets
- Phones
- Watches
- Cameras

Now open another tab and give the URL for another application. You will be redirected to the application without going to the keycloak login screen.



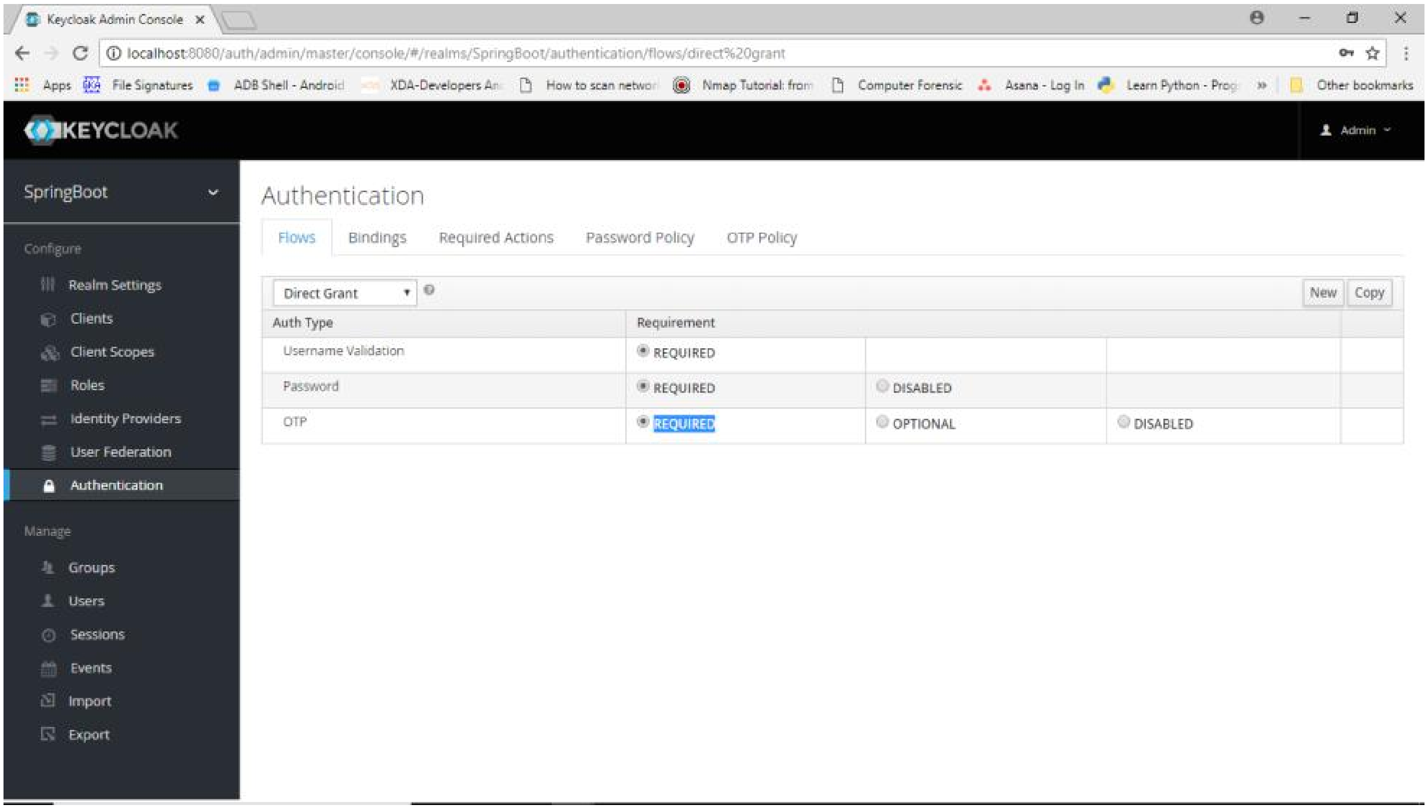


Hello keycloak

Following are the apple products we see in our day-to-day life

- iPad
- iPod
- iPhone

We will add an authentication method i.e. OTP. Go to authentication-flows-direct grant, at OTP mark it to required.

