**2022년도 졸업프로젝트**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **작성자** | 이유민 | **작성일자** | 2022.05.04 |

* **금일 진행 상황**

|  |  |
| --- | --- |
| **진행 상황** | **비고** |
| **질문 및 진행방식에 대한 논의**  **진행 중**  **완료**   1. **이벤트 게시판 기능 구현**   **-Package 구조**    **-EventController**  package com.EmperorPenguin.SangmyungBank.api.event.controller;  import com.EmperorPenguin.SangmyungBank.api.event.model.Event; import com.EmperorPenguin.SangmyungBank.api.event.service.EventService; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*;  import java.time.LocalDateTime; import java.util.List; import java.util.Map;  @RestController @RequestMapping("/cont") public class EventController {   @Autowired  private EventService eventService;   // create event  @PostMapping("/event")  public Event createEvent(@RequestBody Event event)  {  event.setCreatedDate(LocalDateTime.*now*());  return eventService.createEvent(event);  }   // list all events  @GetMapping("/event")  public List<Event> listAllEvents() {  return eventService.listAllEvents();  }   // get event by id  @GetMapping("/event/{id}")  public ResponseEntity<Event> getEventById(@PathVariable Long id) {  return eventService.getEventById(id);  }   @PutMapping("/event/{id}")  public ResponseEntity<Event> updateEvent(  @PathVariable Long id, @RequestBody Event eventDetails) {  return eventService.updateEvent(id, eventDetails);  }   @DeleteMapping("/event/{id}")  public ResponseEntity<Map<String, Boolean>> deleteEvent(@PathVariable Long id) {  return eventService.deleteEvent(id);  } }  **-Event(Entity)**  package com.EmperorPenguin.SangmyungBank.api.event.model;  import com.fasterxml.jackson.annotation.JsonFormat; import lombok.Getter; import lombok.NoArgsConstructor; import lombok.Setter; import org.springframework.data.annotation.CreatedDate;  import javax.persistence.\*; import java.time.LocalDateTime;  @Entity @Table(name = "Event") @Getter @Setter @NoArgsConstructor public class Event {  @Id  @GeneratedValue(strategy = GenerationType.*IDENTITY*)  private Long id;   private String title;   private String content;   @JsonFormat(pattern = "yyyy-MM-dd HH:mm")  @CreatedDate  private LocalDateTime createdDate; }  **-EventRepository**  package com.EmperorPenguin.SangmyungBank.api.event.repository;  import com.EmperorPenguin.SangmyungBank.api.event.model.Event; import org.springframework.data.jpa.repository.JpaRepository;  public interface EventRepository extends JpaRepository<Event, Long> { }  **-EventService**  package com.EmperorPenguin.SangmyungBank.api.event.service;  import com.EmperorPenguin.SangmyungBank.api.event.model.Event; import com.EmperorPenguin.SangmyungBank.api.event.repository.EventRepository; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.boot.context.config.ConfigDataResourceNotFoundException; import org.springframework.data.crossstore.ChangeSetPersister; import org.springframework.http.ResponseEntity; import org.springframework.stereotype.Service; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestBody;  import java.util.HashMap; import java.util.List; import java.util.Map;  @Service public class EventService {   @Autowired  private EventRepository eventRepository;   public Event createEvent(@RequestBody Event event) {  return eventRepository.save(event);  }   public List<Event> listAllEvents() {  return eventRepository.findAll();  }   public ResponseEntity<Event> getEventById(@PathVariable Long id) {  Event event = eventRepository.findById(id)  .orElse(null);   return ResponseEntity.*ok*(event);  }   public ResponseEntity<Event> updateEvent(@PathVariable Long id, @RequestBody Event eventDetails){  Event event = eventRepository.findById(id)  .orElse(null);   event.setTitle(eventDetails.getTitle());  event.setContent(eventDetails.getContent());   Event updateEvent = eventRepository.save(event);  return ResponseEntity.*ok*(updateEvent);  }   public ResponseEntity<Map<String, Boolean>> deleteEvent(@PathVariable Long id) {  Event event = eventRepository.findById(id)  .orElse(null);   eventRepository.delete(event);  Map <String, Boolean> response = new HashMap<>();  response.put("deleted", Boolean.*TRUE*);  return ResponseEntity.*ok*(response);  }  }  **-Postman 동작확인**  **텍스트이(가) 표시된 사진  자동 생성된 설명**  **텍스트이(가) 표시된 사진  자동 생성된 설명**  **텍스트이(가) 표시된 사진  자동 생성된 설명**  **텍스트이(가) 표시된 사진  자동 생성된 설명**  **텍스트이(가) 표시된 사진  자동 생성된 설명**   1. **새소식 게시판 기능 구현**   **-이벤트 게시판과 코드 유사.**   1. **계좌 생성 기능 구현(완료)**   **-Account(Entity)**  문제해결: private Long userId;→ private User userId; user class의 userId를 Foriegn Key(FK)로 참조.  package com.EmperorPenguin.SangmyungBank.api.users.accounts.add.domain.account;  import com.EmperorPenguin.SangmyungBank.api.users.add.domain.User.User; import lombok.Builder; import lombok.Getter; import lombok.NoArgsConstructor; import lombok.Setter;  import javax.persistence.\*;  @Getter @Setter @Entity @NoArgsConstructor public class Account {   @ManyToOne(targetEntity = User.class, fetch = FetchType.*EAGER*)  @JoinColumn(name="userId")  private User userId;   @Id  @GeneratedValue(strategy = GenerationType.