1. **Package: com.userfront.domain**

**1.Appointment.java**

**package** com.userfront.domain;

**import** java.util.Date;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.ManyToOne;

@Entity

**public** **class** Appointment {

@Id

@GeneratedValue(strategy = GenerationType.***AUTO***)

**private** Long id;

**private** Date date;

**private** String location;

**private** String description;

**private** **boolean** confirmed;

@ManyToOne

@JoinColumn(name = "user\_id")

**private** User user;

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** Date getDate() {

**return** date;

}

**public** **void** setDate(Date date) {

**this**.date = date;

}

**public** String getLocation() {

**return** location;

}

**public** **void** setLocation(String location) {

**this**.location = location;

}

**public** String getDescription() {

**return** description;

}

**public** **void** setDescription(String description) {

**this**.description = description;

}

**public** User getUser() {

**return** user;

}

**public** **void** setUser(User user) {

**this**.user = user;

}

**public** **boolean** isConfirmed() {

**return** confirmed;

}

**public** **void** setConfirmed(**boolean** confirmed) {

**this**.confirmed = confirmed;

}

@Override

**public** String toString() {

**return** "Appointment{" +

"id=" + id +

", date=" + date +

", location='" + location + '\'' +

", description='" + description + '\'' +

", user=" + user +

'}';

}

}

**PrimaryAccount.java**

**package** com.userfront.domain;

**import** java.math.BigDecimal;

**import** java.util.List;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Entity;

**import** javax.persistence.FetchType;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.OneToMany;

**import** com.fasterxml.jackson.annotation.JsonIgnore;

@Entity

**public** **class** PrimaryAccount {

@Id

@GeneratedValue(strategy = GenerationType.***AUTO***)

**private** Long id;

**private** **int** accountNumber;

**private** BigDecimal accountBalance;

@OneToMany(mappedBy = "primaryAccount", cascade = CascadeType.***ALL***, fetch = FetchType.***LAZY***)

@JsonIgnore

**private** List<PrimaryTransaction> primaryTransactionList;

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** **int** getAccountNumber() {

**return** accountNumber;

}

**public** **void** setAccountNumber(**int** accountNumber) {

**this**.accountNumber = accountNumber;

}

**public** BigDecimal getAccountBalance() {

**return** accountBalance;

}

**public** **void** setAccountBalance(BigDecimal accountBalance) {

**this**.accountBalance = accountBalance;

}

**public** List<PrimaryTransaction> getPrimaryTransactionList() {

**return** primaryTransactionList;

}

**public** **void** setPrimaryTransactionList(List<PrimaryTransaction> primaryTransactionList) {

**this**.primaryTransactionList = primaryTransactionList;

}

}

**Primary Transaction.java**

**package** com.userfront.domain;

**import** java.math.BigDecimal;

**import** java.util.Date;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.ManyToOne;

@Entity

**public** **class** PrimaryTransaction {

@Id

@GeneratedValue(strategy = GenerationType.***AUTO***)

**private** Long id;

**private** Date date;

**private** String description;

**private** String type;

**private** String status;

**private** **double** amount;

**private** BigDecimal availableBalance;

**public** PrimaryTransaction() {}

**public** PrimaryTransaction(Date date, String description, String type, String status, **double** amount, BigDecimal availableBalance, PrimaryAccount primaryAccount) {

**this**.date = date;

**this**.description = description;

**this**.type = type;

**this**.status = status;

**this**.amount = amount;

**this**.availableBalance = availableBalance;

**this**.primaryAccount = primaryAccount;

}

@ManyToOne

@JoinColumn(name = "primary\_account\_id")

**private** PrimaryAccount primaryAccount;

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** Date getDate() {

**return** date;

}

**public** **void** setDate(Date date) {

**this**.date = date;

}

**public** String getDescription() {

**return** description;

}

**public** **void** setDescription(String description) {

**this**.description = description;

}

**public** String getType() {

**return** type;

}

**public** **void** setType(String type) {

**this**.type = type;

}

**public** String getStatus() {

**return** status;

}

**public** **void** setStatus(String status) {

**this**.status = status;

}

**public** **double** getAmount() {

**return** amount;

}

**public** **void** setAmount(**double** amount) {

**this**.amount = amount;

}

**public** BigDecimal getAvailableBalance() {

**return** availableBalance;

}

**public** **void** setAvailableBalance(BigDecimal availableBalance) {

**this**.availableBalance = availableBalance;

}

**public** PrimaryAccount getPrimaryAccount() {

**return** primaryAccount;

}

**public** **void** setPrimaryAccount(PrimaryAccount primaryAccount) {

**this**.primaryAccount = primaryAccount;

}

}

**Recipient.java**

**package** com.userfront.domain;

**import** java.math.BigDecimal;

**import** java.util.Date;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.ManyToOne;

@Entity

**public** **class** PrimaryTransaction {

@Id

@GeneratedValue(strategy = GenerationType.***AUTO***)

**private** Long id;

**private** Date date;

**private** String description;

**private** String type;

**private** String status;

**private** **double** amount;

**private** BigDecimal availableBalance;

**public** PrimaryTransaction() {}

**public** PrimaryTransaction(Date date, String description, String type, String status, **double** amount, BigDecimal availableBalance, PrimaryAccount primaryAccount) {

**this**.date = date;

**this**.description = description;

**this**.type = type;

**this**.status = status;

**this**.amount = amount;

**this**.availableBalance = availableBalance;

**this**.primaryAccount = primaryAccount;

}

@ManyToOne

@JoinColumn(name = "primary\_account\_id")

**private** PrimaryAccount primaryAccount;

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** Date getDate() {

**return** date;

}

**public** **void** setDate(Date date) {

**this**.date = date;

}

**public** String getDescription() {

**return** description;

}

**public** **void** setDescription(String description) {

**this**.description = description;

}

**public** String getType() {

**return** type;

}

**public** **void** setType(String type) {

**this**.type = type;

}

**public** String getStatus() {

**return** status;

}

**public** **void** setStatus(String status) {

**this**.status = status;

}

**public** **double** getAmount() {

**return** amount;

}

**public** **void** setAmount(**double** amount) {

**this**.amount = amount;

}

**public** BigDecimal getAvailableBalance() {

**return** availableBalance;

}

**public** **void** setAvailableBalance(BigDecimal availableBalance) {

**this**.availableBalance = availableBalance;

}

**public** PrimaryAccount getPrimaryAccount() {

**return** primaryAccount;

}

**public** **void** setPrimaryAccount(PrimaryAccount primaryAccount) {

**this**.primaryAccount = primaryAccount;

}

}

**SavingAccount.java**

**package** com.userfront.domain;

**import** java.math.BigDecimal;

**import** java.util.List;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Entity;

**import** javax.persistence.FetchType;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.OneToMany;

**import** com.fasterxml.jackson.annotation.JsonIgnore;

@Entity

**public** **class** SavingsAccount {

@Id

@GeneratedValue(strategy = GenerationType.***AUTO***)

