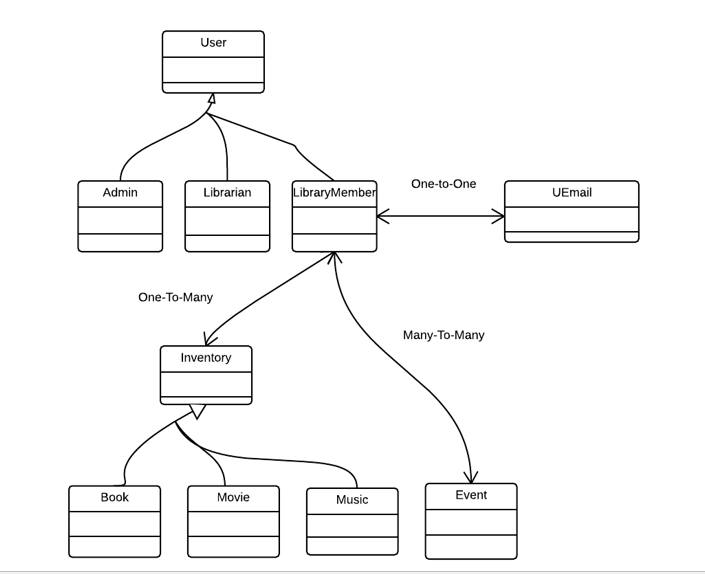
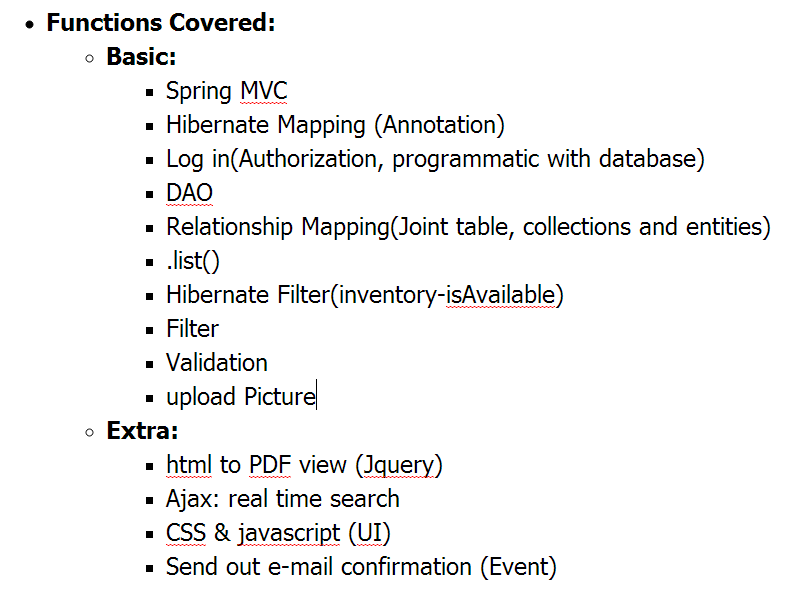
# Final project Report:

# Short summary of the project

This project is implementing a library website. The UML for POJO classes and main features are depicted in the graphs below.





# Summary of the Functionality Performed

The major roles and functions in the website include:

* Admin, who oversees and manages user accounts.
* Librarian, who manages inventories and events. Also, a librarian is responsible for helping Library Members to check out inventories, register for events and return inventories. After adding new events, they can also send out an e-mail to notice library members
* Library Members, who can check their own accounts and can see these available events or inventories only.

Special associations among entities include:

* One-to-one: One user should have only one e-mail; One e-mail could only associate to one user account
* One-to-many: One Inventory (Book, Movie, Music) can only be borrowed by one Library Member, yet one Library Member can borrow multiple inventories.
* Many-to-many: One Event can be registered by many Library Members and One Library Member can register to many events

