**2022년도 졸업프로젝트[Back-end]**

**팀이름: 황제펭귄**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **작성자** | 정경재 | **작성일자** | 2022.02.23 |

* **금일 진행 상황**

|  |  |
| --- | --- |
| **진행 상황** | **비고** |
| **질문 및 진행방식에 대한 논의**  **진행 중**   1. 스프링 스터디 진행.   **완료**  회원 관리 예제 – 백엔드 개발   1. 비즈니스 요구사항 정리   데이터: 회원ID, 이름  기능: 회원 등록, 조회  아직 데이터 저장소가 선정되지 않았다고 가정    컨트롤러: 웹 MVC의 컨트롤러 역할  서비스: 핵심 비즈니스 로직 구현  리포지토리: 데이터베이스에 접근, 도메인 객체를 DB에 저장하고 관리  도메인: 비즈니스 도메인 객체    아직 데이터 저장소가 선정되지 않음, 우선 인터페이스로 구현 클래스를 변경할 수 있도록 설계  [데이터 저장소는 RDB, NoSQL 등등 다양한 저장소를 고민중인 상황으로 가정]  개발을 진행하기 위해서 초기 개발 단계에서는 구현체로 가벼운 메모리 기반의 데이터 저장소 사용   1. 회원 도메인과 리포지토리 만들기   회원 객체 생성  package hello.hellospring.repository;  public class Member {    private Long id;  private String name;  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;  }  }  회원 리포지토리 인터페이스 생성  package hello.hellospring.domain;  public class Member {    private Long id;  private String name;  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;  }  }  회원 리포지토리 메모리 구현체 생성  package hello.hellospring.repository;  import hello.hellospring.domain.Member;  import java.util.\*;  public class MemoryMemberRepository implements MemberRepository {  private static Map<Long, Member> store = new HashMap<>();  private static long sequence = 0L;  @Override  public Member save(Member member) {  member.setId(++sequence);  store.put(member.getId(), member);  return member;  }  @Override  public Optional<Member> findById(Long id) {  return Optional.ofNullable(store.get(id));  }  @Override  public List<Member> findAll() {  return new ArrayList<>(store.values()); }  @Override  public Optional<Member> findByName(String name) {  return store.values().stream()  .filter(member -> member.getName().equals(name))  .findAny();  }  public void clearStore() {  store.clear();  }  }   1. 회원 리포지토리 테스트 케이스 작성   테스트는 서로 의존관계가 없어야한다.  test/java 하위폴더에 MemoryMemberRepositoryTest 클래스생성  package hello.hellospring.repository;  import hello.hellospring.domain.Member;  import org.junit.jupiter.api.AfterEach;  import org.junit.jupiter.api.Test;  import java.util.List;  import java.util.Optional;  import static org.assertj.core.api.Assertions.\*;class MemoryMemberRepositoryTest {  MemoryMemberRepository repository = new MemoryMemberRepository();  @AfterEach  public void afterEach() {  repository.clearStore();  } // 각각의 테스트의 결과가 남아 다른 테스트를할 때 영향을 줄 수있으므로 스토어를 클리어하는 기능을 추가한다.  @Test  public void save() {  //given  Member member = new Member();  member.setName("spring");  //when  repository.save(member);  //then  Member result = repository.findById(member.getId()).get();  assertThat(result).isEqualTo(member);  }  @Test  public void findByName() {  //given  Member member1 = new Member();  member1.setName("spring1");  repository.save(member1);  Member member2 = new Member();  member2.setName("spring2");  repository.save(member2);  //when  Member result = repository.findByName("spring1").get();  //then assertThat(result).isEqualTo(member1);  }  @Test  public void findAll() {  //given  Member member1 = new Member();  member1.setName("spring1");  repository.save(member1);  Member member2 = new Member();  member2.setName("spring2");  repository.save(member2);  //when  List<Member> result = repository.findAll();  //then  assertThat(result.size()).isEqualTo(2);  }  }   1. 회원 서비스 개발   package hello.hellospring.service;  import hello.hellospring.domain.Member;  import hello.hellospring.repository.MemberRepository;  import java.util.List;import java.util.Optional;  public class MemberService {  private final MemberRepository memberRepository = new  MemoryMemberRepository();  /\*\*  \* 회원가입  \*/  public Long join(Member member) {  validateDuplicateMember(member); //중복 회원 검증  memberRepository.save(member);  return member.getId();  }  private void validateDuplicateMember(Member member) {  memberRepository.findByName(member.getName())  .ifPresent(m -> {  throw new IllegalStateException("이미 존재하는 회원입니다.");  });  }  /\*\*  \* 전체 회원 조회  \*/  public List<Member> findMembers() {  return memberRepository.findAll();  }  public Optional<Member> findOne(Long memberId) {  return memberRepository.findById(memberId);  }  }   1. 회원 서비스 테스트   package hello.hellospring.service;  import hello.hellospring.domain.Member;  import hello.hellospring.repository.MemoryMemberRepository;  import org.junit.jupiter.api.BeforeEach;  import org.junit.jupiter.api.Test;  import static org.assertj.core.api.Assertions.\*;  import static org.junit.jupiter.api.Assertions.\*;  class MemberServiceTest { MemberService memberService;  MemoryMemberRepository memberRepository;  @BeforeEach // 각 테스트전에 실행됨  public void beforeEach() {  memberRepository = new MemoryMemberRepository();  memberService = new MemberService(memberRepository);  }  @AfterEach  public void afterEach() {  memberRepository.clearStore();  }  @Test  public void 회원가입() throws Exception {  //Given  Member member = new Member();  member.setName("hello");  //When  Long saveId = memberService.join(member);  //Then  Member findMember = memberRepository.findById(saveId).get();  assertEquals(member.getName(), findMember.getName());  }  @Test  public void 중복\_회원\_예외() throws Exception {  //Given  Member member1 = new Member();  member1.setName("spring");  Member member2 = new Member();  member2.setName("spring"); //When  memberService.join(member1);  IllegalStateException e = assertThrows(IllegalStateException.class,  () -> memberService.join(member2));//예외가 발생해야 한다.  assertThat(e.getMessage()).isEqualTo("이미 존재하는 회원입니다.");  }  }  **진행 예정**   1. **3/1일 8시 Back-End 스터디** |  |
| * **특이사항 / 협업 사항**   매주 화, 수, 목 7시30부터 10시 30분까지 유동적으로 하기. | |