Spring Security

1. Create a spring boot application

Dependencies: spring web, spring security

1. Create one rest controller and get mapping
2. Run the application.
3. Notice that password is generated.
4. When you go to browser and try to access any URL, you are prompted with username and password
5. By default, username is “user” and the password is to be copied from log.

Next task:

Exclude “SecurityAutoConfiguration.class” from @SpringBootApplication

Check now, that we are allowed to access any url without authentication

Next task:

In application.properties: configure your own choice of username and password

spring.security.user.name=jag

spring.security.user.password=jag@123

Instead of giving username and password in application.properties, we should provide them in java code.

I am going to configure user credentials using a configuration class now:

Class MyConfiguration implements WebSecurityConfigurer

Class MyConfiguration extends WebSecurityConfigurerAdapter

Create a configuration class:

package com.ust.spring.config;

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.configuration.WebSecurityConfigurerAdapter;

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

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

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

@Configuration

public class MyConfiguration extends WebSecurityConfigurerAdapter

{

@Autowired

private MyUserDetailsService userDetailsService;

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(userDetailsService);

}

@Bean

public PasswordEncoder passwordEncoder()

{

return NoOpPasswordEncoder.getInstance();

}

}

We need to create MyUserDetailsService that implements UserDetailsService

package com.ust.spring.config;

import java.util.ArrayList;

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 MyUserDetailsService implements UserDetailsService

{

@Override

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

return new User("dinesh", "diny", new ArrayList<>());

}

}

Now, run the project and check with username and passwords as “dinesh” and “diny”

Again, the username and password visible in java code is not good.

So we are going to configure this project to access an user object from database table using spring data jpa

1. Pom.xml, add the spring data jpa starter dependency
2. In application.properties: url, username, password, driver-class-name, dialect
3. Entity class
4. Repository
5. Server???? Already we have a service class, MyUserDetailsService we have to autowire the repository

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

1. Entity class
2. Repository interface
3. Service class

@Entity

public class MyUser extends User

{

@Id

private String username;

private String password;

}

@Repository

public interface UserRepository extends JpaRepository<MyUser,String>

{}

spring.datasource.url=jdbc:mysql://localhost:3306/ust1

spring.datasource.username=root

spring.datasource.password=

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

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

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

package com.ust.spring.entity;

import java.util.ArrayList;

import java.util.Collection;

import javax.persistence.Entity;

import javax.persistence.Id;

import org.springframework.security.core.GrantedAuthority;

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

@Entity

public class MyUser extends User

{

@Id

private String username;

private String password;

public MyUser()

{

super("jag","jag",new ArrayList<>());

}

public MyUser(String username, String password, Collection<? extends GrantedAuthority> authorities) {

super(username, password, authorities);

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

@Override

public String toString() {

return "MyUser [username=" + username + ", password=" + password + "]";

}

}

package com.ust.spring.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.ust.spring.entity.MyUser;

@Repository

public interface UserRepository extends JpaRepository<MyUser, String>

{

}

package com.ust.spring.config;

import java.util.Optional;

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

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;

import com.ust.spring.entity.MyUser;

import com.ust.spring.repository.UserRepository;

@Service

public class MyUserDetailsService implements UserDetailsService

{

@Autowired

private UserRepository ur;

@Override

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

// return new User("dinesh", "diny", new ArrayList<>());

Optional<MyUser> temp = ur.findById(username);

MyUser u=null;

if(temp.isPresent())

{

u=temp.get();

}

return u;

}

}

From postman, when we access a rest api, we will not be able to login using browser.

Every request to rest api, is independent. So how it will remember us after login? We need a token to make it remember us.

Spring Security with JWT

1. Create a spring boot project with following dependencies:

Spring web

Spring data jpa

Mysql driver

Spring security

Jjwt

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

<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-web</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<scope>runtime</scope>

</dependency>

1. Configuration

package com.ust.spring.config;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.BeanIds;

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.http.SessionCreationPolicy;

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

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

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

import com.ust.spring.filter.JwtFilter;

import com.ust.spring.service.MyUserDetailsService;

@Configuration

@EnableWebSecurity

public class MySecurityConfiguration extends WebSecurityConfigurerAdapter {

@Autowired

private JwtFilter jwtFilter;

@Autowired

private MyUserDetailsService userDetailsService;

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(userDetailsService);

}

@Bean(name = BeanIds.AUTHENTICATION\_MANAGER)

@Override

public AuthenticationManager authenticationManagerBean() throws Exception {

return super.authenticationManagerBean();

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.cors().and()

.csrf().disable()

.authorizeRequests().antMatchers("/login")

.permitAll().anyRequest().authenticated()

.and().exceptionHandling().and().sessionManagement()

.sessionCreationPolicy(SessionCreationPolicy.STATELESS);

http.addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);

}

@Bean

public PasswordEncoder passwordEncoder() {

return NoOpPasswordEncoder.getInstance();

}

}

Refer to the project named: 05-jun-security-jwt

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

Spring security “Logout” capabilities:

<https://rwinch.github.io/spring-reference/servlet/authentication/logout.html>

