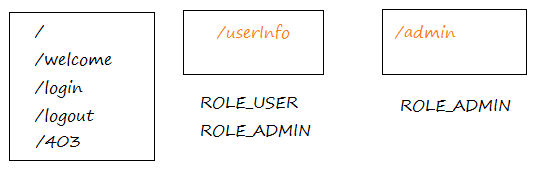
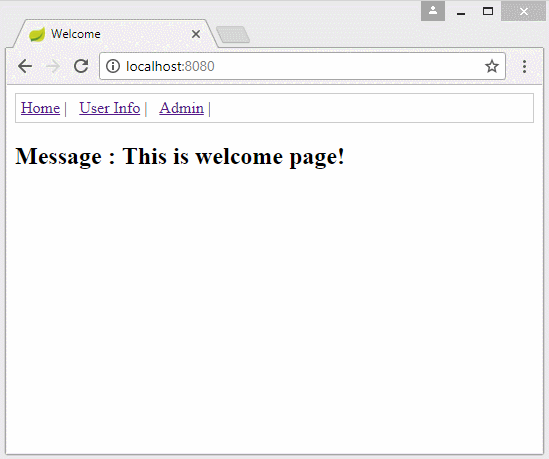
**Лабораторна робота № 29 «Створення додатку з Spring Boot Security та JPA»**

У цій роботі треба створити програму Login, використовуючи  **Spring Boot + Spring Security + JPA + Thymeleaf**.



Користувач зобов'язаний увійти до системи, щоб переглянути захищені сторінки:

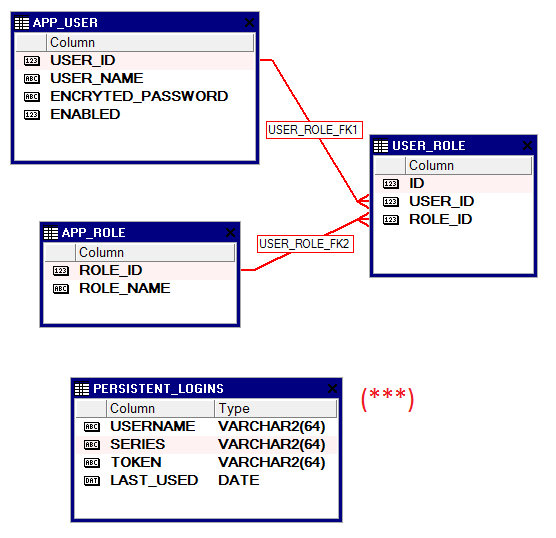


Користувачі, які увійшли до системи, можуть переглядати сторінки тільки в рамках їхньої ролі. Якщо вони переглядатимуть сторінки поза їхньою роллю, доступ буде відмовлено.

## 

## 2- Приготувати базу даних

У базі даних ми маємо 3 таблиці  **APP\_USER, APP\_ROLE, USER\_ROLE** це таблиці, які повинні вас зацікавити. Крім цього, інша таблиця це  **PERSISTENT\_LOGINS** , дана таблиця використовується  **Spring Remember Me API** для зберігання інформації **Token** і часу останнього входу в систему кожного користувача.



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)

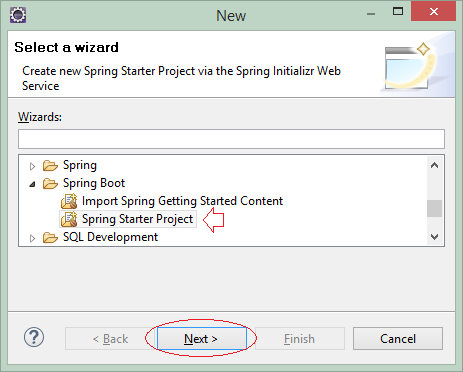
**values** (3, 2, 2);

