Gaurav Panchal 001447046

Shopping cart

A website where a user can register as a buyer or a seller. The seller can create products’ advertisements and make them available for the buyers. The buyer will be able to view all the products and add the products to their cart. Upon registration the user will get a confirmation email and upon adding the product to the cart and while checking out, the buyer will get a downloadable pdf that will contain all the billing information.

Technologies used:

User defined Exceptions: Created exception classes.

Interceptor: to handle error pages

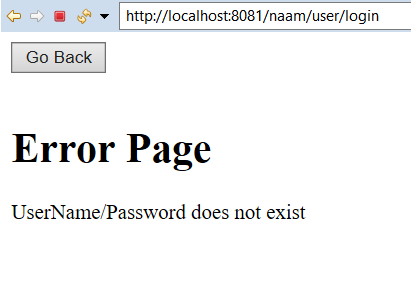
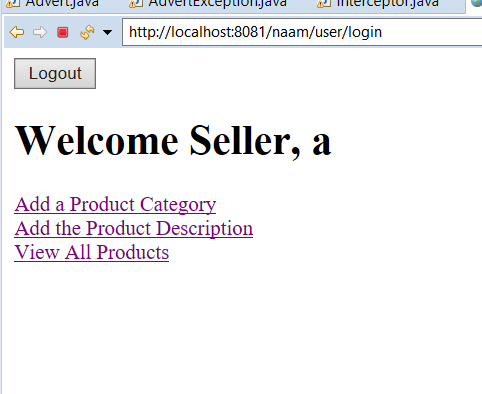
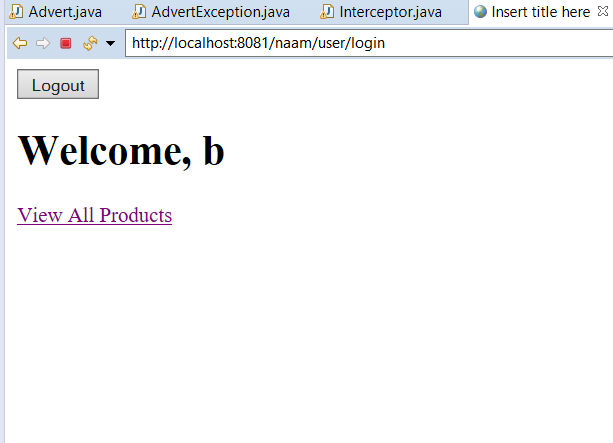
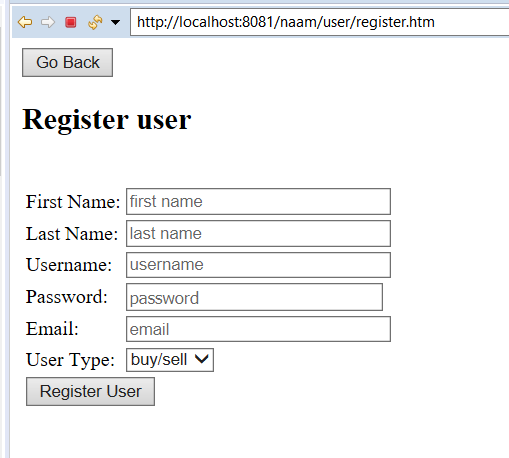
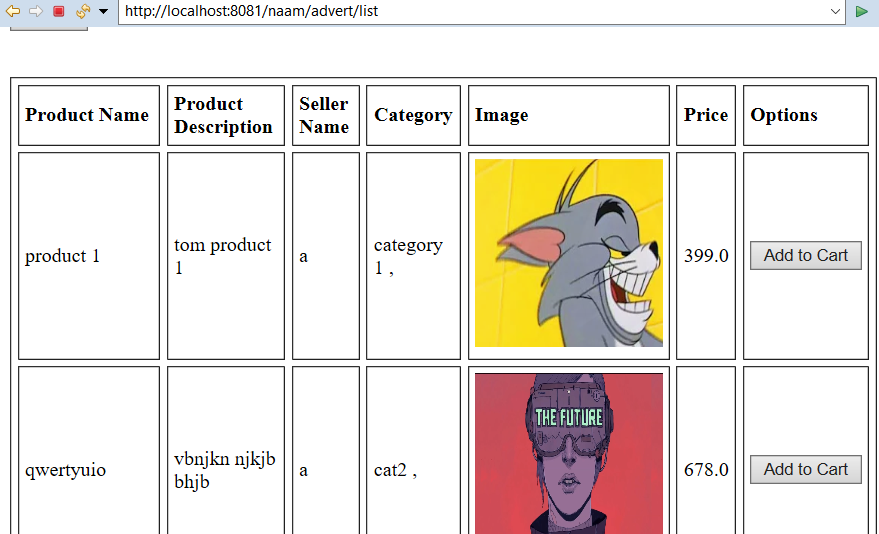
Email, Pdf view

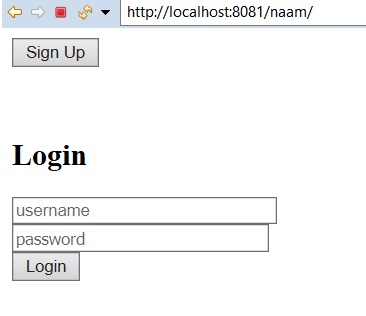
Spring mvc, hibernate,

Brief description of tasks each role could perform:

Buyer: register, login, logout, view products, add to cart, checkout (this will generate a downloadable pdf that has all the billing information)

Seller: register, login, logout, view products, add category, add product.





package com.neu.naam.controller;

import java.io.File;

import java.util.LinkedHashMap;

import java.util.List;

import java.util.Map;

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.validation.BindingResult;

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.SessionAttributes;

import org.springframework.web.multipart.commons.CommonsMultipartFile;

import org.springframework.web.servlet.ModelAndView;

import com.neu.naam.dao.AdvertDAO;

import com.neu.naam.dao.CategoryDAO;

import com.neu.naam.dao.UserDAO;

import com.neu.naam.exception.AdvertException;

import com.neu.naam.pojo.Advert;

import com.neu.naam.pojo.Cart;

import com.neu.naam.pojo.Category;

import com.neu.naam.pojo.User;

@Controller

@RequestMapping("/advert/\*")

public class AdvertController {

@Autowired

@Qualifier("advertDao")

AdvertDAO advertDao;

@Autowired

@Qualifier("categoryDao")

CategoryDAO categoryDao;

@Autowired

@Qualifier("userDao")

UserDAO userDao;

@Autowired

ServletContext servletContext;

@RequestMapping(value = "/advert/add", method = RequestMethod.POST)

public ModelAndView addCategory(@ModelAttribute("advert") Advert advert, BindingResult result) throws Exception {

try {

User u = userDao.get(advert.getPostedBy());

advert.setUser(u);

advert = advertDao.create(advert);

for(Category c: advert.getCategories()){

c = categoryDao.get(c.getTitle());

c.getAdverts().add(advert);

categoryDao.update(c); //to maintain many to many relationship

}

if (advert.getFilename().trim() != "" || advert.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 + "\\" + advert.getFilename());

boolean temp = directory.exists();

if (!temp) {

temp = directory.mkdir();

}

if (temp) {

// We need to transfer to a file

CommonsMultipartFile photoInMemory = advert.getPhoto();

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);

advert.setFilename(localFile.getPath());

System.out.println("File is stored at" + localFile.getPath());

System.out.print("registerNewUser");

Advert a = advertDao.create(advert);

} else {

System.out.println("Failed to create directory!");

}

}

return new ModelAndView("advert-success", "advert", advert);

} catch (AdvertException e) {

System.out.println(e.getMessage());

return new ModelAndView("error", "errorMessage", "error while login");

}

}

@RequestMapping(value = "/advert/list", method = RequestMethod.GET)

public ModelAndView addCategory(HttpServletRequest request) throws Exception {

ModelAndView mav = new ModelAndView("advert-list");

List<Advert> adverts = advertDao.list();

mav.addObject("adverts", adverts);

mav.addObject("cart", new Cart());

return mav;

}

@RequestMapping(value = "/advert/sellerlist", method = RequestMethod.GET)

public ModelAndView addCategories(HttpServletRequest request) throws Exception {

try {

List<Advert> adverts = advertDao.list();

return new ModelAndView("seller-advert-list", "adverts", adverts);

} catch (AdvertException e) {

System.out.println(e.getMessage());

return new ModelAndView("error", "errorMessage", "error while login");

}

}

@RequestMapping(value="/advert/add", method = RequestMethod.GET)

public ModelAndView initializeForm(HttpServletRequest request) throws Exception {

ModelAndView mv = new ModelAndView();

mv.addObject("categories", categoryDao.list());

mv.addObject("advert", new Advert());

mv.setViewName("advert-form");

return mv;

}

}

package com.neu.naam.controller;

import java.text.DateFormat;

import java.util.Date;

import java.util.HashMap;

import java.util.List;

import java.util.Locale;

import java.util.Map;

import javax.servlet.ServletContext;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.springframework.web.bind.ServletRequestUtils;

import org.springframework.web.servlet.ModelAndView;

import org.springframework.web.servlet.View;

import org.springframework.web.servlet.mvc.AbstractController;

import com.neu.naam.dao.AdvertDAO;

import com.neu.naam.dao.CartDAO;

import com.neu.naam.dao.CategoryDAO;

import com.neu.naam.dao.DAO;

import com.neu.naam.dao.UserDAO;

import com.neu.naam.pojo.Cart;

import com.neu.naam.pojo.PDFView;

import com.neu.naam.pojo.User;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

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

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

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

@Controller

@RequestMapping("/cart/\*")

public class BillController extends PDFView{

//private static final Logger logger = LoggerFactory.getLogger(BillController.class);

@Autowired

@Qualifier("advertDao")

AdvertDAO advertDao;

@Autowired

@Qualifier("categoryDao")

CategoryDAO categoryDao;

@Autowired

@Qualifier("userDao")

UserDAO userDao;

@Autowired

@Qualifier("cartDao")

CartDAO cartDao;

@Autowired

ServletContext servletContext;

@RequestMapping(value = "/cart/checkout", method = RequestMethod.POST)

public ModelAndView showPdfReport(@ModelAttribute("cart") Cart cart,

ModelMap model,

BindingResult result,

HttpServletRequest request) throws Exception

{

List<Cart> view=cartDao.list();

model.addAttribute("cartitems", view);