**private** Long id;

**private** **int** accountNumber;

**private** BigDecimal accountBalance;

@OneToMany(mappedBy = "savingsAccount", cascade = CascadeType.***ALL***, fetch = FetchType.***LAZY***)

@JsonIgnore

**private** List<SavingsTransaction> savingsTransactionList;

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** **int** getAccountNumber() {

**return** accountNumber;

}

**public** **void** setAccountNumber(**int** accountNumber) {

**this**.accountNumber = accountNumber;

}

**public** BigDecimal getAccountBalance() {

**return** accountBalance;

}

**public** **void** setAccountBalance(BigDecimal accountBalance) {

**this**.accountBalance = accountBalance;

}

**public** List<SavingsTransaction> getSavingsTransactionList() {

**return** savingsTransactionList;

}

**public** **void** setSavingsTransactionList(List<SavingsTransaction> savingsTransactionList) {

**this**.savingsTransactionList = savingsTransactionList;

}

}

**SavingTransaction.java**

**package** com.userfront.domain;

**import** java.math.BigDecimal;

**import** java.util.Date;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.ManyToOne;

@Entity

**public** **class** SavingsTransaction {

@Id

@GeneratedValue(strategy = GenerationType.***AUTO***)

**private** Long id;

**private** Date date;

**private** String description;

**private** String type;

**private** String status;

**private** **double** amount;

**private** BigDecimal availableBalance;

@ManyToOne

@JoinColumn(name = "savings\_account\_id")

**private** SavingsAccount savingsAccount;

**public** SavingsTransaction() {}

**public** SavingsTransaction(Date date, String description, String type, String status, **double** amount, BigDecimal availableBalance, SavingsAccount savingsAccount) {

**this**.date = date;

**this**.description = description;

**this**.type = type;

**this**.status = status;

**this**.amount = amount;

**this**.availableBalance = availableBalance;

**this**.savingsAccount = savingsAccount;

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** Date getDate() {

**return** date;

}

**public** **void** setDate(Date date) {

**this**.date = date;

}

**public** String getDescription() {

**return** description;

}

**public** **void** setDescription(String description) {

**this**.description = description;

}

**public** String getType() {

**return** type;

}

**public** **void** setType(String type) {

**this**.type = type;

}

**public** String getStatus() {

**return** status;

}

**public** **void** setStatus(String status) {

**this**.status = status;

}

**public** **double** getAmount() {

**return** amount;

}

**public** **void** setAmount(**double** amount) {

**this**.amount = amount;

}

**public** BigDecimal getAvailableBalance() {

**return** availableBalance;

}

**public** **void** setAvailableBalance(BigDecimal availableBalance) {

**this**.availableBalance = availableBalance;

}

**public** SavingsAccount getSavingsAccount() {

**return** savingsAccount;

}

**public** **void** setSavingsAccount(SavingsAccount savingsAccount) {

**this**.savingsAccount = savingsAccount;

}

}

**User.java**

**package** com.userfront.domain;

**import** java.util.Collection;

**import** java.util.HashSet;

**import** java.util.List;

**import** java.util.Set;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.FetchType;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.OneToMany;

**import** javax.persistence.OneToOne;

**import** org.springframework.security.core.GrantedAuthority;

**import** org.springframework.security.core.userdetails.UserDetails;

**import** com.fasterxml.jackson.annotation.JsonIgnore;

**import** com.userfront.domain.security.Authority;

**import** com.userfront.domain.security.UserRole;

@Entity

**public** **class** User **implements** UserDetails{

@Id

@GeneratedValue(strategy = GenerationType.***AUTO***)

@Column(name = "userId", nullable = **false**, updatable = **false**)

**private** Long userId;

**private** String username;

**private** String password;

**private** String firstName;

**private** String lastName;

@Column(name = "email", nullable = **false**, unique = **true**)

**private** String email;

**private** String phone;

**private** **boolean** enabled=**true**;

@OneToOne

**private** PrimaryAccount primaryAccount;

@OneToOne

**private** SavingsAccount savingsAccount;

@OneToMany(mappedBy = "user", cascade = CascadeType.***ALL***, fetch = FetchType.***LAZY***)

@JsonIgnore

**private** List<Appointment> appointmentList;

@OneToMany(mappedBy = "user", cascade = CascadeType.***ALL***, fetch = FetchType.***LAZY***)

**private** List<Recipient> recipientList;

@OneToMany(mappedBy = "user", cascade = CascadeType.***ALL***, fetch = FetchType.***EAGER***)

@JsonIgnore

**private** Set<UserRole> userRoles = **new** HashSet<>();

**public** Set<UserRole> getUserRoles() {

**return** userRoles;

}

**public** **void** setUserRoles(Set<UserRole> userRoles) {

**this**.userRoles = userRoles;

}

**public** Long getUserId() {

**return** userId;

}

**public** **void** setUserId(Long userId) {

**this**.userId = userId;

}

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getPhone() {

**return** phone;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** List<Appointment> getAppointmentList() {

**return** appointmentList;

}

**public** **void** setAppointmentList(List<Appointment> appointmentList) {

**this**.appointmentList = appointmentList;

}

**public** List<Recipient> getRecipientList() {

**return** recipientList;

}

**public** **void** setRecipientList(List<Recipient> recipientList) {

**this**.recipientList = recipientList;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** PrimaryAccount getPrimaryAccount() {

**return** primaryAccount;

}

**public** **void** setPrimaryAccount(PrimaryAccount primaryAccount) {

**this**.primaryAccount = primaryAccount;

}

**public** SavingsAccount getSavingsAccount() {

**return** savingsAccount;

}

**public** **void** setSavingsAccount(SavingsAccount savingsAccount) {

**this**.savingsAccount = savingsAccount;

}

**public** **void** setEnabled(**boolean** enabled) {

**this**.enabled = enabled;

}

@Override

**public** String toString() {

**return** "User{" +

"userId=" + userId +

", username='" + username + '\'' +

", password='" + password + '\'' +

", firstName='" + firstName + '\'' +

", lastName='" + lastName + '\'' +

", email='" + email + '\'' +

", phone='" + phone + '\'' +

", appointmentList=" + appointmentList +

", recipientList=" + recipientList +

", userRoles=" + userRoles +

'}';

}

@Override

**public** Collection<? **extends** GrantedAuthority> getAuthorities() {

Set<GrantedAuthority> authorities = **new** HashSet<>();

userRoles.forEach(ur -> authorities.add(**new** Authority(ur.getRole().getName())));

