* [Home](https://www.javainuse.com/home)
* [Java](https://www.javainuse.com/)
* [Spring](https://www.javainuse.com/)
* [Angular](https://www.javainuse.com/)
* [Full Stack](https://www.javainuse.com/)
* [Apache Camel](https://www.javainuse.com/)
* [Cloud Frameworks](https://www.javainuse.com/)
* [Messaging](https://www.javainuse.com/)
* [Drools](https://www.javainuse.com/)
* [Search Engine](https://www.javainuse.com/)
* [Hazelcast](https://www.javainuse.com/hazelcast)
* [DevOps](https://www.javainuse.com/)

Search Tutorials

Top of Form

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | |  |

Bottom of Form

* [Home](https://www.javainuse.com/home)
* [Java](https://www.javainuse.com/)
* [Spring](https://www.javainuse.com/)
* [Angular](https://www.javainuse.com/)
* [Full Stack](https://www.javainuse.com/)
* [Apache Camel](https://www.javainuse.com/)
* [Cloud Frameworks](https://www.javainuse.com/)
* [Messaging](https://www.javainuse.com/)
* [Drools](https://www.javainuse.com/)
* [Search Engine](https://www.javainuse.com/)
* [Hazelcast](https://www.javainuse.com/hazelcast)
* [DevOps](https://www.javainuse.com/)

Search Tutorials

Top of Form

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | |  |

Bottom of Form

Spring Boot + Security Hello world Example

[**Close X**](https://www.javainuse.com/)

In this post we configure a spring boot application to add basic authorization and authentication.[Spring MVC Security](https://www.javainuse.com/spring/sprsec_authprovider) had created a Simple Spring MVC Security example using Basic Authentication . But as can be seen in that post lot of configuration had to be done. This chapter we see how simple it is for configuring security with Spring Boot. We will be adding the spring security configuration for the [Spring Boot web project](https://www.javainuse.com/spring/SpringBoot_DataJPA) developed previously. We have implemented [Spring Boot Form Security Login Hello World Example](https://www.javainuse.com/spring/boot_form_security) here

Spring Boot Security - Table Of Contents

[**Spring Boot + Simple Security Configuration**](https://www.javainuse.com/spring/sprboot_sec)

[Spring Boot Form Security Login Hello World Example](https://www.javainuse.com/spring/boot_form_security)

[Spring Boot Security - Custom Login Page Example](https://www.javainuse.com/spring/boot_form_security_custom_login)

[Spring Boot Security - JDBC Authentication Example](https://www.javainuse.com/spring/boot_security_jdbc_authentication)

[Spring Boot Security - Creating Users Programmatically Using JdbcUserDetailsManager](https://www.javainuse.com/spring/boot_security_jdbc_authentication_program)

[Spring Boot Security - Password Encoding Using Bcrypt](https://www.javainuse.com/spring/boot_security_jdbc_authentication_bcrypt)

[Spring Boot Security - Enabling CSRF Protection](https://www.javainuse.com/spring/boot_security_csrf)

[Spring Boot Security - Authentication Handler Example](https://www.javainuse.com/spring/boot_form_authentication_handler)

[Spring Boot Security - Introduction to OAuth](https://www.javainuse.com/spring/spring-boot-oauth-introduction)

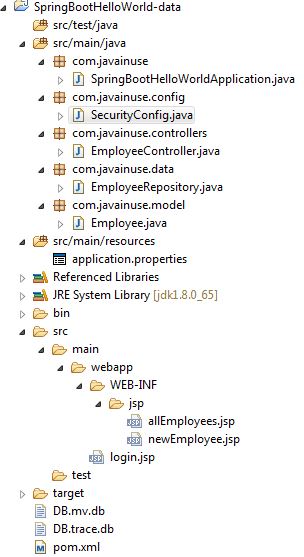
[Spring Boot OAuth2 Part 1 - Getting The Authorization Code](https://www.javainuse.com/spring/spring-boot-oauth-authorization-code)

[Spring Boot OAuth2 Part 2 - Getting The Access Token And Using it to Fetch Data.](https://www.javainuse.com/spring/spring-boot-oauth-access-token)

Video

This tutorial is explained in the below Youtube Video.

Lets Begin-

We will modify the project we developed.The maven project will be as follows-  
  
  
In the Maven we only need the **spring-boot-starter-security** dependency.Maven will be as follows-

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.javainuse</groupId>

<artifactId>SpringBootHelloWorld</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>SpringBootHelloWorld</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.1.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

**<dependency>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-security</artifactId>**

**<scope>test</scope>**

**</dependency>**

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<optional>true</optional>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

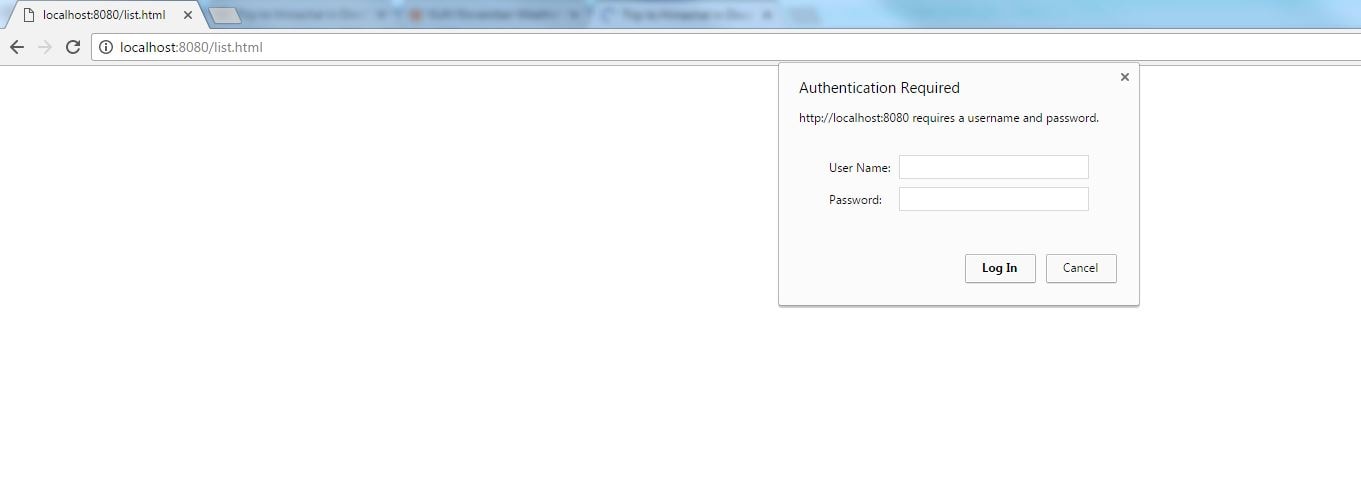
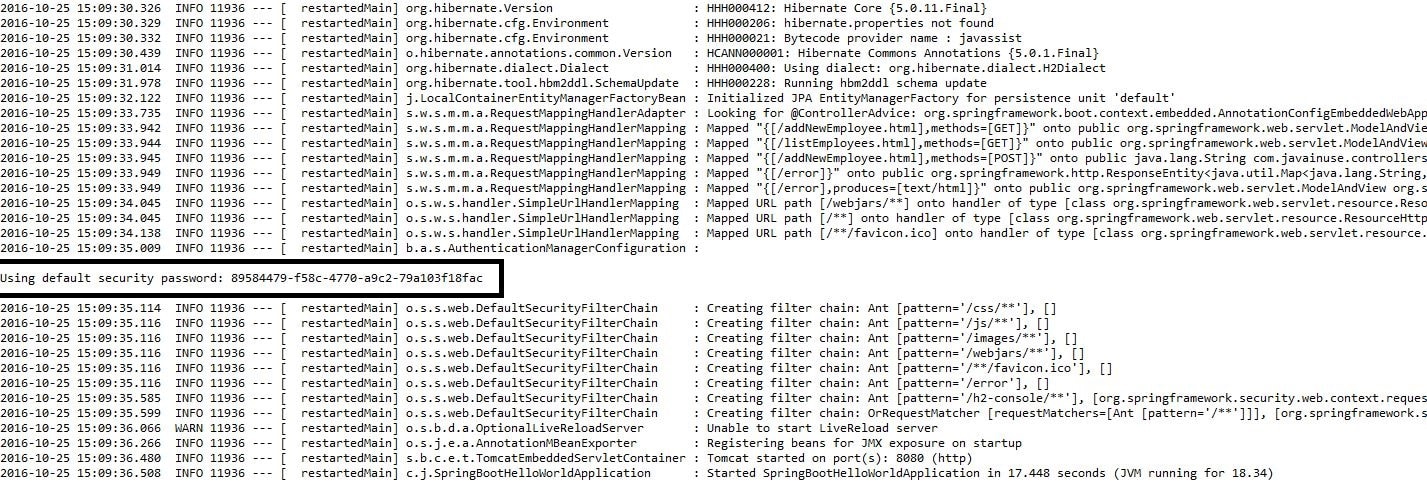
<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