//return new ModelAndView("view","cartitems",view);

View v = new PDFView();

return new ModelAndView(v);

}

}

package com.neu.naam.controller;

import java.util.List;

import javax.servlet.ServletContext;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

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

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.stereotype.Controller;

import org.springframework.validation.BindingResult;

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.servlet.ModelAndView;

import com.neu.naam.dao.AdvertDAO;

import com.neu.naam.dao.CategoryDAO;

import com.neu.naam.dao.DAO;

import com.neu.naam.dao.UserDAO;

import com.neu.naam.dao.CartDAO;

import com.neu.naam.pojo.Advert;

import com.neu.naam.pojo.Cart;

import com.neu.naam.pojo.User;

@Controller

@RequestMapping("/cart/\*")

public class CartController extends DAO{

@Autowired

@Qualifier("advertDao")

AdvertDAO advertDao;

@Autowired

@Qualifier("categoryDao")

CategoryDAO categoryDao;

@Autowired

@Qualifier("userDao")

UserDAO userDao;

@Autowired

@Qualifier("cartDao")

CartDAO cartDao;

@Autowired

ServletContext servletContext;

@RequestMapping(value = "/cart/insert", method = RequestMethod.POST)

public ModelAndView addCategory(@ModelAttribute("cart") Cart cart, BindingResult result, HttpServletRequest request) throws Exception {

HttpSession session = (HttpSession) request.getSession();

User u = (User)session.getAttribute("user");

cart.setUser(u);

u.setCart(cart);

Cart cd=null;

// Cart c = cartDao.insert(cart);

// User user1 = cartDao.update(u);

// getSession().update(u);

List<Cart> c=cartDao.list();

int i=0;

Cart cw = null;

for(Cart cc:c){

if(u.getPersonID()==cc.getId()){

cw = cartDao.updateCart(cc);

i=1;

return new ModelAndView("user-cart","c",cw);

}

}

if (i==0){

cd = cartDao.insert(cart);

}

return new ModelAndView("user-cart","c",cd);

}

}

package com.neu.naam.controller;

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

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.stereotype.Controller;

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.bind.annotation.SessionAttributes;

import org.springframework.web.servlet.ModelAndView;

import com.neu.naam.dao.CategoryDAO;

import com.neu.naam.exception.CategoryException;

import com.neu.naam.pojo.Category;

import com.neu.naam.validator.CategoryValidator;

@Controller

@RequestMapping("/category/\*")

public class CategoryController {

@Autowired

@Qualifier("categoryValidator")

CategoryValidator categoryValidator;

@Autowired

@Qualifier("categoryDao")

CategoryDAO categoryDAO;

@InitBinder

private void initBinder(WebDataBinder binder) {

binder.setValidator(categoryValidator);

}

@RequestMapping(value = "/category/add", method = RequestMethod.POST)

public ModelAndView addCategory(@ModelAttribute("category") Category category, BindingResult result) throws Exception {

categoryValidator.validate(category, result);

if (result.hasErrors()) {

return new ModelAndView("category-form", "category", category);

}

try {

category = categoryDAO.create(category.getTitle());

} catch (CategoryException e) {

System.out.println(e.getMessage());

return new ModelAndView("error", "errorMessage", "error while login");

}

return new ModelAndView("category-success", "category", category);

}

@RequestMapping(value="/category/add", method = RequestMethod.GET)

public ModelAndView initializeForm() throws Exception {

return new ModelAndView("category-form", "category", new Category());

}

}

package com.neu.naam.controller;

import javax.servlet.ServletContext;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

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

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.stereotype.Controller;

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

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

import com.neu.naam.dao.AdvertDAO;

import com.neu.naam.dao.CategoryDAO;

import com.neu.naam.dao.UserDAO;

@Controller

@RequestMapping("/advert/\*")

public class LogoutController {

@Autowired

@Qualifier("advertDao")

AdvertDAO advertDao;

@Autowired

@Qualifier("categoryDao")

CategoryDAO categoryDao;

@Autowired

@Qualifier("userDao")

UserDAO userDao;

@Autowired

ServletContext servletContext;

@RequestMapping(value = "/advert/logout", method = RequestMethod.POST)

public String logout(HttpServletRequest request){

HttpSession httpSession = request.getSession();

httpSession.invalidate();

return "logout";

}

}

package com.neu.naam.controller;

import java.util.HashMap;

import java.util.LinkedHashMap;

import java.util.Map;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

import org.apache.commons.mail.Email;

import org.apache.commons.mail.SimpleEmail;

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

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.stereotype.Controller;

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.servlet.ModelAndView;

import com.neu.naam.dao.UserDAO;

import com.neu.naam.exception.UserEx

ception;

import com.neu.naam.pojo.User;

import com.neu.naam.validator.UserValidator;

@Controller

@RequestMapping("/user/\*")

public class UserController {

@Autowired

@Qualifier("userDao")

UserDAO userDao;

@Autowired

@Qualifier("userValidator")

UserValidator validator;

@InitBinder

private void initBinder(WebDataBinder binder) {

binder.setValidator(validator);

}

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

protected String goToUserHome(HttpServletRequest request) throws Exception {

return "user-home";