---

**Commit**;

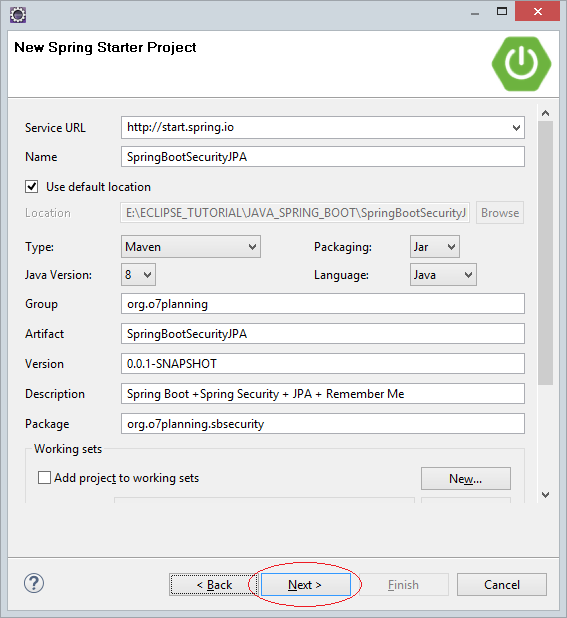
## 3- Створити Spring Boot Project

В  **Eclipse** створити проект  **Spring Boot**.



Ввести:

* **Name** : SpringBootSecurityJPA
* **Group** : org.o7planning
* **Artifact** : SpringBootSecurityJPA
* **Description** : Spring Boot + Spring Security + JPA + Remember Me.
* **Package** : org.o7planning.sbsecurity



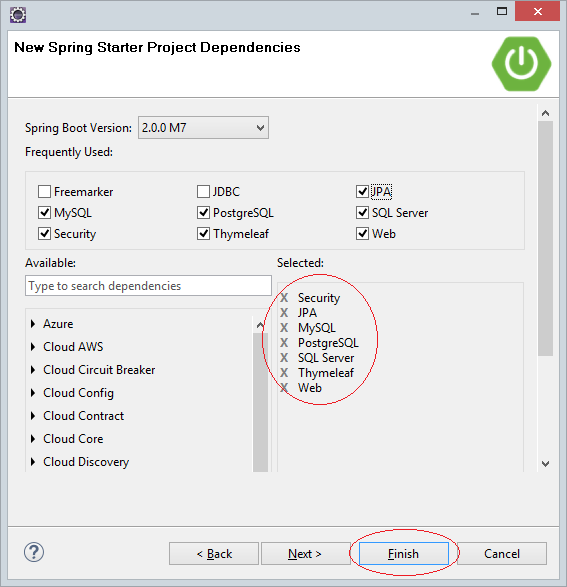
Наступний крок, вам потрібно вибрати технології та бібліотеки які будуть використані (У цій статті ми підключимося до бази даних  **Oracle** , **MySQL** , **SQL Server** або  **Postgres** ).

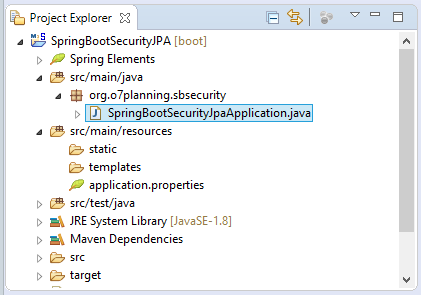
Database Libraries:

* MySQL
* PostgresSQL
* SQL Server

Tech:

* Web
* Thymeleaf
* Security





SpringBootSecurityJpaApplication.java

**package** org.o7planning.sbsecurity;

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

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

@SpringBootApplication

**public** **class** **SpringBootSecurityJpaApplication** {

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

SpringApplication.run(SpringBootSecurityJpaApplication.class, args);

}

}

## 4- Конфігурація pom.xml

Якщо ви використовуєте базу даних  **Oracle** , вам потрібно оголосити потрібну бібліотеку  **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**>SpringBootSecurityJPA</**artifactId**>

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

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

<**name**>SpringBootSecurityJPA</**name**>

<**description**>Spring Boot +Spring Security + JPA + Remember Me</**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-data-jpa</**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.oracle</**groupId**>

