[Thực hành] Thực hành xây dựng JWT

Mục tiêu

Xây dưng và sử dụng được JWT

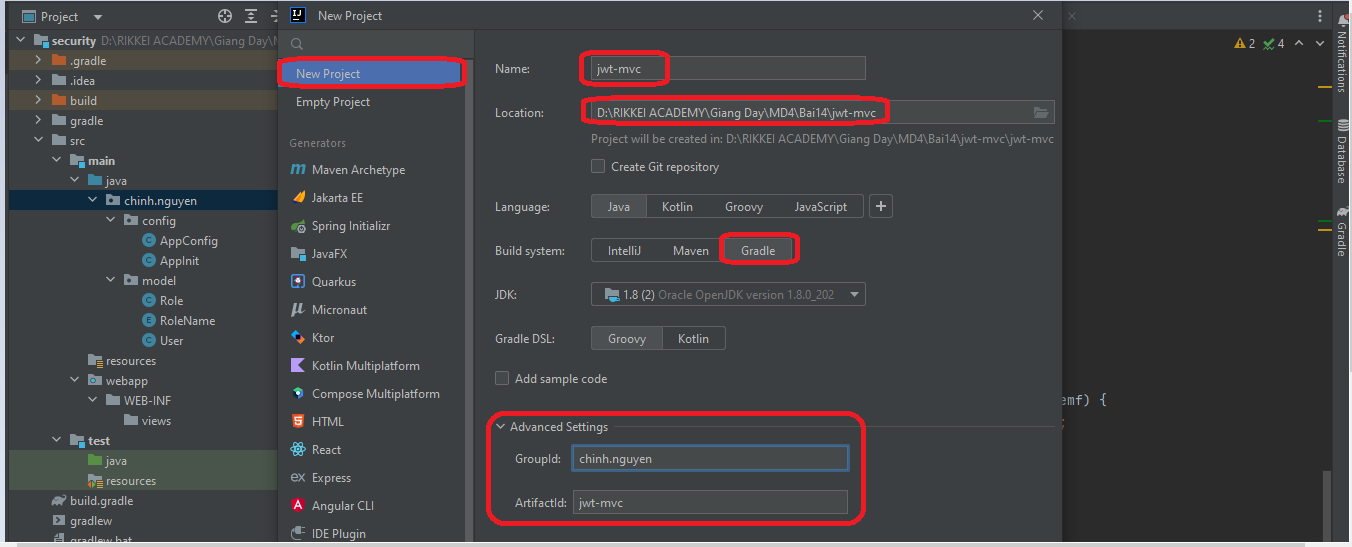
Mô tả

Trong phần này, chúng ta sẽ phát triển một ứng dụng sử dụng Spring Security và Restful API.

Ứng dụng sẽ cho phép truy cập vào các API cần có quyền truy cập.

Hướng dẫn

* Tạo project Gradle như hình:



* Import thư viện vào **build.gradle**

providedCompile group: 'javax.servlet', name: 'javax.servlet-api', version: '4.0.1'  
implementation group: 'org.springframework', name: 'spring-core', version: '5.3.2'  
implementation group: 'org.springframework', name: 'spring-context', version: '5.3.2'  
implementation group: 'org.springframework', name: 'spring-beans', version: '5.3.2'  
implementation group: 'org.springframework', name: 'spring-web', version: '5.3.2'  
implementation group: 'org.springframework', name: 'spring-webmvc', version: '5.3.2'  
//ThymeLeaf Spring5  
implementation group: 'org.thymeleaf', name: 'thymeleaf-spring5', version: '3.0.11.RELEASE'  
implementation group: 'nz.net.ultraq.thymeleaf', name: 'thymeleaf-layout-dialect', version: '2.5.2'  
//Hibernate  
implementation group: 'org.hibernate', name: 'hibernate-core', version: '5.3.0.Final'  
implementation group: 'org.hibernate', name: 'hibernate-entitymanager', version: '5.3.0.Final'  
implementation group: 'org.springframework', name: 'spring-orm', version: '4.3.17.RELEASE'  
implementation group: 'mysql', name: 'mysql-connector-java', version: '8.0.11'  
//jpa  
implementation group: 'org.springframework.data', name: 'spring-data-jpa', version: '2.4.2'  
//Validate  
implementation group: 'javax.validation', name: 'validation-api', version: '2.0.1.Final'  
implementation group: 'org.hibernate', name: 'hibernate-validator', version: '6.0.10.Final'  
//security  
implementation group: 'org.springframework.security', name: 'spring-security-web', version: '5.4.2'  
implementation group: 'org.springframework.security', name: 'spring-security-config', version: '5.4.2'  
implementation group: 'org.springframework.security', name: 'spring-security-taglibs', version: '5.4.2'  
implementation group: 'org.springframework.data', name: 'spring-data-jpa', version: '2.4.2'  
  
implementation group: 'org.springframework', name: 'spring-tx', version: '5.3.2'  
//json vs jwt  
implementation group: 'com.fasterxml.jackson.core', name: 'jackson-databind', version: '2.13.4'  
implementation group: 'io.jsonwebtoken', name: 'jjwt', version: '0.9.0'  
implementation "com.fasterxml.jackson.core:jackson-databind:"

