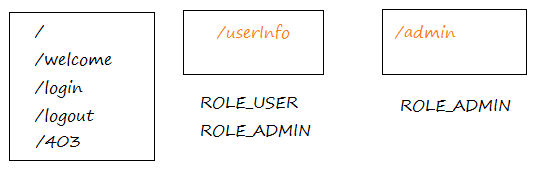
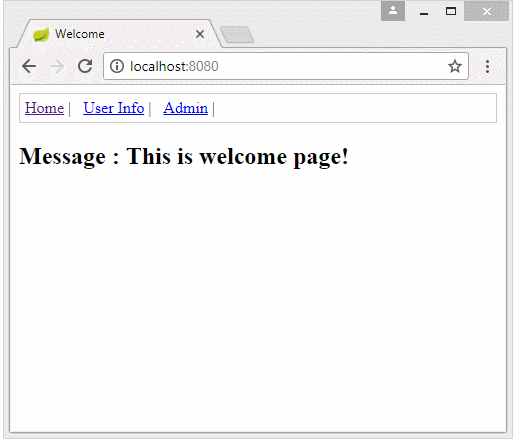
**Лабораторна робота № 28 «Створення форми для логування із використанням Spring Boot Security»**

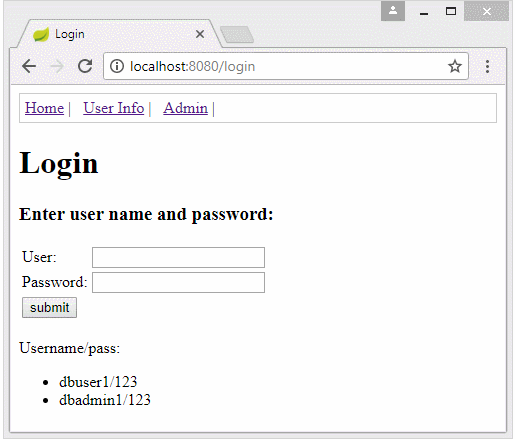
У цій роботі треба створити додаток Login, використовуючи **Spring Boot + Spring Security + JDBC + Thymeleaf**. Принцип роботи **Spring Security**.



Користувач зобов'язаний увійти в систему, щоб переглянути захищені сторінки:

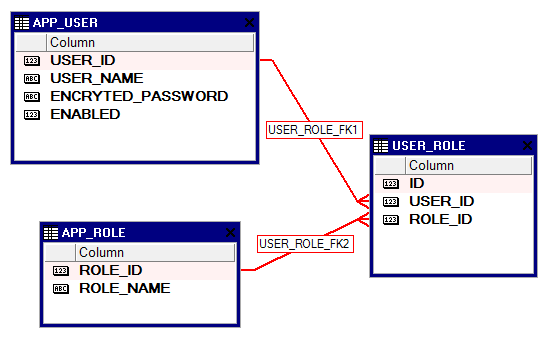


Користувачі, які увійшли в систему, можуть переглядати сторінки тільки в їх рамках. Якщо вони будуть переглядати сторінки поза їх роликами, доступ буде відмовлено.



## 

## 2- Приготувати базу даних



MySQL

-- Create table

**create** **table** APP\_USER

(

USER\_ID BIGINT **not** **null**,

USER\_NAME VARCHAR(36) **not** **null**,

ENCRYTED\_PASSWORD VARCHAR(128) **not** **null**,

ENABLED BIT **not** **null**

) ;

--

**alter** **table** APP\_USER

**add** **constraint** APP\_USER\_PK **primary** key (USER\_ID);

**alter** **table** APP\_USER

**add** **constraint** APP\_USER\_UK **unique** (USER\_NAME);

-- Create table

**create** **table** APP\_ROLE

(

ROLE\_ID BIGINT **not** **null**,

ROLE\_NAME VARCHAR(30) **not** **null**

) ;

--

**alter** **table** APP\_ROLE

**add** **constraint** APP\_ROLE\_PK **primary** key (ROLE\_ID);

**alter** **table** APP\_ROLE

**add** **constraint** APP\_ROLE\_UK **unique** (ROLE\_NAME);

-- Create table

**create** **table** USER\_ROLE

(

ID BIGINT **not** **null**,

USER\_ID BIGINT **not** **null**,

ROLE\_ID BIGINT **not** **null**

);

--

**alter** **table** USER\_ROLE

**add** **constraint** USER\_ROLE\_PK **primary** key (ID);

**alter** **table** USER\_ROLE

**add** **constraint** USER\_ROLE\_UK **unique** (USER\_ID, ROLE\_ID);

**alter** **table** USER\_ROLE

**add** **constraint** USER\_ROLE\_FK1 **foreign** key (USER\_ID)

**references** APP\_USER (USER\_ID);

**alter** **table** USER\_ROLE

**add** **constraint** USER\_ROLE\_FK2 **foreign** key (ROLE\_ID)

**references** APP\_ROLE (ROLE\_ID);

-- Used by Spring Remember Me API.

**CREATE** **TABLE** Persistent\_Logins (

username varchar(64) **not** **null**,

series varchar(64) **not** **null**,

token varchar(64) **not** **null**,

last\_used timestamp **not** **null**,

**PRIMARY** KEY (series)

);