**return** authorities;

}

@Override

**public** **boolean** isAccountNonExpired() {

// **TODO** Auto-generated method stub

**return** **true**;

}

@Override

**public** **boolean** isAccountNonLocked() {

// **TODO** Auto-generated method stub

**return** **true**;

}

@Override

**public** **boolean** isCredentialsNonExpired() {

// **TODO** Auto-generated method stub

**return** **true**;

}

@Override

**public** **boolean** isEnabled() {

**return** enabled;

}

}

**Package: com.userfront.controller**

**HomeController.java**

**package** com.userfront.controller;

**import** java.security.Principal;

**import** java.util.HashSet;

**import** java.util.Set;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** com.userfront.dao.RoleDao;

**import** com.userfront.domain.PrimaryAccount;

**import** com.userfront.domain.SavingsAccount;

**import** com.userfront.domain.User;

**import** com.userfront.domain.security.UserRole;

**import** com.userfront.service.UserService;

@Controller

**public** **class** HomeController {

@Autowired

**private** UserService userService;

@Autowired

**private** RoleDao roleDao;

@RequestMapping("/")

**public** String home() {

**return** "redirect:/index";

}

@RequestMapping("/index")

**public** String index() {

**return** "index";

}

@RequestMapping(value = "/signup", method = RequestMethod.***GET***)

**public** String signup(Model model) {

User user = **new** User();

model.addAttribute("user", user);

**return** "signup";

}

@RequestMapping(value = "/signup", method = RequestMethod.***POST***)

**public** String signupPost(@ModelAttribute("user") User user, Model model) {

**if**(userService.checkUserExists(user.getUsername(), user.getEmail())) {

**if** (userService.checkEmailExists(user.getEmail())) {

model.addAttribute("emailExists", **true**);

}

**if** (userService.checkUsernameExists(user.getUsername())) {

model.addAttribute("usernameExists", **true**);

}

**return** "signup";

} **else** {

Set<UserRole> userRoles = **new** HashSet<>();

userRoles.add(**new** UserRole(user, roleDao.findByName("ROLE\_USER")));

userService.createUser(user, userRoles);

**return** "redirect:/";

}

}

@RequestMapping("/userFront")

**public** String userFront(Principal principal, Model model) {

User user = userService.findByUsername(principal.getName());

PrimaryAccount primaryAccount = user.getPrimaryAccount();

SavingsAccount savingsAccount = user.getSavingsAccount();

model.addAttribute("primaryAccount", primaryAccount);

model.addAttribute("savingsAccount", savingsAccount);

**return** "userFront";

}

}

**AccountController.java**

**package** com.userfront.controller;

**import** java.security.Principal;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** com.userfront.domain.PrimaryAccount;

**import** com.userfront.domain.PrimaryTransaction;

**import** com.userfront.domain.SavingsAccount;

**import** com.userfront.domain.SavingsTransaction;

**import** com.userfront.domain.User;

**import** com.userfront.service.AccountService;

**import** com.userfront.service.TransactionService;

**import** com.userfront.service.UserService;

@Controller

@RequestMapping("/account")

**public** **class** AccountController {

@Autowired

**private** UserService userService;

@Autowired

**private** AccountService accountService;

@Autowired

**private** TransactionService transactionService;

@RequestMapping("/primaryAccount")

**public** String primaryAccount(Model model, Principal principal) {

List<PrimaryTransaction> primaryTransactionList = transactionService.findPrimaryTransactionList(principal.getName());

User user = userService.findByUsername(principal.getName());

PrimaryAccount primaryAccount = user.getPrimaryAccount();

model.addAttribute("primaryAccount", primaryAccount);

model.addAttribute("primaryTransactionList", primaryTransactionList);

**return** "primaryAccount";

}

@RequestMapping("/savingsAccount")

**public** String savingsAccount(Model model, Principal principal) {

List<SavingsTransaction> savingsTransactionList = transactionService.findSavingsTransactionList(principal.getName());

User user = userService.findByUsername(principal.getName());

SavingsAccount savingsAccount = user.getSavingsAccount();

model.addAttribute("savingsAccount", savingsAccount);

model.addAttribute("savingsTransactionList", savingsTransactionList);

**return** "savingsAccount";

}

@RequestMapping(value = "/deposit", method = RequestMethod.***GET***)

**public** String deposit(Model model) {

model.addAttribute("accountType", "");

model.addAttribute("amount", "");

**return** "deposit";

}

@RequestMapping(value = "/deposit", method = RequestMethod.***POST***)

**public** String depositPOST(@ModelAttribute("amount") String amount, @ModelAttribute("accountType") String accountType, Principal principal) {

accountService.deposit(accountType, Double.*parseDouble*(amount), principal);

**return** "redirect:/userFront";

}

@RequestMapping(value = "/withdraw", method = RequestMethod.***GET***)

**public** String withdraw(Model model) {

model.addAttribute("accountType", "");

model.addAttribute("amount", "");

**return** "withdraw";

}

@RequestMapping(value = "/withdraw", method = RequestMethod.***POST***)

**public** String withdrawPOST(@ModelAttribute("amount") String amount, @ModelAttribute("accountType") String accountType, Principal principal) {

accountService.withdraw(accountType, Double.*parseDouble*(amount), principal);

**return** "redirect:/userFront";

}

}

**AppointmentController**

**package** com.userfront.controller;

**import** java.security.Principal;

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** com.userfront.domain.Appointment;

**import** com.userfront.domain.User;

**import** com.userfront.service.AppointmentService;

**import** com.userfront.service.UserService;

@Controller

@RequestMapping("/appointment")

**public** **class** AppointmentController {

@Autowired

**private** AppointmentService appointmentService;

@Autowired

**private** UserService userService;

@RequestMapping(value = "/create",method = RequestMethod.***GET***)

**public** String createAppointment(Model model) {

Appointment appointment = **new** Appointment();

model.addAttribute("appointment", appointment);

model.addAttribute("dateString", "");

**return** "appointment";

}

@RequestMapping(value = "/create",method = RequestMethod.***POST***)

**public** String createAppointmentPost(@ModelAttribute("appointment") Appointment appointment, @ModelAttribute("dateString") String date, Model model, Principal principal) **throws** ParseException {

SimpleDateFormat format1 = **new** SimpleDateFormat("yyyy-MM-dd hh:mm");

Date d1 = format1.parse( date );

appointment.setDate(d1);

User user = userService.findByUsername(principal.getName());

appointment.setUser(user);

appointmentService.createAppointment(appointment);

**return** "redirect:/userFront";

}

}

**TransferController**

**package** com.userfront.controller;

**import** java.security.Principal;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.transaction.annotation.Transactional;

**import** org.springframework.ui.Model;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** com.userfront.domain.PrimaryAccount;

**import** com.userfront.domain.Recipient;

**import** com.userfront.domain.SavingsAccount;

**import** com.userfront.domain.User;

**import** com.userfront.service.TransactionService;

**import** com.userfront.service.UserService;

@Controller

@RequestMapping("/transfer")

**public** **class** TransferController {

@Autowired

**private** TransactionService transactionService;

@Autowired

**private** UserService userService;

@RequestMapping(value = "/betweenAccounts", method = RequestMethod.***GET***)

**public** String betweenAccounts(Model model) {

model.addAttribute("transferFrom", "");

model.addAttribute("transferTo", "");

model.addAttribute("amount", "");

