import javax.sql.DataSource;

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

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.jdbc.datasource.DriverManagerDataSource;

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.config.annotation.web.configurers.ExpressionUrlAuthorizationConfigurer.ExpressionInterceptUrlRegistry;

import org.springframework.security.crypto.password.NoOpPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration //This is also an example of auto-configuration.

//This reads the contents of properties and application does self-configuration.

@EnableWebSecurity //This will generate the login page which is connected to the database

//through jdbc/hibernate/spring-data-jpa code

public class Authenticate extends WebSecurityConfigurerAdapter {

@Value("${spring.datasource.driverClassName}")

private String databaseDriverClassName;

@Value("${spring.datasource.url}")

private String datasourceUrl;

@Value("${spring.datasource.username}")

private String databaseUserName;

@Value("${spring.datasource.passwd}")

private String databasePassword;

@Autowired

DataSource dbds;

@Autowired

public void configAuthentication(AuthenticationManagerBuilder ambd) throws Exception

{

//Authentication is done here.

//Here it will generate the code for doing and performs authentication.

ambd.jdbcAuthentication().dataSource(dbds);

}

/\*Alternatives for NoOpPasswordEncoder

\* BCryptPasswordEncoder

\* SCryptPasswordEncoder

\*/

@Bean

public PasswordEncoder passwordEncoder() {

return NoOpPasswordEncoder.getInstance();

}

@Bean

public DataSource datasource()

{

//It will locate the driver and load it

//Database connection is done here

DriverManagerDataSource ds=new DriverManagerDataSource();

ds.setDriverClassName(databaseDriverClassName);

ds.setUrl(datasourceUrl);

ds.setUsername(databaseUserName);

ds.setPassword(databasePassword);

return ds;

}

@Override

protected void configure(HttpSecurity http) throws Exception

{

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

hasAnyAuthority("ROLE\_ADMIN","ROLE\_USER").antMatchers("/Project/deleteProject").

hasAnyAuthority("ROLE\_ADMIN").antMatchers("/Project/addProject").

hasAnyAuthority("ROLE\_ADMIN").antMatchers("/Project/addNewProject").

hasAnyAuthority("ROLE\_ADMIN").antMatchers("/Project/listProjects").

hasAnyAuthority("ROLE\_ADMIN","ROLE\_USER").anyRequest().authenticated().and().

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

}

}