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

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

</**dependency**>

<!-- SQL Server Mssql-Jdbc Driver -->

<**dependency**>

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

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

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

</**dependency**>

<!-- SQL Server JTDS Driver -->

<**dependency**>

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

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

<**version**>1.3.1</**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**>

<**repositories**>

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

<**repository**>

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

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

</**repository**>

</**repositories**>

<**build**>

<**plugins**>

<**plugin**>

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

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

</**plugin**>

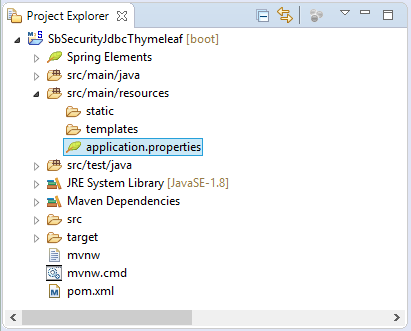
</**plugins**>

</**build**>

</**project**>

## 5- Конфігурація Datasource

Щоб  **Spring** підключився до бази даних, вам потрібно конфігурувати необхідні параметри у файлі  **application.properties** .



application.properties (MySQL)

# ===============================

# DATABASE

# ===============================

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

spring.datasource.url=jdbc:mysql://tran-vmware-pc:3306/Test

spring.datasource.username=root

spring.datasource.password=12345

# ===============================

# JPA / HIBERNATE

# ===============================

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=none

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

application.properties (ORACLE)

# ===============================

# DATABASE

# ===============================

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

spring.datasource.url=jdbc:oracle:thin:@tran-vmware-pc:1521:db12c

spring.datasource.username=Test

spring.datasource.password=12345

# ===============================

# JPA / HIBERNATE

# ===============================

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=none

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.Oracle10gDialect

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

# ===============================

# DATABASE

# ===============================

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

# ===============================

# JPA / HIBERNATE

# ===============================

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=none

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.SQLServer2012Dialect

application.properties (SQL Server + JTDS driver)

# ===============================

# DATABASE

# ===============================

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

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

spring.datasource.username=sa

spring.datasource.password=12345

# ===============================

# JPA / HIBERNATE

# ===============================

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=none

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.SQLServerDialect

application.properties (Postgres)

# ===============================

# DATABASE CONNECTION

# ===============================

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

spring.datasource.url=jdbc:postgresql://tran-vmware-pc:5432/Test

spring.datasource.username=postgres

spring.datasource.password=12345

# ===============================

# JPA / HIBERNATE

# ===============================

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=none

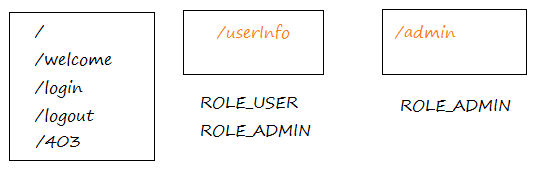
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

# Fix Postgres JPA Error:

# Method org.postgresql.jdbc.PgConnection.createClob() is not yet implemented.

spring.jpa.properties.hibernate.temp.use\_jdbc\_metadata\_defaults=false

## 6- Конфігурація захисту & Remember Me



Ця програма має деякі функції (сторінки), які:

* **/userInfo**

Це сторінка перегляду інформації користувача, ця сторінка вимагає користувача увійти до системи, і мати роль  **ROLE\_ADMIN** або  **ROLE\_USER** .

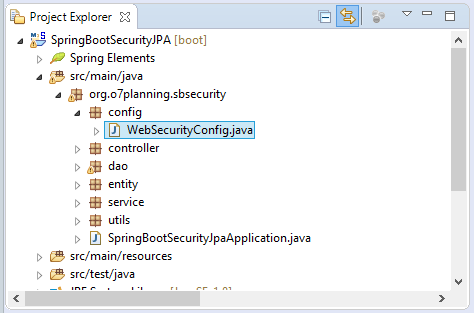
* **/admin**

Це сторінка для адміністратора, що вимагає користувача увійти в систему, і тільки ті, що з ролем   **ROLE\_ADMIN** мають право на доступ.