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

**insert** **into** App\_User (USER\_ID, USER\_NAME, ENCRYTED\_PASSWORD, ENABLED)

**values** (2, 'dbuser1', '$2a$10$PrI5Gk9L.tSZiW9FXhTS8O8Mz9E97k2FZbFvGFFaSsiTUIl.TCrFu', 1);

**insert** **into** App\_User (USER\_ID, USER\_NAME, ENCRYTED\_PASSWORD, ENABLED)

**values** (1, 'dbadmin1', '$2a$10$PrI5Gk9L.tSZiW9FXhTS8O8Mz9E97k2FZbFvGFFaSsiTUIl.TCrFu', 1);

---

**insert** **into** app\_role (ROLE\_ID, ROLE\_NAME)

**values** (1, 'ROLE\_ADMIN');

**insert** **into** app\_role (ROLE\_ID, ROLE\_NAME)

**values** (2, 'ROLE\_USER');

---

**insert** **into** user\_role (ID, USER\_ID, ROLE\_ID)

**values** (1, 1, 1);

**insert** **into** user\_role (ID, USER\_ID, ROLE\_ID)

**values** (2, 1, 2);

**insert** **into** user\_role (ID, USER\_ID, ROLE\_ID)

**values** (3, 2, 2);

---

SQL Server

-- Create table

**create** **table** APP\_USER

(

USER\_ID BIGINT **not** **null**,

USER\_NAME VARCHAR(36) **not** **null**,

ENCRYTED\_PASSWORD VARCHAR(128) **not** **null**,

ENABLED BIT **not** **null**

) ;

--

**alter** **table** APP\_USER

**add** **constraint** APP\_USER\_PK **primary** key (USER\_ID);

**alter** **table** APP\_USER

**add** **constraint** APP\_USER\_UK **unique** (USER\_NAME);

-- Create table

**create** **table** APP\_ROLE

(

ROLE\_ID BIGINT **not** **null**,

ROLE\_NAME VARCHAR(30) **not** **null**

) ;

--

**alter** **table** APP\_ROLE

**add** **constraint** APP\_ROLE\_PK **primary** key (ROLE\_ID);

**alter** **table** APP\_ROLE

**add** **constraint** APP\_ROLE\_UK **unique** (ROLE\_NAME);

-- Create table

**create** **table** USER\_ROLE

(

ID BIGINT **not** **null**,

USER\_ID BIGINT **not** **null**,

ROLE\_ID BIGINT **not** **null**

);

--

**alter** **table** USER\_ROLE

**add** **constraint** USER\_ROLE\_PK **primary** key (ID);

**alter** **table** USER\_ROLE

**add** **constraint** USER\_ROLE\_UK **unique** (USER\_ID, ROLE\_ID);

**alter** **table** USER\_ROLE

**add** **constraint** USER\_ROLE\_FK1 **foreign** key (USER\_ID)

**references** APP\_USER (USER\_ID);

**alter** **table** USER\_ROLE

**add** **constraint** USER\_ROLE\_FK2 **foreign** key (ROLE\_ID)

**references** APP\_ROLE (ROLE\_ID);

-- Used by Spring Remember Me API.

**CREATE** **TABLE** Persistent\_Logins (

username varchar(64) **not** **null**,

series varchar(64) **not** **null**,

token varchar(64) **not** **null**,

last\_used Datetime **not** **null**,

**PRIMARY** KEY (series)

);

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

**insert** **into** App\_User (USER\_ID, USER\_NAME, ENCRYTED\_PASSWORD, ENABLED)

**values** (2, 'dbuser1', '$2a$10$PrI5Gk9L.tSZiW9FXhTS8O8Mz9E97k2FZbFvGFFaSsiTUIl.TCrFu', 1);

**insert** **into** App\_User (USER\_ID, USER\_NAME, ENCRYTED\_PASSWORD, ENABLED)

**values** (1, 'dbadmin1', '$2a$10$PrI5Gk9L.tSZiW9FXhTS8O8Mz9E97k2FZbFvGFFaSsiTUIl.TCrFu', 1);

---

**insert** **into** app\_role (ROLE\_ID, ROLE\_NAME)

**values** (1, 'ROLE\_ADMIN');

**insert** **into** app\_role (ROLE\_ID, ROLE\_NAME)

**values** (2, 'ROLE\_USER');

---

**insert** **into** user\_role (ID, USER\_ID, ROLE\_ID)

**values** (1, 1, 1);

**insert** **into** user\_role (ID, USER\_ID, ROLE\_ID)

**values** (2, 1, 2);

**insert** **into** user\_role (ID, USER\_ID, ROLE\_ID)

**values** (3, 2, 2);

ORACLE

-- Create table

**create** **table** APP\_USER

(

USER\_ID NUMBER(19) **not** **null**,

USER\_NAME VARCHAR2(36) **not** **null**,

ENCRYTED\_PASSWORD VARCHAR2(128) **not** **null**,

ENABLED NUMBER(1) **not** **null**

) ;

--

**alter** **table** APP\_USER

**add** **constraint** APP\_USER\_PK **primary** key (USER\_ID);

**alter** **table** APP\_USER

**add** **constraint** APP\_USER\_UK **unique** (USER\_NAME);

-- Create table

**create** **table** APP\_ROLE

(

ROLE\_ID NUMBER(19) **not** **null**,

ROLE\_NAME VARCHAR2(30) **not** **null**

) ;

--

**alter** **table** APP\_ROLE

**add** **constraint** APP\_ROLE\_PK **primary** key (ROLE\_ID);

**alter** **table** APP\_ROLE

**add** **constraint** APP\_ROLE\_UK **unique** (ROLE\_NAME);

-- Create table

**create** **table** USER\_ROLE

(

ID NUMBER(19) **not** **null**,

USER\_ID NUMBER(19) **not** **null**,

ROLE\_ID NUMBER(19) **not** **null**

);

--

**alter** **table** USER\_ROLE

**add** **constraint** USER\_ROLE\_PK **primary** key (ID);

**alter** **table** USER\_ROLE

**add** **constraint** USER\_ROLE\_UK **unique** (USER\_ID, ROLE\_ID);

**alter** **table** USER\_ROLE

**add** **constraint** USER\_ROLE\_FK1 **foreign** key (USER\_ID)

**references** APP\_USER (USER\_ID);

**alter** **table** USER\_ROLE

**add** **constraint** USER\_ROLE\_FK2 **foreign** key (ROLE\_ID)

**references** APP\_ROLE (ROLE\_ID);

-- Used by Spring Remember Me API.

