STAGE-3

**ServletInitializer**

package com.springboot.moviecruiser;

import org.springframework.boot.builder.SpringApplicationBuilder;

import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

public class ServletInitializer extends SpringBootServletInitializer {

@Override

protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {

return application.sources(SpringbootApplication.class);

}

}

**SpringbootApplication**

package com.springboot.moviecruiser;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringbootApplication {

public static void main(String[] args) {

SpringApplication.run(SpringbootApplication.class, args);

}

}

**MovieController**

package com.springboot.moviecruiser.controller;

import java.math.BigInteger;

import java.util.ArrayList;

import java.util.List;

import java.util.Set;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.security.core.Authentication;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

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

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

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

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

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

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

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

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

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

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

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

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

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

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

import org.springframework.web.bind.support.SessionStatus;

import com.springboot.moviecruiser.movie.Favorites;

import com.springboot.moviecruiser.movie.Movies;

import com.springboot.moviecruiser.movie.User;

import com.springboot.moviecruiser.service.FavoriteService;

import com.springboot.moviecruiser.service.MovieService;

import com.springboot.moviecruiser.service.UserService;

import io.jsonwebtoken.JwtBuilder;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import java.util.Base64;

import java.util.Date;

import java.util.HashMap;

import java.util.Map;

@RestController

@CrossOrigin(origins="\*")

@SessionAttributes("username")

public class MovieController {

private static final Logger LOGGER = LoggerFactory.getLogger(MovieService.class);

@Autowired

MovieService movieService;

@Autowired

UserService userService;

@Autowired

FavoriteService favoriteService;

@CrossOrigin

@RequestMapping(value="/moviesAngular")

@ResponseBody

public List<Movies> getMovies(ModelMap model){

List<Movies> list=movieService.getAllMovies();

model.put("list", list);

return list;

}

@RequestMapping("/adminMovieList")

public List<Movies> viewMoviesAdmin(ModelMap model) {

List<Movies> adminMovieList = new ArrayList<>();

adminMovieList = movieService.getAllMovies();

model.put("movieList", adminMovieList);

return adminMovieList;

}

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

@ResponseBody

public List<Movies> viewFavorites(){

User user = userService.getUserById(1);

Set<Movies> movie=user.getFavorites();

List<Movies> favorites=new ArrayList<Movies>();

for(Movies m:movie) {

favorites.add(m);

}

return favorites;

}

@DeleteMapping("/deleteFavorites/{movieId}")

public ResponseEntity<Movies> removeFavorite(@PathVariable int movieId){

Movies movie=userService.removeFavorites(movieId);

return new ResponseEntity<Movies>(movie,HttpStatus.OK);

}

@PostMapping(value = "/addToFavorites")

public ResponseEntity<Movies> addToFavorites(@RequestBody String id){

Movies movie = movieService.findById(Integer.parseInt(id));

userService.addToFavorites(movie);

return new ResponseEntity<Movies>(movie, HttpStatus.OK);

}

@PutMapping("/editMovie/{movieId}")

public ResponseEntity<Movies> update(@PathVariable int movieId, @RequestBody Movies movie){

Movies result = movieService.updateMovie(movie);

return new ResponseEntity<Movies>(result, HttpStatus.OK);

}

@RequestMapping(value = "/logout")

public String ReturnToLogin(HttpServletRequest request, SessionStatus status) {

HttpSession session = request.getSession(false);

status.setComplete();

session.invalidate();

return "redirect:/login";

}

@GetMapping("/authenticate")

@ResponseBody

private Map<String, String> getToken(@RequestHeader("Authorization") String authHeader) {

Map<String, String> m = new HashMap<>();

String user = getUser(authHeader);

String token = generateJwt(user);

m.put("token", token );

return m;

}

private String getUser(String authHeader) {

String baseStr = authHeader.substring(authHeader.indexOf("Basic ")+6);

byte[] encCred = Base64.getDecoder().decode(baseStr);

String encStr = new String(encCred);

return encStr.substring(0, encStr.indexOf(":"));