**return** "betweenAccounts";

}

@RequestMapping(value = "/betweenAccounts", method = RequestMethod.***POST***)

**public** String betweenAccountsPost(

@ModelAttribute("transferFrom") String transferFrom,

@ModelAttribute("transferTo") String transferTo,

@ModelAttribute("amount") String amount,

Principal principal

) **throws** Exception {

User user = userService.findByUsername(principal.getName());

PrimaryAccount primaryAccount = user.getPrimaryAccount();

SavingsAccount savingsAccount = user.getSavingsAccount();

transactionService.betweenAccountsTransfer(transferFrom, transferTo, amount, primaryAccount, savingsAccount);

**return** "redirect:/userFront";

}

@RequestMapping(value = "/recipient", method = RequestMethod.***GET***)

**public** String recipient(Model model, Principal principal) {

List<Recipient> recipientList = transactionService.findRecipientList(principal);

Recipient recipient = **new** Recipient();

model.addAttribute("recipientList", recipientList);

model.addAttribute("recipient", recipient);

**return** "recipient";

}

@RequestMapping(value = "/recipient/save", method = RequestMethod.***POST***)

**public** String recipientPost(@ModelAttribute("recipient") Recipient recipient, Principal principal) {

User user = userService.findByUsername(principal.getName());

recipient.setUser(user);

transactionService.saveRecipient(recipient);

**return** "redirect:/transfer/recipient";

}

@RequestMapping(value = "/recipient/edit", method = RequestMethod.***GET***)

**public** String recipientEdit(@RequestParam(value = "recipientName") String recipientName, Model model, Principal principal){

Recipient recipient = transactionService.findRecipientByName(recipientName);

List<Recipient> recipientList = transactionService.findRecipientList(principal);

model.addAttribute("recipientList", recipientList);

model.addAttribute("recipient", recipient);

**return** "recipient";

}

@RequestMapping(value = "/recipient/delete", method = RequestMethod.***GET***)

@Transactional

**public** String recipientDelete(@RequestParam(value = "recipientName") String recipientName, Model model, Principal principal){

transactionService.deleteRecipientByName(recipientName);

List<Recipient> recipientList = transactionService.findRecipientList(principal);

Recipient recipient = **new** Recipient();

model.addAttribute("recipient", recipient);

model.addAttribute("recipientList", recipientList);

**return** "recipient";

}

@RequestMapping(value = "/toSomeoneElse",method = RequestMethod.***GET***)

**public** String toSomeoneElse(Model model, Principal principal) {

List<Recipient> recipientList = transactionService.findRecipientList(principal);

model.addAttribute("recipientList", recipientList);

model.addAttribute("accountType", "");

**return** "toSomeoneElse";

}

@RequestMapping(value = "/toSomeoneElse",method = RequestMethod.***POST***)

**public** String toSomeoneElsePost(@ModelAttribute("recipientName") String recipientName, @ModelAttribute("accountType") String accountType, @ModelAttribute("amount") String amount, Principal principal) {

User user = userService.findByUsername(principal.getName());

Recipient recipient = transactionService.findRecipientByName(recipientName);

transactionService.toSomeoneElseTransfer(recipient, accountType, amount, user.getPrimaryAccount(), user.getSavingsAccount());

**return** "redirect:/userFront";

}

}

**UserController**

**package** com.userfront.controller;

**import** java.security.Principal;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.web.bind.annotation.ModelAttribute;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** com.userfront.domain.User;

**import** com.userfront.service.UserService;

@Controller

@RequestMapping("/user")

**public** **class** UserController {

@Autowired

**private** UserService userService;

@RequestMapping(value = "/profile", method = RequestMethod.***GET***)

**public** String profile(Principal principal, Model model) {

User user = userService.findByUsername(principal.getName());

model.addAttribute("user", user);

**return** "profile";

}

@RequestMapping(value = "/profile", method = RequestMethod.***POST***)

**public** String profilePost(@ModelAttribute("user") User newUser, Model model) {

User user = userService.findByUsername(newUser.getUsername());

user.setUsername(newUser.getUsername());

user.setFirstName(newUser.getFirstName());

user.setLastName(newUser.getLastName());

user.setEmail(newUser.getEmail());

user.setPhone(newUser.getPhone());

model.addAttribute("user", user);

userService.saveUser(user);

**return** "profile";

}

}

**Package: com.userfront.dao**

**AppointmentDao**

**package** com.userfront.dao;

**import** java.util.List;

**import** org.springframework.data.repository.CrudRepository;

**import** com.userfront.domain.Appointment;

**public** **interface** AppointmentDao **extends** CrudRepository<Appointment, Long> {

List<Appointment> findAll();

}

**PrimaryAccountDao**

**package** com.userfront.dao;

**import** com.userfront.domain.PrimaryAccount;

**import** org.springframework.data.repository.CrudRepository;

**public** **interface** PrimaryAccountDao **extends** CrudRepository<PrimaryAccount,Long> {

PrimaryAccount findByAccountNumber (**int** accountNumber);

}

**PrimaryTransactionDao**

**package** com.userfront.dao;

**import** java.util.List;

**import** org.springframework.data.repository.CrudRepository;

**import** com.userfront.domain.PrimaryTransaction;

**public** **interface** PrimaryTransactionDao **extends** CrudRepository<PrimaryTransaction, Long> {

List<PrimaryTransaction> findAll();

}

**RecipientDao**

**package** com.userfront.dao;

**import** java.util.List;

**import** org.springframework.data.repository.CrudRepository;

**import** com.userfront.domain.Recipient;

**public** **interface** RecipientDao **extends** CrudRepository<Recipient, Long> {

List<Recipient> findAll();

Recipient findByName(String recipientName);

**void** deleteByName(String recipientName);

}

**RoleDao**

**package** com.userfront.dao;

**import** org.springframework.data.repository.CrudRepository;

**import** com.userfront.domain.security.Role;

**public** **interface** RoleDao **extends** CrudRepository<Role, Integer> {

Role findByName(String name);

}

**SavingAccountDao**

**package** com.userfront.dao;

**import** com.userfront.domain.SavingsAccount;

**import** org.springframework.data.repository.CrudRepository;

**public** **interface** SavingsAccountDao **extends** CrudRepository<SavingsAccount, Long> {

SavingsAccount findByAccountNumber (**int** accountNumber);

}

**SavingTransactionDao**

**package** com.userfront.dao;

**import** java.util.List;

**import** org.springframework.data.repository.CrudRepository;

**import** com.userfront.domain.SavingsTransaction;

**public** **interface** SavingsTransactionDao **extends** CrudRepository<SavingsTransaction, Long> {

List<SavingsTransaction> findAll();

}

**UserDao**

**package** com.userfront.dao;

**import** java.util.List;

**import** org.springframework.data.repository.CrudRepository;

**import** com.userfront.domain.User;

**public** **interface** UserDao **extends** CrudRepository<User, Long> {

User findByUsername(String username);