**CREATE** **TABLE** Persistent\_Logins (

username varchar2(64) **not** **null**,

series varchar2(64) **not** **null**,

token varchar2(64) **not** **null**,

last\_used Date **not** **null**,

**PRIMARY** KEY (series)

);

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

**insert** **into** App\_User (USER\_ID, USER\_NAME, ENCRYTED\_PASSWORD, ENABLED)

**values** (2, 'dbuser1', '$2a$10$PrI5Gk9L.tSZiW9FXhTS8O8Mz9E97k2FZbFvGFFaSsiTUIl.TCrFu', 1);

**insert** **into** App\_User (USER\_ID, USER\_NAME, ENCRYTED\_PASSWORD, ENABLED)

**values** (1, 'dbadmin1', '$2a$10$PrI5Gk9L.tSZiW9FXhTS8O8Mz9E97k2FZbFvGFFaSsiTUIl.TCrFu', 1);

---

**insert** **into** app\_role (ROLE\_ID, ROLE\_NAME)

**values** (1, 'ROLE\_ADMIN');

**insert** **into** app\_role (ROLE\_ID, ROLE\_NAME)

**values** (2, 'ROLE\_USER');

---

**insert** **into** user\_role (ID, USER\_ID, ROLE\_ID)

**values** (1, 1, 1);

**insert** **into** user\_role (ID, USER\_ID, ROLE\_ID)

**values** (2, 1, 2);

**insert** **into** user\_role (ID, USER\_ID, ROLE\_ID)

**values** (3, 2, 2);

---

**Commit**;

Postgres

-- Create table

**create** **table** APP\_USER

(

USER\_ID BIGINT **not** **null**,

USER\_NAME VARCHAR(36) **not** **null**,

ENCRYTED\_PASSWORD VARCHAR(128) **not** **null**,

ENABLED Int **not** **null**

) ;

--

**alter** **table** APP\_USER

**add** **constraint** APP\_USER\_PK **primary** key (USER\_ID);

**alter** **table** APP\_USER

**add** **constraint** APP\_USER\_UK **unique** (USER\_NAME);

-- Create table

**create** **table** APP\_ROLE

(

ROLE\_ID BIGINT **not** **null**,

ROLE\_NAME VARCHAR(30) **not** **null**

) ;

--

**alter** **table** APP\_ROLE

**add** **constraint** APP\_ROLE\_PK **primary** key (ROLE\_ID);

**alter** **table** APP\_ROLE

**add** **constraint** APP\_ROLE\_UK **unique** (ROLE\_NAME);

-- Create table

**create** **table** USER\_ROLE

(

ID BIGINT **not** **null**,

USER\_ID BIGINT **not** **null**,

ROLE\_ID BIGINT **not** **null**

);

--

**alter** **table** USER\_ROLE

**add** **constraint** USER\_ROLE\_PK **primary** key (ID);

**alter** **table** USER\_ROLE

**add** **constraint** USER\_ROLE\_UK **unique** (USER\_ID, ROLE\_ID);

**alter** **table** USER\_ROLE

**add** **constraint** USER\_ROLE\_FK1 **foreign** key (USER\_ID)

**references** APP\_USER (USER\_ID);

**alter** **table** USER\_ROLE

**add** **constraint** USER\_ROLE\_FK2 **foreign** key (ROLE\_ID)

**references** APP\_ROLE (ROLE\_ID);

-- Used by Spring Remember Me API.

**CREATE** **TABLE** Persistent\_Logins (

username varchar(64) **not** **null**,

series varchar(64) **not** **null**,

token varchar(64) **not** **null**,

last\_used timestamp **not** **null**,

**PRIMARY** KEY (series)

);

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

**insert** **into** App\_User (USER\_ID, USER\_NAME, ENCRYTED\_PASSWORD, ENABLED)

**values** (2, 'dbuser1', '$2a$10$PrI5Gk9L.tSZiW9FXhTS8O8Mz9E97k2FZbFvGFFaSsiTUIl.TCrFu', 1);

**insert** **into** App\_User (USER\_ID, USER\_NAME, ENCRYTED\_PASSWORD, ENABLED)

**values** (1, 'dbadmin1', '$2a$10$PrI5Gk9L.tSZiW9FXhTS8O8Mz9E97k2FZbFvGFFaSsiTUIl.TCrFu', 1);

---

**insert** **into** app\_role (ROLE\_ID, ROLE\_NAME)

**values** (1, 'ROLE\_ADMIN');

**insert** **into** app\_role (ROLE\_ID, ROLE\_NAME)

**values** (2, 'ROLE\_USER');

---

**insert** **into** user\_role (ID, USER\_ID, ROLE\_ID)

**values** (1, 1, 1);

**insert** **into** user\_role (ID, USER\_ID, ROLE\_ID)

**values** (2, 1, 2);

**insert** **into** user\_role (ID, USER\_ID, ROLE\_ID)

**values** (3, 2, 2);

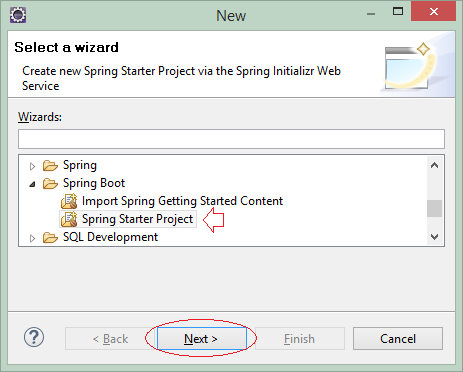
---

**Commit**;

## 

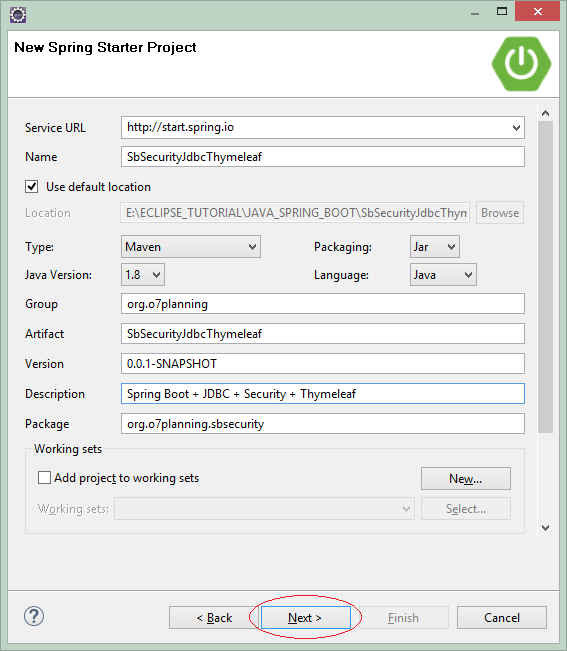
## 3- Створення проекту завантаження Spring