# Screenshots of key screens if available.

|  |  |
| --- | --- |
| Home page of the library website; user will log in from here. And it will check user’s role and redirect them to corresponding page. |  |
| You can see the different pages when login as an admin or librarian or member. |  |
| On the Admin’s browse and search page, the admin can see a full list of users and he can also do a real-time search, which was implemented with ajax. |  |
| On the add user page, the formats of input will be validated and only when the input valid, the create user button will be enabled. |  |
| On the Librarian’s trail. If you go to Browse Inventory page, you can see the full list and click on button to print out the list as a pdf file. |  |
| Since the different inventories has different information requirements for them to be added. A special page is added to get their corresponding numbers and information. |  |
| Delete User page; delete events or inventories are similar |  |
| Picture uploading and displaying are also supported.  For example, in browse event and add event page, librarian can upload or view an image as a poster for the event.  PS: you can click on the button to send out an email reminder to all users that a new event been posted |  |
| The process of helping members to borrow a book or register event or return a book is similar.  You will need to enter target id and user id. As you can see, in database, the status for inventory 3 is updated to not available and owned by user named 8. |  |
| Finally, Library Member’s pages are reutilized the browse event& browse inventory pages from librarian, except filtered based on the user or availability of inventories (or events) |  |

# Copy CONTROLLER (only controllers) Source Codes and PASTE at the end of the Report

|  |  |
| --- | --- |
| Name of Controller and summary | Controller content |
| AdminController:  All Admin related functions | package com.Li.controller;  import java.util.List;  import javax.servlet.http.HttpServletRequest;  import org.hibernate.Query;  import org.hibernate.Session;  import org.hibernate.SessionFactory;  import org.hibernate.cfg.Configuration;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import org.springframework.web.servlet.ModelAndView;  import com.Li.pojos.roles.Admin;  import com.Li.pojos.roles.Librarian;  import com.Li.pojos.roles.LibraryMember;  import com.Li.pojos.roles.UEmail;  import com.Li.pojos.roles.User;  //all Admin-Related Controllers  @Controller  @RequestMapping("admin/")  public class AdminController {  /\*-----------------Page Switch starts------------------------\*/  /\*@Autowired  @Qualifier("userDao")    UserDAO userDao;  @Autowired  @Qualifier("userValidator")  UserValidator validator;  @InitBinder  private void initBinder(WebDataBinder binder) {  binder.setValidator(validator);  }  @Autowired  ServletContext servletContext;\*/  @RequestMapping(value = "/adminMenu", method = RequestMethod.POST)  public ModelAndView adminMenu( HttpServletRequest request) {  String result = request.getParameter("isselect");  String destination= null;  ModelAndView mv = null;  if (result.equals("BrowseUser")) {  destination= "admin/browseUser";  Configuration cfg = new Configuration();  SessionFactory sf = cfg.configure().buildSessionFactory();  Session hibernatesession = sf.openSession();  //HttpSession session = request.getSession(true);  System.out.println("In User Browse");  //String SQL\_QUERY =" from User as o where o.userName=? and o.userPassword=?";  String SQL\_QUERY = " FROM User";  Query query = hibernatesession.createQuery(SQL\_QUERY);  @SuppressWarnings("unchecked")  List<User> userList = query.list();  hibernatesession.close();  mv = new ModelAndView(destination,"userList",userList);  }  else if (result.equals("AddAUser")){  destination= "admin/addUser";  mv = new ModelAndView(destination);  }  else if (result.equals("DeleteUser")){  destination= "admin/deleteUser";  mv = new ModelAndView(destination);  }  return mv;  }  /\*  @RequestMapping(value = "/admin/adminMenu.htm", method = RequestMethod.GET)  public String backAdminMenu(HttpServletRequest request) {    return "admin/adminMenu";  }\*/    @RequestMapping(value = "/admin/adminMenu.htm", method = RequestMethod.GET)  public String ajaxSearch(HttpServletRequest request) {    return "test";  }    @RequestMapping(value = "/admin/addUser.htm", method = RequestMethod.POST)  public String addUser(HttpServletRequest request) {  Configuration cfg = new Configuration();  SessionFactory sf = cfg.configure().buildSessionFactory();  Session hibernatesession = sf.openSession();  //HttpSession session = request.getSession(true);  System.out.println("In User Browse");  //String SQL\_QUERY =" from User as o where o.userName=? and o.userPassword=?";    UEmail newEmail = new UEmail();  String role = request.getParameter("roleSelected");  String userPassword= request.getParameter("userPassword");  String userName = request.getParameter("userName");  /\* if(role.equals("Admin") ){    }\*/  if(role.equals("Librarian") ){  Librarian newUser = new Librarian(userName,userPassword,role);  newEmail.setEmailAddress(request.getParameter("email"));  newUser.setEmail(newEmail);  newUser.setFirstName(request.getParameter("firstName"));  newUser.setLastName(request.getParameter("lastName"));  newUser.setUserPassword(request.getParameter("userPassword"));  newUser.setUserName(request.getParameter("userName"));  newUser.setUserRole(request.getParameter("roleSelected"));  hibernatesession.save(newUser);  hibernatesession.save(newEmail);    }  else if(role.equals("LibraryMember") ) {  LibraryMember newUser = new LibraryMember(userName,userPassword,role);  newEmail.setEmailAddress(request.getParameter("email"));  newUser.setEmail(newEmail);  newUser.setFirstName(request.getParameter("firstName"));  newUser.setLastName(request.getParameter("lastName"));  newUser.setUserPassword(request.getParameter("userPassword"));  newUser.setUserName(request.getParameter("userName"));  newUser.setUserRole(request.getParameter("roleSelected"));  hibernatesession.save(newUser);  hibernatesession.save(newEmail);  }      /\* newEmail.setEmailAddress(request.getParameter("email"));  newUser.setEmail(newEmail);  newUser.setFirstName(request.getParameter("firstName"));  newUser.setLastName(request.getParameter("lastName"));  newUser.setUserPassword(request.getParameter("userPassword"));  newUser.setUserName(request.getParameter("userName"));  newUser.setUserRole(request.getParameter("roleSelected"));  hibernatesession.save(newUser);  hibernatesession.save(newEmail);\*/  hibernatesession.close();    return "admin/addSuccessfully";  }    /\* @RequestMapping(value = "/search.htm", method = RequestMethod.POST)  public ModelAndView seacrhUser(HttpServletRequest request) {  String key = request.getParameter("inputtext");  String searchKey = request.getParameter("searchkey");  List<User> userList = null;  try {  userList = userDao.get(key, searchKey);  } catch (UserException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  return new ModelAndView("search-result", "userList", userList);  }\*/  @RequestMapping(value ="deleteUser.htm",method = RequestMethod.POST)  public ModelAndView deleteInventory(HttpServletRequest request){  ModelAndView mv =null;  int inventoryID = Integer.valueOf(request.getParameter("userID"));  String warning = "NO such Item with this ID";  Configuration cfg = new Configuration();  SessionFactory sf = cfg.configure().buildSessionFactory();  Session hibernatesession = sf.openSession();  // User targetUser = new User();  /\* targetUser.setEmail(email);  targetUser.setFirstName(firstName);  targetUser.setLastName(lastName);\*/  /\* targetUser.setUserId(inventoryID);    hibernatesession.delete(targetUser);  hibernatesession.flush();  hibernatesession.close();\*/    Query q = hibernatesession.createQuery("delete User where id = :inventoryID");    q.setParameter("inventoryID", inventoryID);  q.executeUpdate();    if(request.getParameter("userID").isEmpty()){  mv = new ModelAndView("deleteUser", "warning",warning);  }  else{  mv = new ModelAndView("admin/adminMenu", "warning",warning);    }      return mv;  }  } |
| EventController  All Event related functions | package com.Li.controller;  import java.io.File;  import java.io.IOException;  import javax.servlet.ServletContext;  import javax.servlet.http.HttpServletRequest;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.beans.factory.annotation.Qualifier;  import org.springframework.stereotype.Controller;  import org.springframework.ui.ModelMap;  import org.springframework.validation.BindingResult;  import org.springframework.web.bind.WebDataBinder;  import org.springframework.web.bind.annotation.InitBinder;  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.multipart.commons.CommonsMultipartFile;  import org.springframework.web.servlet.ModelAndView;  import com.Li.dao.EventDAO;  import com.Li.exception.EventException;  import com.Li.pojos.inventory.Event;  import com.Li.validator.EventValidator;  @Controller  public class EventController {  @Autowired  @Qualifier("eventDao")  EventDAO eventDao;  @Autowired  @Qualifier("eventValidator")  EventValidator eventvalidator;  @InitBinder  private void initeventBinder(WebDataBinder eventbinder) {  eventbinder.setValidator(eventvalidator);  }    @Autowired  ServletContext servletContext;  ////////////////////////////////////add event///////////////////////////////////////////  @RequestMapping(value = "librarian/addEvent.htm", method = RequestMethod.GET)  public String createModel(ModelMap model){    Event event = new Event();  System.out.println("iiiiiin get method"+event.getEventID());  model.addAttribute("event",event);  return ("addEvent");  }  @RequestMapping(value = "librarian/addEvent.htm", method = RequestMethod.POST)  public ModelAndView handleUpload(@ModelAttribute("event") Event event,HttpServletRequest request) {  System.out.println("in add event controller");  //event = new Event();  System.out.println("iiiiiin get method"+event.getDescription());  ModelAndView mv;  //eventvalidator.validate(event, result);  try {  event.setDescription(request.getParameter("description"));  if (event.getFilename().trim() != "" || event.getFilename() != null) {  File directory;  String check = File.separator; // Checking if system is linux  // based or windows based by  // checking seprator used.  String path = null;  if (check.equalsIgnoreCase("\\")) {  path = servletContext.getRealPath("").replace("build\\", ""); // gives real path as Lab9/build/web/  // so we need to replace build in the path  }  if (check.equalsIgnoreCase("/")) {  path = servletContext.getRealPath("").replace("build/", "");  path += "/"; // Adding trailing slash for Mac systems.  }  directory = new File(path + "\\" + event.getFilename());  boolean temp = directory.exists();  if (!temp) {  temp = directory.mkdir();  }  if (temp) {  // We need to transfer to a file  CommonsMultipartFile photoInMemory = event.getPoster();  String fileName = photoInMemory.getOriginalFilename();  // could generate file names as well  File localFile = new File(directory.getPath(), fileName);  // move the file from memory to the file  photoInMemory.transferTo(localFile);  event.setDescription(localFile.getPath());  System.out.println("File is stored at" + localFile.getPath());  System.out.print("registerNewUser");  Event u = eventDao.create(event);  } else {  System.out.println("Failed to create directory!");  }  }  } catch (IllegalStateException e) {  System.out.println("\*\*\* IllegalStateException: " + e.getMessage());  } catch (IOException e) {  // TODO Auto-generated catch block  System.out.println("\*\*\* IOException: " + e.getMessage());  } catch (EventException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  return mv = new ModelAndView("addSuccessfully");  }  } |