}

private String generateJwt(String user) {

JwtBuilder builder = Jwts.builder();

builder.setSubject(user);

// Set the token issue time as current time

builder.setIssuedAt(new Date());

// Set the token expiry as 20 minutes from now

builder.setExpiration(new Date((new Date()).getTime() + 1200000));

builder.signWith(SignatureAlgorithm.HS256, "ABC123");

String token = builder.compact();

return token;

}

}

­­­ **Favorites**

package com.springboot.moviecruiser.movie;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.OneToOne;

import javax.persistence.Table;

@Entity

@Table(name = "favorites")

public class Favorites {

@Id

@Column(name = "fav\_id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "user\_id")

private int favUserId;

@Column(name = "movie\_id")

private int favMovieId;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public int getFavUserId() {

return favUserId;

}

public void setFavUserId(int favUserId) {

this.favUserId = favUserId;

}

public int getFavMovieId() {

return favMovieId;

}

public void setFavMovieId(int favMovieId) {

this.favMovieId = favMovieId;

}

**Movies**

package com.springboot.moviecruiser.movie;

import java.math.BigInteger;

import java.util.Date;

import java.util.Set;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.ManyToMany;

import javax.persistence.OneToOne;

import javax.persistence.Table;

@Entity

@Table(name="movie")

public class Movies {

@Id

@Column(name="movie\_id")

int movieId;

@Column(name="mtitle")

String title;

@Column(name="box\_office")

java.math.BigInteger boxOffice;

@Column(name="active\_status")

boolean active;

@Column(name="date\_of\_launch")

Date dateOfLaunch;

@Column(name="genre")

String genre;

@Column(name="has\_teaser")

boolean hasTeaser;

@ManyToMany(mappedBy="favorites")

Set<User> userList;

public int getMovieId() {

return movieId;

}

public void setMovieId(int movieId) {

this.movieId = movieId;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public java.math.BigInteger getBoxOffice() {

return boxOffice;

}

public void setBoxOffice(java.math.BigInteger boxOffice) {

this.boxOffice = boxOffice;

}

public boolean isActive() {

return active;

}

public void setActive(boolean active) {

this.active = active;

}

public Date getDateOfLaunch() {

return dateOfLaunch;

}

public void setDateOfLaunch(Date dateOfLaunch) {

this.dateOfLaunch = dateOfLaunch;

}

public String getGenre() {

return genre;

}

public void setGenre(String genre) {

this.genre = genre;

}

public boolean isHasTeaser() {

return hasTeaser;

}

public void setHasTeaser(boolean hasTeaser) {

this.hasTeaser = hasTeaser;

}

public Movies() {

}

public Movies(int movieId, String title, BigInteger boxOffice, boolean active, Date dateOfLaunch, String genre,

boolean hasTeaser) {

super();

this.movieId = movieId;

this.title = title;

this.boxOffice = boxOffice;

this.active = active;

this.dateOfLaunch = dateOfLaunch;

this.genre = genre;

this.hasTeaser = hasTeaser;

}

}

**Role**

package com.springboot.moviecruiser.movie;

import java.util.Set;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.ManyToMany;

import javax.persistence.Table;

@Entity

@Table(name="role")

public class Role {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name="ro\_id")

int roleId;

@Column(name="ro\_name")

String roleName;

@ManyToMany(mappedBy = "roleList")

private Set<User> userList;

public int getRoleId() {

return roleId;

}

public void setRoleId(int roleId) {

this.roleId = roleId;

}

public String getRoleName() {

return roleName;

}

public void setRoleName(String roleName) {

this.roleName = roleName;

}

public Set<User> getUserList() {

return userList;

}

public void setUserList(Set<User> userList) {

this.userList = userList;

}

@Override

public String toString() {

return "Role [roleId=" + roleId + ", roleName=" + roleName + ", userList=" + userList + "]";

}

}

**User**

package com.springboot.moviecruiser.movie;

import java.util.Set;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.ManyToMany;

import javax.persistence.Table;

@Entity

@Table(name="user")