У **Eclipse** створити проект **Spring Boot**.

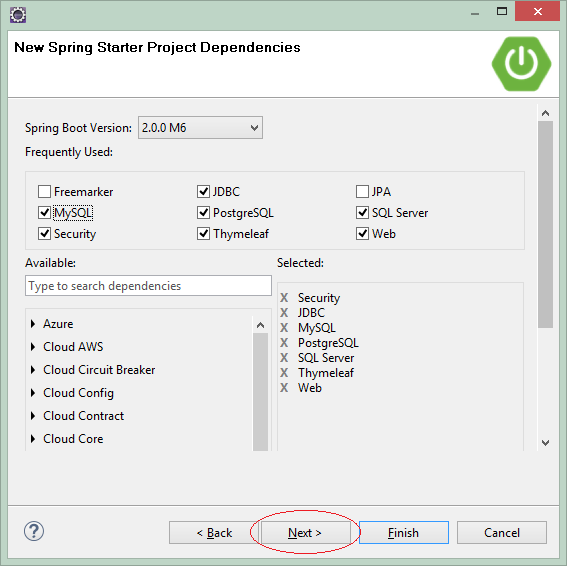


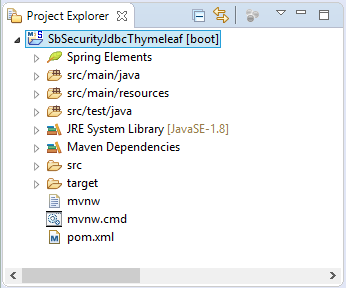
Ввести:

* **Назва** : SbSecurityJdbcThymeleaf
* **Група** : org.o7planning
* **Артефакт** : SbSecurityJdbcThymeleaf
* **Опис** : Spring Boot + JDBC + Security + Thymeleaf
* **Пакет** : org.o7planning.sbsecurity



Наступний крок, вам потрібно вибрати технології та бібліотеки, які будуть використані ( У цій статті ми підключаємося до бази даних **Oracle** , **MySQL** , **SQL Server** або **Postgres** ).





SbSecurityJdbcThymeleafApplication.java

**package** org.o7planning.sbsecurity;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** **SbSecurityJdbcThymeleafApplication** {

**public** **static** **void** **main**(String[] args) {

SpringApplication.run(SbSecurityJdbcThymeleafApplication.class, args);

}

}

## 4- Конфігурація pom.xml

Якщо ви використовуєте базу даних **Oracle** , вам потрібно опублікувати потрібну бібліотеку для **Oracle** у файлі **pom.xml** :

\*\* Oracle \*\*

<**dependencies**>

.....

<**dependency**>

<**groupId**>com.oracle</**groupId**>

<**artifactId**>ojdbc6</**artifactId**>

<**version**>11.2.0.3</**version**>

</**dependency**>

.....

</**dependencies**>

<**repositories**>

....

<!-- Repository for ORACLE JDBC Driver -->

<**repository**>

<**id**>codelds</**id**>

<**url**>https://code.lds.org/nexus/content/groups/main-repo</**url**>

</**repository**>

.....

</**repositories**>

Якщо ви підключаєте до бази даних **SQL Service** , ви можете використовувати одну з 2 бібліотек **JTDS** або **Mssql-Jdbc** :

\*\* SQL Server \*\*

<**dependencies**>

.....

<**dependency**>

<**groupId**>com.microsoft.sqlserver</**groupId**>

<**artifactId**>mssql-jdbc</**artifactId**>

<**scope**>runtime</**scope**>

</**dependency**>

<**dependency**>

<**groupId**>net.sourceforge.jtds</**groupId**>

<**artifactId**>jtds</**artifactId**>

<**scope**>runtime</**scope**>

</**dependency**>

.....

</**dependencies**>

Повний вміст файлу **pom.xml** :

pom.xml

<?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**>org.o7planning</**groupId**>

<**artifactId**>SbSecurityJdbcThymeleaf</**artifactId**>

<**version**>0.0.1-SNAPSHOT</**version**>

<**packaging**>jar</**packaging**>

<**name**>SbSecurityJdbcThymeleaf</**name**>

<**description**>Spring Boot + JDBC + Security + Thymeleaf</**description**>

<**parent**>

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

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

<**version**>2.0.0.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-jdbc</**artifactId**>