User findByEmail(String email);

List<User> findAll();

}

**Package: com.userfront.domain.security**

**Authority:**

**package** com.userfront.domain.security;

**import** org.springframework.security.core.GrantedAuthority;

**public** **class** Authority **implements** GrantedAuthority{

**private** **final** String authority;

**public** Authority(String authority) {

**this**.authority = authority;

}

@Override

**public** String getAuthority() {

**return** authority;

}

}

**Role:**

**package** com.userfront.domain.security;

**import** javax.persistence.\*;

**import** java.util.HashSet;

**import** java.util.Set;

@Entity

**public** **class** Role {

@Id

// @GeneratedValue(strategy = GenerationType.AUTO)

**private** **int** roleId;

**private** String name;

@OneToMany(mappedBy = "role", cascade = CascadeType.***ALL***, fetch = FetchType.***LAZY***)

**private** Set<UserRole> userRoles = **new** HashSet<>();

**public** Role() {

}

**public** **int** getRoleId() {

**return** roleId;

}

**public** **void** setRoleId(**int** roleId) {

**this**.roleId = roleId;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Set<UserRole> getUserRoles() {

**return** userRoles;

}

**public** **void** setUserRoles(Set<UserRole> userRoles) {

**this**.userRoles = userRoles;

}

}

**UserRole**

**package** com.userfront.domain.security;

**import** com.userfront.domain.User;

**import** javax.persistence.\*;

@Entity

@Table(name="user\_role")

**public** **class** UserRole {

@Id

@GeneratedValue(strategy = GenerationType.***AUTO***)

**private** **long** userRoleId;

**public** UserRole(User user, Role role) {

**this**.user = user;

**this**.role = role;

}

@ManyToOne(fetch = FetchType.***EAGER***)

@JoinColumn(name = "user\_id")

**private** User user;

@ManyToOne(fetch = FetchType.***EAGER***)

@JoinColumn(name = "role\_id")

**private** Role role;

**public** UserRole() {}

**public** **long** getUserRoleId() {

**return** userRoleId;

}

**public** **void** setUserRoleId(**long** userRoleId) {

**this**.userRoleId = userRoleId;

}

**public** User getUser() {

**return** user;

}

**public** **void** setUser(User user) {

**this**.user = user;

}

**public** Role getRole() {

**return** role;

}

**public** **void** setRole(Role role) {

**this**.role = role;

}

}

**Package : com.userfront.config**

**RequestFilter:**

**package** com.userfront.config;

**import** javax.servlet.Filter;

**import** javax.servlet.FilterChain;

**import** javax.servlet.FilterConfig;

**import** javax.servlet.ServletRequest;

**import** javax.servlet.ServletResponse;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** org.springframework.core.Ordered;

**import** org.springframework.core.annotation.Order;

**import** org.springframework.stereotype.Component;

@Component

@Order(Ordered.***HIGHEST\_PRECEDENCE***)

**public** **class** RequestFilter **implements** Filter {

**public** **void** doFilter(ServletRequest req, ServletResponse res, FilterChain chain) {

HttpServletResponse response = (HttpServletResponse) res;

HttpServletRequest request = (HttpServletRequest) req;

response.setHeader("Access-Control-Allow-Origin", "http://localhost:4200");

response.setHeader("Access-Control-Allow-Methods", "POST, PUT, GET, OPTIONS, DELETE");

response.setHeader("Access-Control-Allow-Headers", "x-requested-with");

response.setHeader("Access-Control-Max-Age", "3600");

response.setHeader("Access-Control-Allow-Credentials", "true");

**if** (!(request.getMethod().equalsIgnoreCase("OPTIONS"))) {

**try** {

chain.doFilter(req, res);

} **catch**(Exception e) {

e.printStackTrace();

}

} **else** {

System.***out***.println("Pre-flight");

response.setHeader("Access-Control-Allow-Methods", "POST,GET,DELETE");

response.setHeader("Access-Control-Max-Age", "3600");

response.setHeader("Access-Control-Allow-Headers", "authorization, content-type," +

"access-control-request-headers,access-control-request-method,accept,origin,authorization,x-requested-with");

response.setStatus(HttpServletResponse.***SC\_OK***);

}

}

**public** **void** init(FilterConfig filterConfig) {}

**public** **void** destroy() {}

}

**SecurityConfig**

**package** com.userfront.config;

**import** java.security.SecureRandom;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.core.env.Environment;

**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.crypto.bcrypt.BCryptPasswordEncoder;

**import** org.springframework.security.web.util.matcher.AntPathRequestMatcher;

**import** com.userfront.service.UserServiceImpl.UserSecurityService;

@Configuration

@EnableWebSecurity

@EnableGlobalMethodSecurity(prePostEnabled=**true**)

**public** **class** SecurityConfig **extends** WebSecurityConfigurerAdapter {

@Autowired

**private** Environment env;

@Autowired

**private** UserSecurityService userSecurityService;

**private** **static** **final** String ***SALT*** = "salt"; // Salt should be protected carefully

@Bean

**public** BCryptPasswordEncoder passwordEncoder() {

**return** **new** BCryptPasswordEncoder(12, **new** SecureRandom(***SALT***.getBytes()));

}

**private** **static** **final** String[] ***PUBLIC\_MATCHERS*** = {

"/webjars/\*\*",

"/css/\*\*",

"/js/\*\*",

"/images/\*\*",

"/",

"/about/\*\*",

"/contact/\*\*",

"/error/\*\*/\*",

"/console/\*\*",

"/signup"

};

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http

.authorizeRequests().

// antMatchers("/\*\*").

antMatchers(***PUBLIC\_MATCHERS***).

permitAll().anyRequest().authenticated();

http

.csrf().disable().cors().disable()

.formLogin().failureUrl("/index?error").defaultSuccessUrl("/userFront").loginPage("/index").permitAll()

.and()

.logout().logoutRequestMatcher(**new** AntPathRequestMatcher("/logout")).logoutSuccessUrl("/index?logout").deleteCookies("remember-me").permitAll()

.and()

.rememberMe();

}

@Autowired

**public** **void** configureGlobal(AuthenticationManagerBuilder auth) **throws** Exception {

// auth.inMemoryAuthentication().withUser("user").password("password").roles("USER"); //This is in-memory authentication

auth.userDetailsService(userSecurityService).passwordEncoder(passwordEncoder());

}

}

**Package: com.userfront.resource**

AppointmentResource:

**package** com.userfront.resource;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.security.access.prepost.PreAuthorize;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.userfront.domain.Appointment;

**import** com.userfront.service.AppointmentService;

@RestController

@RequestMapping("/api/appointment")

@PreAuthorize("hasRole('ADMIN')")

**public** **class** AppointmentResource {

@Autowired

**private** AppointmentService appointmentService;

@RequestMapping("/all")

**public** List<Appointment> findAppointmentList() {

List<Appointment> appointmentList = appointmentService.findAll();