* Tạo package: **chinh.nguyen.config**
* Tạo class: **AppConfig**

package chinh.nguyen.config;  
  
import org.springframework.beans.BeansException;  
import org.springframework.beans.factory.annotation.Qualifier;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.ApplicationContextAware;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.ComponentScan;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.data.web.config.EnableSpringDataWebSupport;  
import org.springframework.jdbc.datasource.DriverManagerDataSource;  
import org.springframework.orm.jpa.JpaTransactionManager;  
import org.springframework.orm.jpa.JpaVendorAdapter;  
import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;  
import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;  
import org.springframework.security.config.annotation.authentication.configuration.EnableGlobalAuthentication;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.servlet.configuration.EnableWebMvcSecurity;  
import org.springframework.transaction.PlatformTransactionManager;  
import org.springframework.web.servlet.config.annotation.EnableWebMvc;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
import org.thymeleaf.spring5.SpringTemplateEngine;  
import org.thymeleaf.spring5.templateresolver.SpringResourceTemplateResolver;  
import org.thymeleaf.spring5.view.ThymeleafViewResolver;  
import org.thymeleaf.templatemode.TemplateMode;  
import javax.persistence.EntityManager;  
import javax.persistence.EntityManagerFactory;  
import javax.sql.DataSource;  
import java.util.Properties;  
  
  
@EnableWebMvc  
@Configuration  
@EnableSpringDataWebSupport  
@EnableGlobalAuthentication  
@EnableWebSecurity  
@EnableWebMvcSecurity  
public class AppConfig implements WebMvcConfigurer, ApplicationContextAware {  
 private ApplicationContext applicationContext;  
  
 @Override  
 public void setApplicationContext(ApplicationContext applicationContext) throws BeansException {  
 this.applicationContext = applicationContext;  
 }  
  
 //Cấu hình Thymleaf  
 @Bean  
 public SpringResourceTemplateResolver templateResolver() {  
 SpringResourceTemplateResolver templateResolver = new SpringResourceTemplateResolver();  
 templateResolver.setApplicationContext(applicationContext);  
 templateResolver.setPrefix("/WEB-INF/views");  
 templateResolver.setSuffix(".html");  
 templateResolver.setTemplateMode(TemplateMode.*HTML*);  
 templateResolver.setCharacterEncoding("UTF-8");  
 return templateResolver;  
 }  
  
 @Bean  
 public SpringTemplateEngine templateEngine() {  
 SpringTemplateEngine templateEngine = new SpringTemplateEngine();  
 templateEngine.setTemplateResolver(templateResolver());  
 return templateEngine;  
 }  
  
 @Bean  
 public ThymeleafViewResolver viewResolver() {  
 ThymeleafViewResolver viewResolver = new ThymeleafViewResolver();  
 viewResolver.setTemplateEngine(templateEngine());  
 viewResolver.setCharacterEncoding("UTF-8");  
 viewResolver.setContentType("UTF-8");  
 return viewResolver;  
 }  
  
 //Cấu hình JPA  
 @Bean  
 @Qualifier(value = "entityManager")  
 public EntityManager entityManager(EntityManagerFactory entityManagerFactory) {  
 return entityManagerFactory.createEntityManager();  
 }  
  
 @Bean  
 public LocalContainerEntityManagerFactoryBean entityManagerFactory() {  
 LocalContainerEntityManagerFactoryBean em = new LocalContainerEntityManagerFactoryBean();  
 em.setDataSource(dataSource());  
 em.setPackagesToScan("chinh.nguyen.model");  
  
 JpaVendorAdapter vendorAdapter = new HibernateJpaVendorAdapter();  
 em.setJpaVendorAdapter(vendorAdapter);  
 em.setJpaProperties(additionalProperties());  
 return em;  
 }  
  
 @Bean  
 public DataSource dataSource() {  
 DriverManagerDataSource dataSource = new DriverManagerDataSource();  
 dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");  
 dataSource.setUrl("jdbc:mysql://localhost:3306/jwt\_mvc");  
 dataSource.setUsername("root");  
 dataSource.setPassword("Minhtri29092014");  
 return dataSource;  
 }  
  
 @Bean  
 public PlatformTransactionManager transactionManager(EntityManagerFactory emf) {  
 JpaTransactionManager transactionManager = new JpaTransactionManager();  
 transactionManager.setEntityManagerFactory(emf);  
 return transactionManager;  
 }  
  
 public Properties additionalProperties() {  
 Properties properties = new Properties();  
 properties.setProperty("hibernate.hbm2ddl.auto", "update");  
 properties.setProperty("hibernate.dialect", "org.hibernate.dialect.MySQL5Dialect");  
 return properties;  
 }  
}

* Tạo package: **model** trong package: **chinh.nguyen**
* Tạo class enum: **RoleName** (Tạo kiểu dữ liệu cho class Role)

package chinh.nguyen.model;  
  
public enum RoleName {  
 *USER*,*PM*,*ADMIN*}

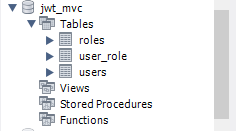
* Tạo class: **Role** trong package: **chinh.nguyen.model**

package chinh.nguyen.model;  
  
import org.hibernate.annotations.NaturalId;  
  
import javax.persistence.\*;  
  
@Entity  
@Table(name = "roles")  
public class Role {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 @Enumerated(EnumType.*STRING*)  
 @NaturalId  
 @Column(length = 60)  
 private RoleName name;  
  
 public Role(Long id, RoleName name) {  
 this.id = id;  
 this.name = name;  
 }  
  
 public Role() {  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public RoleName getName() {  
 return name;  
 }  
  
 public void setName(RoleName name) {  
 this.name = name;  
 }}

* Tạo class: **User** trong package: **model**

package chinh.nguyen.model;  
  
import com.fasterxml.jackson.annotation.JsonIgnore;  
import org.hibernate.annotations.NaturalId;  
  
import javax.persistence.\*;  
import javax.validation.constraints.Email;  
import javax.validation.constraints.NotBlank;  
import javax.validation.constraints.Size;  
import java.util.HashSet;  
import java.util.Set;  
  