| HomeConrtoller  The first controller used for login and identify the user’s role | package com.Li.controller;  import java.text.DateFormat;  import java.util.Date;  import java.util.Locale;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpSession;  import org.hibernate.Query;  import org.hibernate.Session;  import org.hibernate.SessionFactory;  import org.hibernate.cfg.Configuration;  import org.slf4j.Logger;  import org.slf4j.LoggerFactory;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import org.springframework.web.servlet.ModelAndView;  import com.Li.pojos.roles.LibraryMember;  import com.Li.pojos.roles.User;  /\*\*  \* Handles requests for the application home page.  \*/  @Controller  public class HomeController {    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);    /\*\*  \* Simply selects the home view to render by returning its name.  \*/  //Main Page  @RequestMapping(value = "/", method = RequestMethod.GET)  public String home(HttpServletRequest request, Locale locale, Model model) {  logger.info("Welcome home! The client locale is {}.", locale);    Date date = new Date();  DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG, DateFormat.LONG, locale);    String formattedDate = dateFormat.format(date);    model.addAttribute("serverTime", formattedDate );    return "home";  }      @RequestMapping(value = "/goHome", method = RequestMethod.POST)  public String home(HttpServletRequest request) {    return "home";  }    //switch to menu page according to user-role  @RequestMapping(value = "/login.htm", method = RequestMethod.POST)  public ModelAndView loginToMenu(HttpServletRequest request){  ModelAndView mv = null;  String userName = request.getParameter("j\_username");  String userPassword = request.getParameter("j\_password");  Configuration cfg = new Configuration();  SessionFactory sf = cfg.configure().buildSessionFactory();  Session hibernatesession = sf.openSession();      //HttpSession session = request.getSession(true);  System.out.println("In Check login");    //String SQL\_QUERY =" from User as o where o.userName=? and o.userPassword=?";  String SQL\_QUERY = " FROM User WHERE userName= :userName AND userPassword= :userPassword";  Query query = hibernatesession.createQuery(SQL\_QUERY);  query.setParameter("userName",userName);  query.setParameter("userPassword",userPassword);  User result = (User)query.uniqueResult();  HttpSession sessionH = request.getSession();  User signedUser = new User();        if (result == null) {  hibernatesession.close();  mv = new ModelAndView("login-error");  return mv;  }  else if(result != null){  if(result.getUserRole().equals("Admin")){  hibernatesession.close();  signedUser = result;  return mv = new ModelAndView("admin/adminMenu","user",result);  }  else if(result.getUserRole().equals("Librarian")){  hibernatesession.close();  signedUser = result;  return mv = new ModelAndView("librarian/librarianMenu", "user",result);  }  else if(result.getUserRole().equals("LibraryMember") ){    int userId = result.getUserId();  SQL\_QUERY = " FROM LibraryMember WHERE userId= :userId";  query = hibernatesession.createQuery(SQL\_QUERY);  query.setParameter("userId",userId);  LibraryMember member = (LibraryMember)query.uniqueResult();  hibernatesession.close();  signedUser = result;  sessionH.setAttribute("signedUser", result);  return mv = new ModelAndView("member/memberMenu","user",result);  }      }    return mv;    }        /\*public ModelAndView home(HttpServletRequest request, BindingResult result, Map model) {  Configuration cfg = new Configuration();  SessionFactory sf = cfg.configure().buildSessionFactory();  Session hibernatesession = sf.openSession();  HttpSession session = request.getSession(true);  ModelAndView mv = null;  if (result.hasErrors()) {  mv = new ModelAndView("loginform");  return mv;  }    String userName = request.getParameter("j\_username");  String userPassword = request.getParameter("j\_password");    boolean userExists = loginService.checkLogin(userName,userPassword);  if(userExists){  User findUser = loginService.getUser(userExists, userName, userPassword);  if(findUser.getUserRole()== "admin"){    return mv = new ModelAndView("adminMenu","user",findUser);  }  else if(findUser.getUserRole() == "Librarian"){  return mv = new ModelAndView("librarianMenu");  }  else if(findUser.getUserRole() == "LibraryMember"){  return mv = new ModelAndView("memberMenu");  }      }else{  result.rejectValue("userName","invaliduser");  mv = new ModelAndView("login-error");  return mv;  }  return mv;    }\*/        } |
