

<http://baeldung.com>

# A Quick Guide to Using Keycloak with Spring Boot

Last modified: November 14, 2017

by Michael Good (<http://www.baeldung.com/author/michael-good/>)

**Security** (<http://www.baeldung.com/category/security-2/>)

**Spring** (<http://www.baeldung.com/category/spring/>) +

If you have a few years of experience in the Java ecosystem, and you're interested in sharing that experience with the community (and getting paid for your work of course), have a look at the "Write for Us" page (</contribution-guidelines>). Cheers. Eugen

I just announced the new *Spring 5* modules in REST With Spring:

**>> CHECK OUT THE COURSE** (</rest-with-spring-course#new-modules>)

## 1. Overview

In this article, we'll cover the basics of setting up a Keycloak server, how to **connect a Spring Boot application** to it, and how to use it with Spring Security.

## 2. What is Keycloak?

**Keycloak is an open source Identity and Access Management solution targeted towards modern applications and services.**

Keycloak offers features such as Single-Sign-On (SSO), Identity Brokering and Social Login, User Federation, Client Adapters, an Admin Console, and an Account Management Console. To learn more about Keycloak, please visit the official page (<http://www.keycloak.org/>).

In our tutorial, we'll be using the Admin Console of Keycloak for setting up and then connecting to Spring Boot using the Keycloak Client Adapter.

## 3. Setting Up a Keycloak Server

### 3.1. Downloading and Installing Keycloak

There're several distributions to choose from.

However, in this tutorial, we'll be using the standalone version.

Download Keycloak-3.3.0.Final Standalone server distribution (<http://www.keycloak.org/downloads.html>) from the official source.

Once we've downloaded the Standalone server distribution, we can unzip and start Keycloak from the terminal:

```
1 unzip keycloak-3.3.0.Final.zip
2 cd keycloak-3.3.0.Final/bin
3 ./standalone.sh -Djboss.socket.binding.port-offset=100
```

After running `./standalone.sh`, Keycloak will be starting its services. Once we see a line containing *Keycloak 3.3.0.Final (WildFly Core 3.0.1.Final)* started, we'll know its start-up is complete.

Open a browser and visit <http://localhost:8180>. We'll be redirected to <http://localhost:8180/auth> to create an administrative login:



## Welcome to Keycloak

Please create an initial admin user to get started.

Username	<input type="text"/>
Password	<input type="password"/>
Password confirmation	<input type="password"/>
<input type="button" value="Create"/>	

([http://www.baeldung.com/wp-](http://www.baeldung.com/wp-content/uploads/2017/11/createKeycloakAdmin.png)

[Documentation](#) | [Administration Console](#)

[Keycloak Project](#) | [Mailing List](#) | [Report an issue](#)

 | [JBoss Community](#)

content/uploads/2017/11/createKeycloakAdmin.png)

Let's create a user named "initial1" with the password "zaq1!QAZ".

We now see "Welcome to Keycloak".



## Welcome to Keycloak

User created

[Documentation](#) | [Administration Console](#)

[Keycloak Project](#) | [Mailing List](#) | [Report an issue](#)

 | [JBoss Community](#)

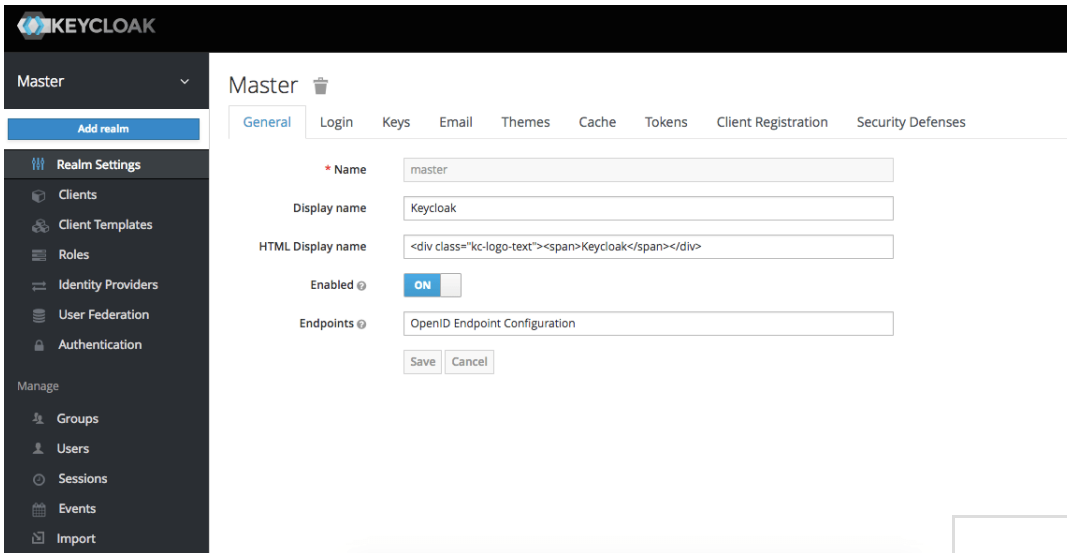
([http://www.baeldung.com/wp-](http://www.baeldung.com/wp-content/uploads/2017/11/welcomeKeycloak.png)

content/uploads/2017/11/welcomeKeycloak.png)

We can now proceed to the Administrative Console.