@Entity  
@Table(name = "users", uniqueConstraints = {  
 @UniqueConstraint(columnNames = {  
 "username"  
 }),  
 @UniqueConstraint(columnNames = {  
 "email"  
 })  
})  
public class User {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 @NotBlank  
 @Size(min = 3, max = 50)  
 private String name;  
 @NotBlank  
 @Size(min = 3,max = 50)  
 private String username;  
 @NaturalId  
 @NotBlank  
 @Size  
 @Email  
 private String email;  
 // @JsonIgnore  
 @ManyToMany(fetch = FetchType.*EAGER*)  
 @JoinTable(name = "user\_role",  
 joinColumns = @JoinColumn(name = "user\_id"),  
 inverseJoinColumns = @JoinColumn(name = "role\_id"))  
 Set<Role> roles = new HashSet<>();  
 @JsonIgnore  
 @NotBlank  
 private String password;  
  
 public User() {  
 }  
  
 public User(Long id, String name, String username, String email, Set<Role> roles, String password) {  
 this.id = id;  
 this.name = name;  
 this.username = username;  
 this.email = email;  
 this.roles = roles;  
 this.password = password;  
 }  
  
 public User(String name, String username, String email, String encode) {  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public Set<Role> getRoles() {  
 return roles;  
 }  
  
 public void setRoles(Set<Role> roles) {  
 this.roles = roles;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 @Override  
 public String toString() {  
 return "User{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 ", username='" + username + '\'' +  
 ", email='" + email + '\'' +  
 ", roles=" + roles +  
 ", password='" + password + '\'' +  
 '}';  
 }  
}

* Tạo database: **jwt\_mvc** trong MySQL, chỉnh sửa các phần user, password với MySQL của các bạn, add Tomcat và chạy chương trình => Quan sát kết quả thu được dưới database.



* Tạo package: **repository** trong package: **chinh.nguyen:**
* Tạo interface: **IRoleRepository** trong package này

package chinh.nguyen.repository;  
  
import chinh.nguyen.model.Role;  
import chinh.nguyen.model.RoleName;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
import java.util.Optional;  
  
public interface IRoleRepository extends JpaRepository<Role, Long> {  
 Optional<Role> findByName(RoleName name);  
}

* Tạo interface: **IUserRepository** trong package: **repository**

package chinh.nguyen.repository;  
  
import chinh.nguyen.model.User;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
import java.util.Optional;  
  
public interface IUserRepository extends JpaRepository<User, Long> {  
 Boolean existsByUsername(String username); //Check username không trùng lặp  
 Boolean existsByEmail(String email); //Check email không trùng lặp  
 Optional<User> findByUsername(String username); //Tìm kiếm username có tồn tại trong DB không?  
}

* Tạo package: **service**
* Tạo package: **role** trong package **service** => Tạo interface: **IRoleService**

package chinh.nguyen.service.role;  
  
import chinh.nguyen.model.Role;  
import chinh.nguyen.model.RoleName;  
  
import java.util.Optional;  
  
public interface IRoleService {  
 Optional<Role> findByName(RoleName name);  
}

* Tạo class: **RoleServiceIMPL** triển khai interface trên:

package chinh.nguyen.service.role;  
  
import chinh.nguyen.model.Role;  
import chinh.nguyen.model.RoleName;  
import chinh.nguyen.repository.IRoleRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
  
import java.util.Optional;  
  
public class RoleServiceIMPL implements IRoleService{  
 @Autowired  
 IRoleRepository repository;  
 @Override  
 public Optional<Role> findByName(RoleName name) {  
 return repository.findByName(name);  
 }  
}

* Tạo package: **user** trong pakage: **service**
* Tạo interface: **IUserService** trong package: **user**

package chinh.nguyen.service.user;  
  
import chinh.nguyen.model.User;  
  
import java.util.Optional;  
  
public interface IUserService {  
 Boolean existsByUsername(String username);  
 Boolean existsByEmail(String email);  
 Optional<User> findByUsername(String username);  
 User save(User user);  
}

* Triển khai interface trên thông qua: **UserServiceIMPL**

package chinh.nguyen.service.user;  
  
import chinh.nguyen.model.User;  
import chinh.nguyen.repository.IUserRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
  
import java.util.Optional;  
  
public class UserServiceIMPL implements IUserService{  
 @Autowired  
 IUserRepository userRepository;  
 @Override  
 public Boolean existsByUsername(String username) {  
 return userRepository.existsByUsername(username);  
 }  
  
 @Override  
 public Boolean existsByEmail(String email) {  
 return userRepository.existsByEmail(email);  
 }  
   
 @Override  
 public Optional<User> findByUsername(String username) {  
 return userRepository.findByUsername(username);  
 }  
  
 @Override  
 public User save(User user) {  
 return userRepository.save(user);  
 }  
}

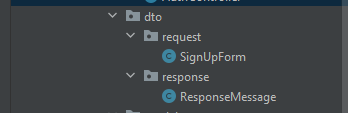
* Tạo package: **dto** trong package: **chinh.nguyen =>** Tạo 2 package con **request** và **response** trong package **dto**
* Tạo class: **SignUpForm** trong package: **request: *(Class này được tạo ra mới mục đích chia nhỏ từ class User gốc để hứng dữ liệu form đăng ký từ FrontEnd)***

package chinh.nguyen.dto.request;  
  
import java.util.Set;  
  
public class SignUpForm {  
 private String name;  
 private String username;  
 private String email;  
 private String password;  
 Set<String> roles;  
  
 public SignUpForm() {  
 }  
  
 public SignUpForm(String name, String username, String email, String password, Set<String> roles) {  
 this.name = name;  
 this.username = username;  
 this.email = email;  
 this.password = password;  
 this.roles = roles;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public Set<String> getRoles() {  
 return roles;  
 }  
  