| LibrarianController  All libraian related functions | package com.Li.controller;  import java.io.File;  import java.io.IOException;  import java.util.ArrayList;  import java.util.HashSet;  import java.util.List;  import java.util.Set;  import javax.servlet.ServletContext;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpSession;  import org.apache.commons.mail.DefaultAuthenticator;  import org.apache.commons.mail.Email;  import org.apache.commons.mail.EmailException;  import org.apache.commons.mail.SimpleEmail;  import org.hibernate.Criteria;  import org.hibernate.HibernateException;  import org.hibernate.Query;  import org.hibernate.Session;  import org.hibernate.SessionFactory;  import org.hibernate.cfg.Configuration;  import org.hibernate.criterion.Projections;  import org.hibernate.criterion.Restrictions;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.beans.factory.annotation.Qualifier;  import org.springframework.stereotype.Controller;  import org.springframework.ui.ModelMap;  import org.springframework.validation.BindingResult;  import org.springframework.web.bind.WebDataBinder;  import org.springframework.web.bind.annotation.InitBinder;  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.multipart.commons.CommonsMultipartFile;  import org.springframework.web.servlet.ModelAndView;  import com.Li.dao.BookDAO;  import com.Li.dao.EventDAO;  import com.Li.dao.InventoryDAO;  import com.Li.dao.MovieDAO;  import com.Li.dao.MusicDAO;  import com.Li.exception.EventException;  import com.Li.exception.InventoryException;  import com.Li.pojos.inventory.Book;  import com.Li.pojos.inventory.Event;  import com.Li.pojos.inventory.Inventory;  import com.Li.pojos.inventory.Movie;  import com.Li.pojos.inventory.Music;  import com.Li.pojos.roles.LibraryMember;  import com.Li.pojos.roles.UEmail;  import com.Li.pojos.roles.User;  import com.Li.validator.BookValidator;  import com.Li.validator.EventValidator;  import com.Li.validator.InventoryValidator;  import com.Li.validator.MovieValidator;  import com.Li.validator.MusicValidator;  //menu switch page  @Controller  //@RequestMapping("librarian")  public class LibrarianController {    //autowired start  @Autowired  @Qualifier("musicDao")  MusicDAO musicDao;  @Autowired  @Qualifier("musicValidator")  MusicValidator mvalidator;  @InitBinder  private void initMBinder(WebDataBinder mbinder) {  mbinder.setValidator(mvalidator);  }  @Autowired  @Qualifier("movieDao")  MovieDAO movieDao;  @Autowired  @Qualifier("movieValidator")  MovieValidator movievalidator;  @InitBinder  private void initMovieBinder(WebDataBinder moviebinder) {  moviebinder.setValidator(movievalidator);  }      @Autowired  @Qualifier("eventDao")  EventDAO eventDao;  @Autowired  @Qualifier("eventValidator")  EventValidator eventvalidator;  @InitBinder  private void initeventBinder(WebDataBinder eventbinder) {  eventbinder.setValidator(eventvalidator);  }      @Autowired  @Qualifier("bookDao")  BookDAO bookDao;    @Autowired  @Qualifier("bookValidator")  BookValidator validator;  @InitBinder  private void initBinder(WebDataBinder binder) {  binder.setValidator(validator);  }  @Autowired  ServletContext servletContext;    @Autowired  @Qualifier("inventoryDao")  InventoryDAO inventoryDao;    /\* @Autowired  @Qualifier("inventoryValidator")  InventoryValidator inventoryValidator;  @InitBinder  private void initInventoryBinder(WebDataBinder inventorybinder) {  inventorybinder.setValidator(inventoryValidator);  }  \*/    //autowired ends    //menu switch    @RequestMapping(value = "librarianMenu.htm", method = RequestMethod.POST)  public ModelAndView adminMenu( HttpServletRequest request) {    String result = request.getParameter("isselect");  ModelAndView mv = null;  /\*List<Book> bookList = null;  List<Music> musicList = null;  List<Movie> movieList = null; \*/    if (result.equals("Browse Inventory")) {  List<Inventory> inventoryList = null;  try {  inventoryList = inventoryDao.list();  } catch (InventoryException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  mv = new ModelAndView("browseInventory","inventoryList",inventoryList);  }    else if (result.equals("Add Inventory")){  mv = new ModelAndView("numberToAdd");  }  else if (result.equals("Delete Inventory/Event")){  mv = new ModelAndView("deleteInventory");  }  else if (result.equals("Browse Event")){  List<Event> events=null;  try {  events= eventDao.list();  } catch (EventException e) {  e.printStackTrace();  }  mv = new ModelAndView("browseEvent","events",events);  }  else if (result.equals("Add Event")){  /////////////////////////////////////////////////////////////  System.out.println("in get method");  mv = new ModelAndView("addEvent");  }  else if (result.equals("Borrow Inventory/Event")){  System.out.println("inborrow");  mv = new ModelAndView("borrow");  }  else{  System.out.println("outside options");}      return mv;  }  /\*-----------------Inventory related------------------------\*/    @RequestMapping(value ="librarian/addInventoryNumber.htm",method = RequestMethod.POST)  public ModelAndView goToAddNumber(HttpServletRequest request){  ModelAndView mv = null;  int numBook = Integer.valueOf(request.getParameter("numBook"));  int numMovies = Integer.valueOf(request.getParameter("numMovies"));  int numMusic = Integer.valueOf(request.getParameter("numMusic"));  int[] numberAdd = new int[3];  numberAdd[0] = numBook;  numberAdd[1] = numMovies;  numberAdd[2] = numMusic;  System.out.println(numberAdd[0]);  mv = new ModelAndView("addInventoryNumber","numberAdd",numberAdd);  return mv;  }      @RequestMapping(value ="librarian/addSuccessfully.htm",method = RequestMethod.POST)  public ModelAndView addInventory(HttpServletRequest request, int[] numberAdd){  ModelAndView mv = null;  Configuration cfg = new Configuration();  SessionFactory sf = cfg.configure("hibernate.cfg.xml").buildSessionFactory();  Session session = sf.openSession();  String[] isbn;  String[] title;  String[] authors;    String[] musicTitle;  String[] musicAuthor;  String[] musicGenre;    String[] movieTitle;  String[] movieActor;  String[] movieActress;  String[] movieGenre;  String[] movieYear;  String[] movieDescription;      ArrayList<Book> bookList = new ArrayList<Book>();  ArrayList<Music> musicList = new ArrayList<Music>();  ArrayList<Movie> movieList = new ArrayList<Movie>();  int number = Integer.valueOf(request.getParameter("numBook"));  System.out.println(number);  int numberMusic = Integer.valueOf(request.getParameter("numMusic"));  int numberMovie = Integer.valueOf(request.getParameter("numMovies"));  for (int i = 0; i < number; i++) {  Book book = new Book();  isbn = request.getParameterValues("isbn");  title = request.getParameterValues("title");  authors = request.getParameterValues("authors");    book.setIsbn(isbn[i]);  book.setTitle(title[i]);  book.setAuthor(authors[i]);    bookList.add(book);    }  for (int i = 0; i < numberMusic; i++){  Music music = new Music();  musicTitle = request.getParameterValues("musictitle");  musicAuthor = request.getParameterValues("musicauthor");  musicGenre = request.getParameterValues("musicgenre");    music.setAuthor(musicTitle[i]);  music.setGenre(musicGenre[i]);  music.setAuthor(musicAuthor[i]);    musicList.add(music);  }    for (int i = 0; i < numberMovie; i++){  Movie movie = new Movie();  movieTitle = request.getParameterValues("movietitle");  movieActor = request.getParameterValues("movieactor");  movieActress = request.getParameterValues("movieactress");  movieGenre = request.getParameterValues("movieGenre");  movieYear = request.getParameterValues("movieyear");  movieDescription = request.getParameterValues("moviedescription");    movie.setTitle(movieTitle[i]);  movie.setActor(movieActor[i]);  movie.setGenre(movieGenre[i]);  movie.setActress(movieActress[i]);  movie.setYear(Integer.valueOf(movieYear[i]));  movie.setTitle(movieTitle[i]);  movie.setTitle(movieDescription[i]);    movieList.add(movie);  }    try{  session.beginTransaction();  for(Book book:bookList){  session.save(book);  }  for(Movie movie:movieList){  session.save(movie);  }  for(Music music:musicList){  session.save(music);  }  session.getTransaction().commit();  }catch(HibernateException e){  System.out.println("Cannot create inventories! " + e);  session.getTransaction().rollback();  }finally{  session.close();  }    mv = new ModelAndView("addSuccessfully","numberAdd",numberAdd);  return mv;        }    @RequestMapping(value ="deleteInventory.htm",method = RequestMethod.POST)  public ModelAndView deleteInventory(HttpServletRequest request){  ModelAndView mv =null;  int inventoryID = Integer.valueOf(request.getParameter("inventoryID"));  String warning = "NO such Item with this ID";  String inventoryType = request.getParameter("isselect");  if(inventoryType.equals("Event")){  int eventID = Integer.valueOf(request.getParameter("eventID"));  try {  Event event = eventDao.getEvent(eventID);  if (event == null){  mv = new ModelAndView("deleteInventory", "warning",warning);  }  else{  eventDao.delete(event);  mv = new ModelAndView("/librarian/librarianMenu");  }    } catch (EventException e) {  e.printStackTrace();  }    }  else if(inventoryType.equals("Inventory")){}  Inventory inventory;  try {    inventory = inventoryDao.get(inventoryID);    if(inventory == null){    mv = new ModelAndView("deleteInventory", "warning",warning);  }    else{  inventoryDao.delete(inventory);  mv = new ModelAndView("/librarian/librarianMenu");  }    } catch (InventoryException e) {  e.printStackTrace();  }      return mv;  }    @RequestMapping(value ="borrowInventory.htm",method = RequestMethod.POST)  public ModelAndView borrowInventory(HttpServletRequest request){  ModelAndView mv =null;  int inventoryID = Integer.valueOf(request.getParameter("inventoryID"));  String warning = "NO such Item with this ID";  String inventoryType = request.getParameter("isselect");  Integer userID = Integer.valueOf(request.getParameter("userID"));  Configuration cfg = new Configuration();  SessionFactory sf = cfg.configure().buildSessionFactory();  Session hibernatesession = sf.openSession();  Query query2=hibernatesession.createQuery("from LibraryMember where userId= :userId");  query2.setParameter("userId",userID);  LibraryMember member= (LibraryMember)query2.uniqueResult();  if(inventoryType.equals("Event")){    try {    Query query=hibernatesession.createQuery("from Event where eventID= :eventID");  query.setParameter("eventID",inventoryID);  Event event = (Event)query.uniqueResult();  System.out.println("event"+event.getEventID());  //Event event = eventDao.getEvent(eventID);  if (event == null){  warning = "no such event";  mv = new ModelAndView("borrow", "warning",warning);  }  else{  /\*System.out.println("member:"+member.getUserId());  System.out.println("event:"+event.getEventID());\*/  event.getRegisteredUsers().add(member);  member.getEventsRegistered().add(event);  //eventDao.update(event);  warning = "registered the event successfully!";  mv = new ModelAndView("borrow", "warning",warning);  }    } catch (Exception e) {  e.printStackTrace();  }    }  else if(inventoryType.equals("Inventory")){    Inventory inventory;    try {    inventory = inventoryDao.get(inventoryID);  if(inventory == null){    mv = new ModelAndView("borrow", "warning",warning);  }    else{    inventory.setUser(member);  inventory.setAvailable(false);  inventoryDao.update(inventory);  System.out.println("inventory availablility after update"+inventory.isAvailable());  warning = "borrowed book successfully!";  mv = new ModelAndView("borrow", "warning",warning);  }    } catch (InventoryException e) {  e.printStackTrace();  }    }    else if(inventoryType.equals("ReturnInventory")){    try {  Inventory inventory;  inventory = inventoryDao.get(inventoryID);    if(inventory == null){    mv = new ModelAndView("borrow", "warning",warning);  }    else{  inventory.setAvailable(true);  inventory.setUser(null);  inventoryDao.update(inventory);  warning = "Returned book successfully!";  mv = new ModelAndView("borrow", "warning",warning);  }    } catch (InventoryException e) {  e.printStackTrace();  }  }    hibernatesession.close();  return mv;  }    /\*ReturnInventory  \* Event related  \* \*/  @RequestMapping(value ="librarian/sendEmail.htm", method = RequestMethod.POST)  public ModelAndView sendEmail(HttpServletRequest request) throws EmailException{  ModelAndView mv = new ModelAndView();  Email email = new SimpleEmail();  try {  email.setHostName("smtp.googlemail.com");  email.setSmtpPort(587);  email.setAuthenticator(  new DefaultAuthenticator("lig098abc@gmail.com", "54lilyGL"));  email.setSSLOnConnect(true); // disable in case of EmailException  email.setFrom("CarnegieLibrary@lib.com", "Carnegie Library");  email.setSubject("Library News");  email.setMsg("We are pleased to inform you that a new event posted in the Library.");  email.addTo("lig098abc@gmail.com");  email.send();  } catch(EmailException ee) {  ee.printStackTrace();  }  /\*  HttpSession session = request.getSession();  String userName = (String) session.getAttribute("userName");  Configuration config = new Configuration();  SessionFactory sf = config.configure().buildSessionFactory();  Session hsession = sf.openSession();  Query query = hsession.createQuery("SELECT UEmail.emailAddress from UEmail");  List<String> users = query.list();  System.out.println("chech user id"+users.get(2).toString());  // Query query2 = hsession.createQuery("SELECT from Event where ");  Criteria crit = hsession.createCriteria(User.class);  List<User> users = crit.list();  Criteria crit2 = hsession.createCriteria(Event.class);  Event event = (Event)crit2.uniqueResult();  Criteria crit2 = hsession.createCriteria(Event.class);  Event event = (Event)crit2.uniqueResult();  // System.out.println("chech event"+event.getDescription());  crit2.setProjection(Projections.max("eventID"));  if(event == null){  System.out.print("no event to send");  }  else{  \*/  // for(String u:users){  /\*// System.out.print("Email:"+u);  Email email = new SimpleEmail();  email.setHostName("smtp.googlemail.com");//If a server is capable of sending email, then you don't need the authentication. In this case, an email server needs to be running on that machine. Since we are running this application on the localhost and we don't have a email server, we are simply asking gmail to relay this email.  //email.setSmtpPort(465);  String authuser = "user";  String authpwd = "pass";  email.setAuthenticator(new DefaultAuthenticator(authuser, authpwd));  // properties to configure encryption  MailSession mailSession = email.getMailSession();  getProperties().put("mail.smtps.auth", "true");  email.getMailSession().getProperties().put("mail.debug", "true");  email.getMailSession().getProperties().put("mail.smtps.port", "587");  email.getMailSession().getProperties().put("mail.smtps.socketFactory.port", "587");  email.getMailSession().getProperties().put("mail.smtps.socketFactory.class", "javax.net.ssl.SSLSocketFactory");  email.getMailSession().getProperties().put("mail.smtps.socketFactory.fallback", "false");  email.getMailSession().getProperties().put("mail.smtp.starttls.enable", "true");  //email.setAuthenticator(new DefaultAuthenticator("User","User"));  email.setSSLOnConnect(true);  email.setFrom("ClientService@library.edu");//This email will appear in the from field of the sending email. It doesn't have to be a real email address.This could be used for phishing/spoofing!  email.setSubject("Notice Mail");//+event.getDescription()  email.setMsg("There is a new Event in carnegie library:"+"/t");  try {  email.addTo("lig098abc@gmail.com");  } catch (Exception e) {  // TODO Auto-generated catch block  e.printStackTrace();  }//Will come from the database  email.send();    // }  // }  // hsession.close();  \*/  mv = new ModelAndView("librarian/librarianMenu");  return mv;    }  /\*////////////////////////////////////add event///////////////////////////////////////////  @RequestMapping(value = "librarian/addEvent.htm", method = RequestMethod.GET)  public ModelAndView createModel(){  return new ModelAndView("addEvent","event",new Event());  }  @RequestMapping(value = "librarian/addEvent.htm", method = RequestMethod.POST)  public ModelAndView handleUpload(@ModelAttribute("event") Event event,BindingResult result,HttpServletRequest request) {  System.out.println("in add event controller");  ModelAndView mv;  try {  if (event.getFilename().trim() != "" || event.getFilename() != null) {  File directory;  String check = File.separator; // Checking if system is linux  // based or windows based by  // checking seprator used.  String path = null;  if (check.equalsIgnoreCase("\\")) {  path = servletContext.getRealPath("").replace("build\\", ""); // gives real path as Lab9/build/web/  // so we need to replace build in the path  }  if (check.equalsIgnoreCase("/")) {  path = servletContext.getRealPath("").replace("build/", "");  path += "/"; // Adding trailing slash for Mac systems.  }  directory = new File(path + "\\" + event.getFilename());  boolean temp = directory.exists();  if (!temp) {  temp = directory.mkdir();  }  if (temp) {  // We need to transfer to a file  CommonsMultipartFile photoInMemory = event.getPoster();  String fileName = photoInMemory.getOriginalFilename();  // could generate file names as well  File localFile = new File(directory.getPath(), fileName);  // move the file from memory to the file  photoInMemory.transferTo(localFile);  event.setDescription(localFile.getPath());  System.out.println("File is stored at" + localFile.getPath());  System.out.print("registerNewUser");  Event u = eventDao.create(event);  } else {  System.out.println("Failed to create directory!");  }  }  } catch (IllegalStateException e) {  System.out.println("\*\*\* IllegalStateException: " + e.getMessage());  } catch (IOException e) {  // TODO Auto-generated catch block  System.out.println("\*\*\* IOException: " + e.getMessage());  } catch (EventException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  return mv = new ModelAndView("addSuccessfully");  }  \*/  } |
| MemberController  All member related functions | package com.Li.controller;  import java.util.HashSet;  import java.util.List;  import java.util.Set;  import javax.servlet.ServletContext;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpSession;  import org.hibernate.Criteria;  import org.hibernate.Query;  import org.hibernate.Session;  import org.hibernate.SessionFactory;  import org.hibernate.cfg.Configuration;  import org.hibernate.criterion.Restrictions;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.beans.factory.annotation.Qualifier;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.WebDataBinder;  import org.springframework.web.bind.annotation.InitBinder;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import org.springframework.web.servlet.ModelAndView;  import com.Li.dao.BookDAO;  import com.Li.dao.EventDAO;  import com.Li.dao.InventoryDAO;  import com.Li.dao.MovieDAO;  import com.Li.dao.MusicDAO;  import com.Li.dao.UserDAO;  import com.Li.exception.BookException;  import com.Li.exception.EventException;  import com.Li.exception.InventoryException;  import com.Li.pojos.inventory.Event;  import com.Li.pojos.inventory.Inventory;  import com.Li.pojos.roles.LibraryMember;  import com.Li.pojos.roles.User;  import com.Li.validator.BookValidator;  import com.Li.validator.EventValidator;  import com.Li.validator.InventoryValidator;  import com.Li.validator.MovieValidator;  import com.Li.validator.MusicValidator;  //menu switch page  @Controller  //@RequestMapping("librarian")  public class MemberController {    //menu switch    @RequestMapping(value = "memberMenu.htm", method = RequestMethod.POST)  public ModelAndView adminMenu( HttpServletRequest request, User user) throws InventoryException {  String result = request.getParameter("isselect");  ModelAndView mv = null;    if (result.equals("Browse My Events")) {  User u =(User) request.getSession().getAttribute("signedUser");  int userID = u.getUserId();  Configuration cfg = new Configuration();  SessionFactory sf = cfg.configure().buildSessionFactory();  Session hibernatesession = sf.openSession();    Query query2=hibernatesession.createQuery("from LibraryMember where userId= :userId");  query2.setParameter("userId",userID);  System.out.println("userId"+userID);  LibraryMember member= (LibraryMember)query2.uniqueResult();  System.out.println("here"+member.getUserId());  hibernatesession.close();  Set<Event> events = member.getEventsRegistered();  mv = new ModelAndView("browseEvent","events",events);  }  else if (result.equals("Browse My Book&Medias")){  User u =(User) request.getSession().getAttribute("signedUser");  int userID = u.getUserId();  Configuration cfg = new Configuration();  SessionFactory sf = cfg.configure().buildSessionFactory();  Session hibernatesession = sf.openSession();  Query query2=hibernatesession.createQuery("from LibraryMember where userId= :userId");  query2.setParameter("userId",userID);  System.out.println("userId"+userID);  LibraryMember member= (LibraryMember)query2.uniqueResult();  System.out.println("here"+member.getUserId());  Set<Inventory> inventoryList = member.getBorrowedInventory();  System.out.println("here2"+inventoryList.isEmpty());  mv = new ModelAndView("browseInventory","inventoryList",inventoryList);  hibernatesession.close();  }  else if (result.equals("Browse All available Events")){  try {    Configuration cfg = new Configuration();  SessionFactory sf = cfg.configure().buildSessionFactory();  Session hibernatesession = sf.openSession();  Query query2=hibernatesession.createQuery("from Event where isExpried= :isExpried");  query2.setParameter("isExpried",false);  List<Event> events = (List<Event>)query2.list();  mv = new ModelAndView("browseEvent","events",events);  } catch (Exception e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    }  else if (result.equals("Browse All available Book&Medias")){  Configuration cfg = new Configuration();  SessionFactory sf = cfg.configure().buildSessionFactory();  Session hibernatesession = sf.openSession();  Query query2=hibernatesession.createQuery("from Inventory where isAvailable = true");  List<Inventory> inventoryList = (List<Inventory>)query2.list();  System.out.println("inventoryList check"+inventoryList.get(0));  mv = new ModelAndView("browseInventory","inventoryList",inventoryList);  }    return mv;  }  /\*-----------------Inventory related------------------------\*/    } |