Lets not add any security configuration and run this code.  
Goto-**http://localhost:8080/listEmployees.html**  
  
  
We can see that no security configuration is added still it asks for username password. This is expected behaviour. We will see that the password is system generated when we run the boot application.  
  
  
**Thus by just adding the spring boot security starter dependency the basic security has already been configured by default.**

Lets customize the security configuration by writing our own authorization and authentication. For this create a new class SecurityConfig that extends the WebSecurityConfigurerAdapter and overrides its methods.

package com.javainuse.config;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

public void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication().withUser("javainuse")

.password("javainuse").roles("USER");

}

@Override

public void configure(HttpSecurity http) throws Exception {

http.antMatcher("/\*\*").authorizeRequests().anyRequest().hasRole("USER")

.and().formLogin().loginPage("/login.jsp")

.failureUrl("/login.jsp?error=1").loginProcessingUrl("/login")

.permitAll().and().logout()

.logoutSuccessUrl("/listEmployees.html");

}

}

We have added a new page named login.jsp. Users will get redirected to this page for adding credentials.

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>

<html>

<head>

<title>Employees Login</title>

</head>

<body>

<h1><strong>Employees Login</strong></h1>

<c:url value="/login" var="login"/>

<form:form action="" method="post">

<label>Username:</label> <input type="text" name="username" />

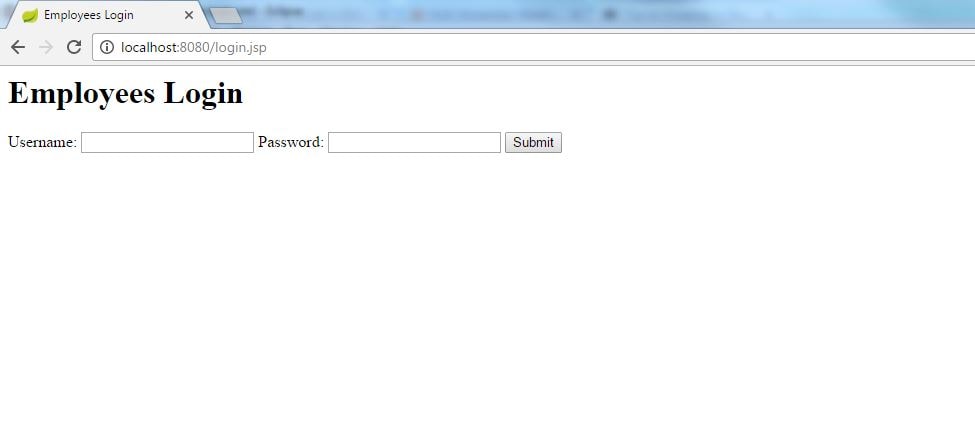
<label>Password:</label> <input type="text" name="password" />

<input type="submit"/>

</form:form>

</body>

</html>

This are the only changes required. Run the application. Goto-**http://localhost:8080/listEmployees.html**  
User will be automatically redirected to localhost:8080/login.jsp  
  
  
On entering the username and password as 'javainuse' the user will be able to visit all the other URLs correctly.

Download Source Code

Download it -  
[Spring Boot + Security Configuration](https://www.javainuse.com/zip/spring/boot/SpringBootHelloWorld-data_security.rar)

See Also

[Spring Boot Hello World Application- Create simple controller and jsp view using Maven](https://www.javainuse.com/spring/SpringBoot_HelloWorld)

[Spring Boot Hello World Application- Create simple controller and jsp view using Gradle](https://www.javainuse.com/spring/SpringBoot_HelloWorld_gradle)

[Spring Boot Tutorial-Spring Data JPA](https://www.javainuse.com/spring/SpringBoot_DataJPA)

[Pagination using Spring Boot Simple Example](https://www.javainuse.com/spring/SpringBootUsingPagination)

[Spring Boot + ActiveMQ Hello world Example](https://www.javainuse.com/spring/sprboot_activemq)

[Spring Boot + Swagger Example Hello World Example](https://www.javainuse.com/spring/boot_swagger)

[Spring Boot + Swagger- Understanding the various Swagger Annotations](https://www.javainuse.com/spring/boot_swagger_annotations)

[Spring Boot Main Menu](https://www.javainuse.com/spring/sprboot)

[Spring Boot Interview Questions](https://www.javainuse.com/spring/SpringBootInterviewQuestions)