* **/. /welcome, /login, /logout, /403**

Всі інші сторінки програми не вимагають користувача входити до системи.

Клас  **WebSecurityConfig**  використовується для конфігурації захисту програми. Він анотований (annotate) за допомогою  **@Configuration** , Дана інструкція говорить  **Spring** що він є класом конфігурації, тому він буде аналізований за допомогою  **Spring** під час запуску цієї програми.



WebSecurityConfig.java

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

**import** javax.sql.DataSource;

**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;

**import** org.springframework.security.web.authentication.rememberme.JdbcTokenRepositoryImpl;

**import** org.springframework.security.web.authentication.rememberme.PersistentTokenRepository;

@Configuration

@EnableWebSecurity

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

@Autowired

**private** UserDetailsServiceImpl userDetailsService;

@Autowired

**private** DataSource dataSource;

@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");

// Config Remember Me.

http.authorizeRequests().and() //

.rememberMe().tokenRepository(this.persistentTokenRepository()) //

.tokenValiditySeconds(1 \* 24 \* 60 \* 60); // 24h

}

@Bean

**public** PersistentTokenRepository **persistentTokenRepository**() {

JdbcTokenRepositoryImpl db = **new** **JdbcTokenRepositoryImpl**();

db.setDataSource(dataSource);

**return** db;

}

}

Що таке опція "Remember Me"?

Користувач заходить на веб-сторінку і входить до системи. Після, користувач вимикає браузер і заходить в веб-сайт у певний момент (наприклад, на інший день), і він повинен ще раз увійти в систему, це створює непотрібну проблему. Опція  **"Remember Me"** дозволяє веб-сайту  **"запам'ятати"**  інформацію користувача для автоматичного входу в систему, коли користувач заходить до веб-сайту наступного разу.

Коли користувач входить до системи з опцією  **"Remember Me"** , **Spring**  збереже іноформацію часу останнього входу і  **token** . **Token** є рядком (string) який був кодований (encryted), він містить потрібну інформацію для того, щоб  **Spring** автоматично входив до системи, коли користувач заходив на вебсайт наступного разу.  
  
Існують 2 поширені способи для  **Spring,** щоб зберегти цю інформацію:

1. **Memory**
2. **Database**

\*\*WebSecurityConfig\*\*

// Token stored in Table (Persistent\_Logins)

@Bean

**public** PersistentTokenRepository **persistentTokenRepository**() {

JdbcTokenRepositoryImpl db = **new** **JdbcTokenRepositoryImpl**();

db.setDataSource(this.dataSource);

**return** db;

}

// Token stored in Memory (Of Web Server).

@Bean

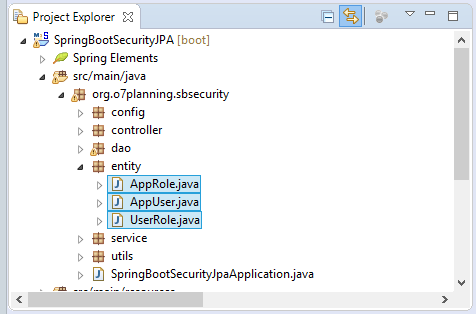
**public** PersistentTokenRepository **persistentTokenRepository**() {

InMemoryTokenRepositoryImpl memory = **new** **InMemoryTokenRepositoryImpl**();

**return** memory;

}

## 7- Класи Entity



AppRole.java

**package** org.o7planning.sbsecurity.entity;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

**import** javax.persistence.UniqueConstraint;

@Entity

@Table(name = "App\_Role", //

uniqueConstraints = { //

@UniqueConstraint(name = "APP\_ROLE\_UK", columnNames = "Role\_Name") })

**public** **class** **AppRole** {

@Id

@GeneratedValue

@Column(name = "Role\_Id", nullable = false)

**private** Long roleId;

@Column(name = "Role\_Name", length = 30, nullable = false)