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name="us\_id")

int userId;

@Column(name="us\_username")

String username;

@Column(name="us\_password")

String password;

@ManyToMany(fetch=FetchType.EAGER)

@JoinTable(name = "user\_role",

joinColumns = @JoinColumn(name = "ur\_us\_id"),

inverseJoinColumns = @JoinColumn(name = "ur\_ro\_id"))

private Set<Role> roleList;

@ManyToMany(fetch=FetchType.EAGER)

@JoinTable(name ="favorites",joinColumns = @JoinColumn(name ="user\_id",referencedColumnName="us\_id"),

inverseJoinColumns = @JoinColumn(name ="movie\_id",referencedColumnName="movie\_id"))

private Set<Movies> favorites;

public Set<Movies> getFavorites() {

return favorites;

}

public void setFavorites(Set<Movies> favorites) {

this.favorites = favorites;

}

public int getUserId() {

return userId;

}

public void setUserId(int userId) {

this.userId = userId;

}

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;

}

public Set<Role> getRoleList() {

return roleList;

}

public void setRoleList(Set<Role> roleList) {

this.roleList = roleList;

}

//

// @Override

// public String toString() {

// return "User [userId=" + userId + ", username=" + username + ", password=" + password + ", roleList=" + roleList

// + "]";

// }

}

**FavoritesRepository**

package com.springboot.moviecruiser.repository;

import java.util.Set;

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

import org.springframework.stereotype.Repository;

import com.springboot.moviecruiser.movie.Favorites;

@Repository

public interface FavoritesRepository extends JpaRepository<Favorites, Integer>{

Set<Favorites> findByFavUserId(int id);

void deleteById(int mov\_id);

}

**MovieRepository**

package com.springboot.moviecruiser.repository;

import java.util.List;

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

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

import org.springframework.stereotype.Repository;

import com.springboot.moviecruiser.movie.Movies;

@Repository

public interface MovieRepository extends JpaRepository<Movies, Integer>{

@Query(value="FROM Movies")

List<Movies> getAllMovies();

List<Movies> findByActive(boolean value);

Movies findByTitle(String Movietitle);

}

**UserRepository**

package com.springboot.moviecruiser.repository;

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

import com.springboot.moviecruiser.movie.User;

public interface UserRepository extends JpaRepository<User, Integer>{

User findByUsername(String name);

User findByUserId(int id);

}

**AppUser**

package com.springboot.moviecruiser.security;

import java.util.ArrayList;

import java.util.Collection;

import java.util.List;

import java.util.Set;

import org.springframework.security.core.GrantedAuthority;

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

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

import com.springboot.moviecruiser.movie.Role;

import com.springboot.moviecruiser.movie.User;

public class AppUser implements UserDetails {

private static final long serialVersionUID = 1L;

private User user;

private List<SimpleGrantedAuthority> authorities;

// private Collection<? extends GrantedAuthority> authorities;

public AppUser(User user) {

this.user = user;

Set<Role> roles = user.getRoleList();

this.authorities = new ArrayList<>();

for (Role role : roles) {

authorities.add(new SimpleGrantedAuthority(role.getRoleName()));

}

}

@Override

public Collection<? extends GrantedAuthority> getAuthorities() {

return authorities;

}

@Override

public String getPassword() {

return user.getPassword();

}

@Override

public String getUsername() {

return user.getUsername();

}

@Override

public boolean isAccountNonExpired() {

return true;

}

@Override

public boolean isAccountNonLocked() {

return true;

}

@Override

public boolean isCredentialsNonExpired() {

return true;

}

@Override

public boolean isEnabled() {

return true;

}

}

**AppUserDetailsService**

package com.springboot.moviecruiser.security;

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.springboot.moviecruiser.movie.User;

import com.springboot.moviecruiser.repository.UserRepository;

@Service

public class AppUserDetailsService implements UserDetailsService{

@Autowired

UserRepository userRepository;

@Override

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

User user = userRepository.findByUsername(username);

if(user == null) {

throw new UsernameNotFoundException(username);

}

AppUser appUser = new AppUser(user);

return appUser;

}

}

**JWTAuthFilter**

package com.springboot.moviecruiser.security;

import java.io.IOException;

import java.util.ArrayList;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

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

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.web.authentication.www.BasicAuthenticationFilter;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jws;

import io.jsonwebtoken.JwtException;

import io.jsonwebtoken.Jwts;

public class JWTAuthFilter extends BasicAuthenticationFilter {

@Override

protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain chain)

throws IOException, ServletException {

String header = request.getHeader("Authorization");

UsernamePasswordAuthenticationToken authentication = getAuthentication(request);

SecurityContextHolder.getContext().setAuthentication(authentication);

chain.doFilter(request, response);

}

public JWTAuthFilter(AuthenticationManager authenticationManager) {

super(authenticationManager);

}

private UsernamePasswordAuthenticationToken getAuthentication(HttpServletRequest request) {

String token = request.getHeader("Authorization");

if (token != null) {

// parse the token.

Jws<Claims> jws;

try {

jws = Jwts.parser().setSigningKey("ABC123").parseClaimsJws(token.replace("Bearer ", ""));

String user = jws.getBody().getSubject();

ArrayList<SimpleGrantedAuthority> arr = new ArrayList<>();

// Get the role list from UserRepo

arr.add(new SimpleGrantedAuthority("ROLE\_USER"));

if (user != null) {

return new UsernamePasswordAuthenticationToken(user, null, arr);

}

} catch (JwtException ex) {

return null;

}

return null;

}

return null;

}

}

**SecurityConfig**

package com.springboot.moviecruiser.security;

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.crypto.password.PasswordEncoder;

import org.springframework.security.web.authentication.www.BasicAuthenticationFilter;

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter{

@Autowired

AppUserDetailsService userDetailsService;

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

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

}

@Bean

public PasswordEncoder getPasswordEncoder() {

return new BCryptPasswordEncoder();

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.csrf().disable()

.authorizeRequests()

.antMatchers("/customerMovieList")

.hasRole("USER")

.antMatchers("/deleteFavorites/\*\*")

.hasRole("USER")

.antMatchers("/viewFavorites")

.hasRole("USER")

.antMatchers("/addToFavorites")

.hasRole("USER")

.antMatchers("/adminMovieList")

.hasRole("ADMIN");

//http.formLogin();

//http.addFilter(new BasicAuthenticationFilter(authenticationManager()));

http.addFilter(new JWTAuthFilter(authenticationManager()));

}

// @Override

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

// http.authorizeRequests()

// .antMatchers("/adminMovieList").hasRole("admin")

// .antMatchers("/customerMovieList").hasRole("user")

// .and()

// .formLogin().successHandler(myAuthenticationSuccessHandler())

// .and()

// .logout()

// .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))

// .logoutSuccessUrl("/login");

//

// }

//

// @Bean

// protected AuthenticationSuccessHandler myAuthenticationSuccessHandler() {

// return new UrlSuccessHandler();

// }

}

**FavoriteService**

package com.springboot.moviecruiser.service;

import java.util.Set;

import javax.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import com.springboot.moviecruiser.movie.Favorites;

import com.springboot.moviecruiser.repository.FavoritesRepository;

@Service

public class FavoriteService {

@Autowired

FavoritesRepository favoritesRepository;

@Transactional

public Set<Favorites> getFavorites(int id){

return favoritesRepository.findByFavUserId(id);

}

@Transactional

public void addToFavorites(Favorites favorite) {

favoritesRepository.save(favorite);

}

@Transactional

public void deleteFavorites(int movieid) {

favoritesRepository.deleteById(movieid);

}

}

**MovieService**

package com.springboot.moviecruiser.service;

import java.util.List;

import javax.transaction.Transactional;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Service;

import com.springboot.moviecruiser.movie.Movies;

import com.springboot.moviecruiser.repository.MovieRepository;

@Service

public class MovieService {

@Autowired

MovieRepository movieRepository;