 public void setRoles(Set<String> roles) {  
 this.roles = roles;  
 }  
}

* Tạo class: **ResponseMessage** trong package: **response** (***Class này tạo ra nhằm mục đích định nghĩa dữ liệu trả về thông qua các response tại các API***)

package chinh.nguyen.dto.response;  
  
public class ResponseMessage {  
 private String message;  
  
 public ResponseMessage() {  
 }  
  
 public ResponseMessage(String message) {  
 this.message = message;  
 }  
  
 public String getMessage() {  
 return message;  
 }  
  
 public void setMessage(String message) {  
 this.message = message;  
 }  
}

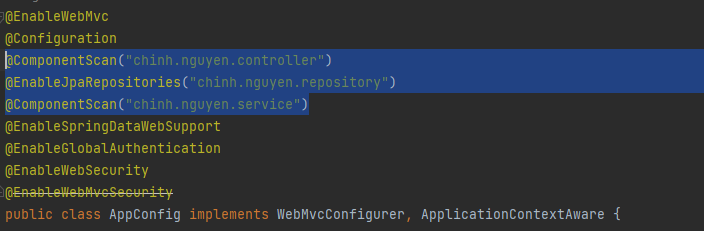
* Cấu trúc dự án sau khi tạo:



* Config các đường dẫn sang service và repository tiêm các sự phụ thuộc thông qua @Bean vào **AppConfig**

@ComponentScan("chinh.nguyen.controller")  
@EnableJpaRepositories("chinh.nguyen.repository")  
@ComponentScan("chinh.nguyen.service")

* Cấu trúc dự án:

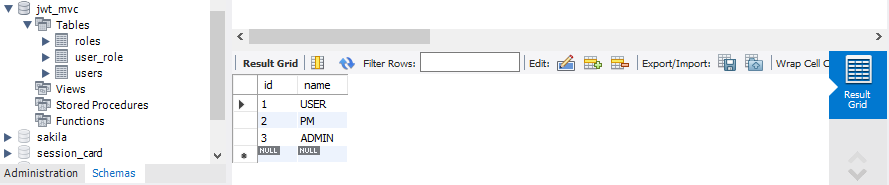


@Bean  
IRoleService roleService() {  
 return new RoleServiceIMPL();  
}  
  
@Bean  
IUserService userService() {  
 return new UserServiceIMPL();  
}  
  
@Bean  
public PasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
}

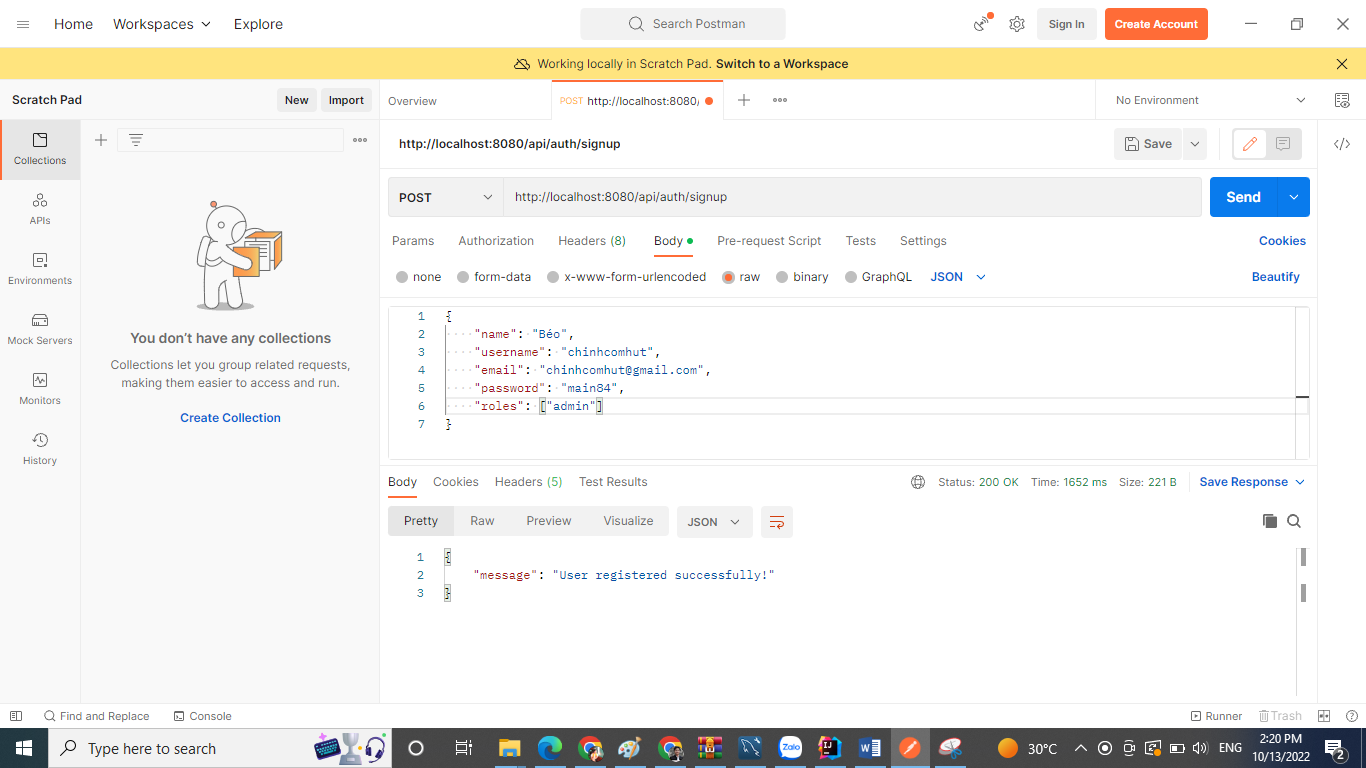
* Tạo package: **controller** trong package: **chinh.nguyen**
* Tạo class: **AuthController** trong package: **controller**
* Thêm vào code để tạo API : đăng ký:

package chinh.nguyen.controller;  
  
import chinh.nguyen.dto.request.SignUpForm;  
import chinh.nguyen.dto.response.ResponseMessage;  
import chinh.nguyen.model.Role;  
import chinh.nguyen.model.RoleName;  
import chinh.nguyen.model.User;  
import chinh.nguyen.service.role.IRoleService;  
import chinh.nguyen.service.user.IUserService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
import javax.validation.Valid;  
import java.util.HashSet;  
import java.util.Set;  
  