</**dependency**>

<**dependency**>

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

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

</**dependency**>

<**dependency**>

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

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

</**dependency**>

<**dependency**>

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

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

</**dependency**>

<**dependency**>

<**groupId**>mysql</**groupId**>

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

<**scope**>runtime</**scope**>

</**dependency**>

<**dependency**>

<**groupId**>org.postgresql</**groupId**>

<**artifactId**>postgresql</**artifactId**>

<**scope**>runtime</**scope**>

</**dependency**>

<**dependency**>

<**groupId**>com.microsoft.sqlserver</**groupId**>

<**artifactId**>mssql-jdbc</**artifactId**>

<**scope**>runtime</**scope**>

</**dependency**>

<**dependency**>

<**groupId**>net.sourceforge.jtds</**groupId**>

<**artifactId**>jtds</**artifactId**>

<**scope**>runtime</**scope**>

</**dependency**>

<**dependency**>

<**groupId**>com.oracle</**groupId**>

<**artifactId**>ojdbc6</**artifactId**>

<**version**>11.2.0.3</**version**>

</**dependency**>

<**dependency**>

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

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

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

</**dependency**>

<**dependency**>

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

<**artifactId**>spring-security-test</**artifactId**>

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

</**dependency**>

</**dependencies**>

<**build**>

<**plugins**>

<**plugin**>

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

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

</**plugin**>

</**plugins**>

</**build**>

<**repositories**>

<!-- Repository for ORACLE JDBC Driver -->

<**repository**>

<**id**>codelds</**id**>

<**url**>https://code.lds.org/nexus/content/groups/main-repo</**url**>

</**repository**>

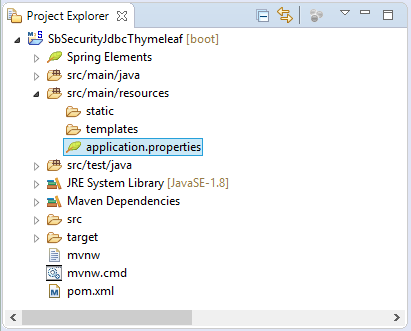
</**repositories**>

</**project**>

## 

## 5- Джерело даних конфігурації

Щоб **Spring** підключився до бази даних, потрібно налаштувати необхідні параметри файлу **application.properties** .



application.properties (ORACLE)

# Database Connection

spring.datasource.driver-class-name=oracle.jdbc.driver.OracleDriver

spring.datasource.url=jdbc:oracle:thin:@localhost:1521:db12c

spring.datasource.username=Test

spring.datasource.password=test001

# Other properties ..

application.properties (SQL Server + драйвер JTDS)

# Database Connection

spring.datasource.driver-class-name=net.sourceforge.jtds.jdbc.Driver

spring.datasource.url=jdbc:jtds:sqlserver://localhost:1433/simplehr;instance=SQLEXPRESS

spring.datasource.username=sa

spring.datasource.password=12345

# Other properties ..

application.properties (SQL Server + драйвер Mssql-jdbc)

# Database Connection

spring.datasource.driver-class-name=com.microsoft.sqlserver.jdbc.SQLServerDriver

spring.datasource.url=jdbc:sqlserver://tran-vmware-pc\\SQLEXPRESS:1433;databaseName=Test

spring.datasource.username=sa

spring.datasource.password=12345

# Other properties ..

application.properties (MySQL)

# Database Connection

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/mydatabase

spring.datasource.username=root

spring.datasource.password=12345

# Other properties ..

application.properties (Postgres)

# Database Connection

spring.datasource.driver-class-name=org.postgresql.Driver

spring.datasource.url=jdbc:postgresql://localhost:5432/mydatabase

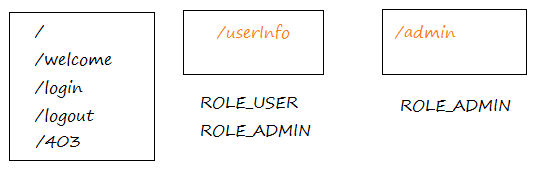
spring.datasource.username=postgres

spring.datasource.password=12345

# Other properties .

## 

## 6- Конфігурація захисту



Дана програма має деякі функції (сторінки), які:

* **/userInfo**

Ця сторінка переглядає інформацію користувача, дана сторінка вимагає входу користувача в систему та має роль **ROLE\_ADMIN** або **ROLE\_USER** .

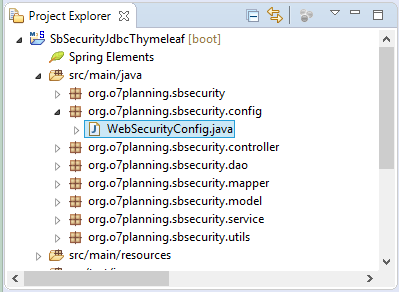
* **/ адмін**

Ця сторінка для адміністратора вимагає входу користувача в систему, і тільки люди з роллю **ROLE\_ADMIN** мають право доступу.

* **/. /вітаємо, /вхід, /вихід, /403**

Усі інші сторінки в додатку не вимагають від користувача входити в систему.

Клас **WebSecurityConfig** використовується для конфігурації захисту додатків. Він анотований (анотований) за допомогою **@Configuration** , Дана анотація говорить **Spring** , що він є класом конфігурації, тому він буде проаналізований за допомогою **Spring** під час запуску цього додатка.



WebSecurityConfig.java

**package** org.o7planning.sbsecurity.config;

**import** org.o7planning.sbsecurity.service.UserDetailsServiceImpl;

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

**import** org.springframework.context.annotation.Bean;

**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;

**import** org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@Configuration

@EnableWebSecurity

**public** **class** **WebSecurityConfig** **extends** **WebSecurityConfigurerAdapter** {

@Autowired

UserDetailsServiceImpl userDetailsService;

@Bean

**public** BCryptPasswordEncoder **passwordEncoder**() {

BCryptPasswordEncoder bCryptPasswordEncoder = **new** **BCryptPasswordEncoder**();

**return** bCryptPasswordEncoder;

}

@Autowired

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

// Setting Service to find User in the database.

// And Setting PassswordEncoder

auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());

}

@Override

**protected** **void** **configure**(HttpSecurity http) **throws** Exception {

http.csrf().disable();

// The pages does not require login

http.authorizeRequests().antMatchers("/", "/login", "/logout").permitAll();

// /userInfo page requires login as ROLE\_USER or ROLE\_ADMIN.

// If no login, it will redirect to /login page.

http.authorizeRequests().antMatchers("/userInfo").access("hasAnyRole('ROLE\_USER', 'ROLE\_ADMIN')");

// For ADMIN only.

http.authorizeRequests().antMatchers("/admin").access("hasRole('ROLE\_ADMIN')");

// When the user has logged in as XX.