Popular Posts

* [E-commerce Website - Online Book Store using Angular 8 + Spring Boot](https://www.javainuse.com/fullstack/ecommerce)
* [Spring Boot +JSON Web Token(JWT) Hello World Example](https://www.javainuse.com/spring/boot-jwt)
* [Angular 7 + Spring Boot Application Hello World Example](https://www.javainuse.com/spring/ang7-hello)
* [Build a Real Time Chat Application using Spring Boot + WebSocket + RabbitMQ](https://www.javainuse.com/spring/boot-websocket-chat)
* [Pivotal Cloud Foundry Tutorial - Deploy Spring Boot Application Hello World Example](https://www.javainuse.com/pcf/pcf-hello)
* [Deploying Spring Based WAR Application to Docker](https://www.javainuse.com/devOps/docker/docker-war)
* [EIP patterns using Apache Camel](https://www.javainuse.com/camel/camel_EIP)
* [Spring Cloud- Netflix Eureka + Ribbon Simple Example](https://www.javainuse.com/spring/spring_ribbon)
* [Spring Cloud- Netflix Hystrix Circuit Breaker Simple Example](https://www.javainuse.com/spring/spring_hystrix_circuitbreaker)
* [Spring Boot + Swagger Example Hello World Example](https://www.javainuse.com/spring/boot_swagger)
* [Spring Boot Batch Simple example](https://www.javainuse.com/spring/bootbatch)
* [Spring Boot + Apache Kafka Example](https://www.javainuse.com/spring/spring-boot-apache-kafka-hello-world)
* [Spring Boot Admin Simple Example](https://www.javainuse.com/spring/boot-admin)
* [Spring Boot Security - Introduction to OAuth](https://www.javainuse.com/spring/spring-boot-oauth-introduction)
* [Spring Boot OAuth2 Part 1 - Getting The Authorization Code](https://www.javainuse.com/spring/spring-boot-oauth-authorization-code)
* [Spring Boot OAuth2 Part 2 - Getting The Access Token And Using it to Fetch Data.](https://www.javainuse.com/spring/spring-boot-oauth-access-token)
* [JBoss Drools Hello World-Stateful Knowledge Session using KieSession](https://www.javainuse.com/drools_hello_kie)
* [Understand Drools Stateful vs Stateless Knowledge Session](https://www.javainuse.com/drools_states)
* [JBoss Drools- Understanding Drools Decision Table using Simple Example](https://www.javainuse.com/drools/drools_decision)

See Also

* [Spring Boot Interview Questions](https://www.javainuse.com/spring/SpringBootInterviewQuestions)
* [Spring Batch Interview Questions](https://www.javainuse.com/spring/sprbatch_interview)
* [Spring AOP Interview Questions](https://www.javainuse.com/spring/spring-AOP-interview-quesions)
* [Angular 2 Interview Questions](https://www.javainuse.com/angular/ang2_intvw)
* [Apache Camel Interview Questions](https://www.javainuse.com/camel/Apache_Camel_Questions)
* [JBoss Fuse Interview Questions](https://www.javainuse.com/camel/JBoss_Fuse_Questions)
* [Drools Interview Questions](https://www.javainuse.com/drools/drools_intvw)
* [Java 8 Interview Questions](https://www.javainuse.com/java/java8_intvw)
* [Spring Cloud Interview Questions](https://www.javainuse.com/spring/spring-cloud-interview-questions)
* [Microservices Interview Questions](https://www.javainuse.com/spring/microservices-interview-quesions)
* [Java HashMap and ConcurrentHashMap Interview Questions](https://www.javainuse.com/java/java_map_intvw)
* [Mule ESB frequently asked interview questions](https://www.javainuse.com/misc/muleintvw)
* [Apache Kafka Interview Questions](https://www.javainuse.com/misc/apache-kafka-interview-questions)
* [Tosca Testing Tool Interview Questions](https://www.javainuse.com/misc/tosca-testing-tool-interview-questions)
* [Top Maven Build Tool Interview Questions](https://www.javainuse.com/misc/maven-interview-questions)
* [Top Gradle Build Tool Interview Questions](https://www.javainuse.com/misc/gradle-interview-questions)
* [Miscellaneous Topics](https://www.javainuse.com/misc)

© Copyright JavaInUse. All Rights Reserved.

* [Home](https://www.javainuse.com/home)
* [Java](https://www.javainuse.com/)
* [Spring](https://www.javainuse.com/)
* [Angular](https://www.javainuse.com/)
* [Full Stack](https://www.javainuse.com/)
* [Apache Camel](https://www.javainuse.com/)
* [Cloud Frameworks](https://www.javainuse.com/)
* [Messaging](https://www.javainuse.com/)
* [Drools](https://www.javainuse.com/)
* [Search Engine](https://www.javainuse.com/)
* [Hazelcast](https://www.javainuse.com/hazelcast)
* [DevOps](https://www.javainuse.com/)

Search Tutorials

Top of Form

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | |  |

Bottom of Form

Spring Boot Form Security Login Example

In a previous post we had developed a [Spring Boot Security Hello World Application](https://www.javainuse.com/spring/sprboot_sec)  
It used http basic security. Besides being not visually helpful it also has some other drawbacks like once logged in, then to log out user has to close the browser.  
In this example we make use of the login form provided by Spring Security for authenticating users. We will be adding the security to the [Spring Boot Form Handling Example](https://www.javainuse.com/spring/spring-boot-form-handling) we had created before.

Spring Boot Security - Table Of Contents

[Spring Boot + Simple Security Configuration](https://www.javainuse.com/spring/sprboot_sec)

[**Spring Boot Form Security Login Hello World Example**](https://www.javainuse.com/spring/boot_form_security)

[Spring Boot Security - Custom Login Page Example](https://www.javainuse.com/spring/boot_form_security_custom_login)

[Spring Boot Security - Database Authentication using JDBC](https://www.javainuse.com/spring/boot_security_jdbc_authentication)

[Spring Boot Security - Creating Users Programmatically Using JdbcUserDetailsManager](https://www.javainuse.com/spring/boot_security_jdbc_authentication_program)

[Spring Boot Security - Password Encoding Using Bcrypt](https://www.javainuse.com/spring/boot_security_jdbc_authentication_bcrypt)

[Spring Boot Security - Enabling CSRF Protection](https://www.javainuse.com/spring/boot_security_csrf)

[Spring Boot Security - Authentication Handler Example](https://www.javainuse.com/spring/boot_form_authentication_handler)

[Spring Boot Security - Introduction to OAuth](https://www.javainuse.com/spring/spring-boot-oauth-introduction)

[Spring Boot OAuth2 Part 1 - Getting The Authorization Code](https://www.javainuse.com/spring/spring-boot-oauth-authorization-code)

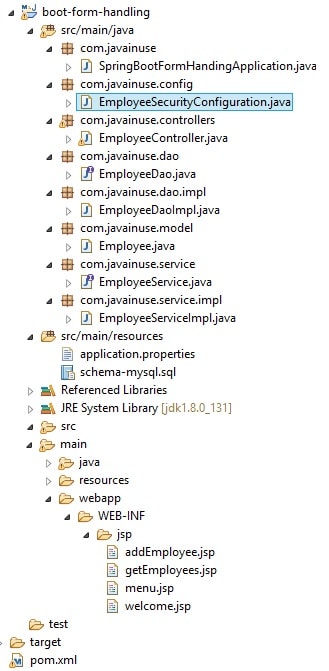
[Spring Boot OAuth2 Part 2 - Getting The Access Token And Using it to Fetch Data.](https://www.javainuse.com/spring/spring-boot-oauth-access-token)

Video

This tutorial is explained in the below Youtube Video.

Lets Begin-

Maven Project will be as follows-

  
In the Maven we need to add the spring boot security dependency to the existing dependencies.