@RestController  
@RequestMapping("/api/auth")  
public class AuthController {  
 @Autowired  
 PasswordEncoder passwordEncoder;  
 @Autowired  
 IUserService userService;  
 @Autowired  
 IRoleService roleService;  
 @PostMapping("/signup")  
 public ResponseEntity<?> registerUser(@Valid @RequestBody SignUpForm signUpRequest) {  
 if (userService.existsByUsername(signUpRequest.getUsername())) {  
 return new ResponseEntity<>(new ResponseMessage("Fail -> Username is already taken!"),  
 HttpStatus.*OK*);  
 }  
  
 if (userService.existsByEmail(signUpRequest.getEmail())) {  
 return new ResponseEntity<>(new ResponseMessage("Fail -> Email is already in use!"),  
 HttpStatus.*OK*);  
 }  
  
 // Creating user's account  
 User user = new User();  
 user.setName(signUpRequest.getName());  
 user.setUsername(signUpRequest.getUsername());  
 user.setEmail(signUpRequest.getEmail());  
 user.setPassword(passwordEncoder.encode(signUpRequest.getPassword()));  
 System.*out*.println("user ROW1"+user.toString());  
 Set<String> strRoles = signUpRequest.getRoles();  
 Set<Role> roles = new HashSet<>();  
  
 strRoles.forEach(role -> {  
 switch (role) {  
 case "admin":  
 Role adminRole = roleService.findByName(RoleName.*ADMIN*)  
 .orElseThrow(() -> new RuntimeException("Fail! -> Cause: User Role not find."));  
 roles.add(adminRole);  
  
 break;  
 case "pm":  
 Role pmRole = roleService.findByName(RoleName.*PM*)  
 .orElseThrow(() -> new RuntimeException("Fail! -> Cause: User Role not find."));  
 roles.add(pmRole);  
  
 break;  
 default:  
 Role userRole = roleService.findByName(RoleName.*USER*)  
 .orElseThrow(() -> new RuntimeException("Fail! -> Cause: User Role not find."));  
 roles.add(userRole);  
 }  
 });  
  
 user.setRoles(roles);  
 userService.save(user);  
 System.*out*.println("user"+user.toString());  
 return new ResponseEntity<>(new ResponseMessage("User registered successfully!"), HttpStatus.*OK*);  
 }  
}

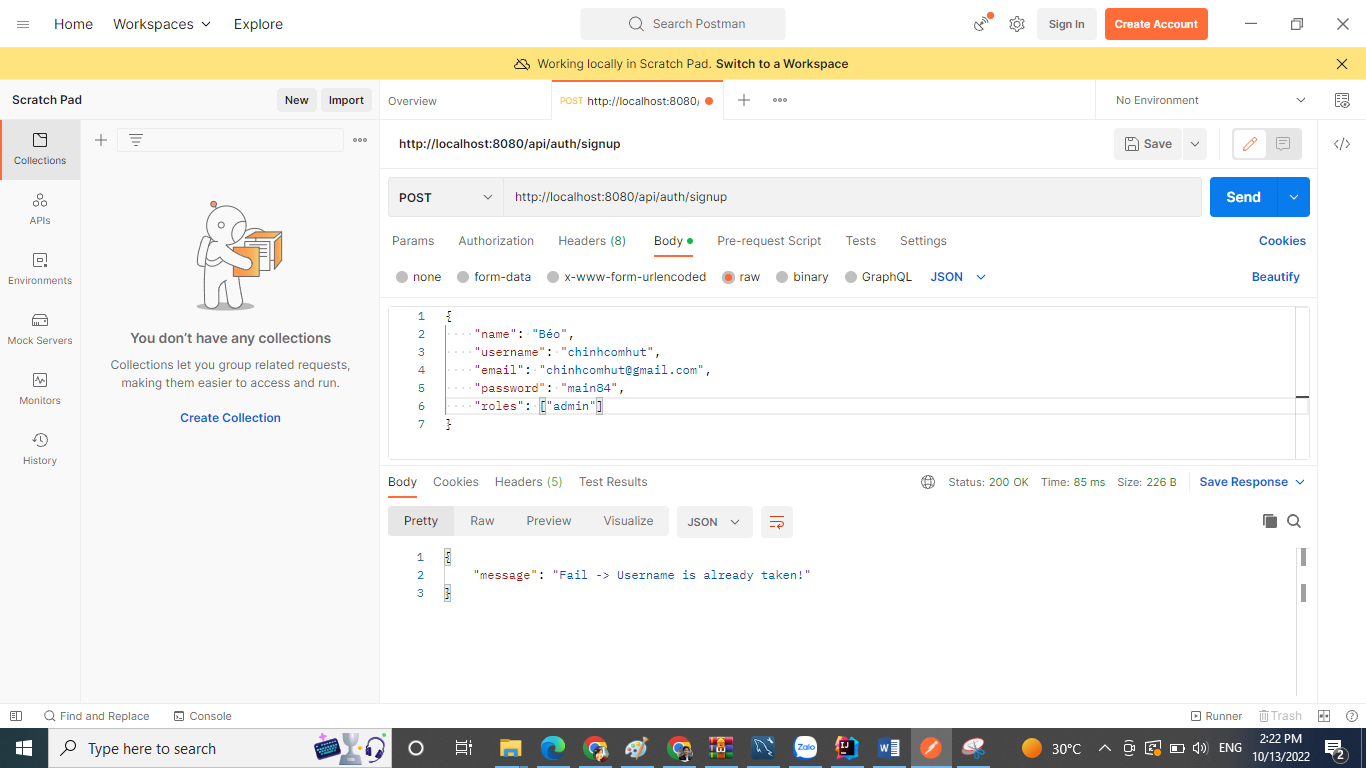
* Fix data trong bảng **Role: (Dữ liệu phân quyền)**