*IDENTITY*)  private Long accountNumber;   @Column(columnDefinition = "text",nullable = false)  private String accountPassword;   @Column(nullable = false)  private Long balance;   @Builder  public Account(User userId, String accountPassword, Long balance)  {  this.userId = userId;  this.accountPassword = accountPassword;  this.balance = balance;  } }   1. **이체 기능 구현**   **-Package 구조**    **-TransactionController**  package com.EmperorPenguin.SangmyungBank.api.users.accounts.transaction.controller;  import com.EmperorPenguin.SangmyungBank.api.users.accounts.transaction.domain.transactionForm.TransactionForm; import com.EmperorPenguin.SangmyungBank.api.users.accounts.transaction.service.AccountTransactionService; import lombok.RequiredArgsConstructor; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; 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;  @RequiredArgsConstructor @RequestMapping("/users/accounts") @RestController public class TransactionController {   private final AccountTransactionService accountTransactionService;   @PostMapping(path="/transaction")  public ResponseEntity<HttpStatus> sendBalance(@RequestBody TransactionForm transactionForm){  boolean check = accountTransactionService.transaction(transactionForm);  if(!check)  return ResponseEntity.*ok*(HttpStatus.*BAD\_REQUEST*);  else  return ResponseEntity.*ok*(HttpStatus.*OK*);  } }  **-AccountTransactionRepository**  package com.EmperorPenguin.SangmyungBank.api.users.accounts.transaction.domain.repository;  import com.EmperorPenguin.SangmyungBank.api.users.accounts.add.domain.account.Account; import com.EmperorPenguin.SangmyungBank.api.users.accounts.transaction.domain.transactionForm.TransactionForm; import org.springframework.stereotype.Repository;  import javax.persistence.EntityManager; import java.util.List; import java.util.Optional;  @Repository public class AccountTransactionRepository {   private final EntityManager em;   public AccountTransactionRepository(EntityManager em) {  this.em = em;  }   public Optional<Account> findByAccountNumber(Long accountNumber)  {  List<Account> result = em.createQuery("select m from Account m where m.accountNumber = :accountNumber", Account.class)  .setParameter("accountNumber",accountNumber)  .getResultList();  return result.stream().findAny();  }   public void updateMyBalance(TransactionForm transactionForm)  {  em.createQuery("update Account m set m.balance = m.balance - :balance where m.accountNumber =:accountNumber")  .setParameter("balance", transactionForm.getBalance())  .setParameter("accountNumber", transactionForm.getMyAccountNumber())  .executeUpdate();  }   public void sendBalance(TransactionForm transactionForm)  {  em.createQuery("update Account m set m.balance = m.balance + :balance where m.accountNumber =:accountNumber")  .setParameter("balance", transactionForm.getBalance())  .setParameter("accountNumber", transactionForm.getSendAccountNumber())  .executeUpdate();  } }  **-TransactionForm**  package com.EmperorPenguin.SangmyungBank.api.users.accounts.transaction.domain.transactionForm;  import lombok.Builder; import lombok.Getter; import lombok.NoArgsConstructor; import lombok.Setter;  @Getter @Setter @Builder @NoArgsConstructor public class TransactionForm {   private Long myAccountNumber;  private Long sendAccountNumber;  private String accountPassword;  private Long balance;   @Builder  public TransactionForm(Long myAccountNumber, Long sendAccountNumber ,String accountPassword, Long balance)  {  this.myAccountNumber = myAccountNumber;  this.sendAccountNumber = sendAccountNumber;  this.accountPassword = accountPassword;  this.balance = balance;  }  }  **-TransactionService**  package com.EmperorPenguin.SangmyungBank.api.users.accounts.transaction.service;  import com.EmperorPenguin.SangmyungBank.api.users.accounts.add.domain.account.Account;  import com.EmperorPenguin.SangmyungBank.api.users.accounts.transaction.domain.transactionForm.TransactionForm; import com.EmperorPenguin.SangmyungBank.api.users.accounts.transaction.domain.repository.AccountTransactionRepository; import lombok.RequiredArgsConstructor; import org.springframework.stereotype.Service;  import javax.transaction.Transactional;  @RequiredArgsConstructor @Transactional @Service public class AccountTransactionService {   private final AccountTransactionRepository accountTransactionRepository;   public boolean transaction (TransactionForm transactionForm)  {  Account dbAccount = accountTransactionRepository.findByAccountNumber(transactionForm.getMyAccountNumber())  .orElse(null);  if(dbAccount == null)  return false;  else  if(transactionForm.getAccountPassword().equals(dbAccount.getAccountPassword()))  if ( dbAccount.getBalance() >= transactionForm.getBalance())  {  accountTransactionRepository.updateMyBalance(transactionForm);  accountTransactionRepository.sendBalance(transactionForm);  return true;  }  else  return false;  else  return false;   } }  **-TransactionTest**  package com.EmperorPenguin.SangmyungBank.api.user.account.transaction;  import com.EmperorPenguin.SangmyungBank.api.users.accounts.transaction.domain.transactionForm.TransactionForm; import com.EmperorPenguin.SangmyungBank.api.users.accounts.transaction.service.AccountTransactionService; import org.junit.jupiter.api.Test; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.boot.test.context.SpringBootTest;  import javax.transaction.Transactional;  import static org.assertj.core.api.AssertionsForClassTypes.*assertThat*;  @Transactional @SpringBootTest public class transactionTest {  @Autowired  AccountTransactionService accountTransactionService;  @Test  void 계좌이체(){  //given  Long myAccountNum = 2L;  String password = "1344";  Long target = 1L;  Long balance = 100L;   // when  TransactionForm transactionForm = TransactionForm.*builder*()  .myAccountNumber(myAccountNum)  .accountPassword(password)  .sendAccountNumber(target)  .balance(balance)  .build();  //then  *assertThat*(true).isEqualTo(accountTransactionService.transaction(transactionForm));   }  }  **진행 예정** | re |
| * **특이사항 / 협업 사항** | |