**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. **FrontForm 생성**   package com.EmperorPenguin.SangmyungBank.api.users.login.domain.frontForm;  import lombok.Builder; import lombok.Getter; import lombok.NoArgsConstructor; import lombok.Setter;  @Getter @Setter @Builder @NoArgsConstructor public class FrontForm {   private String loginId;  private String userName;   @Builder  public FrontForm(String loginId, String userName) {  this.loginId = loginId;  this.userName = userName;  } }   1. **UserloginController를 FrontForm을 적용해서 내부 로직 변경.**   package com.EmperorPenguin.SangmyungBank.api.users.login.controller;  import com.EmperorPenguin.SangmyungBank.api.users.login.domain.frontForm.FrontForm; import com.EmperorPenguin.SangmyungBank.api.users.login.domain.loginForm.LoginForm; import com.EmperorPenguin.SangmyungBank.api.users.login.service.UserLoginService; import lombok.RequiredArgsConstructor; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*; import com.EmperorPenguin.SangmyungBank.api.users.login.service.SessionService; import com.EmperorPenguin.SangmyungBank.api.users.add.domain.User.User; import javax.servlet.http.HttpServletResponse;   @RequiredArgsConstructor @RequestMapping("/users") @RestController public class UserLoginController {   private final UserLoginService userLoginService;  private final SessionService sessionService;    @RequestMapping (path="/login",method={RequestMethod.*GET*,RequestMethod.*POST*})  public ResponseEntity<FrontForm> authUser(@RequestBody LoginForm loginForm,HttpServletResponse response)  {  User auth = userLoginService.authenticate(loginForm);  if (auth == null) {  return ResponseEntity.*status*(HttpStatus.*NOT\_FOUND*)  .body(null);  }  else {  sessionService.CreateSession(auth, response);  FrontForm frontForm = FrontForm.*builder*()  .loginId(auth.getLoginId())  .userName(auth.getName())  .build();  return ResponseEntity.*status*(HttpStatus.*OK*)  .body(frontForm);  }   }   1. **HomeController에 Logout function 추가**   package com.EmperorPenguin.SangmyungBank.api.home;  import com.EmperorPenguin.SangmyungBank.api.users.login.service.SessionService; 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.RequestMapping; import org.springframework.web.bind.annotation.RestController;  import javax.servlet.http.HttpServletRequest;  @RequiredArgsConstructor @RequestMapping("") @RestController public class HomeContoller {   private final SessionService sessionService;   @PostMapping("/logout")  public ResponseEntity<HttpStatus> logout(HttpServletRequest request){  sessionService.expire(request);  return ResponseEntity.*ok*(HttpStatus.*OK*);  }  }   1. **Postman을 통한 Logout test**   -세션이 만료되어 Connection이 close로 바뀌고, 쿠키가 사라지는 것 확인.   1. **계좌 생성(미완료)**   문제점:Postman internal server 500 error, H2 Database에 테이블에 User\_Id\_User\_Id가 생성되는 문제 발생.  **-Package 구조**    **-Account(Entity)**  package com.EmperorPenguin.SangmyungBank.api.users.account.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="user\_userId")  private Long userId;   @Id  @GeneratedValue(strategy = GenerationType.*IDENTITY*)  private Long accountNumber;   @Column(columnDefinition = "text",nullable = false)  private String accountPassword;   @Column(columnDefinition = "Bigint", nullable = true)  private Long balance;   @Builder  public Account(Long userId, String accountPassword, Long balance)  {  this.userId = userId;  this.accountPassword = accountPassword;  this.balance = balance;  } }  **-AccountForm**  package com.EmperorPenguin.SangmyungBank.api.users.account.domain.accountForm.controller;  import com.EmperorPenguin.SangmyungBank.api.users.account.domain.account.Account; import com.EmperorPenguin.SangmyungBank.api.users.account.domain.accountForm.AccountForm; import com.EmperorPenguin.SangmyungBank.api.users.account.service.AccountAddService; 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") @RestController public class AccountAddController {   private final AccountAddService accountAddService;   @PostMapping(path="/accounts/add")  public ResponseEntity<HttpStatus> addAccount(@RequestBody AccountForm accountForm){  Account resultAccount = accountAddService.register(accountForm);  if(resultAccount == null)  return ResponseEntity.