* Dùng: **POSTMANT** để test API



* Test cả các trường hợp username trùng và email trùng



* Bây giờ chúng ta sẽ tạo chức năng LOGIN => Chức năng này khó nhất trong JWT. Không giống như các chức năng login trước đây các bạn HV đã học. Chức năng LOGIN này có kết hợp với tính năng bảo mật. Mã hóa thông tin người dùng thành chuỗi TOKEN. Ngoài ra kết hợp với Authentication Security.
* Tạo package: **security** trong package: **chinh.nguyen**
* Tạo 2 package con : **jwt** và package: **userprincipal** trong package **security:**
* Tạo class: **UserPrinciple** trong package: **userprincipal**: (Class này tạo ra với nhiệm vụ tạo User đăng nhập hiện tại build lên hệ thống thông qua Authen security của Spring bằng cách kế thừa interface: **UserDetails** của hệ thống) .

package chinh.nguyen.security.userprincipal;  
  
import chinh.nguyen.model.User;  
import com.fasterxml.jackson.annotation.JsonIgnore;  
import org.springframework.security.core.GrantedAuthority;  
import org.springframework.security.core.authority.SimpleGrantedAuthority;  
import org.springframework.security.core.userdetails.UserDetails;  
  
import java.util.Collection;  
import java.util.List;  
import java.util.Objects;  
import java.util.stream.Collectors;  
  
public class UserPrinciple implements UserDetails {  
 private static final long *serialVersionUID* = 1L;  
  
 private Long id;  
  
 private String name;  
  
 private String username;  
  
 private String email;  
  
  
 @JsonIgnore  
 private String password;  
  
 private Collection<? extends GrantedAuthority> roles;  
  
 public UserPrinciple(Long id, String name,  
 String username, String email, String password,  
 Collection<? extends GrantedAuthority> roles) {  
 this.id = id;  
 this.name = name;  
 this.username = username;  
 this.email = email;  
 this.password = password;  
 this.roles = roles;  
 }  
//Lấy User hiện tại và build user trong Authentication  
 public static UserPrinciple build(User user) {  
 List<GrantedAuthority> authorities = user.getRoles().stream().map(role ->  
 new SimpleGrantedAuthority(role.getName().name())  
 ).collect(Collectors.*toList*());  
  
 return new UserPrinciple(  
 user.getId(),  
 user.getName(),  
 user.getUsername(),  
 user.getEmail(),  
 user.getPassword(),  
 authorities  
 );  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public static long getSerialVersionUID() {  
 return *serialVersionUID*;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public Collection<? extends GrantedAuthority> getRoles() {  
 return roles;  
 }  
  
 public void setRoles(Collection<? extends GrantedAuthority> roles) {  
 this.roles = roles;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
  
  
 @Override  
 public String getUsername() {  
 return username;  
 }  
  
 @Override  
 public String getPassword() {  
 return password;  
 }  
  
 @Override  
 public Collection<? extends GrantedAuthority> getAuthorities() {  
 return roles;  
 }  
  
 @Override  
 public boolean isAccountNonExpired() {  
 return true;  
 }  
  
 @Override  
 public boolean isAccountNonLocked() {  
 return true;  
 }  
  
 @Override  
 public boolean isCredentialsNonExpired() {  
 return true;  
 }  
  
 @Override  
 public boolean isEnabled() {  
 return true;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 UserPrinciple user = (UserPrinciple) o;  
 return Objects.*equals*(id, user.id);  
 }  
}

* Tạo class: **UserDetailsServiceIMPL** (Class này có nhiệm vụ tìm **username** có tồn tại trong database hay không nếu tồn tại thì thực hiện triển khai build **user** trên hệ thống security) thông qua:

return UserPrinciple.*build*(user);

package chinh.nguyen.security.userprincipal;  
  
import chinh.nguyen.model.User;  
import chinh.nguyen.repository.IUserRepository;  
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 javax.transaction.Transactional;  
@Service  
public class UserDetailsServiceIMPL implements UserDetailsService {  
 @Autowired  
 IUserRepository userRepository;  
  
 @Override  
 @Transactional  
 public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {  
  
 User user = userRepository.findByUsername(username).orElseThrow(  
 () -> new UsernameNotFoundException("User Not Found with -> username or email : " + username));  
  
 return UserPrinciple.*build*(user);  
 }  
}

* Tạo class: **JwtProvider (Class này tạo ra với nhiệm vụ mã hóa thông tin đăng nhập thành chuỗi TOKEN – Kiểm tra TOKEN có hợp lệ hay không? – Lấy thông tin người dùng từ chính chuỗi TOKEN đã mã hóa –** *CHÍNH NĐ Hà Nội ngày cuối thu – lá vàng rơi lả tả…*

package chinh.nguyen.security.jwt;  
  
import chinh.nguyen.security.userprincipal.UserPrinciple;  
import io.jsonwebtoken.\*;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.security.core.Authentication;  
import org.springframework.stereotype.Component;  
  
import java.util.Date;  
  