**return** appointmentList;

}

@RequestMapping("/{id}/confirm")

**public** **void** confirmAppointment(@PathVariable("id") Long id) {

appointmentService.confirmAppointment(id);

}

}

**UserResource:**

**package** com.userfront.resource;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.security.access.prepost.PreAuthorize;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.userfront.domain.PrimaryTransaction;

**import** com.userfront.domain.SavingsTransaction;

**import** com.userfront.domain.User;

**import** com.userfront.service.TransactionService;

**import** com.userfront.service.UserService;

@RestController

@RequestMapping("/api")

@PreAuthorize("hasRole('ADMIN')")

**public** **class** UserResource {

@Autowired

**private** UserService userService;

@Autowired

**private** TransactionService transactionService;

@RequestMapping(value = "/user/all", method = RequestMethod.***GET***)

**public** List<User> userList() {

**return** userService.findUserList();

}

@RequestMapping(value = "/user/primary/transaction", method = RequestMethod.***GET***)

**public** List<PrimaryTransaction> getPrimaryTransactionList(@RequestParam("username") String username) {

**return** transactionService.findPrimaryTransactionList(username);

}

@RequestMapping(value = "/user/savings/transaction", method = RequestMethod.***GET***)

**public** List<SavingsTransaction> getSavingsTransactionList(@RequestParam("username") String username) {

**return** transactionService.findSavingsTransactionList(username);

}

@RequestMapping("/user/{username}/enable")

**public** **void** enableUser(@PathVariable("username") String username) {

userService.enableUser(username);

}

@RequestMapping("/user/{username}/disable")

**public** **void** diableUser(@PathVariable("username") String username) {

userService.disableUser(username);

}

}

**Package: com.userfront.service**

**AccountService**

**package** com.userfront.service;

**import** java.security.Principal;

**import** com.userfront.domain.PrimaryAccount;

**import** com.userfront.domain.PrimaryTransaction;

**import** com.userfront.domain.SavingsAccount;

**import** com.userfront.domain.SavingsTransaction;

**public** **interface** AccountService {

PrimaryAccount createPrimaryAccount();

SavingsAccount createSavingsAccount();

**void** deposit(String accountType, **double** amount, Principal principal);

**void** withdraw(String accountType, **double** amount, Principal principal);

}

**AppointmentService:**

**package** com.userfront.service;

**import** java.util.List;

**import** com.userfront.domain.Appointment;

**public** **interface** AppointmentService {

Appointment createAppointment(Appointment appointment);

List<Appointment> findAll();

Appointment findAppointment(Long id);

**void** confirmAppointment(Long id);

}

**TransactionService:**

**package** com.userfront.service;

**import** java.security.Principal;

**import** java.util.List;

**import** com.userfront.domain.PrimaryAccount;

**import** com.userfront.domain.PrimaryTransaction;

**import** com.userfront.domain.Recipient;

**import** com.userfront.domain.SavingsAccount;

**import** com.userfront.domain.SavingsTransaction;

**public** **interface** TransactionService {

List<PrimaryTransaction> findPrimaryTransactionList(String username);

List<SavingsTransaction> findSavingsTransactionList(String username);

**void** savePrimaryDepositTransaction(PrimaryTransaction primaryTransaction);

**void** saveSavingsDepositTransaction(SavingsTransaction savingsTransaction);

**void** savePrimaryWithdrawTransaction(PrimaryTransaction primaryTransaction);

**void** saveSavingsWithdrawTransaction(SavingsTransaction savingsTransaction);

**void** betweenAccountsTransfer(String transferFrom, String transferTo, String amount, PrimaryAccount primaryAccount, SavingsAccount savingsAccount) **throws** Exception;

List<Recipient> findRecipientList(Principal principal);

Recipient saveRecipient(Recipient recipient);

Recipient findRecipientByName(String recipientName);

**void** deleteRecipientByName(String recipientName);

**void** toSomeoneElseTransfer(Recipient recipient, String accountType, String amount, PrimaryAccount primaryAccount, SavingsAccount savingsAccount);

}

**UserService:**

**package** com.userfront.service;

**import** java.util.List;

**import** java.util.Set;

**import** com.userfront.domain.User;

**import** com.userfront.domain.security.UserRole;

**public** **interface** UserService {

User findByUsername(String username);

User findByEmail(String email);

**boolean** checkUserExists(String username, String email);

**boolean** checkUsernameExists(String username);

**boolean** checkEmailExists(String email);

**void** save (User user);

User createUser(User user, Set<UserRole> userRoles);

User saveUser (User user);

List<User> findUserList();

**void** enableUser (String username);

**void** disableUser (String username);

}

Package: com.userfront.service.UserServiceImpl

AccountServiceImpl:

**package** com.userfront.service.UserServiceImpl;

**import** java.math.BigDecimal;

**import** java.security.Principal;

**import** java.util.Date;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.userfront.dao.PrimaryAccountDao;

**import** com.userfront.dao.SavingsAccountDao;

**import** com.userfront.domain.PrimaryAccount;

**import** com.userfront.domain.PrimaryTransaction;

**import** com.userfront.domain.SavingsAccount;

**import** com.userfront.domain.SavingsTransaction;

**import** com.userfront.domain.User;

**import** com.userfront.service.AccountService;

**import** com.userfront.service.TransactionService;

**import** com.userfront.service.UserService;

@Service

**public** **class** AccountServiceImpl **implements** AccountService {

**private** **static** **int** *nextAccountNumber* = 11223145;

@Autowired

**private** PrimaryAccountDao primaryAccountDao;

@Autowired

**private** SavingsAccountDao savingsAccountDao;

@Autowired

**private** UserService userService;

@Autowired

**private** TransactionService transactionService;

**public** PrimaryAccount createPrimaryAccount() {

PrimaryAccount primaryAccount = **new** PrimaryAccount();

primaryAccount.setAccountBalance(**new** BigDecimal(0.0));

primaryAccount.setAccountNumber(accountGen());

primaryAccountDao.save(primaryAccount);

**return** primaryAccountDao.findByAccountNumber(primaryAccount.getAccountNumber());

}

**public** SavingsAccount createSavingsAccount() {

SavingsAccount savingsAccount = **new** SavingsAccount();

savingsAccount.setAccountBalance(**new** BigDecimal(0.0));

savingsAccount.setAccountNumber(accountGen());

savingsAccountDao.save(savingsAccount);

**return** savingsAccountDao.findByAccountNumber(savingsAccount.getAccountNumber());

}

**public** **void** deposit(String accountType, **double** amount, Principal principal) {

User user = userService.findByUsername(principal.getName());

**if** (accountType.equalsIgnoreCase("Primary")) {

PrimaryAccount primaryAccount = user.getPrimaryAccount();

primaryAccount.setAccountBalance(primaryAccount.getAccountBalance().add(**new** BigDecimal(amount)));

primaryAccountDao.save(primaryAccount);

Date date = **new** Date();