*ok*(HttpStatus.*BAD\_REQUEST*);  else  return ResponseEntity.*ok*(HttpStatus.*OK*);  }  }  **-AccountAddController**  package com.EmperorPenguin.SangmyungBank.api.users.account.domain.accountForm.controller;  import com.EmperorPenguin.SangmyungBank.api.users.account.domain.account.Account; import com.EmperorPenguin.SangmyungBank.api.users.account.domain.accountForm.AccountForm; import com.EmperorPenguin.SangmyungBank.api.users.account.service.AccountAddService; 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") @RestController public class AccountAddController {   private final AccountAddService accountAddService;   @PostMapping(path="/accounts/add")  public ResponseEntity<HttpStatus> addAccount(@RequestBody AccountForm accountForm){  Account resultAccount = accountAddService.register(accountForm);  if(resultAccount == null)  return ResponseEntity.*ok*(HttpStatus.*BAD\_REQUEST*);  else  return ResponseEntity.*ok*(HttpStatus.*OK*);  }  }  **-AccountAddRepository**  package com.EmperorPenguin.SangmyungBank.api.users.account.domain.repository;  import com.EmperorPenguin.SangmyungBank.api.users.account.domain.account.Account; import org.springframework.stereotype.Repository;  import javax.persistence.EntityManager; import java.util.List; import java.util.Optional;  @Repository public class AccountAddRepository {   private final EntityManager em;   public AccountAddRepository(EntityManager em) {  this.em = em;  }   public Account save(Account account){  em.persist(account);  return account;  }   public List<Account> findByUserId(String userId) {  List<Account> userAccounts = em.createQuery("select m from Account m where m.userId =:userId",Account.class)  .setParameter("userId", userId)  .getResultList();  return userAccounts;  }   public Optional<Account> findByAccountNumber(Long accountNumber) {  List<Account> userAccount = em.createQuery("select m from Account m where m.accountNumber =:accountNumber",Account.class)  .setParameter("accountNumber", accountNumber)  .getResultList();  return userAccount.stream().findAny();  } }  **-AccountAddService**  package com.EmperorPenguin.SangmyungBank.api.users.account.service;  import com.EmperorPenguin.SangmyungBank.api.users.account.domain.account.Account; import com.EmperorPenguin.SangmyungBank.api.users.account.domain.accountForm.AccountForm; import com.EmperorPenguin.SangmyungBank.api.users.account.domain.repository.AccountAddRepository; import com.EmperorPenguin.SangmyungBank.api.users.add.domain.User.User; import com.EmperorPenguin.SangmyungBank.api.users.login.domain.repository.UserLoginRepository; import org.springframework.stereotype.Service;  import javax.transaction.Transactional;  @Transactional @Service public class AccountAddService {   private final AccountAddRepository accountAddRepository;  private final UserLoginRepository userLoginRepository;   public AccountAddService(AccountAddRepository accountAddRepository, UserLoginRepository userLoginRepository){  this.accountAddRepository = accountAddRepository;  this.userLoginRepository = userLoginRepository;  }  //Id가 DB에 존재하면,계좌발행  public Account register(AccountForm accountForm){  User dbUser = userLoginRepository.findByLoginId(accountForm.getLoginId()).get();  if(dbUser == null)  return null;  else {  Account account = Account.*builder*()  .userId(dbUser.getUserId())  .accountPassword(accountForm.getAccountPassword())  .balance(0L)  .build();  Account result = accountAddRepository.save(account);  return result;  }  } }  **진행 예정** | re |
| * **특이사항 / 협업 사항** | |