**private** String roleName;

**public** Long **getRoleId**() {

**return** roleId;

}

**public** **void** **setRoleId**(Long roleId) {

this.roleId = roleId;

}

**public** String **getRoleName**() {

**return** roleName;

}

**public** **void** **setRoleName**(String roleName) {

this.roleName = roleName;

}

}

AppUser.java

**package** org.o7planning.sbsecurity.entity;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

**import** javax.persistence.UniqueConstraint;

@Entity

@Table(name = "App\_User", //

uniqueConstraints = { //

@UniqueConstraint(name = "APP\_USER\_UK", columnNames = "User\_Name") })

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

@Id

@GeneratedValue

@Column(name = "User\_Id", nullable = false)

**private** Long userId;

@Column(name = "User\_Name", length = 36, nullable = false)

**private** String userName;

@Column(name = "Encryted\_Password", length = 128, nullable = false)

**private** String encrytedPassword;

@Column(name = "Enabled", length = 1, nullable = false)

**private** boolean enabled;

**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;

}

**public** boolean **isEnabled**() {

**return** enabled;

}

**public** **void** **setEnabled**(boolean enabled) {

this.enabled = enabled;

}

}

UserRole.java

**package** org.o7planning.sbsecurity.entity;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.FetchType;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.ManyToOne;

**import** javax.persistence.Table;

**import** javax.persistence.UniqueConstraint;

@Entity

@Table(name = "User\_Role", //

uniqueConstraints = { //

@UniqueConstraint(name = "USER\_ROLE\_UK", columnNames = { "User\_Id", "Role\_Id" }) })

**public** **class** **UserRole** {

@Id

@GeneratedValue

@Column(name = "Id", nullable = false)

**private** Long id;

@ManyToOne(fetch = FetchType.LAZY)

@JoinColumn(name = "User\_Id", nullable = false)

**private** AppUser appUser;

@ManyToOne(fetch = FetchType.LAZY)

@JoinColumn(name = "Role\_Id", nullable = false)

**private** AppRole appRole;

**public** Long **getId**() {

**return** id;

}

**public** **void** **setId**(Long id) {

this.id = id;

}

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

**return** appUser;

}

**public** **void** **setAppUser**(AppUser appUser) {

this.appUser = appUser;

}

**public** AppRole **getAppRole**() {

**return** appRole;

}

**public** **void** **setAppRole**(AppRole appRole) {

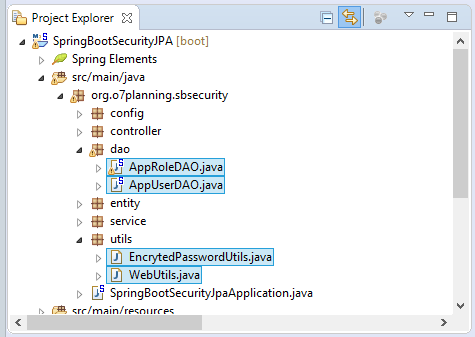
this.appRole = appRole;

}

}

## 

## 8- DAO, WebUtils



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

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

AppUserDAO.java

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

**import** javax.persistence.EntityManager;

**import** javax.persistence.NoResultException;

**import** javax.persistence.Query;

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

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

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

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

@Repository

@Transactional

**public** **class** **AppUserDAO** {

@Autowired

**private** EntityManager entityManager;

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

**try** {

String sql = "Select e from " + AppUser.class.getName() + " e " //

+ " Where e.userName = :userName ";

Query query = entityManager.createQuery(sql, AppUser.class);

query.setParameter("userName", userName);

**return** (AppUser) query.getSingleResult();

} **catch** (NoResultException e) {

**return** null;

}

}

}

AppRoleDAO.java

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

**import** java.util.List;

**import** javax.persistence.EntityManager;

**import** javax.persistence.Query;

**import** org.o7planning.sbsecurity.entity.UserRole;

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

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

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

@Repository

@Transactional

**public** **class** **AppRoleDAO** {

@Autowired

**private** EntityManager entityManager;

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

String sql = "Select ur.appRole.roleName from " + UserRole.class.getName() + " ur " //

+ " where ur.appUser.userId = :userId ";

Query query = this.entityManager.createQuery(sql, String.class);

query.setParameter("userId", userId);

**return** query.getResultList();

}

}

-

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);

}

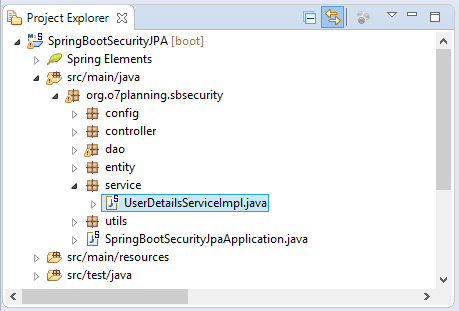
}

## 

## 9- UserDetailsService

***UserDetailsService*** це центральний інтерфейс у **Spring Security** . Це сервіс для пошуку  "Акаунт користувача та ролі того користувача" . Використовується **Spring Security**  щоразу, коли користувач входить у систему. Тому потрібно написати клас застосування (implements) даного інтерфейсу.

Тут я створюю клас  **UserDetailsServiceImpl** застосування (implements) інтерфейсу  **UserDetailsService** . Клас  **UserDetailsServiceImpl** анотований за допомогою  **@Service** щоб сказати  **Spring** керувати ним 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.entity.AppUser;

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

**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;

}

}

## 10- Controllers

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) {

// After user login successfully.

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);

}

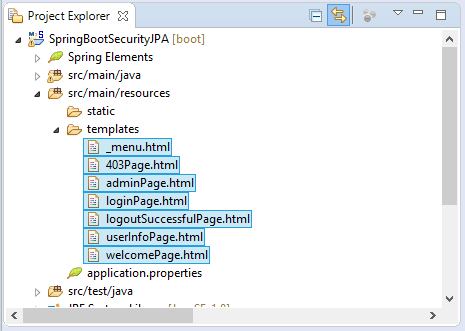
**return** "403Page";

}

}

## 

## 11- Thymeleaf Template



**\_menu.html**  використовується як частина веб-сторінки, він занурений в інші сторінки, щоб створити **Menu** сторінки.

\_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**>Remember Me?</**td**>

<**td**><**input** type="checkbox" name="remember-me" /></**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 доступ до сторінки не авторизований (Не входить у їхню роль), система відобразить зміст сторінки  **/403**  для оповіщення заборони доступу до сторінки (Access Denied).

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**>

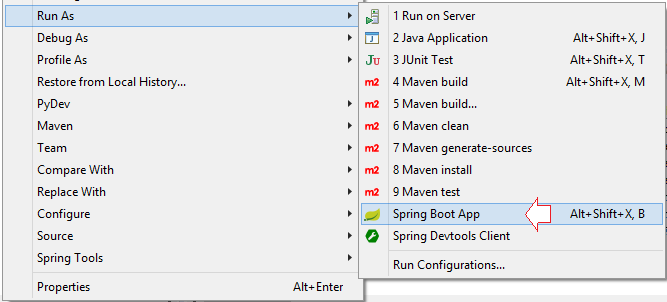
</**html**>

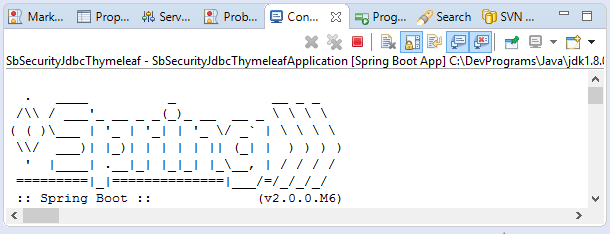
## 

## 12- Запуск програми

Натиснути на праву кнопку миші на Project, вибрати:

* **Run As/Spring Boot App**





Увійти в систему з участю  **ROLE\_USER** :