PrimaryTransaction primaryTransaction = **new** PrimaryTransaction(date, "Deposit to Primary Account", "Account", "Finished", amount, primaryAccount.getAccountBalance(), primaryAccount);

transactionService.savePrimaryDepositTransaction(primaryTransaction);

} **else** **if** (accountType.equalsIgnoreCase("Savings")) {

SavingsAccount savingsAccount = user.getSavingsAccount();

savingsAccount.setAccountBalance(savingsAccount.getAccountBalance().add(**new** BigDecimal(amount)));

savingsAccountDao.save(savingsAccount);

Date date = **new** Date();

SavingsTransaction savingsTransaction = **new** SavingsTransaction(date, "Deposit to savings Account", "Account", "Finished", amount, savingsAccount.getAccountBalance(), savingsAccount);

transactionService.saveSavingsDepositTransaction(savingsTransaction);

}

}

**public** **void** withdraw(String accountType, **double** amount, Principal principal) {

User user = userService.findByUsername(principal.getName());

**if** (accountType.equalsIgnoreCase("Primary")) {

PrimaryAccount primaryAccount = user.getPrimaryAccount();

primaryAccount.setAccountBalance(primaryAccount.getAccountBalance().subtract(**new** BigDecimal(amount)));

primaryAccountDao.save(primaryAccount);

Date date = **new** Date();

PrimaryTransaction primaryTransaction = **new** PrimaryTransaction(date, "Withdraw from Primary Account", "Account", "Finished", amount, primaryAccount.getAccountBalance(), primaryAccount);

transactionService.savePrimaryWithdrawTransaction(primaryTransaction);

} **else** **if** (accountType.equalsIgnoreCase("Savings")) {

SavingsAccount savingsAccount = user.getSavingsAccount();

savingsAccount.setAccountBalance(savingsAccount.getAccountBalance().subtract(**new** BigDecimal(amount)));

savingsAccountDao.save(savingsAccount);

Date date = **new** Date();

SavingsTransaction savingsTransaction = **new** SavingsTransaction(date, "Withdraw from savings Account", "Account", "Finished", amount, savingsAccount.getAccountBalance(), savingsAccount);

transactionService.saveSavingsWithdrawTransaction(savingsTransaction);

}

}

**private** **int** accountGen() {

**return** ++*nextAccountNumber*;

}

}

**AppointmentServiceImpl**

**package** com.userfront.service.UserServiceImpl;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.userfront.dao.AppointmentDao;

**import** com.userfront.domain.Appointment;

**import** com.userfront.service.AppointmentService;

@Service

**public** **class** AppointmentServiceImpl **implements** AppointmentService {

@Autowired

**private** AppointmentDao appointmentDao;

**public** Appointment createAppointment(Appointment appointment) {

**return** appointmentDao.save(appointment);

}

**public** List<Appointment> findAll() {

**return** appointmentDao.findAll();

}

**public** Appointment findAppointment(Long id) {

**return** appointmentDao.findOne(id);

}

**public** **void** confirmAppointment(Long id) {

Appointment appointment = findAppointment(id);

appointment.setConfirmed(**true**);

appointmentDao.save(appointment);

}

}

**package** com.userfront.service.UserServiceImpl;

**import** java.math.BigDecimal;

**import** java.security.Principal;

**import** java.util.Date;

**import** java.util.List;

**import** java.util.stream.Collectors;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.userfront.dao.PrimaryAccountDao;

**import** com.userfront.dao.PrimaryTransactionDao;

**import** com.userfront.dao.RecipientDao;

**import** com.userfront.dao.SavingsAccountDao;

**import** com.userfront.dao.SavingsTransactionDao;

**import** com.userfront.domain.PrimaryAccount;

**import** com.userfront.domain.PrimaryTransaction;

**import** com.userfront.domain.Recipient;

**import** com.userfront.domain.SavingsAccount;

**import** com.userfront.domain.SavingsTransaction;

**import** com.userfront.domain.User;

**import** com.userfront.service.TransactionService;

**import** com.userfront.service.UserService;

@Service

**public** **class** TransactionServiceImpl **implements** TransactionService {

@Autowired

**private** UserService userService;

@Autowired

**private** PrimaryTransactionDao primaryTransactionDao;

@Autowired

**private** SavingsTransactionDao savingsTransactionDao;

@Autowired

**private** PrimaryAccountDao primaryAccountDao;

@Autowired

**private** SavingsAccountDao savingsAccountDao;

@Autowired

**private** RecipientDao recipientDao;

**public** List<PrimaryTransaction> findPrimaryTransactionList(String username){

User user = userService.findByUsername(username);

List<PrimaryTransaction> primaryTransactionList = user.getPrimaryAccount().getPrimaryTransactionList();

**return** primaryTransactionList;

}

**public** List<SavingsTransaction> findSavingsTransactionList(String username) {

User user = userService.findByUsername(username);

List<SavingsTransaction> savingsTransactionList = user.getSavingsAccount().getSavingsTransactionList();

**return** savingsTransactionList;

}

**public** **void** savePrimaryDepositTransaction(PrimaryTransaction primaryTransaction) {

primaryTransactionDao.save(primaryTransaction);

}

**public** **void** saveSavingsDepositTransaction(SavingsTransaction savingsTransaction) {

savingsTransactionDao.save(savingsTransaction);

}

**public** **void** savePrimaryWithdrawTransaction(PrimaryTransaction primaryTransaction) {

primaryTransactionDao.save(primaryTransaction);

}

**public** **void** saveSavingsWithdrawTransaction(SavingsTransaction savingsTransaction) {

savingsTransactionDao.save(savingsTransaction);

}

**public** **void** betweenAccountsTransfer(String transferFrom, String transferTo, String amount, PrimaryAccount primaryAccount, SavingsAccount savingsAccount) **throws** Exception {

**if** (transferFrom.equalsIgnoreCase("Primary") && transferTo.equalsIgnoreCase("Savings")) {

primaryAccount.setAccountBalance(primaryAccount.getAccountBalance().subtract(**new** BigDecimal(amount)));

savingsAccount.setAccountBalance(savingsAccount.getAccountBalance().add(**new** BigDecimal(amount)));

primaryAccountDao.save(primaryAccount);

savingsAccountDao.save(savingsAccount);

Date date = **new** Date();

PrimaryTransaction primaryTransaction = **new** PrimaryTransaction(date, "Between account transfer from "+transferFrom+" to "+transferTo, "Account", "Finished", Double.*parseDouble*(amount), primaryAccount.getAccountBalance(), primaryAccount);

primaryTransactionDao.save(primaryTransaction);

} **else** **if** (transferFrom.equalsIgnoreCase("Savings") && transferTo.equalsIgnoreCase("Primary")) {

primaryAccount.setAccountBalance(primaryAccount.getAccountBalance().add(**new** BigDecimal(amount)));

savingsAccount.setAccountBalance(savingsAccount.getAccountBalance().subtract(**new** BigDecimal(amount)));

primaryAccountDao.save(primaryAccount);

savingsAccountDao.save(savingsAccount);