// But access a page that requires role YY,

// AccessDeniedException will be thrown.

http.authorizeRequests().and().exceptionHandling().accessDeniedPage("/403");

// Config for Login Form

http.authorizeRequests().and().formLogin()//

// Submit URL of login page.

.loginProcessingUrl("/j\_spring\_security\_check") // Submit URL

.loginPage("/login")//

.defaultSuccessUrl("/userAccountInfo")//

.failureUrl("/login?error=true")//

.usernameParameter("username")//

.passwordParameter("password")

// Config for Logout Page

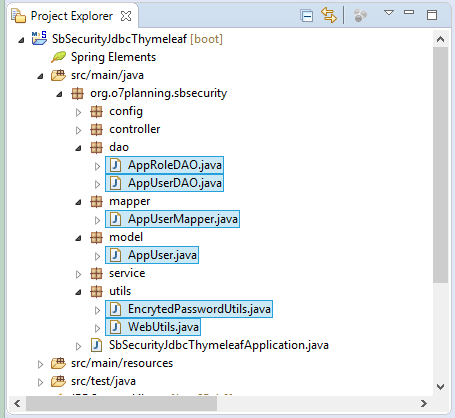
.and().logout().logoutUrl("/logout").logoutSuccessUrl("/logoutSuccessful");

}

}

## 

## 7- Модель, Mapper, DAO, WebUtils



Клас **AppUser** містить запис у таблиці бази даних **APP\_USER .**

AppUser.java

**package** org.o7planning.sbsecurity.model;

**public** **class** **AppUser** {

**private** Long userId;

**private** String userName;

**private** String encrytedPassword;

**public** **AppUser**() {

}

**public** **AppUser**(Long userId, String userName, String encrytedPassword) {

this.userId = userId;

this.userName = userName;

this.encrytedPassword = encrytedPassword;

}

**public** Long **getUserId**() {

**return** userId;

}

**public** **void** **setUserId**(Long userId) {

this.userId = userId;

}

**public** String **getUserName**() {

**return** userName;

}

**public** **void** **setUserName**(String userName) {

this.userName = userName;

}

**public** String **getEncrytedPassword**() {

**return** encrytedPassword;

}

**public** **void** **setEncrytedPassword**(String encrytedPassword) {

this.encrytedPassword = encrytedPassword;

}

@Override

**public** String **toString**() {

**return** this.userName + "/" + this.encrytedPassword;

}

}

Клас **AppUserMapper** використовується для компонування (відображення) столбців у таблиці **APP\_USER** з полями (полем) у класі **AppUser** (Основна в команді запиту).

AppUserMapper.java

**package** org.o7planning.sbsecurity.mapper;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** org.o7planning.sbsecurity.model.AppUser;

**import** org.springframework.jdbc.core.RowMapper;

**public** **class** **AppUserMapper** **implements** **RowMapper**<AppUser> {

**public** **static** **final** String BASE\_SQL //

= "Select u.User\_Id, u.User\_Name, u.Encryted\_Password From App\_User u ";

@Override

**public** AppUser **mapRow**(ResultSet rs, int rowNum) **throws** SQLException {

Long userId = rs.getLong("User\_Id");

String userName = rs.getString("User\_Name");

String encrytedPassword = rs.getString("Encryted\_Password");

**return** **new** **AppUser**(userId, userName, encrytedPassword);

}

}

Класи **DAO** (Data Access Object) є класами, які використовуються для доступу до бази даних, наприклад Query, Insert, Update, Delete. Класи **DAO** зазвичай анотуються за допомогою **@Repository** , щоб сказати **Spring** керувати ними як **Spring BEAN** .

Клас **AppUserDAO** використовується для маніпуляції з таблицею **APP\_USER** . Він має метод пошуку користувача в базі даних відповідного імені користувача.

AppUserDAO.java

**package** org.o7planning.sbsecurity.dao;

**import** javax.sql.DataSource;

**import** org.o7planning.sbsecurity.mapper.AppUserMapper;

**import** org.o7planning.sbsecurity.model.AppUser;

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

**import** org.springframework.dao.EmptyResultDataAccessException;

**import** org.springframework.jdbc.core.support.JdbcDaoSupport;

**import** org.springframework.stereotype.Repository;

**import** org.springframework.transaction.annotation.Transactional;

@Repository

@Transactional

**public** **class** **AppUserDAO** **extends** **JdbcDaoSupport** {

@Autowired

**public** **AppUserDAO**(DataSource dataSource) {

this.setDataSource(dataSource);

}

**public** AppUser **findUserAccount**(String userName) {

// Select .. from App\_User u Where u.User\_Name = ?

String sql = AppUserMapper.BASE\_SQL + " where u.User\_Name = ? ";

Object[] params = **new** **Object**[] { userName };

AppUserMapper mapper = **new** **AppUserMapper**();

**try** {

AppUser userInfo = this.getJdbcTemplate().queryForObject(sql, params, mapper);

**return** userInfo;

} **catch** (EmptyResultDataAccessException e) {

**return** null;

}

}

}

AppRoleDAO.java

**package** org.o7planning.sbsecurity.dao;

**import** java.util.List;

**import** javax.sql.DataSource;

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

**import** org.springframework.jdbc.core.support.JdbcDaoSupport;

**import** org.springframework.stereotype.Repository;

**import** org.springframework.transaction.annotation.Transactional;

@Repository

@Transactional

**public** **class** **AppRoleDAO** **extends** **JdbcDaoSupport** {

@Autowired

**public** **AppRoleDAO**(DataSource dataSource) {

this.setDataSource(dataSource);

}

**public** List<String> **getRoleNames**(Long userId) {

String sql = "Select r.Role\_Name " //

+ " from User\_Role ur, App\_Role r " //

+ " where ur.Role\_Id = r.Role\_Id and ur.User\_Id = ? ";

Object[] params = **new** **Object**[] { userId };

List<String> roles = this.getJdbcTemplate().queryForList(sql, params, String.class);

**return** roles;

}

}

-

WebUtils.java

**package** org.o7planning.sbsecurity.utils;

**import** java.util.Collection;

**import** org.springframework.security.core.GrantedAuthority;

**import** org.springframework.security.core.userdetails.User;

**public** **class** **WebUtils** {

**public** **static** String **toString**(User user) {

StringBuilder sb = **new** **StringBuilder**();

sb.append("UserName:").append(user.getUsername());

Collection<GrantedAuthority> authorities = user.getAuthorities();

**if** (authorities != null && !authorities.isEmpty()) {

sb.append(" (");

boolean first = true;

**for** (GrantedAuthority a : authorities) {

**if** (first) {

sb.append(a.getAuthority());

first = false;

} **else** {

sb.append(", ").append(a.getAuthority());

}

}

sb.append(")");

}

**return** sb.toString();

}

}

EncrytedPasswordUtils.java

**package** org.o7planning.sbsecurity.utils;

**import** org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

**public** **class** **EncrytedPasswordUtils** {

// Encryte Password with BCryptPasswordEncoder

**public** **static** String **encrytePassword**(String password) {

BCryptPasswordEncoder encoder = **new** **BCryptPasswordEncoder**();

**return** encoder.encode(password);

}

**public** **static** **void** **main**(String[] args) {

String password = "123";

String encrytedPassword = encrytePassword(password);

System.out.println("Encryted Password: " + encrytedPassword);

}

}

## 

## 8- UserDetailsService

***UserDetailsService*** – це центральний інтерфейс у  **Spring Security**. Це сервіс для пошуку "Акаунт користувача і ролі користувача" . Використовується  **Spring Security** кожен раз, коли користувач входить в систему. Тому вам потрібно написати клас застосування (реалізації) цього інтерфейсу.

Тут я створюю клас **UserDetailsServiceImpl** застосування (реалізує) інтерфейсу **UserDetailsService**. Клас **UserDetailsServiceImpl** анотований за допомогою **@Service** , щоб сказати **Spring** керувати їм як **Spring BEAN**.

UserDetailsServiceImpl.java

**package** org.o7planning.sbsecurity.service;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.o7planning.sbsecurity.dao.AppUserDAO;

**import** org.o7planning.sbsecurity.dao.AppRoleDAO;

**import** org.o7planning.sbsecurity.model.AppUser;

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

**import** org.springframework.security.core.GrantedAuthority;

**import** org.springframework.security.core.authority.SimpleGrantedAuthority;

**import** org.springframework.security.core.userdetails.User;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** org.springframework.security.core.userdetails.UserDetailsService;

**import** org.springframework.security.core.userdetails.UsernameNotFoundException;

**import** org.springframework.stereotype.Service;

@Service

**public** **class** **UserDetailsServiceImpl** **implements** **UserDetailsService** {

@Autowired

**private** AppUserDAO appUserDAO;

@Autowired

**private** AppRoleDAO appRoleDAO;

@Override

**public** UserDetails **loadUserByUsername**(String userName) **throws** UsernameNotFoundException {

AppUser appUser = this.appUserDAO.findUserAccount(userName);

**if** (appUser == null) {

System.out.println("User not found! " + userName);

**throw** **new** **UsernameNotFoundException**("User " + userName + " was not found in the database");

}

System.out.println("Found User: " + appUser);

// [ROLE\_USER, ROLE\_ADMIN,..]

List<String> roleNames = this.appRoleDAO.getRoleNames(appUser.getUserId());

List<GrantedAuthority> grantList = **new** **ArrayList**<GrantedAuthority>();

**if** (roleNames != null) {

**for** (String role : roleNames) {

// ROLE\_USER, ROLE\_ADMIN,..

GrantedAuthority authority = **new** **SimpleGrantedAuthority**(role);

grantList.add(authority);

}

}

UserDetails userDetails = (UserDetails) **new** **User**(appUser.getUserName(), //

appUser.getEncrytedPassword(), grantList);

**return** userDetails;

}

}

## 9- Контролери

MainController.java

**package** org.o7planning.sbsecurity.controller;

**import** java.security.Principal;

**import** org.o7planning.sbsecurity.utils.WebUtils;

**import** org.springframework.security.core.Authentication;

**import** org.springframework.security.core.userdetails.User;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

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

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

@Controller

**public** **class** **MainController** {

@RequestMapping(value = { "/", "/welcome" }, method = RequestMethod.GET)

**public** String **welcomePage**(Model model) {

model.addAttribute("title", "Welcome");

model.addAttribute("message", "This is welcome page!");

**return** "welcomePage";

}

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

**public** String **adminPage**(Model model, Principal principal) {

User loginedUser = (User) ((Authentication) principal).getPrincipal();

String userInfo = WebUtils.toString(loginedUser);

model.addAttribute("userInfo", userInfo);

**return** "adminPage";

}

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

**public** String **loginPage**(Model model) {

**return** "loginPage";

}

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

**public** String **logoutSuccessfulPage**(Model model) {

model.addAttribute("title", "Logout");

**return** "logoutSuccessfulPage";

}

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

**public** String **userInfo**(Model model, Principal principal) {

// (1) (en)

// After user login successfully.