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.javainuse</groupId>

<artifactId>boot-form-handling</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>boot-form-handling</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.5.2.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-jdbc</artifactId>

</dependency>

**<dependency>**

**<groupId>org.springframework.boot</groupId>**

**<artifactId>spring-boot-starter-security</artifactId>**

**</dependency>**

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

<version>5.1.21</version>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

Next we configure the Spring Security. In the configuration we specify which urls are to be intercepted, and are to be accessed by which users and having which roles. Next we create users along with passwords and specify them a role.  
We will be creating two logins

|  |  |  |  |
| --- | --- | --- | --- |
| **Username** | **Role** | **Pages Accessible** | **Pages not Accessible** |
| javainuse | USER ADMIN | Welcome page Show All Employees Page Add Employee | None |
| employee | USER | Welcome page Show All Employees Page | Add Employee |

package com.javainuse.config;

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

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.builders.WebSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration

@EnableWebSecurity

public class EmployeeSecurityConfiguration extends WebSecurityConfigurerAdapter {

@Override

public void configure(WebSecurity web) throws Exception {

web.ignoring().antMatchers("/resources/\*\*");

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.authorizeRequests().antMatchers("/").permitAll().antMatchers("/welcome")

.hasAnyRole("USER", "ADMIN").antMatchers("/getEmployees").hasAnyRole("USER", "ADMIN")

.antMatchers("/addNewEmployee").hasAnyRole("ADMIN").anyRequest().authenticated().and().formLogin()

.permitAll().and().logout().permitAll();

http.csrf().disable();

}

@Autowired

public void configureGlobal(AuthenticationManagerBuilder authenticationMgr) throws Exception {

authenticationMgr.inMemoryAuthentication().withUser("employee").password("employee")

.authorities("ROLE\_USER").and().withUser("javainuse").password("javainuse")

.authorities("ROLE\_USER", "ROLE\_ADMIN");

}

}

These are the only java changes required.  
The other change is on the JSP side. Spring Security provides a default login and a logout page. The login page will be called automatically when spring intercepts any url which is authenticated. We add code to the menu.jsp to add the logout submenu which ends the user session and logs him out.

<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 20px;">

<a href="/welcome">Home</a> |

<a href="/addNewEmployee">Add

Employee</a> |   <a

href="/getEmployees">Show

Employees</a> |   <u><h2 style="color: red;">

<a onclick="document.forms['logoutForm'].submit()">Logout</a>

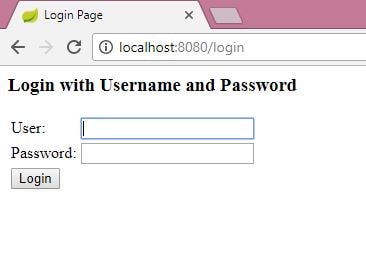
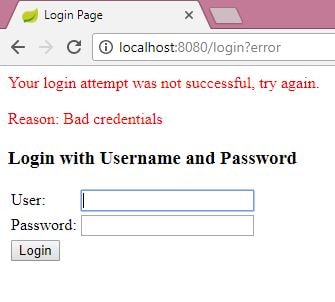
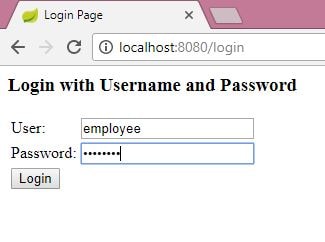
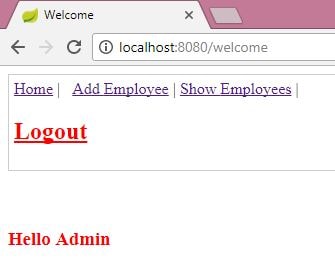
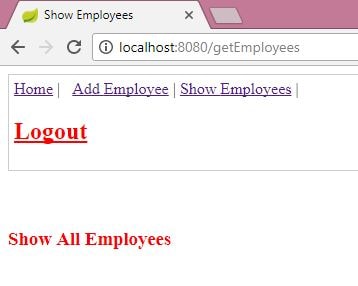
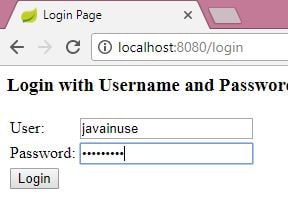
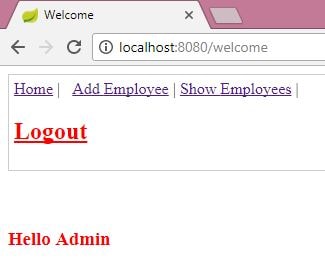
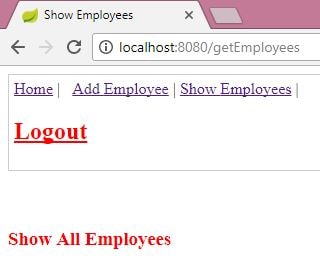
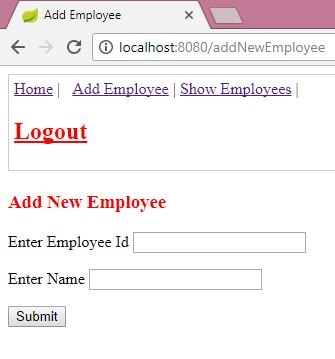
</h3></u>

<form id="logoutForm" method="POST" action="/logout">

</form>

</div>

These are the only changes required.

* Go to localhost:8080/welcome, we will be redirected to the default login page.  
  
* Enter wrong password.  
  
* Enter the correct credentials as employee and employee. User will be able to view the welcome and show all employees pages but not the add employee page.  
    
    
    
    
    
  On Add Employee get the security exception  
  
* Logout. And login using the credentials using javainuse and javainuse User will be able to view all pages including the add employee page.  
    
    
    
    
    
    
  

Download Source Code

Download it -  
[Spring Boot Form Security](https://www.javainuse.com/zip/spring/boot/boot-security.rar)

See Also

[Spring Boot Hello World Application- Create simple controller and jsp view using Maven](https://www.javainuse.com/spring/SpringBoot_HelloWorld)

[Spring Boot Tutorial-Spring Data JPA](https://www.javainuse.com/spring/SpringBoot_DataJPA)

[Spring Boot + Simple Security Configuration](https://www.javainuse.com/spring/sprboot_sec)

[Pagination using Spring Boot Simple Example](https://www.javainuse.com/spring/SpringBootUsingPagination)

[Spring Boot + ActiveMQ Hello world Example](https://www.javainuse.com/spring/sprboot_activemq)

[Spring Boot + Swagger Example Hello World Example](https://www.javainuse.com/spring/boot_swagger)

[Spring Boot + Swagger- Understanding the various Swagger Annotations](https://www.javainuse.com/spring/boot_swagger_annotations)

[Spring Boot Main Menu](https://www.javainuse.com/spring/sprboot)

[Spring Boot Interview Questions](https://www.javainuse.com/spring/SpringBootInterviewQuestions)

Popular Posts

* [E-commerce Website - Online Book Store using Angular 8 + Spring Boot](https://www.javainuse.com/fullstack/ecommerce)
* [Spring Boot +JSON Web Token(JWT) Hello World Example](https://www.javainuse.com/spring/boot-jwt)
* [Angular 7 + Spring Boot Application Hello World Example](https://www.javainuse.com/spring/ang7-hello)
* [Build a Real Time Chat Application using Spring Boot + WebSocket + RabbitMQ](https://www.javainuse.com/spring/boot-websocket-chat)
* [Pivotal Cloud Foundry Tutorial - Deploy Spring Boot Application Hello World Example](https://www.javainuse.com/pcf/pcf-hello)
* [Deploying Spring Based WAR Application to Docker](https://www.javainuse.com/devOps/docker/docker-war)
* [EIP patterns using Apache Camel](https://www.javainuse.com/camel/camel_EIP)
* [Spring Cloud- Netflix Eureka + Ribbon Simple Example](https://www.javainuse.com/spring/spring_ribbon)
* [Spring Cloud- Netflix Hystrix Circuit Breaker Simple Example](https://www.javainuse.com/spring/spring_hystrix_circuitbreaker)
* [Spring Boot + Swagger Example Hello World Example](https://www.javainuse.com/spring/boot_swagger)
* [Spring Boot Batch Simple example](https://www.javainuse.com/spring/bootbatch)
* [Spring Boot + Apache Kafka Example](https://www.javainuse.com/spring/spring-boot-apache-kafka-hello-world)
* [Spring Boot Admin Simple Example](https://www.javainuse.com/spring/boot-admin)
* [Spring Boot Security - Introduction to OAuth](https://www.javainuse.com/spring/spring-boot-oauth-introduction)
* [Spring Boot OAuth2 Part 1 - Getting The Authorization Code](https://www.javainuse.com/spring/spring-boot-oauth-authorization-code)
* [Spring Boot OAuth2 Part 2 - Getting The Access Token And Using it to Fetch Data.](https://www.javainuse.com/spring/spring-boot-oauth-access-token)
* [JBoss Drools Hello World-Stateful Knowledge Session using KieSession](https://www.javainuse.com/drools_hello_kie)
* [Understand Drools Stateful vs Stateless Knowledge Session](https://www.javainuse.com/drools_states)
* [JBoss Drools- Understanding Drools Decision Table using Simple Example](https://www.javainuse.com/drools/drools_decision)

See Also

* [Spring Boot Interview Questions](https://www.javainuse.com/spring/SpringBootInterviewQuestions)
* [Spring Batch Interview Questions](https://www.javainuse.com/spring/sprbatch_interview)
* [Spring AOP Interview Questions](https://www.javainuse.com/spring/spring-AOP-interview-quesions)
* [Angular 2 Interview Questions](https://www.javainuse.com/angular/ang2_intvw)
* [Apache Camel Interview Questions](https://www.javainuse.com/camel/Apache_Camel_Questions)
* [JBoss Fuse Interview Questions](https://www.javainuse.com/camel/JBoss_Fuse_Questions)
* [Drools Interview Questions](https://www.javainuse.com/drools/drools_intvw)
* [Java 8 Interview Questions](https://www.javainuse.com/java/java8_intvw)
* [Spring Cloud Interview Questions](https://www.javainuse.com/spring/spring-cloud-interview-questions)
* [Microservices Interview Questions](https://www.javainuse.com/spring/microservices-interview-quesions)
* [Java HashMap and ConcurrentHashMap Interview Questions](https://www.javainuse.com/java/java_map_intvw)
* [Mule ESB frequently asked interview questions](https://www.javainuse.com/misc/muleintvw)
* [Apache Kafka Interview Questions](https://www.javainuse.com/misc/apache-kafka-interview-questions)
* [Tosca Testing Tool Interview Questions](https://www.javainuse.com/misc/tosca-testing-tool-interview-questions)
* [Top Maven Build Tool Interview Questions](https://www.javainuse.com/misc/maven-interview-questions)
* [Top Gradle Build Tool Interview Questions](https://www.javainuse.com/misc/gradle-interview-questions)
* [Miscellaneous Topics](https://www.javainuse.com/misc)

© Copyright JavaInUse. All Rights Reserved.  
[Privacy Policy](https://www.javainuse.com/privacy)

[Privacy Policy](https://www.javainuse.com/privacy)

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

* [Home](https://www.javainuse.com/home)
* [Java](https://www.javainuse.com/)
* [Spring](https://www.javainuse.com/)
* [Angular](https://www.javainuse.com/)
* [Full Stack](https://www.javainuse.com/)
* [Apache Camel](https://www.javainuse.com/)
* [Cloud Frameworks](https://www.javainuse.com/)
* [Messaging](https://www.javainuse.com/)
* [Drools](https://www.javainuse.com/)
* [Search Engine](https://www.javainuse.com/)
* [Hazelcast](https://www.javainuse.com/hazelcast)
* [DevOps](https://www.javainuse.com/)

Search Tutorials

Top of Form

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | |  |

Bottom of Form

Spring Boot Form Security Example - Creating a custom Login Page

In a previous post we had implemented [Spring Boot Security for a Form Application](https://www.javainuse.com/spring/boot_form_security).  
It made use of the default Spring Login Page. In this tutorial we will adding our own **custom login web page**. On log out we will be directed to this login page with some logout message.

Spring Boot Security - Table Of Contents

[Spring Boot + Simple Security Configuration](https://www.javainuse.com/spring/sprboot_sec)

[Spring Boot Form Security Login Hello World Example](https://www.javainuse.com/spring/boot_form_security)

[**Spring Boot Security - Custom Login Page Example**](https://www.javainuse.com/spring/boot_form_security_custom_login)

[Spring Boot Security - JDBC Authentication Example](https://www.javainuse.com/spring/boot_security_jdbc_authentication)

[Spring Boot Security - Creating Users Programmatically Using JdbcUserDetailsManager](https://www.javainuse.com/spring/boot_security_jdbc_authentication_program)

[Spring Boot Security - Password Encoding Using Bcrypt](https://www.javainuse.com/spring/boot_security_jdbc_authentication_bcrypt)

[Spring Boot Security - Enabling CSRF Protection](https://www.javainuse.com/spring/boot_security_csrf)

[Spring Boot Security - Authentication Handler Example](https://www.javainuse.com/spring/boot_form_authentication_handler)

[Spring Boot Security - Introduction to OAuth](https://www.javainuse.com/spring/spring-boot-oauth-introduction)

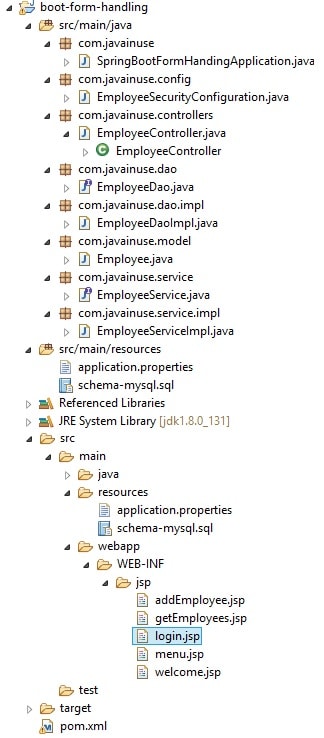
[Spring Boot OAuth2 Part 1 - Getting The Authorization Code](https://www.javainuse.com/spring/spring-boot-oauth-authorization-code)

[Spring Boot OAuth2 Part 2 - Getting The Access Token And Using it to Fetch Data.](https://www.javainuse.com/spring/spring-boot-oauth-access-token)

Video

This tutorial is explained in the below Youtube Video.