}

@RequestMapping(value = "/user/login", method = RequestMethod.POST)

protected String loginUser(HttpServletRequest request) throws Exception {

HttpSession session = (HttpSession) request.getSession();

try {

System.out.print("loginUser");

User u = userDao.get(request.getParameter("username"), request.getParameter("password"));

if(u == null){

System.out.println("UserName/Password does not exist");

session.setAttribute("errorMessage", "UserName/Password does not exist");

return "error";

}

else if(u.getUsertype().equals("Buyer")){

session.setAttribute("user", u);

return "buyer-home";

}

else if(!(u.getUsertype().equals("Buyer"))&&!(u.getUsertype().equals("Seller"))){

session.setAttribute("errorMessage", "UserName/Password does not exist");

return "error";

}

else{

session.setAttribute("user", u);

return "user-home";

}

//return "user-home";

} catch (UserException e) {

System.out.println("Exception: " + e.getMessage());

session.setAttribute("errorMessage", "error while login");

return "error";

}

}

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

protected ModelAndView registerUser() throws Exception {

System.out.print("registerUser");

ModelAndView mav = new ModelAndView("register-user");

Map<String,String> usertype = new LinkedHashMap<String,String>();

usertype.put("Buyer", "Buyer");

usertype.put("Seller", "Seller");

mav.addObject("usertype", usertype);

mav.addObject("user", new User());

return mav;

//return new ModelAndView("", "user", new User());

}

@RequestMapping(value = "/user/register", method = RequestMethod.POST)

protected ModelAndView registerNewUser(HttpServletRequest request, @ModelAttribute("user") User user, BindingResult result) throws Exception {

validator.validate(user, result);

if (result.hasErrors()) {

ModelAndView mav = new ModelAndView("register-user");

Map<String,String> usertype = new LinkedHashMap<String,String>();

usertype.put("Buyer", "Buyer");

usertype.put("Seller", "Seller");

mav.addObject("usertype", usertype);

mav.addObject("user", user);

return mav;

//return new ModelAndView("register-user", "user", user);

}

try {

System.out.print("registerNewUser");

User u = userDao.register(user);

request.getSession().setAttribute("user", u);

Email email= new SimpleEmail();

email.setHostName("smtp.googlemail.com");

email.setSmtpPort(465);

email.setAuthentication("laaqq1414@gmail.com", "qQ1!1111");

email.setSSLOnConnect(true);

email.setFrom("doNotReply"+user.getEmail().getEmailAddress());

email.setSubject("Sign Up Successful");

email.setMsg("Welcome to the Store\n\n Your account has been successfully created.");

email.addTo(user.getEmail().getEmailAddress());

email.send();

return new ModelAndView("account-success", "user", u);

} catch (UserException e) {

System.out.println("Exception: " + e.getMessage());

return new ModelAndView("error", "errorMessage", "error while login");

}

}

protected Map referenceData(HttpServletRequest request) throws Exception {

Map referenceData = new HashMap();

Map<String,String> usertype = new LinkedHashMap<String,String>();

usertype.put("Buyer", "Buyer");

usertype.put("Seller", "Seller");

referenceData.put("usertype", usertype);

return referenceData;

}

}

package com.neu.naam.dao;

//import java.util.HashSet;

import java.util.List;

//import java.util.Set;

import org.hibernate.HibernateException;

import org.hibernate.Query;

import com.neu.naam.exception.AdvertException;

import com.neu.naam.pojo.Advert;

//import com.neu.naam.pojo.Category;

public class AdvertDAO extends DAO {

public Advert create(Advert advert)

throws AdvertException {

try {

begin();

getSession().save(advert);

commit();

return advert;

} catch (HibernateException e) {

rollback();

//throw new AdException("Could not create advert", e);

throw new AdvertException("Exception while creating advert: " + e.getMessage());

}

}

public void delete(Advert advert)

throws AdvertException {

try {

begin();

getSession().delete(advert);

commit();

} catch (HibernateException e) {

rollback();

throw new AdvertException("Could not delete advert", e);

}

}

public List<Advert> list() throws AdvertException{

try {

begin();

Query query = getSession().createQuery("from Advert");

List<Advert> listOfAdverts = query.list();

commit();

return listOfAdverts;

} catch (HibernateException ex) {

rollback();

throw new AdvertException("Could not delete advert", ex);

}

}

}

package com.neu.naam.dao;

import java.util.List;

import org.hibernate.HibernateException;

import org.hibernate.Query;

import com.neu.naam.exception.CartException;

import com.neu.naam.exception.CategoryException;

import com.neu.naam.pojo.Advert;

import com.neu.naam.pojo.Cart;

import com.neu.naam.pojo.Category;

import com.neu.naam.pojo.User;

public class CartDAO extends DAO {

public CartDAO(){

}

public Cart insert(Cart cart) throws CartException {

try{

begin();

getSession().save(cart);

commit();

return cart;

} catch (HibernateException ex){

rollback();

throw new CartException("Unable to save the cart\n Exception: ", ex);

}

}

public void update(Cart cart) throws CategoryException {

try {

begin();

getSession().update(cart);

commit();

} catch (HibernateException ex) {

rollback();

throw new CategoryException("Unable to save the cart\nHibernate Exception: ", ex);