@Component  
public class JwtProvider {  
 //LỚP Logger giúp ta ghi log bắt các trường hợp ngoại lệ  
 private static final Logger *logger* = LoggerFactory.*getLogger*(JwtProvider.class);  
 private String jwtSecret="jwt.chinh.nguyen";

private int jwtExpiration = 86400; //Set thời gian sống cho token  
//HÀM TIẾN HÀNH MÃ HÓA USER THÀNH CHUỖI TOKEN -> SẼ ĐƯỢC GỌI TẠI API LOGIN TRÊN CONTROLLER  
 public String generateJwtToken(Authentication authentication) {  
  
 UserPrinciple userPrincipal = (UserPrinciple) authentication.getPrincipal();  
  
 return Jwts.*builder*()  
 .setSubject((userPrincipal.getUsername()))  
 .setIssuedAt(new Date())  
 .setExpiration(new Date(new Date().getTime() + jwtExpiration \* 1000))  
 .signWith(SignatureAlgorithm.*HS512*, jwtSecret)  
 .compact();  
 }  
//HÀM TIẾN HAH KIỂM TRA TÍNH HỢP LỆ CỦA TOKEN ĐANG ĐĂNG NHẬP  
 public boolean validateJwtToken(String authToken) {  
 try {  
 Jwts.*parser*().setSigningKey(jwtSecret).parseClaimsJws(authToken);  
 return true;  
 } catch (SignatureException e) {  
 *logger*.error("Invalid JWT signature -> Message: {} ", e);  
 } catch (MalformedJwtException e) {  
 *logger*.error("Invalid JWT token -> Message: {}", e);  
 } catch (ExpiredJwtException e) {  
 *logger*.error("Expired JWT token -> Message: {}", e);  
 } catch (UnsupportedJwtException e) {  
 *logger*.error("Unsupported JWT token -> Message: {}", e);  
 } catch (IllegalArgumentException e) {  
 *logger*.error("JWT claims string is empty -> Message: {}", e);  
 }  
  
 return false;  
 }  
 //Lay lai thong nguoi dung tu chinh Token tao ra  
 public String getUserNameFromJwtToken(String token) {  
  
 String userName = Jwts.*parser*()  
 .setSigningKey(jwtSecret)  
 .parseClaimsJws(token)  
 .getBody().getSubject();  
 return userName;  
 }  
}

* Tạo class: **JwtAuthTokenFilter** (Class này tạo ra với mục đích triển khai lớp **JwtProvider =>** Tiến hành kiểm tra xem TOKEN có hợp lệ hay không? Nếu hợp lệ sẽ set User đó vào trong **authentication**
* Tạo class: **JwtAuthEntryPoint** => Lớp này có tác dụng báo ngoại lệ liên quan tới vấn đề của Authentication thất bại.

package chinh.nguyen.security.jwt;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.security.core.AuthenticationException;  
import org.springframework.security.web.AuthenticationEntryPoint;  
  
import javax.servlet.ServletException;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
  
public class JwtAuthEntryPoint implements AuthenticationEntryPoint {  
 private static final Logger *logger* = LoggerFactory.*getLogger*(JwtAuthEntryPoint.class);  
  
 @Override  
 public void commence(HttpServletRequest request,  
 HttpServletResponse response,  
 AuthenticationException e)  
 throws IOException, ServletException {  
  
 *logger*.error("Unauthorized error. Message - {}", e.getMessage());  
 response.sendError(HttpServletResponse.*SC\_UNAUTHORIZED*, "Error -> Unauthorized");  
 }  
}

* Tạo class: **SecurityConfig** trong package: security (Lớp này tạo ra mục đích phân tích lấy ra username và mật khẩu đã encode config các đường dẫn và tiêm các sự phụ thuộc của các lớp trong package: **jwt**

package chinh.nguyen.security;  
  
import chinh.nguyen.security.jwt.JwtAuthEntryPoint;  
import chinh.nguyen.security.jwt.JwtAuthTokenFilter;  
import chinh.nguyen.security.userprincipal.UserDetailsServiceIMPL;  
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.annotation.authentication.builders.AuthenticationManagerBuilder;  
import org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;  
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.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;  
  
@Configuration  
@EnableWebSecurity  
@EnableGlobalMethodSecurity(prePostEnabled = true)  
public class SecurityConfig extends WebSecurityConfigurerAdapter {  
 @Autowired  
 UserDetailsServiceIMPL userDetailsService;  
  
 @Autowired  
 private JwtAuthEntryPoint unauthorizedHandler;  
  
 @Bean  
 public JwtAuthTokenFilter authenticationJwtTokenFilter() {  
 return new JwtAuthTokenFilter();  
 }  
  
 @Override  
 public void configure(AuthenticationManagerBuilder authenticationManagerBuilder) throws Exception {  
 authenticationManagerBuilder  
 .userDetailsService(userDetailsService)  
 .passwordEncoder(passwordEncoder());  
 }  
  
 @Bean  
 @Override  
 public AuthenticationManager authenticationManagerBean() throws Exception {  
 return super.authenticationManagerBean();  
 }  
  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http.cors().and().csrf().disable().  
 authorizeRequests()  
 .antMatchers("/api/auth/\*\*").permitAll()  
 .anyRequest().authenticated()  
 .and()  
 .exceptionHandling()  
 .authenticationEntryPoint(unauthorizedHandler).and()  
 .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.*STATELESS*);  
  
 http.addFilterBefore(authenticationJwtTokenFilter(), UsernamePasswordAuthenticationFilter.class);  
 }  
}

* Thêm code sau vào AppConfig: Scan luồng chạy xuống package: **security**

@ComponentScan("chinh.nguyen.security")

* Quay lại package: **dto.request** => Tạo class: **SignInForm** để hứng dữ liệu từ FrontEnd

package chinh.nguyen.dto.request;  
  
public class SignInForm {  
 private String username;  
 private String password;  
  
 public SignInForm() {  
 }  
  
 public SignInForm(String username, String password) {  
 this.username = username;  
 this.password = password;  
 }  
  