// (vi)

// Sau khi user login thanh cong se co principal

String userName = principal.getName();

System.out.println("User Name: " + userName);

User loginedUser = (User) ((Authentication) principal).getPrincipal();

String userInfo = WebUtils.toString(loginedUser);

model.addAttribute("userInfo", userInfo);

**return** "userInfoPage";

}

@RequestMapping(value = "/403", method = RequestMethod.GET)

**public** String **accessDenied**(Model model, Principal principal) {

**if** (principal != null) {

User loginedUser = (User) ((Authentication) principal).getPrincipal();

String userInfo = WebUtils.toString(loginedUser);

model.addAttribute("userInfo", userInfo);

String message = "Hi " + principal.getName() //

+ "<br> You do not have permission to access this page!";

model.addAttribute("message", message);

}

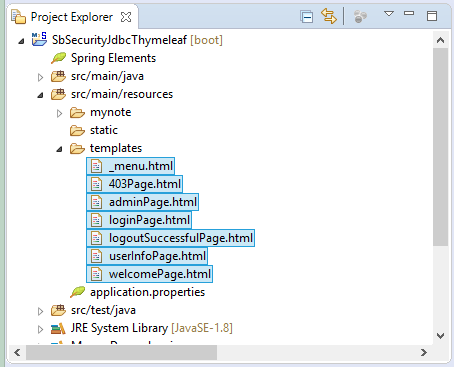
**return** "403Page";

}

}

## 

## 10- Шаблон Thymeleaf



**\_menu.html** використовується як частина веб-сторінки, він завантажений в інші сторінки, щоб створити сторінку **меню** .

\_menu.html

<**div** xmlns:th="http://www.thymeleaf.org"

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

<**a** th:href="@{/}">Home</**a**>

| &nbsp;

<**a** th:href="@{/userInfo}">User Info</**a**>

| &nbsp;

<**a** th:href="@{/admin}">Admin</**a**>

| &nbsp;

<**a** th:if="${#request.userPrincipal != null}" th:href="@{/logout}">Logout</**a**>

</**div**>

WelcomePage.html

<!DOCTYPE **HTML**>

<**html** xmlns:th="http://www.thymeleaf.org">

<**head**>

<**title** th:utext="${title}"></**title**>

</**head**>

<**body**>

<!-- Include \_menu.html -->

<**th:block** th:include="/\_menu"></**th:block**>

<**h2**>Message : <**span** th:utext="${message}"></**span**></**h2**>

</**body**>

</**html**>

loginPage.html

<!DOCTYPE **HTML**>

<**html** xmlns:th="http://www.thymeleaf.org">

<**head**>

<**title**>Login</**title**>

</**head**>

<**body**>

<!-- Include \_menu.html -->

<**th:block** th:include="/\_menu"></**th:block**>

<**h1**>Login</**h1**>

<!-- /login?error=true -->

<**div** th:if="${#request.getParameter('error') == 'true'}"

style="color:red;margin:10px 0px;">

Login Failed!!!<**br** />

Reason :

<**span** th:if="${#session!= null and #session.getAttribute('SPRING\_SECURITY\_LAST\_EXCEPTION') != null}"

th:utext="${#session.getAttribute('SPRING\_SECURITY\_LAST\_EXCEPTION').message}">

Static summary

</**span**>

</**div**>

<**h3**>Enter user name and password:</**h3**>

<**form** name='f' th:action="@{/j\_spring\_security\_check}" method='POST'>

<**table**>

<**tr**>

<**td**>User:</**td**>

<**td**><**input** type='text' name='username' value=''></**td**>

</**tr**>

<**tr**>

<**td**>Password:</**td**>

<**td**><**input** type='password' name='password' /></**td**>

</**tr**>

<**tr**>

<**td**><**input** name="submit" type="submit" value="submit" /></**td**>

</**tr**>

</**table**>

</**form**>

<**br**>

Username/pass:

<**ul**>

<**li**>dbuser1/123</**li**>

<**li**>dbadmin1/123</**li**>

</**ul**>

</**body**>

</**html**>

logoutSuccessfulPage.html

<!DOCTYPE **HTML**>

<**html** xmlns:th="http://www.thymeleaf.org">

<**head**>

<**title**>Logout</**title**>

</**head**>

<**body**>

<!-- Include \_menu.html -->

<**th:block** th:include="/\_menu"></**th:block**>

<**h1**>Logout Successful!</**h1**>

</**body**>

</**html**>

userInfoPage.html

<!DOCTYPE **HTML**>

<**html** xmlns:th="http://www.thymeleaf.org">

<**head**>

<**title**>User Info</**title**>

</**head**>

<**body**>

<!-- Include \_menu.html -->

<**th:block** th:include="/\_menu"></**th:block**>

<**h2**>User Info Page</**h2**>

<**h3**>Welcome : <**span** th:utext="${#request.userPrincipal.name}"></**span**></**h3**>

<**b**>This is protected page!</**b**>

<**br**/><**br**/>

<**div** th:if="${userInfo != null}" th:utext="${userInfo}"></**div**>

</**body**>

</**html**>

adminPage.html

<!DOCTYPE **HTML**>

<**html** xmlns:th="http://www.thymeleaf.org">

<**head**>

<**title** th:utext="${title}"></**title**>

</**head**>

<**body**>

<!-- Include \_menu.html -->

<**th:block** th:include="/\_menu"></**th:block**>

<**h2**>Admin Page</**h2**>

<**h3**>Welcome :

<**span** th:utext="${#request.userPrincipal.name}"></**span**>

</**h3**>

<**b**>This is protected page!</**b**>

<**br**/><**br**/>

<**div** th:if="${userInfo != null}" th:utext="${userInfo}"></**div**>

</**body**>

</**html**>

Якщо користувач увійшов до системи, але сторінка не авторизована (Не входить до її ролі), система показує вміст сторінки **/403** для оповіщення забороненого доступу до сторінки (Доступ заборонено).

403Page.html

<!DOCTYPE **HTML**>

<**html** xmlns:th="http://www.thymeleaf.org">

<**head**>

<**title**>Access Denied</**title**>

</**head**>

<**body**>

<!-- Include \_menu.html -->

<**th:block** th:include="/\_menu"></**th:block**>

<**h3** th:if="${message != null}" th:utext="${message}" style="color: red;"></**h3**>

<**div** th:if="${userInfo != null}" th:utext="${userInfo}"></**div**>

</**body**>

</**html**>

## 11- Запуск додатку

* Натиснути **кнопку As/Spring Boot**