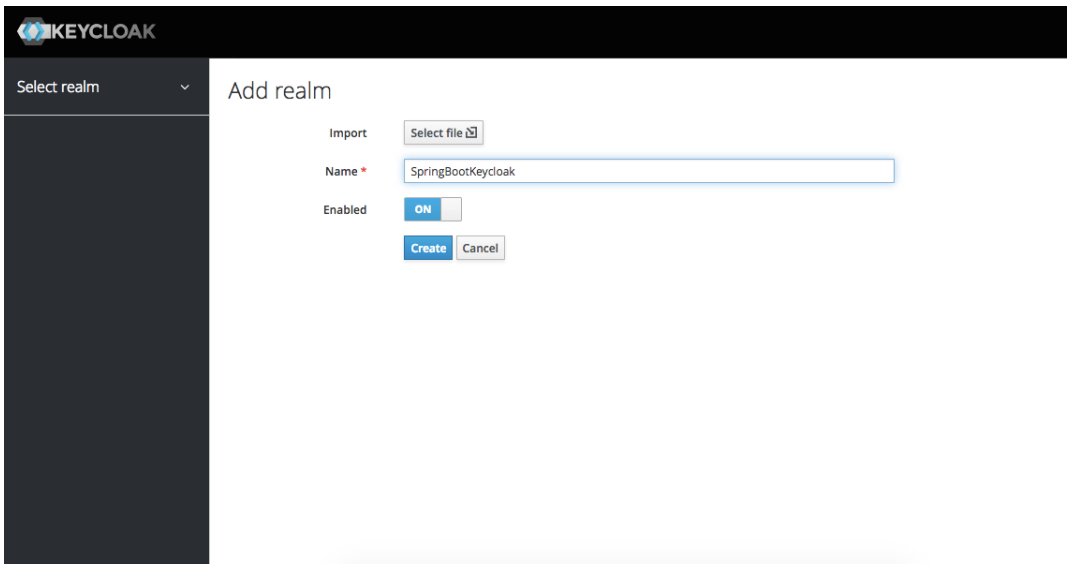
## 3.2. Creating a Realm

Let's navigate our mouse into the upper left upper corner to discover the "Create a Realm" button:



(<http://www.baeldung.com/wp-content/uploads/2017/11/addRealmKeycloak.png>)

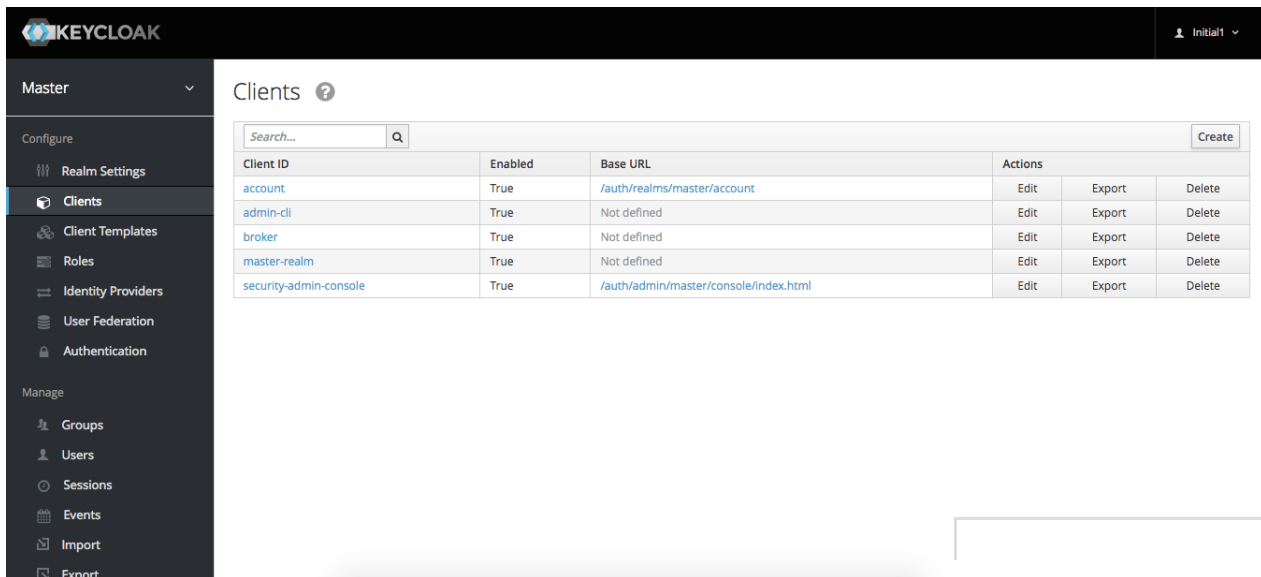
We're naming it "*SpringBootKeycloak*":



(<http://www.baeldung.com/wp-content/uploads/2017/11/realmSpringBootKeycloak.png>)

### 3.3. Creating a Client

Now we'll navigate to the Clients page. As we can see in the image below, **Keycloak comes with Clients that are already built in**:



(<http://www.baeldung.com/wp-content/uploads/2017/11/clientsPageKeycloak.png>)

But we need to add a client to our application, so we click "Create". We'll call the new Client "login-app":

### Add Client

Import

Client ID \*

Client Protocol

Client Template

Root URL

(<http://www.baeldung.com/wp-content/uploads/2017/11/addClientKeycloak.png>)

In the next screen, for this tutorial, we'll be leaving all the defaults except the "Valid Redirect URIs field". We'll be redirected to the port 8081:

Authorization Enabled ☐ OFF

Root URL

\* Valid Redirect URIs  +

Base URL

Admin URL

Web Origins  +

> Fine Grain OpenID Connect Configuration ?