 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;  
 }  
}

* Tạo class: **JwtResponse** trong package: **response** (Class này tạo ra nhằm mục đích định nghĩa kiểu dữ liệu trả về sau khi login trong này bao gồm các trường mà chúng ta muốn lấy ra từ User login bao gồm cả chuỗi TOKEN) ***/\* Chính NĐ Hà Nội mưa rơi rả rích \*/***

package chinh.nguyen.dto.response;  
  
import org.springframework.security.core.GrantedAuthority;  
  
import java.util.Collection;  
  
public class JwtResponse {  
 private Long id;  
 private String token;  
 private String type = "Bearer";  
 private String name;  
 private String email;  
  
 private Collection<? extends GrantedAuthority> roles;  
  
  
  
 public JwtResponse(String jwt, Long id, String name, String email, Collection<? extends GrantedAuthority> authorities) {  
 this.token = jwt;  
 this.id = id;  
 this.name = name;  
 this.email = email;  
 this.roles = authorities;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getToken() {  
 return token;  
 }  
  
 public void setToken(String token) {  
 this.token = token;  
 }  
  
 public String getType() {  
 return type;  
 }  
  
 public void setType(String type) {  
 this.type = type;  
 }  
  
 public void setRoles(Collection<? extends GrantedAuthority> roles) {  
 this.roles = roles;  
 }  
  
  
  
 public Collection<? extends GrantedAuthority> getRoles() {  
 return roles;  
 }  
}

* Quay lại package: **controller** => Thêm các đoạn code liên quan tới API login vào class: **AuthController** như sau:

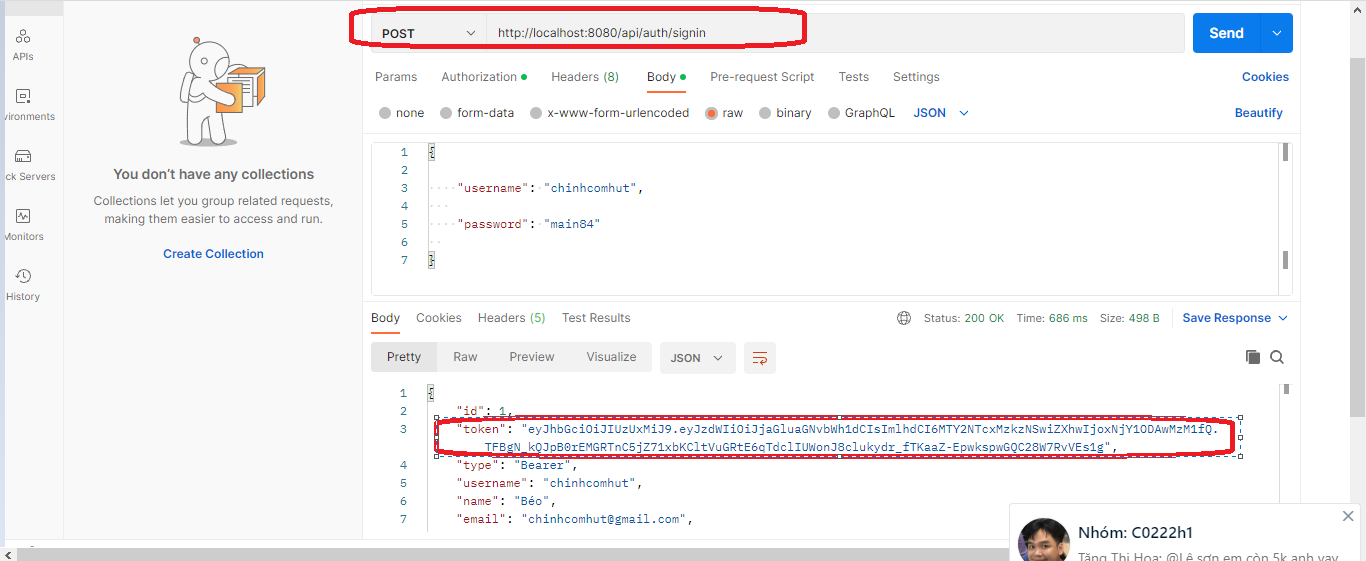
@Autowired  
JwtProvider jwtProvider;  
@Autowired  
AuthenticationManager authenticationManager;

@PostMapping("/signin")  
public ResponseEntity<?> authenticateUser(@Valid @RequestBody SignInForm loginRequest) {  
 Authentication authentication = authenticationManager.authenticate(  
 new UsernamePasswordAuthenticationToken(loginRequest.getUsername(), loginRequest.getPassword()));  
  
 SecurityContextHolder.*getContext*().setAuthentication(authentication);  
  
 String jwt = jwtProvider.generateJwtToken(authentication);  
 UserPrinciple userDetails = (UserPrinciple) authentication.getPrincipal();  
  
 return ResponseEntity.*ok*(new JwtResponse(jwt,  
 userDetails.getId() , userDetails.getName(), userDetails.getEmail() ,  
 userDetails.getAuthorities()  
 ));  
}

* **Chú ý: Bài thực hành này tương đối dài nên anh em kiểm tra lại tất cả cá @Annotation ở đầu các class nhé, có thể trong quá trình tổng hợp anh làm thiếu 1 @Annotation nào đó khiến project thực hành không chạy được. Anh em check lại theo link GIT tham khảo ở bên dưới.**

Sau khi làm đầy đủ các bước trên ta thực hiện bước test API login:

<http://localhost:8080/api/auth/signin>



### <https://github.com/nguyendongminhtri/MD4-JWT-SpringMVC-14-10-2022.git>