}

}

public User update(User user) throws CategoryException {

try {

begin();

getSession().update(user);

commit();

} catch (HibernateException e) {

rollback();

throw new CategoryException("Could not save the user", e);

}

return user;

}

public List<Cart> list(){

begin();

Query q = getSession().createQuery("from Cart");

List<Cart> cart1 = q.list();

commit();

return cart1;

}

public Cart updateCart(Cart cart) throws CategoryException {

try {

begin();

getSession().update(cart);

commit();

return cart;

} catch (HibernateException e) {

rollback();

throw new CategoryException("Could not save the cart", e);

}

}

}

package com.neu.naam.dao;

import java.util.List;

import org.hibernate.HibernateException;

import org.hibernate.Query;

import com.neu.naam.exception.CategoryException;

import com.neu.naam.pojo.Category;

public class CategoryDAO extends DAO {

public Category get(String title) throws CategoryException {

try {

begin();

Query query=getSession().createQuery("from Category where title= :title");

query.setString("title",title);

Category category=(Category)query.uniqueResult();

commit();

return category;

} catch (HibernateException e) {

rollback();

throw new CategoryException("Unable to obtain the category " + title + " " + e.getMessage());

}

}

public List<Category> list() throws CategoryException {

try {

begin();

Query query = getSession().createQuery("from Category");

List<Category> categoryList = query.list();

commit();

return categoryList;

} catch (HibernateException e) {

rollback();

throw new CategoryException("Unable to list the categories", e);

}

}

public Category create(String title) throws CategoryException {

try {

begin();

Category category = new Category(title);

getSession().save(category);

commit();

return category;

} catch (HibernateException e) {

rollback();

throw new CategoryException("Unable to create a category: " + e.getMessage());

}

}

public void update(Category category) throws CategoryException {

try {

begin();

getSession().update(category);

commit();

} catch (HibernateException e) {

rollback();

throw new CategoryException("Could not save the category", e);

}

}

public void delete(Category category) throws CategoryException {

try {

begin();

getSession().delete(category);

commit();

} catch (HibernateException e) {

rollback();

throw new CategoryException("Could not delete the category", e);

}

}

}

package com.neu.naam.dao;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.hibernate.HibernateException;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

public class DAO {

private static final Logger log = Logger.getAnonymousLogger();

private static final ThreadLocal sessionThread = new ThreadLocal();

private static final SessionFactory sessionFactory = new Configuration().configure("hibernate.cfg.xml").buildSessionFactory();

protected DAO() {

}

public static Session getSession()

{

Session session = (Session) DAO.sessionThread.get();

if (session == null)

{

session = sessionFactory.openSession();

DAO.sessionThread.set(session);

}

return session;

}

protected void begin() {

getSession().beginTransaction();

}

protected void commit() {

getSession().getTransaction().commit();

}

protected void rollback() {

try {

getSession().getTransaction().rollback();

} catch (HibernateException e) {

log.log(Level.WARNING, "Cannot rollback", e);

}

try {

getSession().close();

} catch (HibernateException e) {

log.log(Level.WARNING, "Cannot close", e);

}

DAO.sessionThread.set(null);

}

public static void close() {

getSession().close();

DAO.sessionThread.set(null);

}

}

package com.neu.naam.dao;

import org.hibernate.HibernateException;

import org.hibernate.Query;

import com.neu.naam.exception.UserException;

import com.neu.naam.pojo.Email;

import com.neu.naam.pojo.User;

public class UserDAO extends DAO {

public UserDAO() {

}

public User get(String username, String password) throws UserException {

try {

begin();

Query q = getSession().createQuery("from User where username = :username and password = :password");

q.setString("username", username);

q.setString("password", password);

User user = (User) q.setMaxResults(1).uniqueResult();

//System.out.println(user.getUsertype());

commit();

return user;

} catch (HibernateException e) {

rollback();

throw new UserException("Could not get user " + username, e);

}

}

public User get(int userId) throws UserException {

try {

begin();

Query q = getSession().createQuery("from User where personID= :personID");

q.setInteger("personID", userId);

User user = (User) q.setMaxResults(1).uniqueResult();

commit();

return user;

} catch (HibernateException e) {

rollback();

throw new UserException("Could not get user " + userId, e);

}

}

public User register(User u)

throws UserException {

try {

begin();

System.out.println("inside DAO");

Email email = new Email(u.getEmail().getEmailAddress());

User user = new User(u.getUsername(), u.getPassword(), u.getUsertype());

user.setFirstName(u.getFirstName());

user.setLastName(u.getLastName());

user.setEmail(email);

email.setUser(user);

getSession().save(user);

commit();

return user;

} catch (HibernateException e) {

rollback();

throw new UserException("Exception while creating user: " + e.getMessage());

}

}

public void delete(User user) throws UserException {

try {

begin();

getSession().delete(user);

commit();

} catch (HibernateException e) {

rollback();

throw new UserException("Could not delete user " + user.getUsername(), e);

}

}

}

package com.neu.naam.exception;

public class AdvertException extends Exception

{

public AdvertException(String message)

{

super("AdvertException-"+ message);

}

public AdvertException(String message, Throwable cause)

{

super("AdvertException-"+ message,cause);

}

}

package com.neu.naam.exception;

public class CartException extends Exception

{

public CartException(String message)

{

super("CartException-"+ message);

}

public CartException(String message, Throwable cause)

{

super("CartException-"+ message,cause);

}

}

package com.neu.naam.exception;

public class CategoryException extends Exception {

public CategoryException(String message)

{

super("CategoryException-"+message);

}

public CategoryException(String message, Throwable cause)

{

super("CategoryException-"+message,cause);

}

}

package com.neu.naam.exception;

public class UserException extends Exception {

public UserException(String message)

{

super("UserException-"+message);

}

public UserException(String message, Throwable cause)

{

super("UserException-"+message,cause);

}

}

package com.neu.naam.interceptor;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.springframework.web.servlet.ModelAndView;

import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;

import com.neu.naam.pojo.User;

public class Interceptor extends HandlerInterceptorAdapter {

private String signout;

public String getsignout() {

return signout;

}

public void setsignout(String signout) {

this.signout = signout;

}

/\*\*

\* This implementation always returns <code>true</code>.

\*/

@Override

public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)

throws Exception {

System.out.println("Inside User Interceptor");

HttpSession session = request.getSession();

if(session != null){

Object o = session.getAttribute("user");

if(o != null){

try{

User user = (User) o;

return true;

}

catch(Exception e){

System.out.println("Exception in Recruiterinterceptor");

e.printStackTrace();

}}else{

response.sendRedirect(signout);

return false;

}

}

response.sendRedirect(signout);

return false;

}

/\*\*

\* This implementation is empty.

\*/

@Override

public void postHandle(

HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView)

throws Exception {

super.postHandle(request, response, handler, modelAndView);

}

}

package com.neu.naam.pojo;

import java.util.ArrayList;

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.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.ManyToMany;

import javax.persistence.ManyToOne;

import javax.persistence.PrimaryKeyJoinColumn;

import javax.persistence.Table;

import javax.persistence.Transient;

import org.springframework.web.multipart.commons.CommonsMultipartFile;

@Entity

@Table(name="advert\_table")

public class Advert {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

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

private long id;

@Column(name="title")

private String title;

@Column(name="message")

private String message;

@Column(name = "filename")

private String filename;

@Column(name = "price")

private Float price;

public Float getPrice() {

return price;

}

public void setPrice(Float price) {

this.price = price;

}

@ManyToOne

private User user;

@ManyToMany(mappedBy="adverts")

private Set<Category> categories = new HashSet<Category>();

@ManyToMany(mappedBy="adverts")

private Set<Cart> cart = new HashSet<Cart>();

@Transient

int postedBy;

public String getFilename() {

return filename;

}

public void setFilename(String filename) {

String name="";

for(int i=filename.length()-1;i>-1&&(int)filename.charAt(i)!=92;i--) {

name+=filename.charAt(i);

}

StringBuilder input1 = new StringBuilder();

// append a string into StringBuilder input1

input1.append(name);

// reverse StringBuilder input1

input1 = input1.reverse();

this.filename = input1.toString();

}

public CommonsMultipartFile getPhoto() {

return photo;

}

public void setPhoto(CommonsMultipartFile photo) {

this.photo = photo;

}

@Transient

private CommonsMultipartFile photo;

public Advert(String title, String message, User user, Category catergory, Cart cart) {

this.title = title;

this.message = message;

this.user = user;

this.categories.add(catergory);

this.cart.add(cart);

}

public Advert() {

}

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

public Set<Category> getCategories() {

return categories;

}

public void setCategories(Set<Category> categories) {

this.categories = categories;

}

public int getPostedBy() {

return postedBy;

}

public void setPostedBy(int postedBy) {

this.postedBy = postedBy;

}

}

package com.neu.naam.pojo;

import java.util.ArrayList;

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.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.ManyToMany;

import javax.persistence.ManyToOne;

import javax.persistence.OneToOne;

import javax.persistence.PrimaryKeyJoinColumn;

import javax.persistence.Table;

import javax.persistence.Transient;

import org.hibernate.annotations.GenericGenerator;

import org.hibernate.annotations.Parameter;

@Entity

@Table(name="cart\_table")

@PrimaryKeyJoinColumn(name = "personID")

public class Cart{

@Id

@GeneratedValue(generator = "generator")

@GenericGenerator(name = "generator", strategy = "foreign", parameters = @Parameter(name = "property", value = "user"))

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

private long id;

@OneToOne

@PrimaryKeyJoinColumn

private User user;

@ManyToMany

@JoinTable (

name="cart\_advert\_table",

joinColumns = {@JoinColumn(name="cartID", nullable = false, updatable = false, referencedColumnName="cartID")},

inverseJoinColumns = {@JoinColumn(name="advertID", referencedColumnName="advertID" )}

)

private Set<Advert> adverts = new HashSet<Advert>();

@Column(name="title")

private String title;

@Column(name="category")

private String category;

@Column(name="filename")

private String filename;

@Column(name="totalprice")

private String totalprice;

public Cart(){

}

public Cart(String title, String category, String filename, String totalprice) {

this.title = title;

this.category = category;

this.filename = filename;

this.totalprice = totalprice;

}

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

public Set<Advert> getAdverts() {

return adverts;

}

public void setAdverts(Set<Advert> adverts) {

this.adverts = adverts;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getCategory() {

return category;

}

public void setCategory(String category) {

this.category = category;

}

public String getFilename() {

return filename;

}

public void setFilename(String filename) {

this.filename = filename;

}

public String getTotalprice() {

return totalprice;

}

public void setTotalprice(String totalprice) {

this.totalprice = totalprice;

}

}

package com.neu.naam.pojo;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.ManyToMany;

import javax.persistence.Table;

@Entity

@Table(name="category\_table")

public class Category {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

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

private long categoryId;

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

private String title;

@ManyToMany

@JoinTable (

name="category\_advert\_table",

joinColumns = {@JoinColumn(name="categoryID", nullable = false, updatable = false, referencedColumnName="categoryID")},

inverseJoinColumns = {@JoinColumn(name="advertID", referencedColumnName="advertID" )}

)

private Set<Advert> adverts = new HashSet<Advert>();

public Category(String title) {

this.title = title;

}

public Category() {

}

public long getCategoryId() {

return categoryId;

}

public void setCategoryId(long categoryId) {

this.categoryId = categoryId;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public Set<Advert> getAdverts() {

return adverts;

}

public void setAdverts(Set<Advert> adverts) {

this.adverts = adverts;

}

@Override

public String toString(){

return title;

}

}

package com.neu.naam.pojo;

import java.awt.Color;

import java.io.UnsupportedEncodingException;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.List;

import java.util.Map;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.view.document.AbstractPdfView;

import com.lowagie.text.Document;

import com.lowagie.text.Font;

import com.lowagie.text.Paragraph;

import com.lowagie.text.Phrase;

import com.lowagie.text.pdf.PdfPCell;

import com.lowagie.text.pdf.PdfPTable;

import com.lowagie.text.pdf.PdfWriter;

public class PDFView extends AbstractPdfView {

@Override

protected void buildPdfDocument(Map<String, Object> model, Document pdfdoc, PdfWriter pdfwriter, HttpServletRequest request,

HttpServletResponse response) throws Exception {

Font myFont = new Font(Font.HELVETICA, 20, Font.BOLDITALIC, Color.BLACK);

List<Cart> cartlist = (List<Cart>) model.get("cartitems");

Paragraph title = new Paragraph("Thank you for shopping at the Store", myFont);

Phrase producttitle = new Phrase("Below you will find a summary of the item you have purchased");

String inp="";

for (Cart cart\_item : cartlist) {

inp+=cart\_item.getTitle()+cart\_item.getCategory();

}

String hash=makeSHA1Hash(inp);

Phrase thank = new Phrase("Transaction code: "+ hash +"\n\nThank you for shopping with us");

PdfPTable table = new PdfPTable(3);

table.setWidthPercentage(100.0f);

float width[]=new float[] {3.0f, 2.0f, 1.0f};

table.setWidths(width);

table.setSpacingBefore(12);

PdfPCell cell = new PdfPCell();

cell.setBackgroundColor(Color.gray);

cell.setPadding(4);

cell.setPhrase(new Phrase("Product Title", myFont));

table.addCell(cell);

cell.setPhrase(new Phrase("Category", myFont));

table.addCell(cell);

cell.setPhrase(new Phrase("Price", myFont));

table.addCell(cell);

for (Cart cart\_item : cartlist) {

table.addCell(cart\_item.getTitle());

table.addCell(cart\_item.getCategory());

table.addCell(String.valueOf(cart\_item.getTotalprice()));

}

pdfdoc.add(title);

pdfdoc.add(producttitle);

pdfdoc.add(table);

pdfdoc.add(thank);

}

public String makeSHA1Hash(String input)

throws NoSuchAlgorithmException, UnsupportedEncodingException

{

MessageDigest md = MessageDigest.getInstance("SHA1");

md.reset();

byte[] buffer = input.getBytes("UTF-8");

md.update(buffer);

byte[] digest = md.digest();

String hexStr = "";

for (int i = 0; i < digest.length; i++) {

hexStr += Integer.toString( ( digest[i] & 0xff ) + 0x100, 16).substring( 1 );

}

return hexStr;

}

}

package com.neu.naam.pojo;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Inheritance;

import javax.persistence.InheritanceType;

import javax.persistence.Table;

@Entity

@Table(name="person\_table")

@Inheritance(strategy=InheritanceType.JOINED) //table per subclass

public class Person {

@Id

@GeneratedValue(strategy=GenerationType.IDENTITY)

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

private long personID;

@Column(name = "firstName")

private String firstName;

@Column(name ="lastName")

private String lastName;

public Person(){

}

public long getPersonID() {

return personID;

}

public void setPersonID(long personID) {

this.personID = personID;

}

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;

}

}

package com.neu.naam.pojo;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.ManyToMany;

import javax.persistence.OneToMany;

import javax.persistence.OneToOne;

import javax.persistence.PrimaryKeyJoinColumn;

import javax.persistence.Table;

@Entity

@Table(name = "user\_table")

@PrimaryKeyJoinColumn(name = "personID")

public class User extends Person {

@Column(name = "userName")

private String username;

@Column(name = "password")

private String password;

@OneToOne(mappedBy = "user", cascade = CascadeType.ALL)

private Email email;

@OneToOne(mappedBy = "user", cascade = CascadeType.ALL)

private Cart cart;

public Cart getCart() {

return cart;

}

public void setCart(Cart cart) {

this.cart = cart;

}

@Column(name = "usertype")

private String usertype;

public User(String username, String password, String usertype) {

this.username = username;

this.password = password;

this.usertype = usertype;

}

public User() {

}

public String getUsertype() {

return usertype;

}

public void setUsertype(String usertype) {

this.usertype = usertype;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public Email getEmail() {

return email;

}

public void setEmail(Email email) {

this.email = email;

}

}

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory name="sessionFactory">

<property name="hibernate.connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/shoppingcart</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root</property>

<property name="hibernate.hbm2ddl.auto">update</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</property>

<property name="hibernate.show\_sql">true</property>

<mapping class="com.neu.naam.pojo.User"/>

<mapping class="com.neu.naam.pojo.Email"/>

<mapping class="com.neu.naam.pojo.Person"/>

<mapping class="com.neu.naam.pojo.Advert"/>

<mapping class="com.neu.naam.pojo.Category"/>

<mapping class="com.neu.naam.pojo.Cart"/>

</session-factory>

</hibernate-configuration>

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

<c:set var="contextPath" value="${pageContext.request.contextPath}" />

<h1> Your Account has been successfully created. Please go back to the home page to login</h1><br><br>

<a href="${contextPath}/"><input type="submit" value="Home"></a><br/>

</body>

</html>

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<title>Add Advert Form</title>

</head>

<body>

<c:set var="contextPath" value="${pageContext.request.contextPath}" />

<a href="${contextPath}/user/">Seller Home</a><br/>

<h2>Posting a New Advert</h2>

<form:form action="${contextPath}/advert/add" method="post"

commandName="advert" enctype="multipart/form-data">

<table>

<tr>

<td>Poster Name:</td>

<td>${sessionScope.user.username}</td>

<td><form:hidden path="postedBy"

value="${sessionScope.user.personID}" /></td>

</tr>

<tr>

<td>Select a Category:</td>

<td><form:select path="categories" items="${categories}"

multiple="true" required="required" /></td>

</tr>

<tr>

<td>Product Name:</td>

<td><form:input type="text" path="title" size="30" required="required"/>

<font color="red"><form:errors path="title" /></font></td>

</tr>

<tr>

<td>Product Image:</td>

<td><input type="file" name="photo" required="required"/>

</td>

</tr>

<tr>

<td>Image Name:</td>

<td><form:input type="text" path="filename" required="required"/>

<font color="red"><form:errors path="filename" /></font></td>

</tr>

<tr>

<td>Product Description:</td>

<td><form:input type="text" path="message" size="30" required="required"/>

<font color="red"><form:errors path="message" /></font></td>

</tr>

<tr>

<td>Product Price:</td>

<td><form:input type="number" path="price" size="30" required="required"/>

<font color="red"><form:errors path="price" /></font></td>

</tr>

<tr>

<td colspan="2"><input type="submit" value="Post Advert" /></td>

</tr>

</table>

</form:form>

</body>

</html>

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>List Adverts</title>

</head>

<body>

<c:set var="contextPath" value="${pageContext.request.contextPath}" />

<a href="${contextPath}/"><input type="submit" value="Logout"></a><br/><br>

<br>

<form:form action="${contextPath}/cart/insert" method="post" commandName="cart">

<table border="1" cellpadding="5" cellspacing="5">

<tr>

<td><b>Product Name</b></td>

<td><b>Product Description</b></td>

<td><b>Seller Name</b></td>

<td><b>Category</b></td>

<td><b>Image</b></td>

<td><b>Price</b></td>

<td><b>Options</b></td>

</tr>

<c:forEach var="adv" items="${adverts}">

<form:hidden path="title" value="${adv.title}"/>

<form:hidden path="totalprice" value="${adv.price}"/>

<form:hidden path="category" value="${adv.categories}"/>

<form:hidden path="filename" value="${adv.filename}"/>

<tr>

<td>${adv.title}</td>

<td>${adv.message}</td>

<td>${adv.user.username}</td>

<td><c:forEach var="categ" items="${adv.categories}">

${categ} ,

</c:forEach></td>

<td><img height="150" width="150" src="${adv.filename}" /></td>

<td>${adv.price}</td>

<td><input type="submit" value="Add to Cart" /></td>

</tr>

</c:forEach>

</table>

</form:form>

</body>

</html>

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Advert Created Successfully</title>

</head>

<body>

<c:set var="contextPath" value="${pageContext.request.contextPath}" />

<a href="${contextPath}/user/">Home</a><br/>

<h2>Advert Posted Successfully: ${advert.title}</h2>

</body>

</html>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>sbeve</title>

</head>

<body>

<a href="user/register.htm"><input type="submit" value="Sign Up"></a><br/><br><br>

<h2>Login</h2>

<form action="user/login" method="post">

<input name="username" size="30" required="required" placeholder="username"/>

<br>

<input type="password" name="password" size="30" required="required" placeholder="password"/>

<br>

<input type="submit" value="Login" />

</form>

</body>

</html>