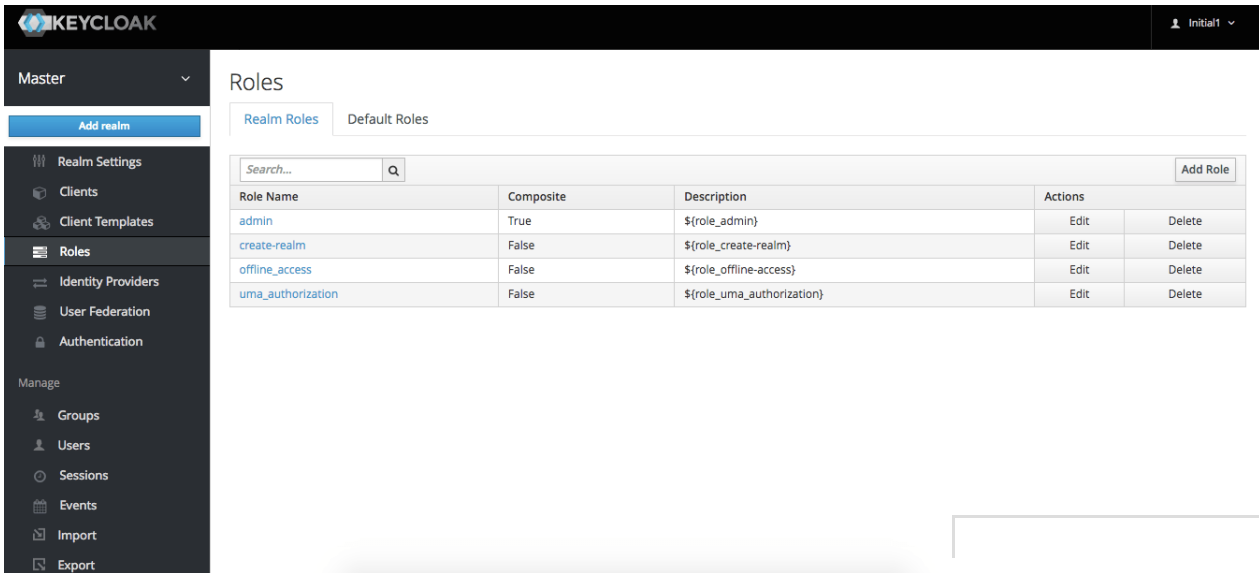
([http://www.baeldung.com/wp-](http://www.baeldung.com/wp-content/uploads/2017/11/validRedirectURI.png)

[content/uploads/2017/11/validRedirectURI.png](http://www.baeldung.com/wp-content/uploads/2017/11/validRedirectURI.png))

### 3.4. Creating a Role and a User

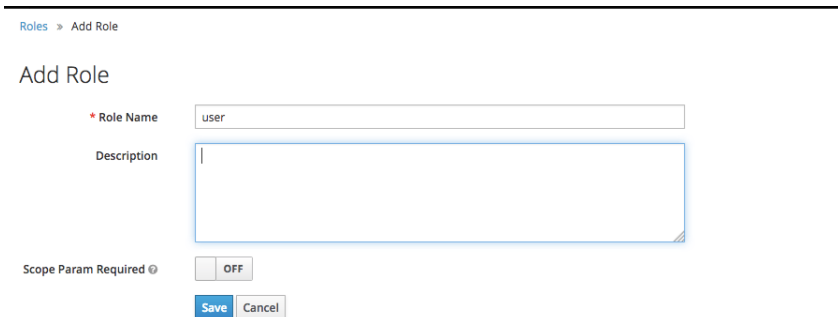
**Keycloak uses the Role-Based Access. Therefore, each user must have a role.**

We need to navigate to the "Role" page:



(<http://www.baeldung.com/wp-content/uploads/2017/11/rolesPageKeycloak.png>)

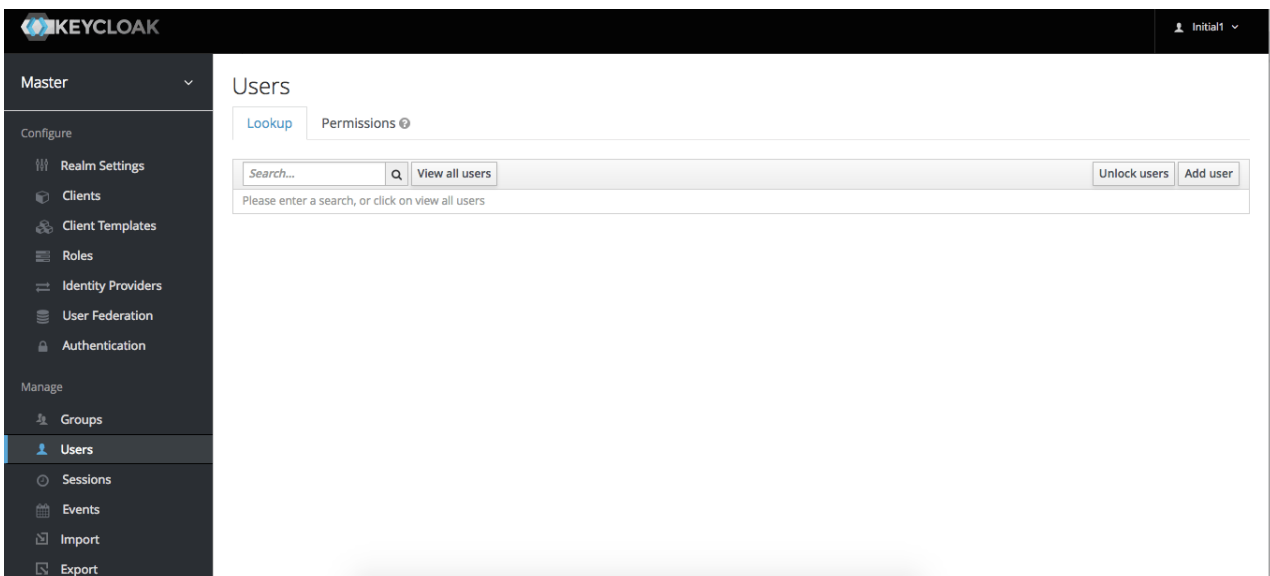
Then, we add the "user" role:



(<http://www.baeldung.com/wp-content/uploads/2017/11/addRoleKeycloak.png>)

content/uploads/2017/11/addRoleKeycloak.png)

Now we've got a role that can be assigned to users, but there are no users yet. So let's go the "Users" page and add one:



(<http://www.baeldung.com/wp-content/uploads/2017/11/usersPageKeycloak.png>)

We add the user "user1":

[Users](#) > Add user

## Add user

ID

Created At

Username \*

Email

First Name

Last Name

User Enabled ☒

Email Verified ☐

Required User Actions

(http://www.baeldung.com/wp-

content/uploads/2017/11/addUserKeycloak.png)

Once the user is created, we'll be shown this page:

Master

Configure

- Realm Settings
- Clients
- Client Templates
- Roles
- Identity Providers
- User Federation
- Authentication

Manage

- Groups
- Users**
- Sessions
- Events
- Import
- Export

User1

Details Attributes Credentials Role Mappings Groups Consents Sessions

ID

Created At 10/23/17 8:53:33 AM

Username

Email

First Name

Last Name

User Enabled ☒

Email Verified ☐

Required User Actions

Impersonate user

(http://www.baeldung.com/wp-content/uploads/2017/11/userCreatedKeycloak.png)

We can now go to the "Credentials" tab. We'll be setting the password to "xsw2@WSX".

User1

Details Attributes **Credentials** Role Mappings Groups Consents Sessions

Manage Password

New Password

Password Confirmation

Temporary ☒

(http://www.baeldung.com/wp-content/uploads/2017/11/setPasswordKeycloak.png)

We navigate to the "Role Mappings" tab. We'll be assigning the user role:

Users » user1

User1 

Details	Attributes	Credentials	Role Mappings	Groups	Consents	Sessions
<div> <div> <div>Realm Roles</div> <div>Available Roles ⓘ</div> <div> admin create-realm <b>user</b> </div> <div>Add selected &gt;</div> </div> <div> <div>Assigned Roles ⓘ</div> <div> offline_access uma_authorization </div> <div>&lt;&lt; Remove selected</div> </div> <div> <div>Effective Roles ⓘ</div> <div> offline_access uma_authorization </div> </div> </div> <div> <div>Client Roles</div> <div>Select client to view roles for client</div> <div> <div></div> </div> </div>						

(http://www.baeldung.com/wp-

content/uploads/2017/11/roleMappingKeycloak.png)

## 4. Creating a Spring Boot Application

### 4.1. Dependencies

The latest Spring Boot Keycloak Starter dependencies can be found on Maven Central (<http://search.maven.org/#search%7Cga%7C1%7Ca%3A%22keycloak-spring-boot-starter%22>).

**The Keycloak Spring Boot adapter capitalizes on Spring Boot's auto-configuration**, so all we need to do is add the Keycloak Spring Boot starter to our project.

Within the dependencies XML element, we need the following to run Keycloak with Spring Boot:

```

1 <dependency>
2   <groupId>org.keycloak</groupId>
3   <artifactId>keycloak-spring-boot-starter</artifactId>
4 </dependency>

```

After the dependencies XML element, we need to specify *dependencyManagement* for Keycloak:

```

1 <dependencyManagement>
2   <dependencies>
3     <dependency>
4       <groupId>org.keycloak.bom</groupId>
5       <artifactId>keycloak-adapter-bom</artifactId>
6       <version>3.3.0.Final</version>
7       <type>pom</type>
8       <scope>import</scope>
9     </dependency>
10   </dependencies>
11 </dependencyManagement>

```

The following embedded containers are supported now and don't require any extra dependencies if using Spring Boot Keycloak Starter:

- Tomcat
- Undertow
- Jetty

### 4.2. Thymeleaf Web Pages

We're using Thymeleaf for our web pages.

We've got three pages:

- *external.html* – an externally facing web page for the public
- *customers.html* – an internally facing page that will have its access restricted to only authenticated users with the role "user."

- *layout.html* – a simple layout, consisting of two fragments, that is used for both the externally facing page and the internally facing page

The code for the Thymeleaf templates is available on Github (<https://github.com/eugenp/tutorials/tree/master/spring-boot-keycloak/src/main/resources/templates>).

### 4.3. Controller

The web controller maps the internal and external URLs to the appropriate Thymeleaf templates:

```
1 | @GetMapping(path = "/")
2 | public String index() {
3 |     return "external";
4 | }
5 |
6 | @GetMapping(path = "/customers")
7 | public String customers(Model model) {
8 |     addCustomers();
9 |     model.addAttribute("customers", customerDAO.findAll());
10 |    return "customers";
11 | }
```

For the path */customers*, we're retrieving all customers from a repository and adding the result as an attribute to *Model*. Later on, we iterate through the results in Thymeleaf.

### 4.4. Keycloak Configuration

Here's the basic, mandatory configuration:

```
1 | keycloak.auth-server-url=http://localhost:8180/auth
2 | keycloak.realm=SpringBootKeycloak
3 | keycloak.resource=login-app
4 | keycloak.public-client=true
```

As we recall, we started Keycloak on port *8180*, hence the path specified in *keycloak.auth-server-url*. We enter the realm name we created in the Keycloak admin console.

The value we specify in *keycloak.resource* matches the client we named in the admin console.

Here are security constraints we'll be using:

```
1 | keycloak.security-constraints[0].authRoles[0]=user
2 | keycloak.security-constraints[0].securityCollections[0].patterns[0]=/customers/*
```

The above security restraints state ensure every request to */customers/\** will only be authorized if the one requesting it's an authenticated user with the role "user".

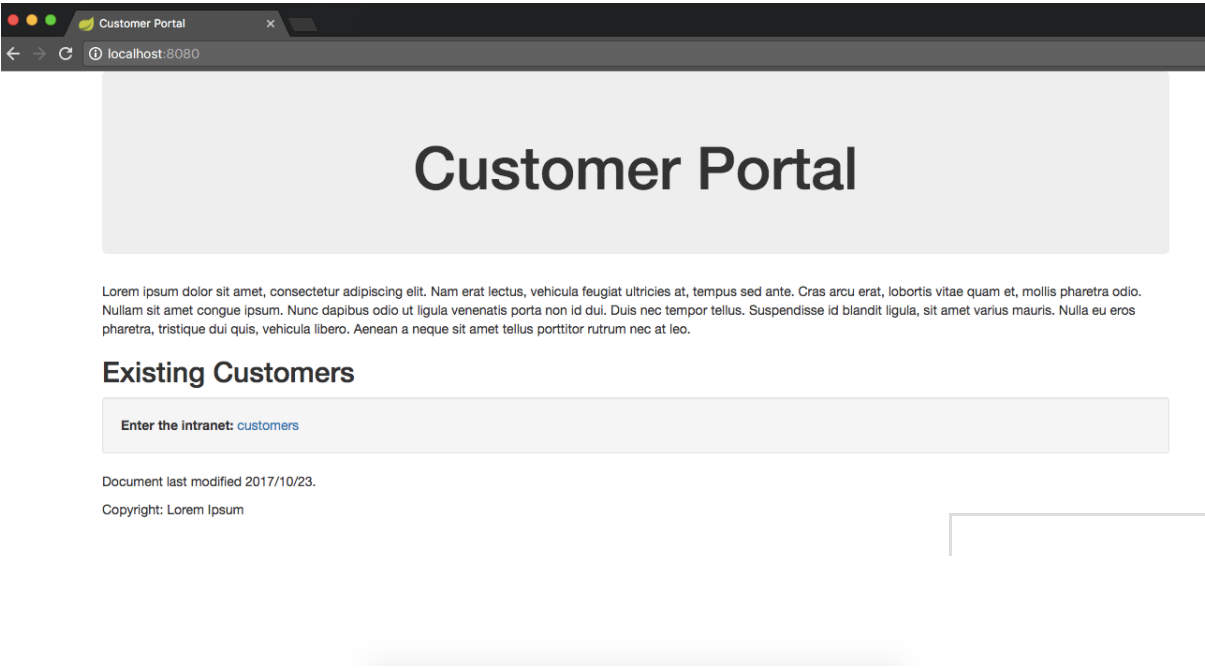
### 4.5. Demonstration

Now, we're ready to test our application. To run a Spring Boot application, we can start it easily through an IDE like Spring Tool Suite (STS) or run this command in the terminal:

```
1 | mvn clean spring-boot:run
```

We visit *localhost:8080*:

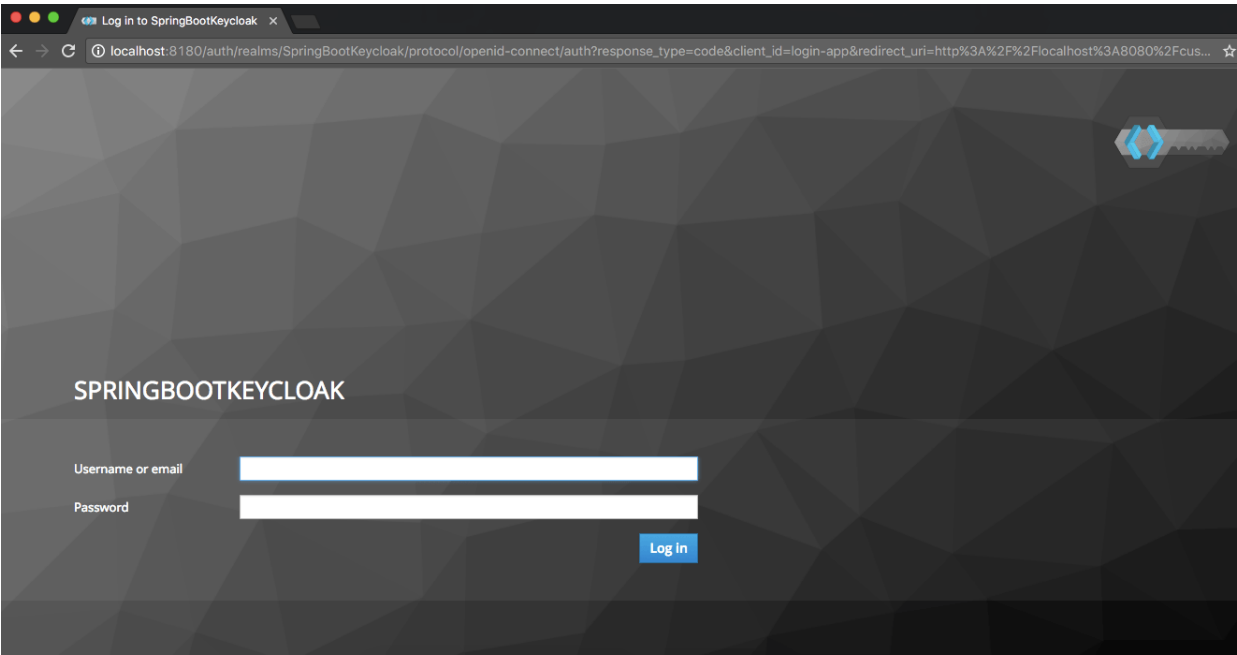




(<http://www.baeldung.com/wp-content/uploads/2017/11/externalFacingKeycloakPage.png>)

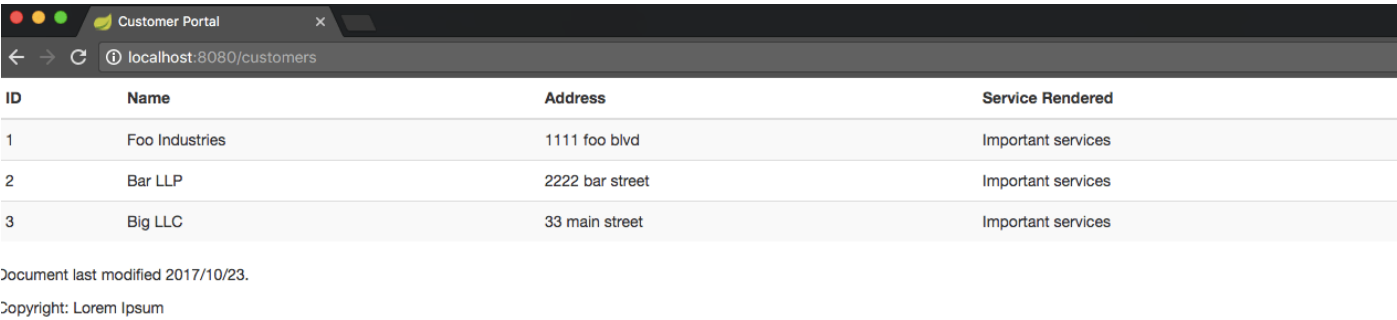
Now we click “customers” to enter the intranet, which is the location of sensitive information.

We can see that we’ve been redirected to authenticate through Keycloak to see if we’re authorized to view this content:



(<http://www.baeldung.com/wp-content/uploads/2017/11/keyCloakloginPage.png>)

Once we authenticate and our authorization is checked by Keycloak, we’re redirected to the restricted customers’ page:



(<http://www.baeldung.com/wp-content/uploads/2017/11/internalPageKeycloakApp.png>)

Now we've finished the set up of connecting Spring Boot with Keycloak and demonstrating how it works.

Now we'll be reviewing how to use Spring Security in conjunction with our existing application.

## 5. Spring Security

There is a Keycloak Spring Security Adapter, and it's **already included in our Spring Boot Keycloak Starter dependency**. We'll now see how to integrate Spring Security with Keycloak.

### 5.1. Dependency

To use Spring Security with Spring Boot, we must add this dependency:

```
1 <dependency>
2   <groupId>org.springframework.boot</groupId>
3   <artifactId>spring-boot-starter-security</artifactId>
4   <version>1.5.3</version>
5 </dependency>
```

The latest Spring Boot Starter Security release can be found on Maven Central (<http://search.maven.org/#search%7Cgav%7C1%7Cg%3A%22org.springframework.boot%22%20AND%20a%3A%22spring-boot-starter-security%22>).

### 5.2. Configuration class

Keycloak provides a *KeycloakWebSecurityConfigurerAdapter* as a convenient base class for creating a *WebSecurityConfigurer* (<http://docs.spring.io/spring-security/site/docs/4.0.x/apidocs/org/springframework/security/config/annotation/web/WebSecurityConfigurer.html>) instance which is convenient because a configuration class extending *WebSecurityConfigurerAdapter* is needed for any application secured by Spring Security:

```

1  @Configuration
2  @EnableWebSecurity
3  @ComponentScan(basePackageClasses = KeycloakSecurityComponents.class)
4  class SecurityConfig extends KeycloakWebSecurityConfigurerAdapter {
5
6      @Autowired
7      public void configureGlobal(
8          AuthenticationManagerBuilder auth) throws Exception {
9
10         KeycloakAuthenticationProvider keycloakAuthenticationProvider
11             = keycloakAuthenticationProvider();
12         keycloakAuthenticationProvider.setGrantedAuthoritiesMapper(
13             new SimpleAuthorityMapper());
14         auth.authenticationProvider(keycloakAuthenticationProvider);
15     }
16
17     @Bean
18     public KeycloakSpringBootConfigResolver keycloakConfigResolver() {
19         return new KeycloakSpringBootConfigResolver();
20     }
21
22     @Bean
23     @Override
24     protected SessionAuthenticationStrategy sessionAuthenticationStrategy() {
25         return new RegisterSessionAuthenticationStrategy(
26             new SessionRegistryImpl());
27     }
28
29     @Override
30     protected void configure(HttpSecurity http) throws Exception {
31         super.configure(http);
32         http.authorizeRequests()
33             .antMatchers("/customers*")
34             .hasRole("user")
35             .anyRequest()
36             .permitAll();
37     }
38 }

```

Please note the code above:

- *configureGlobal*: tasks the *SimpleAuthorityMapper* to make sure roles are not prefixed with *ROLE\_*
- *keycloakConfigResolver*: this defines that we want to use the Spring Boot properties file support instead of the default *keycloak.json*

### 5.3. application.properties

Because we've set up the security constraints with Spring Security, we can remove the previous security constraints we placed in *application.properties*.

Now we'll add this to our *application.properties*:

```
1 | keycloak.principal-attribute=preferred_username
```

### 5.4. Controller

To make use of a user's username, we're updating our controller to inject the *Principal*:

```

1  @GetMapping(path = "/customers")
2  public String customers(Principal principal, Model model){
3      addCustomers();
4      model.addAttribute("customers", customerDAO.findAll());
5      model.addAttribute("username", principal.getName());
6      return "customers";
7  }

```

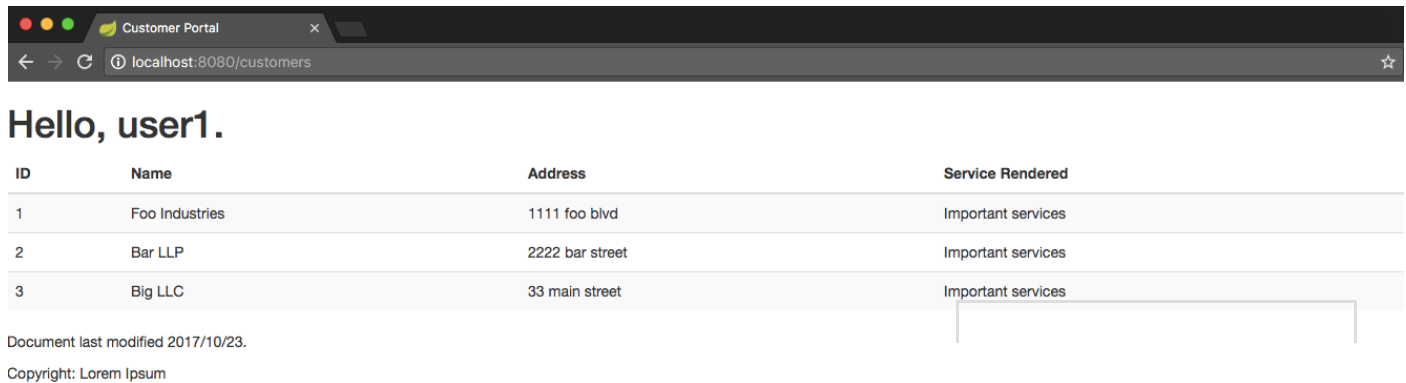
### 5.5. Thymeleaf

Under the div container, we'll add this one line to greet the user:

```
1 | <h1>Hello, <span th:text='${username}'>--name--</span>.</h1>
```

## 5.6. Demo

Now, after we authenticate and are taken to the internal customers' page, we'll see:



ID	Name	Address	Service Rendered
1	Foo Industries	1111 foo blvd	Important services
2	Bar LLP	2222 bar street	Important services
3	Big LLC	33 main street	Important services

Document last modified 2017/10/23.  
Copyright: Lorem Ipsum

(<http://www.baeldung.com/wp-content/uploads/2017/11/springSecurityKeycloak.png>)

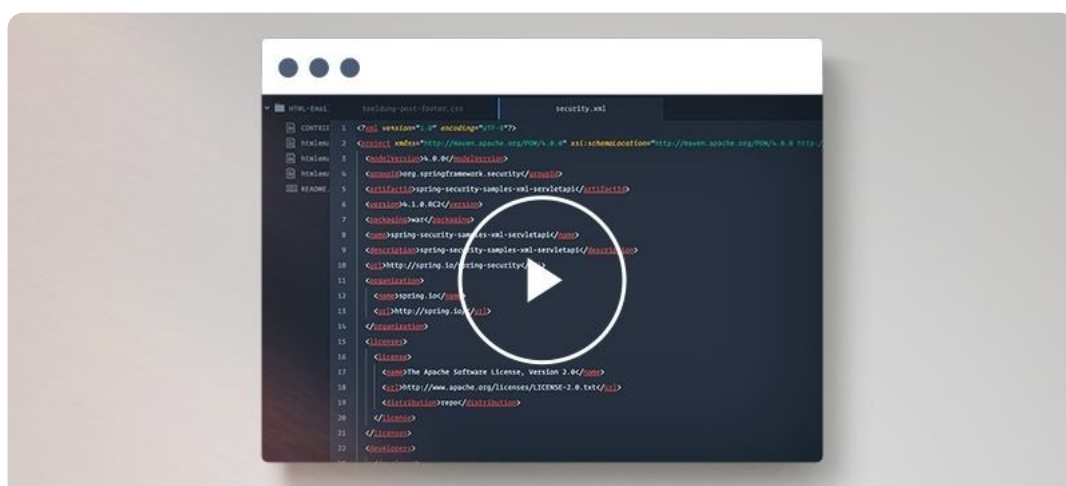
## 6 Conclusion

In this tutorial, we've configured a Keycloak server and used it with a Spring Boot Application.

We've also seen how to set up Spring Security and use it in conjunction with Keycloak. A working version of the code shown in this article is available over on Github (<https://github.com/eugenp/tutorials/tree/master/spring-boot-keycloak>).

**I just announced the new Spring 5 modules in REST With Spring:**

**>> CHECK OUT THE LESSONS (</rest-with-spring-course#new-modules>)**



**Learn the basics of securing a REST API with Spring**  
**Get access to the video lesson!**

## Email Address

## Access &gt;&gt;

Sort by: newest

Petr Vich

Hi,

Guest

I have tried this tutorial and I have problem with returning to my application. In my log is exception

```
javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target
```

Could you please help me?

Thanks.

Petr

1 month 26 days ago ^

Michael Good

(http://www.michaelcgood.com)

(http://www.baeldung.com/author/michael-good/)

Author

Hi Petr, I am sorry to hear you are having problems with the tutorial. Are you perhaps trying to use an SSL certificate with an instance of Keycloak? While this wasn't part of the tutorial, that's what your error suggests. If so, your trust store may be misconfigured. Please try the following links as a launching point if you need more information: – <http://lists.jboss.org/pipermail/keycloak-user/2016-April/005679.html> – <https://issues.jboss.org/browse/KEYCLOAK-2362> If you are not trying to use an SSL certificate, please consider reviewing these settings: – the port you are running on is 8180 (step 3.1 of the tutorial) – that the application.properties is... Read more »

1 month 21 days ago

khushbu

Guest

Could you please tell me why are you using port number 8081, in the redirect url setting of keycloak. Actually I tried the same example but used redirect uri to localhost:8080. And I am hitting invalid parameter redirect uri. I tried to use your example only.

1 month 11 days ago

CATEGORIES

[SPRING \(HTTP://WWW.BAELDUNG.COM/CATEGORY/SPRING/\)](http://www.baeldung.com/category/spring/)

[REST \(HTTP://WWW.BAELDUNG.COM/CATEGORY/REST/\)](http://www.baeldung.com/category/rest/)

[JAVA \(HTTP://WWW.BAELDUNG.COM/CATEGORY/JAVA/\)](http://www.baeldung.com/category/java/)

[SECURITY \(HTTP://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/\)](http://www.baeldung.com/category/security-2/)

[PERSISTENCE \(HTTP://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/\)](http://www.baeldung.com/category/persistence/)

[JACKSON \(HTTP://WWW.BAELDUNG.COM/CATEGORY/JACKSON/\)](http://www.baeldung.com/category/jackson/)

[HTTPCLIENT \(HTTP://WWW.BAELDUNG.COM/CATEGORY/HTTP/\)](http://www.baeldung.com/category/http/)

ABOUT

[ABOUT BAELDUNG \(HTTP://WWW.BAELDUNG.COM/ABOUT/\)](http://www.baeldung.com/about/)

[THE COURSES \(HTTP://COURSES.BAELDUNG.COM\)](http://courses.baeldung.com/)

[META BAELDUNG \(HTTP://META.BAELDUNG.COM/\)](http://meta.baeldung.com/)

<http://www.baeldung.com/spring-boot-keycloak>

13/13