@Transactional

public List<Movies> getAllMovies(){

return movieRepository.getAllMovies();

}

@Transactional

public List<Movies> findAllCustomerMovies(){

return movieRepository.findByActive(true);

}

@Transactional

public void saveMovie(Movies movie) {

movieRepository.save(movie);

}

@Transactional

public Movies getMoviesByName(String title) {

return movieRepository.findByTitle(title);

}

@Transactional

public Movies findById(int id) {

return movieRepository.findById(id).get();

}

public Movies updateMovie(Movies movie) {

Movies editResult = movieRepository.findById(movie.getMovieId()).get();

editResult.setTitle(movie.getTitle());

editResult.setGenre(movie.getGenre());

editResult.setBoxOffice(movie.getBoxOffice());

movieRepository.save(editResult);

return editResult;

}

}

**UserService**

package com.springboot.moviecruiser.service;

import java.util.Set;

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

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

import org.springframework.stereotype.Service;

import com.springboot.moviecruiser.movie.Movies;

import com.springboot.moviecruiser.movie.User;

import com.springboot.moviecruiser.repository.MovieRepository;

import com.springboot.moviecruiser.repository.UserRepository;

@Service

public class UserService {

@Autowired

UserRepository userRepository;

@Autowired

MovieRepository movieRepository;

public User getUserByName(String name) {

return userRepository.findByUsername(name);

}

public User getUserById(int id) {

return userRepository.findByUserId(id);

}

public Movies removeFavorites(int id) {

User user = userRepository.findByUserId(1);

Set<Movies> favorites = user.getFavorites();

Movies movie = movieRepository.findById(id).get();

favorites.remove(movie);

user.setFavorites(favorites);

userRepository.save(user);

return movie;

}

public Movies addToFavorites(Movies movie) {

User user = userRepository.findByUserId(1);

Set<Movies> favorites = user.getFavorites();

favorites.add(movie);

user.setFavorites(favorites);

userRepository.save(user);

return movie;

}

}

**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>com.springboot</groupId>

<artifactId>moviecruiser</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<name>moviecruiser</name>

<!-- FIXME change it to the project's website -->

<url>http://www.example.com</url>

<parent>

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

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

<version>2.4.2</version>

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

</parent>

<properties>

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

<maven.compiler.source>1.7</maven.compiler.source>

<maven.compiler.target>1.7</maven.compiler.target>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>bootstrap</artifactId>

<version>3.3.6</version>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>jquery</artifactId>

<version>1.9.1</version>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

<scope>provided</scope>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

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

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

<scope>provided</scope>

</dependency>

<dependency>

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

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

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>mysql</groupId>

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-security-web</artifactId>

</dependency>

<dependency>

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

<artifactId>spring-security-config</artifactId>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.0</version>

</dependency>

</dependencies>

<build>

<pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->

<plugins>

<!-- clean lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#clean\_Lifecycle -->

<plugin>

<artifactId>maven-clean-plugin</artifactId>

<version>3.1.0</version>

</plugin>

<!-- default lifecycle, jar packaging: see https://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin\_bindings\_for\_jar\_packaging -->

<plugin>

<artifactId>maven-resources-plugin</artifactId>

<version>3.0.2</version>

</plugin>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.0</version>

</plugin>

<plugin>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.1</version>

</plugin>

<plugin>

<artifactId>maven-jar-plugin</artifactId>

<version>3.0.2</version>

</plugin>

<plugin>

<artifactId>maven-install-plugin</artifactId>

<version>2.5.2</version>

</plugin>

<plugin>

<artifactId>maven-deploy-plugin</artifactId>

<version>2.8.2</version>

</plugin>

<!-- site lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#site\_Lifecycle -->

<plugin>

<artifactId>maven-site-plugin</artifactId>

<version>3.7.1</version>

</plugin>

<plugin>

<artifactId>maven-project-info-reports-plugin</artifactId>

<version>3.0.0</version>

</plugin>

<plugin>

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

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

</plugin>

</plugins>

</pluginManagement>

</build>

</project>