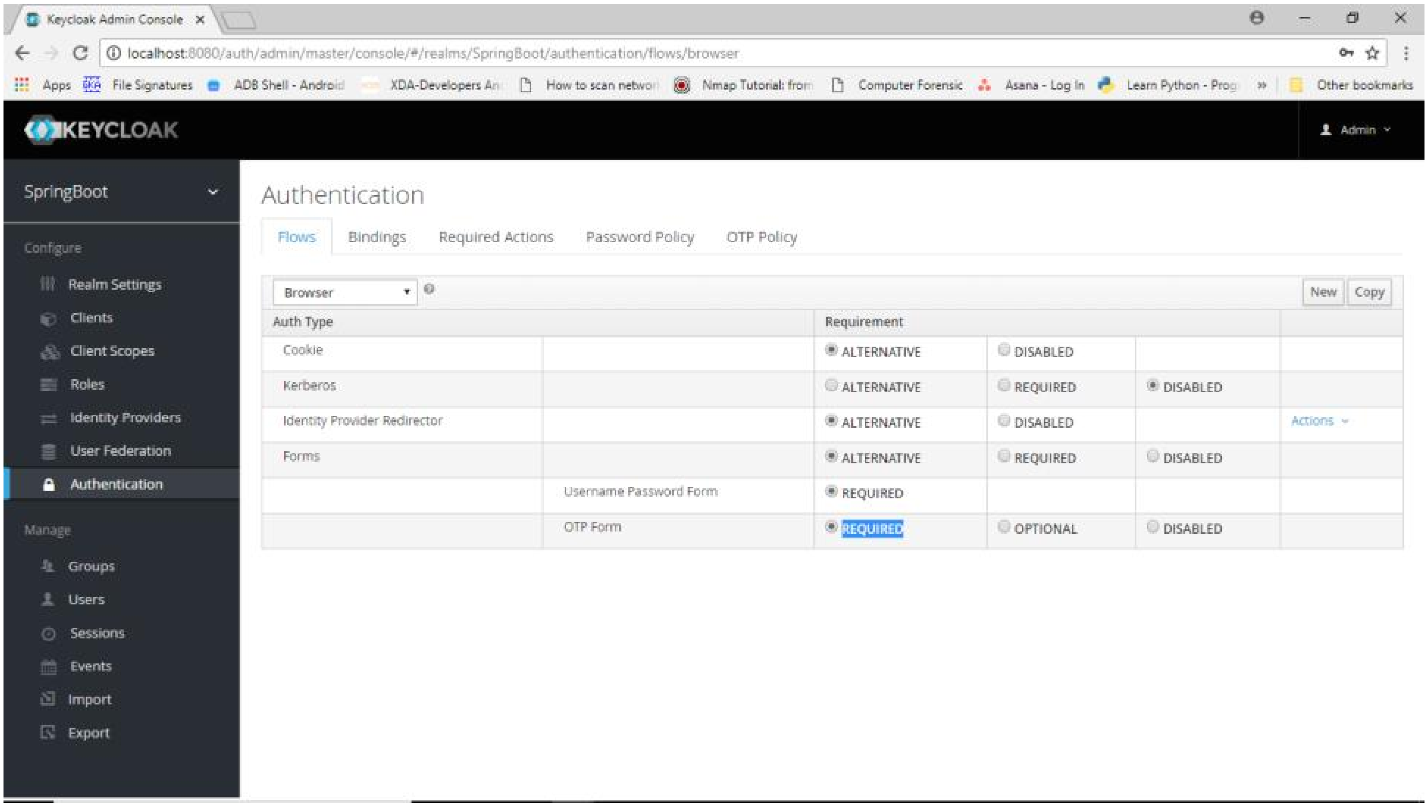


Authentication

Direct Grant

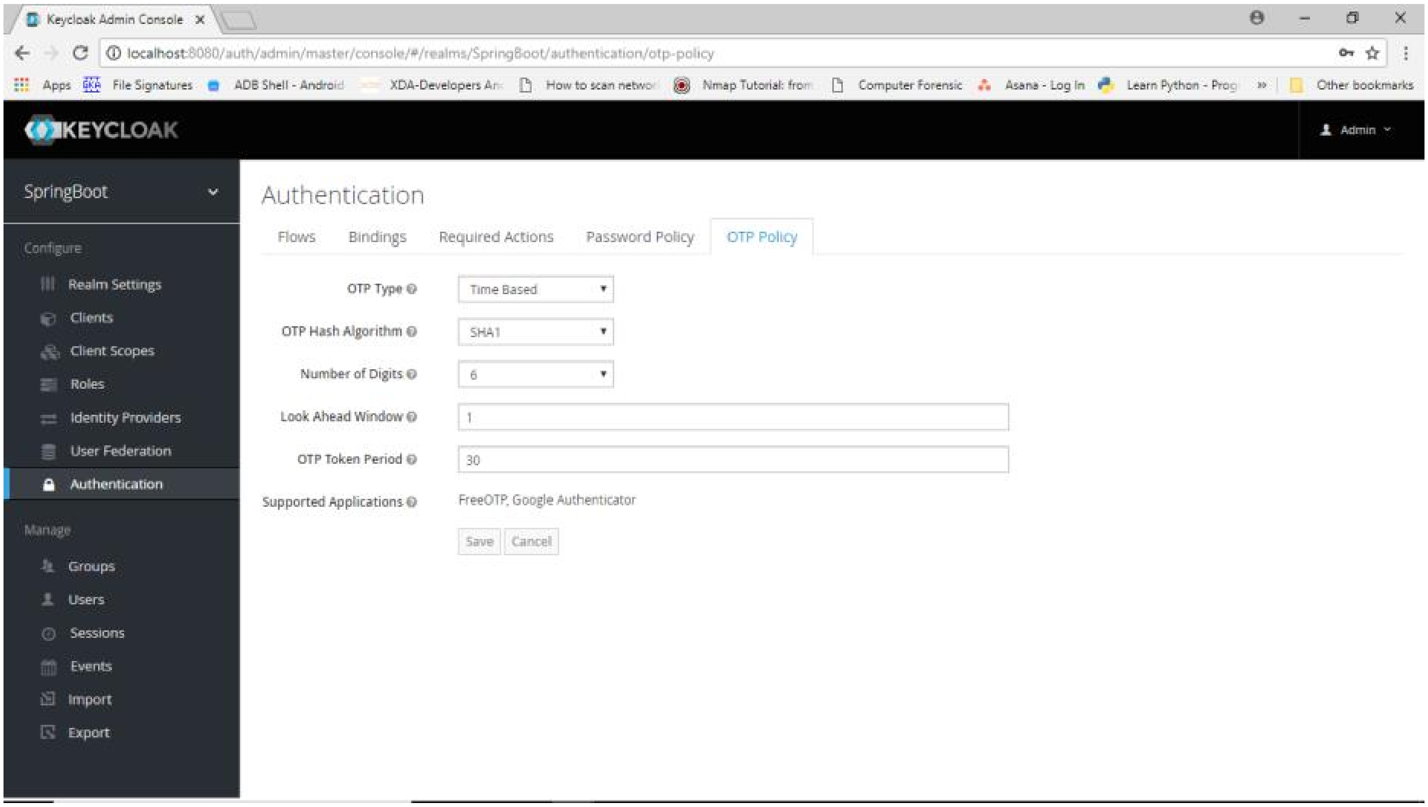
| Auth Type | Requirement | | |
|---|---|---|---|
| Username Validation | ● REQUIRED | | |
| Password | ● REQUIRED | ○ DISABLED | |
| OTP | ● REQUIRED | ○ OPTIONAL | ○ DISABLED |

In browser flow mark OTP to required.



| Auth Type | | Requirement | | | |
|---|---|---|---|---|---|
| Cookie | | ● ALTERNATIVE | ○ DISABLED | | |
| Kerberos | | ○ ALTERNATIVE | ○ REQUIRED | ● DISABLED | |
| Identity Provider Redirector | | ● ALTERNATIVE | ○ DISABLED | | Actions |
| Forms | | ● ALTERNATIVE | ○ REQUIRED | ○ DISABLED | |
| | Username Password Form | ● REQUIRED | | | |
| | OTP Form | ● REQUIRED | ○ OPTIONAL | ○ DISABLED | |

In OTP policy you can see the hash algorithms, time period etc.



OTP Type: Time Based

OTP Hash Algorithm: SHA1

Number of Digits: 6

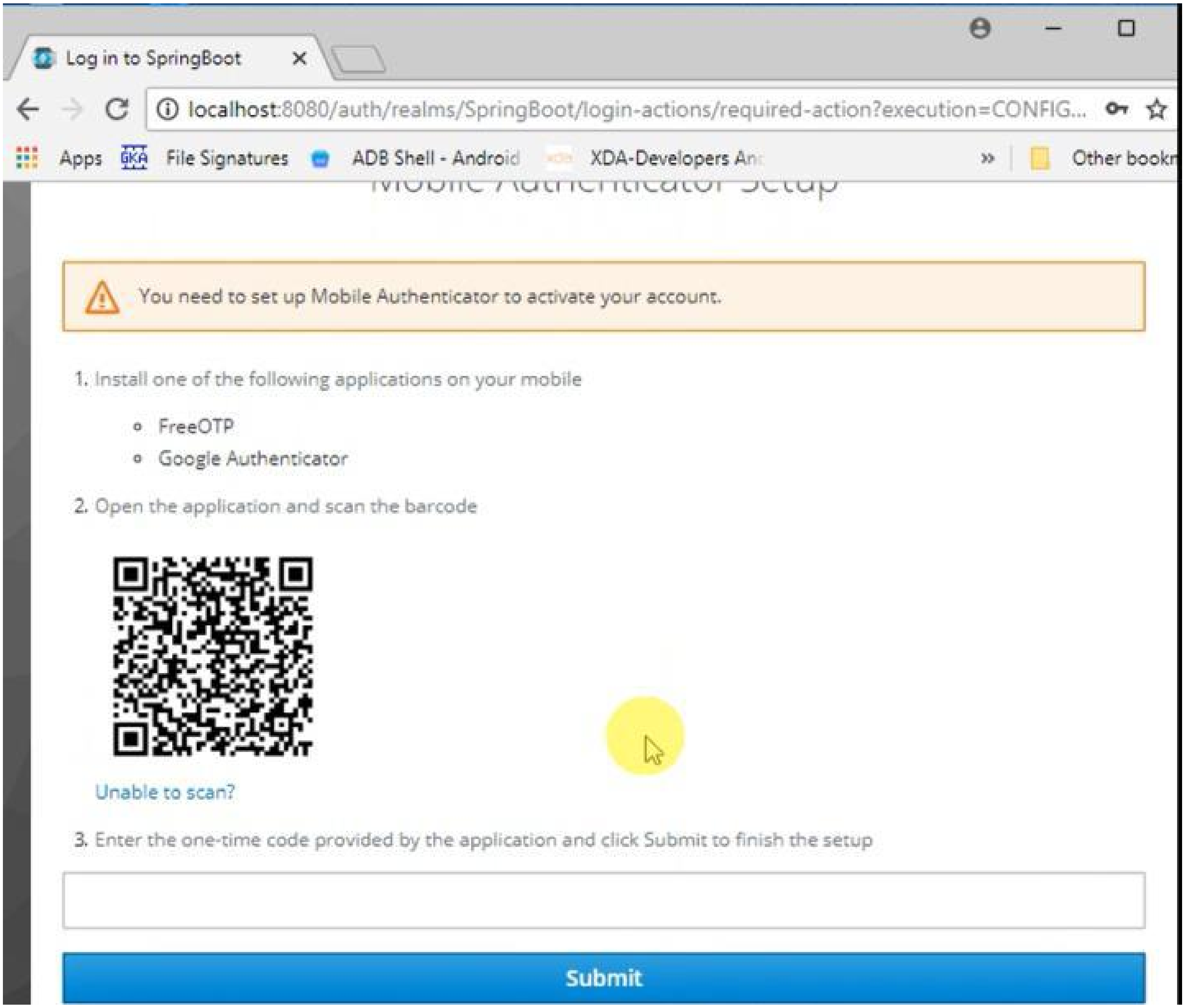
Look Ahead Window: 1

OTP Token Period: 30

Supported Applications: FreeOTP, Google Authenticator

You need to install FreeOTP application or Google Authenticator application on your mobile.

Now re-run keycloak, open browser and give a URL. After keycloak login page you will be redirected to this screen


Log in to SpringBoot
localhost:8080/auth/realms/SpringBoot/login-actions/required-action?execution=CONFIG...
Apps
File Signatures
ADB Shell - Android
XDA-Developers An
Other bookm
Mobile Authenticator Setup
Unable to scan?
Submit

You need to set up Mobile Authenticator to activate your account.

1. Install one of the following applications on your mobile
   - FreeOTP
   - Google Authenticator
2. Open the application and scan the barcode
3. Enter the one-time code provided by the application and click Submit to finish the setup

Now scan the barcode with the application that you have installed on your mobile and enter the OTP.

You will be redirected to your application.

IDENTITY ACCESS MANAGEMENT (HTTPS://WWW.COMAKEIT.COM/TAG/IDENTITY-ACCESS-MANAGEMENT/)

KEYCLOAK (HTTPS://WWW.COMAKEIT.COM/TAG/KEYCLOAK/) KEYCLOAK DEMO (HTTPS://WWW.COMAKEIT.COM/TAG/KEYCLOAK-DEMO/)

KEYCLOAK TUTORIAL (HTTPS://WWW.COMAKEIT.COM/TAG/KEYCLOAK-TUTORIAL/)

♡ 5 (https://www.comakeit.com/wp-admin/admin-ajax.php?action=process_simple_like&nonce=c4038368c9&post_id=21920&disabled=true&is_comment=0)

(//www.facebook.com/sharer.php?u=https://www.comakeit.com/quick-guide-using-keycloak-identity-access-management/)

(//twitter.com/share?url=https://www.comakeit.com/quick-guide-using-keycloak-identity-access-management/)

(//www.linkedin.com/shareArticle?mini=true&url=https://www.comakeit.com/quick-guide-using-keycloak-identity-access-management/&title=A%20Quick%20Guide%20to%20Usi

SEARCH

Enter your keywords...

Careers (https://www.comakeit.com/work-comakeit/) | Contact us (https://www.comakeit.com/contact-us/) | Privacy Policy (https://www.comakeit.com/privacy-policy/) | Sitemap (https://www.comakeit.com/sitemap/) | Blog Sitemap (https://www.comakeit.com/blog/sitemap/)

(https://www.facebook.com/comakeit) (https://twitter.com/comakeit)

(https://www.linkedin.com/company/comakeit)