**2022년도 졸업프로젝트[Back-end]**

**팀이름: 황제펭귄**

주제: TLS를 구현을 통한 홈페이지 제작 및 보안 채널 로그인 기능 구현

|  |  |  |  |
| --- | --- | --- | --- |
| **작성자** | 이유민 | **작성일자** | 2022.04.27 |

* **금일 진행 상황**

|  |  |
| --- | --- |
| **진행 상황** | **비고** |
| **질문 및 진행방식에 대한 논의**  **진행 중**  **완료**   1. **기존 프로젝트에서 Front-End의 확인에서 사용된 어노테이션을 제거.** -@Notnull,@NotEmpty annotation 삭제:빈칸 검증은 프론트측에서 진행.   -UserRepositoryimp 삭제.  -BaseTimeEntity에 @JsonFormat 이용.  -common package 이름:users.  -web package 이름 변경:controller.   1. **회원가입 등록 과정 JSON 데이터 전송 POSTMAN으로 확인.**   문제점:Spring Security 동작으로 인해 권한이 없는 사용자가 보내는 JSON 데이터 수신 불가.  **해결 방법: spring Security의존성 잠시 제거**  plugins {  id 'org.springframework.boot' version '2.6.6'  id 'io.spring.dependency-management' version '1.0.11.RELEASE'  id 'java' }  group = 'com.EmperorPenguin' version = '0.0.1-SNAPSHOT' sourceCompatibility = '11'  configurations {  compileOnly {  extendsFrom annotationProcessor  } }  repositories {  mavenCentral() }  dependencies {  implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  //implementation 'org.springframework.boot:spring-boot-starter-oauth2-client'  //implementation 'org.springframework.boot:spring-boot-starter-security'  implementation 'org.springframework.boot:spring-boot-starter-web'  //implementation 'org.thymeleaf.extras:thymeleaf-extras-springsecurity5'  compileOnly 'org.projectlombok:lombok'  runtimeOnly 'com.h2database:h2'  annotationProcessor 'org.projectlombok:lombok'  testImplementation 'org.springframework.boot:spring-boot-starter-test'  //testImplementation 'org.springframework.security:spring-security-test' }  tasks.named('test') {  useJUnitPlatform() }  **수정 후 controller 메커니즘의 동작:**  package com.EmperorPenguin.SangmyungBank.api.users.add.web;   import com.EmperorPenguin.SangmyungBank.api.users.add.domain.User.User; import com.EmperorPenguin.SangmyungBank.api.users.add.service.UserService; import lombok.RequiredArgsConstructor; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*;  import java.time.LocalDateTime;  @RequiredArgsConstructor @RequestMapping("/users") @RestController public class UserController {   private final UserService userService;   @PostMapping(path="/add")  public ResponseEntity<HttpStatus> addUser(@RequestBody User user)  {  user.setCreatedDate(LocalDateTime.*now*());  user.setModifiedDate(LocalDateTime.*now*());  User savedUser = userService.register(user);  if (savedUser != null)  return ResponseEntity.*ok*(HttpStatus.*OK*);  return ResponseEntity.*ok*(HttpStatus.*BAD\_REQUEST*);  }   1. **로그인 기능 제작**   **Package 구조**  -각 기능에 해당하는 packege구조로 설계  **텍스트이(가) 표시된 사진  자동 생성된 설명**  **Controller의 구조와 실제 동작**  package com.EmperorPenguin.SangmyungBank.api.users.login.controller;  import com.EmperorPenguin.SangmyungBank.api.users.login.domain.loginForm.LoginForm; import com.EmperorPenguin.SangmyungBank.api.users.login.service.UserAuthService; import lombok.RequiredArgsConstructor; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*;  @RequiredArgsConstructor @RequestMapping("/users") @RestController public class UserAuthController {   private final UserAuthService userAuthService;   @GetMapping(path="/login")  public ResponseEntity<HttpStatus> authUser(@RequestBody LoginForm loginForm)  {  boolean auth = userAuthService.authenticate(loginForm);  if (auth == true)  return ResponseEntity.*ok*(HttpStatus.*OK*);  else  return ResponseEntity.*ok*(HttpStatus.*BAD\_REQUEST*);  } }  **Postman 동작 확인**  텍스트이(가) 표시된 사진  자동 생성된 설명  **Compile Complete**  **텍스트이(가) 표시된 사진  자동 생성된 설명**   1. **Bug Fix** AddUserService에서 entity였던 유저의 속성이 unique email,phonenumber에 대한 unique check가 없어 발생했던 http request 500번 error 수정.   package com.EmperorPenguin.SangmyungBank.api.users.add.service;  import com.EmperorPenguin.SangmyungBank.api.users.add.domain.User.Role; import com.EmperorPenguin.SangmyungBank.api.users.add.domain.User.User; import com.EmperorPenguin.SangmyungBank.api.users.add.domain.repository.UserAddRepository; import org.springframework.stereotype.Service;  import java.util.Optional; import javax.transaction.Transactional;  @Transactional @Service public class UserAddService {   private final UserAddRepository userAddRepository; // private final PasswordEncoder passwordEncoder;   public UserAddService(UserAddRepository userAddRepository ) {  this.userAddRepository = userAddRepository;  }   // Id를 통해 DB에 있는지 검색을 하고 있다면 존재한다는 error를 발생  // 아니라면 등록과정을 진행한다.  public boolean ValidateDuplicateUser(User saveUser) {  User user = userAddRepository.findByLoginId(saveUser.getLoginId())  .orElse(null);   if(user == null & userAddRepository.RepeatEmailCheck(saveUser.getEmail())  & userAddRepository.RepeatPhoneNumber(saveUser.getPhoneNumber()))  return true;  else  return false;  }   /\*  회원 등록을 위한 작업  \*/  public User register(User savedUser)  {  if(ValidateDuplicateUser(savedUser))  {  // String encodedPassword = passwordEncoder.encode(password);  savedUser.setRole(Role.*USER*);  return userAddRepository.save(savedUser);  }  else  return null;  }   public Optional<User> findOne(String userId) {  return userAddRepository.findByLoginId(userId);  } }  **진행 예정** |  |
| * **특이사항 / 협업 사항** | |