Lets Begin-

We will be modifying the code we developed in the previous [Spring Boot Security for a Form Application](https://www.javainuse.com/spring/boot_form_security)  
Maven Project will be as follows-  
  
  
The Custom login page login.jsp is as follows-

<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>

<c:set var="contextPath" value=""/>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- The above 3 meta tags \*must\* come first in the head; any other head content must come \*after\* these tags -->

<meta name="description" content="">

<meta name="author" content="">

<title>Log in with your credentials</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

</head>

<body>

<div class="container">

<form method="POST" action="/login" class="form-signin">

<h2 class="form-heading">Log in</h2>

<div class="form-group ">

<span></span>

<input name="username" type="text" class="form-control" placeholder="Username"

autofocus="true"/>

<input name="password" type="password" class="form-control" placeholder="Password"/>

<span></span>

<button class="btn btn-lg btn-primary btn-block" type="submit">Log In</button>

</div>

</form>

</div>

<!-- /container -->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script></body>

</html>

Next in the Controller, add a GET method which returns the custom login page with RequestMapping of /login.

package com.javainuse.controllers;

import java.util.List;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.servlet.ModelAndView;

import com.javainuse.model.Employee;

import com.javainuse.service.EmployeeService;

@Controller

public class EmployeeController {

@Autowired

EmployeeService employeeService;

@RequestMapping("/welcome")

public ModelAndView firstPage() {

return new ModelAndView("welcome");

}

@RequestMapping(value = "/addNewEmployee", method = RequestMethod.GET)

public ModelAndView show() {

return new ModelAndView("addEmployee", "emp", new Employee());

}

@RequestMapping(value = "/addNewEmployee", method = RequestMethod.POST)

public ModelAndView processRequest(@ModelAttribute("emp") Employee emp) {

employeeService.insertEmployee(emp);

List<Employee> employees = employeeService.getAllEmployees();

ModelAndView model = new ModelAndView("getEmployees");

model.addObject("employees", employees);

return model;

}

@RequestMapping("/getEmployees")

public ModelAndView getEmployees() {

List<Employee> employees = employeeService.getAllEmployees();

ModelAndView model = new ModelAndView("getEmployees");

model.addObject("employees", employees);

return model;

}

**@RequestMapping(value = "/login", method = RequestMethod.GET)**

**public String login(Model model, String error, String logout) {**

**if (error != null)**

**model.addAttribute("errorMsg", "Your username and password are invalid.");**

**if (logout != null)**

**model.addAttribute("msg", "You have been logged out successfully.");**

**return "login";**

**}**

}

Finally modify the Spring Security configuration where we specify the custom Login Page to be called during login.

package com.javainuse.config;

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

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.builders.WebSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration

@EnableWebSecurity

public class EmployeeSecurityConfiguration extends WebSecurityConfigurerAdapter {

@Override

public void configure(WebSecurity web) throws Exception {

web.ignoring().antMatchers("/resources/\*\*");

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.authorizeRequests().antMatchers("/").permitAll().antMatchers("/welcome").hasAnyRole("USER", "ADMIN")

.antMatchers("/getEmployees").hasAnyRole("USER", "ADMIN").antMatchers("/addNewEmployee")

.hasAnyRole("ADMIN").anyRequest().authenticated()

.and().formLogin().**loginPage("/login")**.permitAll()

.and().logout().permitAll();

http.csrf().disable();

}

@Autowired

public void configureGlobal(AuthenticationManagerBuilder authenticationMgr) throws Exception {

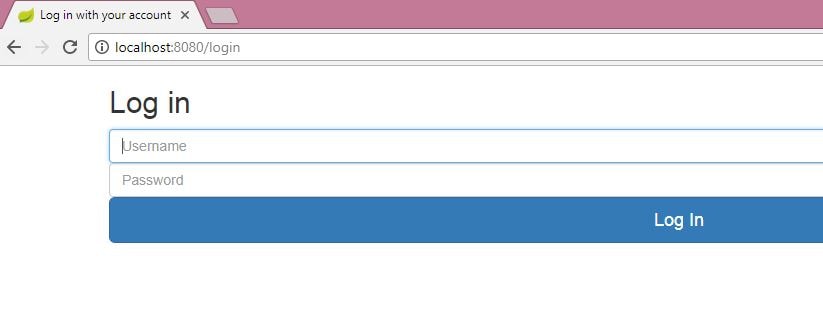
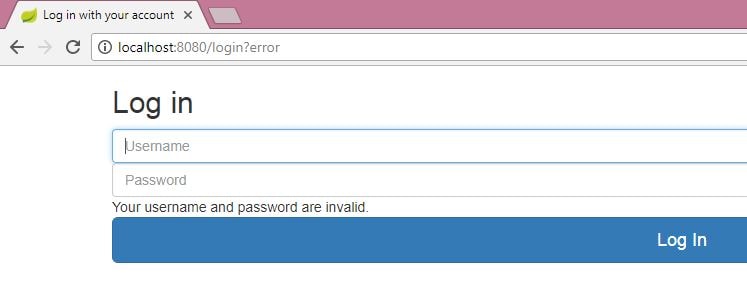
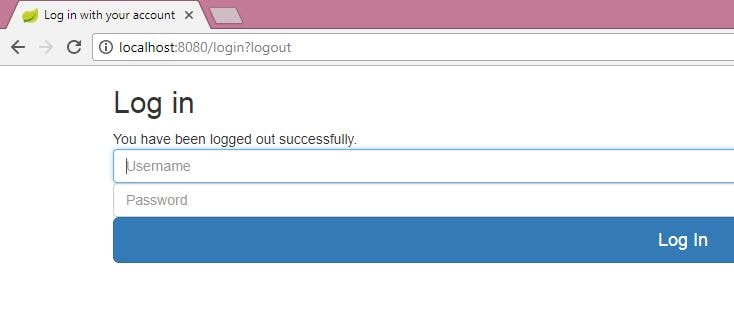
authenticationMgr.inMemoryAuthentication().withUser("admin").password("admin").authorities("ROLE\_USER").and()

.withUser("javainuse").password("javainuse").authorities("ROLE\_USER", "ROLE\_ADMIN");

}

}

These are the only changes required.

* Go to localhost:8080/welcome, we will be redirected to the custom login page.  
  
* Enter wrong password.  
  
* On log out we get the screen as follows-  
  

Download Source Code

Download it -  
[Spring Boot Security - Custom Login Page Example](https://www.javainuse.com/zip/spring/sec/boot-security-custom-login-page.rar)

Popular Posts

* [E-commerce Website - Online Book Store using Angular 8 + Spring Boot](https://www.javainuse.com/fullstack/ecommerce)
* [Spring Boot +JSON Web Token(JWT) Hello World Example](https://www.javainuse.com/spring/boot-jwt)
* [Angular 7 + Spring Boot Application Hello World Example](https://www.javainuse.com/spring/ang7-hello)
* [Build a Real Time Chat Application using Spring Boot + WebSocket + RabbitMQ](https://www.javainuse.com/spring/boot-websocket-chat)
* [Pivotal Cloud Foundry Tutorial - Deploy Spring Boot Application Hello World Example](https://www.javainuse.com/pcf/pcf-hello)
* [Deploying Spring Based WAR Application to Docker](https://www.javainuse.com/devOps/docker/docker-war)
* [EIP patterns using Apache Camel](https://www.javainuse.com/camel/camel_EIP)
* [Spring Cloud- Netflix Eureka + Ribbon Simple Example](https://www.javainuse.com/spring/spring_ribbon)
* [Spring Cloud- Netflix Hystrix Circuit Breaker Simple Example](https://www.javainuse.com/spring/spring_hystrix_circuitbreaker)
* [Spring Boot + Swagger Example Hello World Example](https://www.javainuse.com/spring/boot_swagger)
* [Spring Boot Batch Simple example](https://www.javainuse.com/spring/bootbatch)
* [Spring Boot + Apache Kafka Example](https://www.javainuse.com/spring/spring-boot-apache-kafka-hello-world)
* [Spring Boot Admin Simple Example](https://www.javainuse.com/spring/boot-admin)
* [Spring Boot Security - Introduction to OAuth](https://www.javainuse.com/spring/spring-boot-oauth-introduction)
* [Spring Boot OAuth2 Part 1 - Getting The Authorization Code](https://www.javainuse.com/spring/spring-boot-oauth-authorization-code)
* [Spring Boot OAuth2 Part 2 - Getting The Access Token And Using it to Fetch Data.](https://www.javainuse.com/spring/spring-boot-oauth-access-token)
* [JBoss Drools Hello World-Stateful Knowledge Session using KieSession](https://www.javainuse.com/drools_hello_kie)
* [Understand Drools Stateful vs Stateless Knowledge Session](https://www.javainuse.com/drools_states)
* [JBoss Drools- Understanding Drools Decision Table using Simple Example](https://www.javainuse.com/drools/drools_decision)

See Also

* [Spring Boot Interview Questions](https://www.javainuse.com/spring/SpringBootInterviewQuestions)
* [Spring Batch Interview Questions](https://www.javainuse.com/spring/sprbatch_interview)
* [Spring AOP Interview Questions](https://www.javainuse.com/spring/spring-AOP-interview-quesions)
* [Angular 2 Interview Questions](https://www.javainuse.com/angular/ang2_intvw)
* [Apache Camel Interview Questions](https://www.javainuse.com/camel/Apache_Camel_Questions)
* [JBoss Fuse Interview Questions](https://www.javainuse.com/camel/JBoss_Fuse_Questions)
* [Drools Interview Questions](https://www.javainuse.com/drools/drools_intvw)
* [Java 8 Interview Questions](https://www.javainuse.com/java/java8_intvw)
* [Spring Cloud Interview Questions](https://www.javainuse.com/spring/spring-cloud-interview-questions)
* [Microservices Interview Questions](https://www.javainuse.com/spring/microservices-interview-quesions)
* [Java HashMap and ConcurrentHashMap Interview Questions](https://www.javainuse.com/java/java_map_intvw)
* [Mule ESB frequently asked interview questions](https://www.javainuse.com/misc/muleintvw)
* [Apache Kafka Interview Questions](https://www.javainuse.com/misc/apache-kafka-interview-questions)
* [Tosca Testing Tool Interview Questions](https://www.javainuse.com/misc/tosca-testing-tool-interview-questions)
* [Top Maven Build Tool Interview Questions](https://www.javainuse.com/misc/maven-interview-questions)
* [Top Gradle Build Tool Interview Questions](https://www.javainuse.com/misc/gradle-interview-questions)
* [Miscellaneous Topics](https://www.javainuse.com/misc)

© Copyright JavaInUse. All Rights Reserved.  
[Privacy Policy](https://www.javainuse.com/privacy)

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

Spring Boot Security - Database Authentication Example

[**Close X**](https://www.javainuse.com/)

In a previous post we had implemented [Spring Boot Security - Creating a custom login page](https://www.javainuse.com/spring/boot_form_security_custom_login).  
Till now we were making use of in memory configuration for authenticating users and associated roles. In this example we will authenticate users and roles against database tables.

Spring Boot Security - Table Of Contents

[Spring Boot + Simple Security Configuration](https://www.javainuse.com/spring/sprboot_sec)

[Spring Boot Form Security Login Hello World Example](https://www.javainuse.com/spring/boot_form_security)

[Spring Boot Security - Custom Login Page Example](https://www.javainuse.com/spring/boot_form_security_custom_login)

[**Spring Boot Security - JDBC Authentication Example**](https://www.javainuse.com/spring/boot_security_jdbc_authentication)

[Spring Boot Security - Creating Users Programmatically Using JdbcUserDetailsManager](https://www.javainuse.com/spring/boot_security_jdbc_authentication_program)

[Spring Boot Security - Password Encoding Using Bcrypt](https://www.javainuse.com/spring/boot_security_jdbc_authentication_bcrypt)

[Spring Boot Security - Enabling CSRF Protection](https://www.javainuse.com/spring/boot_security_csrf)

[Spring Boot Security - Authentication Handler Example](https://www.javainuse.com/spring/boot_form_authentication_handler)

[Spring Boot Security - Introduction to OAuth](https://www.javainuse.com/spring/spring-boot-oauth-introduction)

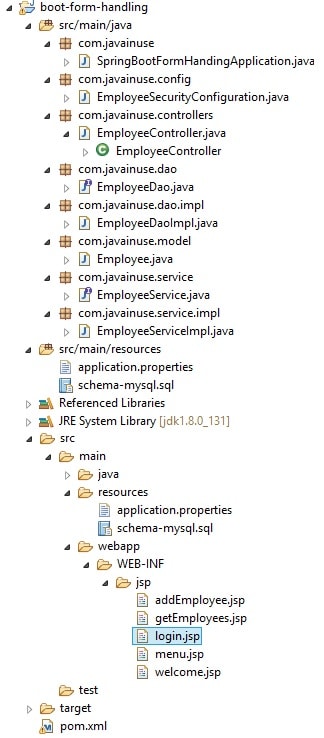
[Spring Boot OAuth2 Part 1 - Getting The Authorization Code](https://www.javainuse.com/spring/spring-boot-oauth-authorization-code)

[Spring Boot OAuth2 Part 2 - Getting The Access Token And Using it to Fetch Data.](https://www.javainuse.com/spring/spring-boot-oauth-access-token)

Video

This tutorial is explained in the below Youtube Video.

Lets Begin-

We will be modifying the code we developed in the previous [Spring Boot Security - Creating a custom login page](https://www.javainuse.com/spring/boot_form_security_custom_login)  
Maven Project will be as follows-  
  
  
By default spring security expects tables named **users table** for storing username, passwords and **authorities table** for storing the associated roles. In the schema-mysql.sql add these schemas and insert statements

DROP TABLE IF EXISTS employee;

DROP TABLE IF EXISTS users;

DROP TABLE IF EXISTS authorities;

CREATE TABLE employee (

empId VARCHAR(10) NOT NULL,

empName VARCHAR(100) NOT NULL

);

create table users (

username varchar(50) not null primary key,

password varchar(120) not null,

enabled boolean not null

);

create table authorities (

username varchar(50) not null,

authority varchar(50) not null,

foreign key (username) references users (username)

);

insert into users(username, password, enabled)values('javainuse','javainuse',true);

insert into authorities(username,authority)values('javainuse','ROLE\_ADMIN');

insert into users(username, password, enabled)values('employee','employee',true);

insert into authorities(username,authority)values('javainuse','ROLE\_USER');

Spring Boot JDBC runs this script before starting the application

Finally modify the Spring Security configuration to switch to jdbc authentication.

package com.javainuse.config;

import javax.sql.DataSource;

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

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.builders.WebSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration

@EnableWebSecurity

public class EmployeeSecurityConfiguration extends WebSecurityConfigurerAdapter {

@Autowired

DataSource dataSource;

**//Enable jdbc authentication**

**@Autowired**

**public void configAuthentication(AuthenticationManagerBuilder auth) throws Exception {**

**auth.jdbcAuthentication().dataSource(dataSource);**

**}**

@Override

public void configure(WebSecurity web) throws Exception {

web.ignoring().antMatchers("/resources/\*\*");

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.authorizeRequests().antMatchers("/").permitAll().antMatchers("/welcome").hasAnyRole("USER", "ADMIN")

.antMatchers("/getEmployees").hasAnyRole("USER", "ADMIN").antMatchers("/addNewEmployee")

.hasAnyRole("ADMIN").anyRequest().authenticated().and().formLogin().**loginPage("/login")**.permitAll()

.and().logout().permitAll();

http.csrf().disable();

}

**//remove this in memory authentication configuration**

// @Autowired

//public void configureGlobal(AuthenticationManagerBuilder authenticationMgr) throws Exception {

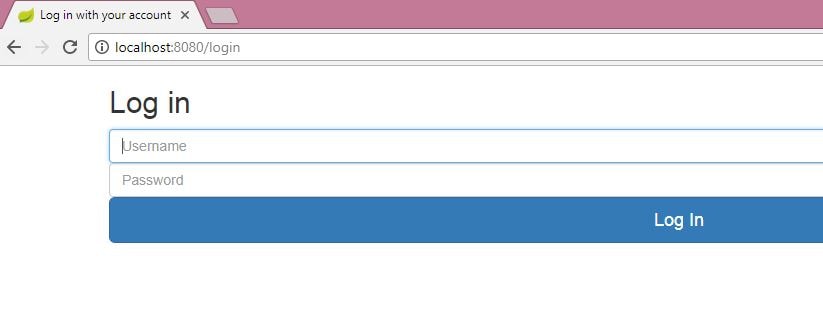
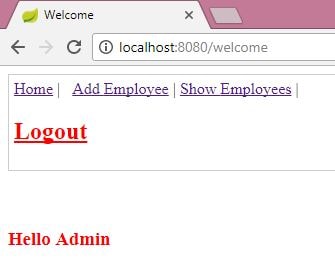
// authenticationMgr.inMemoryAuthentication().withUser("admin").password("admin").authorities("ROLE\_USER").and()

// .withUser("javainuse").password("javainuse").authorities("ROLE\_USER", "ROLE\_ADMIN");

//}

}

These are the only changes required.

* Go to localhost:8080/welcome, we will be redirected to the custom login page.  
  
* Enter the user javainuse and password javainuse  
    
  So our application is working good and getting correctly authenticated using database tables.

Download Source Code

Download it -  
[Spring Boot Security - Database Authentication](https://www.javainuse.com/zip/spring/sec/boot-security-jdbc-authentication.rar)

Popular Posts

* [E-commerce Website - Online Book Store using Angular 8 + Spring Boot](https://www.javainuse.com/fullstack/ecommerce)
* [Spring Boot +JSON Web Token(JWT) Hello World Example](https://www.javainuse.com/spring/boot-jwt)
* [Angular 7 + Spring Boot Application Hello World Example](https://www.javainuse.com/spring/ang7-hello)
* [Build a Real Time Chat Application using Spring Boot + WebSocket + RabbitMQ](https://www.javainuse.com/spring/boot-websocket-chat)
* [Pivotal Cloud Foundry Tutorial - Deploy Spring Boot Application Hello World Example](https://www.javainuse.com/pcf/pcf-hello)
* [Deploying Spring Based WAR Application to Docker](https://www.javainuse.com/devOps/docker/docker-war)
* [EIP patterns using Apache Camel](https://www.javainuse.com/camel/camel_EIP)
* [Spring Cloud- Netflix Eureka + Ribbon Simple Example](https://www.javainuse.com/spring/spring_ribbon)
* [Spring Cloud- Netflix Hystrix Circuit Breaker Simple Example](https://www.javainuse.com/spring/spring_hystrix_circuitbreaker)
* [Spring Boot + Swagger Example Hello World Example](https://www.javainuse.com/spring/boot_swagger)
* [Spring Boot Batch Simple example](https://www.javainuse.com/spring/bootbatch)
* [Spring Boot + Apache Kafka Example](https://www.javainuse.com/spring/spring-boot-apache-kafka-hello-world)
* [Spring Boot Admin Simple Example](https://www.javainuse.com/spring/boot-admin)
* [Spring Boot Security - Introduction to OAuth](https://www.javainuse.com/spring/spring-boot-oauth-introduction)
* [Spring Boot OAuth2 Part 1 - Getting The Authorization Code](https://www.javainuse.com/spring/spring-boot-oauth-authorization-code)
* [Spring Boot OAuth2 Part 2 - Getting The Access Token And Using it to Fetch Data.](https://www.javainuse.com/spring/spring-boot-oauth-access-token)
* [JBoss Drools Hello World-Stateful Knowledge Session using KieSession](https://www.javainuse.com/drools_hello_kie)
* [Understand Drools Stateful vs Stateless Knowledge Session](https://www.javainuse.com/drools_states)
* [JBoss Drools- Understanding Drools Decision Table using Simple Example](https://www.javainuse.com/drools/drools_decision)

See Also

* [Spring Boot Interview Questions](https://www.javainuse.com/spring/SpringBootInterviewQuestions)
* [Spring Batch Interview Questions](https://www.javainuse.com/spring/sprbatch_interview)
* [Spring AOP Interview Questions](https://www.javainuse.com/spring/spring-AOP-interview-quesions)
* [Angular 2 Interview Questions](https://www.javainuse.com/angular/ang2_intvw)
* [Apache Camel Interview Questions](https://www.javainuse.com/camel/Apache_Camel_Questions)
* [JBoss Fuse Interview Questions](https://www.javainuse.com/camel/JBoss_Fuse_Questions)
* [Drools Interview Questions](https://www.javainuse.com/drools/drools_intvw)
* [Java 8 Interview Questions](https://www.javainuse.com/java/java8_intvw)
* [Spring Cloud Interview Questions](https://www.javainuse.com/spring/spring-cloud-interview-questions)
* [Microservices Interview Questions](https://www.javainuse.com/spring/microservices-interview-quesions)
* [Java HashMap and ConcurrentHashMap Interview Questions](https://www.javainuse.com/java/java_map_intvw)
* [Mule ESB frequently asked interview questions](https://www.javainuse.com/misc/muleintvw)
* [Apache Kafka Interview Questions](https://www.javainuse.com/misc/apache-kafka-interview-questions)
* [Tosca Testing Tool Interview Questions](https://www.javainuse.com/misc/tosca-testing-tool-interview-questions)
* [Top Maven Build Tool Interview Questions](https://www.javainuse.com/misc/maven-interview-questions)
* [Top Gradle Build Tool Interview Questions](https://www.javainuse.com/misc/gradle-interview-questions)
* [Miscellaneous Topics](https://www.javainuse.com/misc)

© Copyright JavaInUse. All Rights Reserved.  
[Privacy Policy](https://www.javainuse.com/privacy)