Date date = **new** Date();

SavingsTransaction savingsTransaction = **new** SavingsTransaction(date, "Between account transfer from "+transferFrom+" to "+transferTo, "Transfer", "Finished", Double.*parseDouble*(amount), savingsAccount.getAccountBalance(), savingsAccount);

savingsTransactionDao.save(savingsTransaction);

} **else** {

**throw** **new** Exception("Invalid Transfer");

}

}

**public** List<Recipient> findRecipientList(Principal principal) {

String username = principal.getName();

List<Recipient> recipientList = recipientDao.findAll().stream() //convert list to stream

.filter(recipient -> username.equals(recipient.getUser().getUsername())) //filters the line, equals to username

.collect(Collectors.*toList*());

**return** recipientList;

}

**public** Recipient saveRecipient(Recipient recipient) {

**return** recipientDao.save(recipient);

}

**public** Recipient findRecipientByName(String recipientName) {

**return** recipientDao.findByName(recipientName);

}

**public** **void** deleteRecipientByName(String recipientName) {

recipientDao.deleteByName(recipientName);

}

**public** **void** toSomeoneElseTransfer(Recipient recipient, String accountType, String amount, PrimaryAccount primaryAccount, SavingsAccount savingsAccount) {

**if** (accountType.equalsIgnoreCase("Primary")) {

primaryAccount.setAccountBalance(primaryAccount.getAccountBalance().subtract(**new** BigDecimal(amount)));

primaryAccountDao.save(primaryAccount);

Date date = **new** Date();

PrimaryTransaction primaryTransaction = **new** PrimaryTransaction(date, "Transfer to recipient "+recipient.getName(), "Transfer", "Finished", Double.*parseDouble*(amount), primaryAccount.getAccountBalance(), primaryAccount);

primaryTransactionDao.save(primaryTransaction);

} **else** **if** (accountType.equalsIgnoreCase("Savings")) {

savingsAccount.setAccountBalance(savingsAccount.getAccountBalance().subtract(**new** BigDecimal(amount)));

savingsAccountDao.save(savingsAccount);

Date date = **new** Date();

SavingsTransaction savingsTransaction = **new** SavingsTransaction(date, "Transfer to recipient "+recipient.getName(), "Transfer", "Finished", Double.*parseDouble*(amount), savingsAccount.getAccountBalance(), savingsAccount);

savingsTransactionDao.save(savingsTransaction);

}

}

}

**package** com.userfront.service.UserServiceImpl;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**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** com.userfront.dao.UserDao;

**import** com.userfront.domain.User;

@Service

**public** **class** UserSecurityService **implements** UserDetailsService {

/\*\* The application logger \*/

**private** **static** **final** Logger ***LOG*** = LoggerFactory.*getLogger*(UserSecurityService.**class**);

@Autowired

**private** UserDao userDao;

@Override

**public** UserDetails loadUserByUsername(String username) **throws** UsernameNotFoundException {

User user = userDao.findByUsername(username);

**if** (**null** == user) {

***LOG***.warn("Username {} not found", username);

**throw** **new** UsernameNotFoundException("Username " + username + " not found");

}

**return** user;

}

}

**package** com.userfront.service.UserServiceImpl;

**import** java.util.List;

**import** java.util.Set;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.userfront.dao.RoleDao;

**import** com.userfront.dao.UserDao;

**import** com.userfront.domain.User;

**import** com.userfront.domain.security.UserRole;

**import** com.userfront.service.AccountService;

**import** com.userfront.service.UserService;

@Service

@Transactional

**public** **class** UserServiceImpl **implements** UserService{

**private** **static** **final** Logger ***LOG*** = LoggerFactory.*getLogger*(UserService.**class**);

@Autowired

**private** UserDao userDao;

@Autowired

**private** RoleDao roleDao;

@Autowired

**private** BCryptPasswordEncoder passwordEncoder;

@Autowired

**private** AccountService accountService;

**public** **void** save(User user) {

userDao.save(user);

}

**public** User findByUsername(String username) {

**return** userDao.findByUsername(username);

}

**public** User findByEmail(String email) {

**return** userDao.findByEmail(email);

}

**public** User createUser(User user, Set<UserRole> userRoles) {

User localUser = userDao.findByUsername(user.getUsername());

**if** (localUser != **null**) {

***LOG***.info("User with username {} already exist. Nothing will be done. ", user.getUsername());

} **else** {

String encryptedPassword = passwordEncoder.encode(user.getPassword());

user.setPassword(encryptedPassword);

**for** (UserRole ur : userRoles) {

roleDao.save(ur.getRole());

}

user.getUserRoles().addAll(userRoles);

user.setPrimaryAccount(accountService.createPrimaryAccount());

user.setSavingsAccount(accountService.createSavingsAccount());

localUser = userDao.save(user);

}

**return** localUser;

}

**public** **boolean** checkUserExists(String username, String email){

**if** (checkUsernameExists(username) || checkEmailExists(username)) {

**return** **true**;

} **else** {

**return** **false**;

}

}

**public** **boolean** checkUsernameExists(String username) {

**if** (**null** != findByUsername(username)) {

**return** **true**;

}

**return** **false**;

}

**public** **boolean** checkEmailExists(String email) {

**if** (**null** != findByEmail(email)) {

**return** **true**;

}

**return** **false**;

}

**public** User saveUser (User user) {

**return** userDao.save(user);

}

**public** List<User> findUserList() {

**return** userDao.findAll();

}

**public** **void** enableUser (String username) {

User user = findByUsername(username);

user.setEnabled(**true**);

userDao.save(user);

}

**public** **void** disableUser (String username) {

User user = findByUsername(username);

user.setEnabled(**false**);

System.***out***.println(user.isEnabled());

userDao.save(user);

System.***out***.println(username + " is disabled.");

}

}

**Application.properties:**

# ===============================

# = DATA SOURCE

# ===============================

# Set here configurations for the database connection

# Connection url for the database "netgloo\_blog"

spring.datasource.url = jdbc:mysql://localhost:3306/OnlineBanking

# Username and secret

spring.datasource.username = root

spring.datasource.password = \*\*\*\*\*\*\*\*\*\*\*

# Keep the connection alive if idle for a long time (needed in production)

spring.datasource.testWhileIdle = true

spring.main.allow-circular-references=true

spring.datasource.validationQuery = SELECT 1

# ===============================

# = JPA / HIBERNATE

# ===============================

# Use spring.jpa.properties.\* for Hibernate native properties (the prefix is

# stripped before adding them to the entity manager).

# Show or not log for each sql query

spring.jpa.show-sql = true

# Hibernate ddl auto (create, create-drop, update): with "update" the database

# schema will be automatically updated accordingly to java entities found in

# the project

spring.jpa.hibernate.ddl-auto = update

# Allows Hibernate to generate SQL optimized for a particular DBMS

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect