INFO 6250 Final Project Report
Pet Adoption & Homeless Animal Report Application

1. Summary

The core idea is from my final project for AED class. More families have kept pets or plan to have pets and adoption is always a better option than buying a pet and can be done in a more convenient way. This application should provide more convenience to find pets that are individually listed by pet owners. Homeless animals can cause potential transmitted diseases and need to be rescued in time. Users can report homeless animal immediately and those animals can be rescued on time by animal health care organization.

2. Functionality

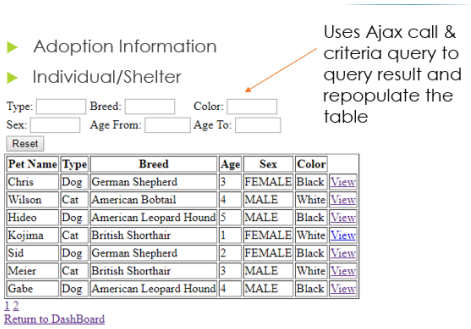
This application provides functions for both users and shelter employee to manage their pet/animal for adoption list. Users can also view the adoption information listed by other individual user or by the shelter. Users can report homeless animals to shelter and employees can arrange the rescue by priority and add the rescued animals later for adoption.

3. Technologies

This web application uses spring mvc and hibernate. Requests are mapped by annotations in controllers. All of the concepts come from lecture or lab.

3.1 For registering new users, I implement the email activations taught in lab.

3.2 For searching the desired pets, I use Ajax call to perform the search. The DAO class used for searching pets uses criteria query in hibernate. It makes the search case insensitive and can be filtered dynamically.



Interface

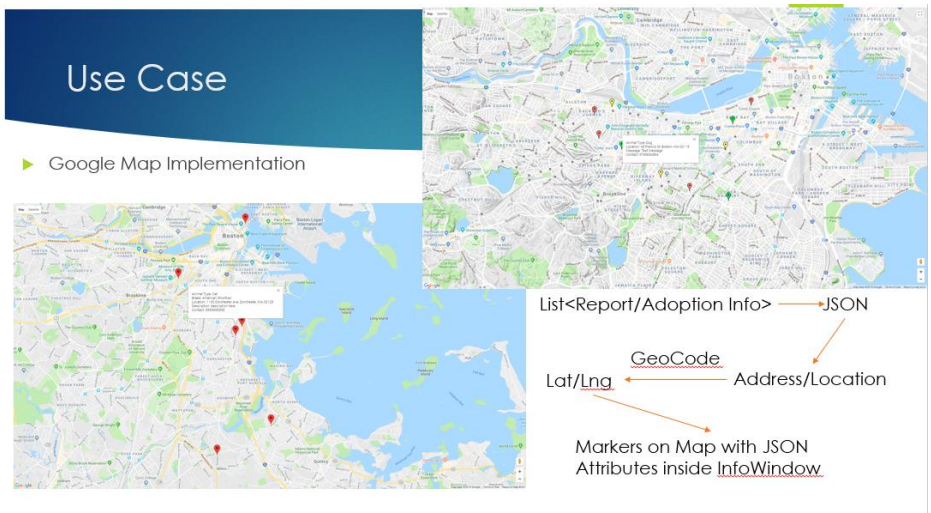
```
@RequestMapping(value = "/ajaxservice.htm", method = RequestMethod.GET)
@ResponseBody
public String ajaxService(HttpServletRequest request, PetDAO petDao) {
    String type = request.getParameter("type");
    String breed = request.getParameter("breed");
    String color = request.getParameter("color");
    String agefrom = request.getParameter("agefrom");
    String ageto = request.getParameter("ageto");
    String sex = request.getParameter("sex");

    String uid = String.valueOf(request.getSession().getAttribute("userid"));
    int userid = Integer.parseInt(uid);
    List<Pet> petlist = new ArrayList<Pet>();
    try {
        petlist = petDao.query(userid, type, breed, color, agefrom, ageto, sex);
    } catch (Exception e) {
        e.printStackTrace();
    }
    JSONArray jarray = new JSONArray();
    for (Pet p : petlist) {
        JSONObject j = new JSONObject();
        j.put("petid", p.getPetId());
        j.put("petname", p.getPetName());
        j.put("petage", p.getPetAge());
        j.put("pettype", p.getAnimalType());
        j.put("sex", p.getGender());
        j.put("breed", p.getBreed());
        j.put("color", p.getColor());
        jarray.put(j);
    }
    // System.out.println(petlist.size());
    return jarray.toString();
}
```

Ajax Ccontroller

```java
public List<Pet> query(int userid, String type, String breed, String color, String agefrom, String ageto,
        String sex) throws Exception {
    try {
        begin();
        // System.out.println("from Pet where petownerid != :ownerId and listed =
        // true"+query);
        CriteriaBuilder cb = getSession().getCriteriaBuilder();
        CriteriaQuery<Pet> cq = cb.createQuery(Pet.class);
        Root e = cq.from(Pet.class);
        cq.select(e);
        Predicate p = cb.conjunction();
        p = cb.and(p, cb.notEqual(e.join("owner"), userid));
        if (!type.equals("")) {
            p = cb.and(p, cb.like(e.get("animalType"), type));
        }
        if (!breed.equals("")) {
            p = cb.and(p, cb.like(e.get("breed"), "%" + breed + "%"));
        }
        if (!color.equals("")) {
            p = cb.and(p, cb.like(e.get("color"), color));
        }
        if (!agefrom.equals("")) {
            p = cb.and(p, cb.greaterThanOrEqualTo(e.get("petAge"), agefrom));
        }
        if (!ageto.equals("")) {
            p = cb.and(p, cb.lessThanOrEqualTo(e.get("petAge"), ageto));
        }
        if (!sex.equals("")) {
            gender g = null;
            if (sex.equalsIgnoreCase("female") || sex.equalsIgnoreCase("f")) {
                g = gender.FEMALE;
            }
            if (sex.equalsIgnoreCase("male") || sex.equalsIgnoreCase("m")) {
                g = gender.MALE;
            }
            p = cb.and(p, cb.equal(e.get("gender"), g));
        }
        cq.where(p);
        cq.select(e);
        List<Pet> list = getSession().createQuery(cq).getResultList();
        System.out.println(list.size());
        close();
        return list;
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    return null;
```

Criteria Query

3.3 I also use google map api for showing the location of pets and homeless reports.



```
function ajaxEvent() {
    //alert("called");
    var xmlHttp;

    $.ajax({
        type : "GET",
        url : "../mapservice.htm?",
        dataType : 'json',

        success : function(response) {
            popMap(response);
        },
        error : function() {
            alert('error');
        }
    });

    function popMap(data) {
        $.each(data, function(key, val) {

            codeAddress(val);

        });

    }

}
```

```
function codeAddress(val) {
    geocoder
            .geocode(
                    {
                        address : val.location
                    },
                    function(results, status) {
                        if (status == google.maps.GeocoderStatus.OK) {
                            map.setCenter(results[0].geometry.location);
                            var marker = new google.maps.Marker({
                                map : map,
                                position : results[0].geometry.location
                            });
                            marker
                                    .setIcon('http://maps.google.com/mapfiles/ms/icons/green-dot.png');
                            if (val.priority === 'MEDIUM') {
                                marker
                                        .setIcon('http://maps.google.com/mapfiles/ms/icons/yellow-dot.png');
                            } else if (val.priority === 'HIGH') {
                                marker
                                        .setIcon('http://maps.google.com/mapfiles/ms/icons/red-dot.png');
                            }

                            var infoWindow = new google.maps.InfoWindow({
                                content : "<p>Animal Type: " + val.type
                                        + "<br />" + "Location: "
                                        + val.location + "<br />Message: "
                                        + val.description
                                        + "<br />Contact: " + val.contact
                                        + "</p>"
                            });
                            google.maps.event.addListener(marker, 'click',
                                    function() {
                                        if (lastWindow) {
                                            lastWindow.close();
                                        }
                                        lastWindow = infoWindow;
                                        infoWindow.open(map, marker);
                                    });
                        } else {
                            alert('Geocode was not successful for the following reason: '
                                    + status);
                        }
                    });
}
```

3.4 Excel view provided by Spring frame work is also used in this project for shelter employees to export the pending reports.

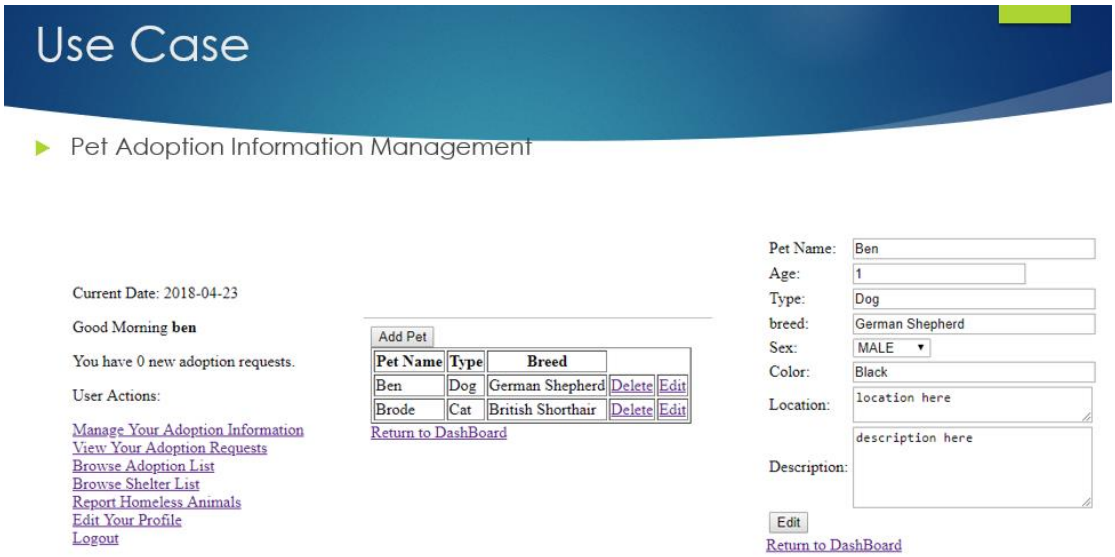
```
@RequestMapping(value = "/report/export.xls", method = RequestMethod.GET)
public ModelAndView exportReports(HttpServletRequest request, ReportDAO reportDao) {
    List<HomelessReport> reportList = null;
    try {
        reportList = reportDao.getList(status.PENDING);
    }catch(Exception e) {
        e.printStackTrace();
    }

    return new ModelAndView(new XlsView(), "list", reportList);
}
```

```
public class XlsView extends AbstractExcelView{

    @Override
    protected void buildExcelDocument(Map<String, Object> model, HSSFWorkbook workbook, HttpServletRequest request,
            HttpServletResponse response) throws Exception {
        // TODO Auto-generated method stub
        List<HomelessReport> list = (List<HomelessReport>)model.get("list");

        Sheet sheet = workbook.createSheet("Reports");
        Row header = sheet.createRow(0);
        header.createCell(0).setCellValue("ReportID");
        header.createCell(1).setCellValue("Priority");
        header.createCell(2).setCellValue("Message");
        header.createCell(3).setCellValue("AnimalType");
        header.createCell(4).setCellValue("HealthCondition");
        header.createCell(5).setCellValue("ReportDate");
        header.createCell(6).setCellValue("Location");
        header.createCell(7).setCellValue("Aggressive");
        header.createCell(8).setCellValue("InDanger");
        int rowNum=1;
        for(HomelessReport r : list) {
            Row row = sheet.createRow(rowNum++);
            row.createCell(0).setCellValue(r.getReportid());
            row.createCell(1).setCellValue(r.getPriority().toString());
            row.createCell(2).setCellValue(r.getMessage());
            row.createCell(3).setCellValue(r.getAnimaltype());
            row.createCell(4).setCellValue(r.getHealthcondition().toString());
            row.createCell(5).setCellValue(r.getReportdate());
            row.createCell(6).setCellValue(r.getLocation());
            row.createCell(7).setCellValue(r.isAggressive());
            row.createCell(8).setCellValue(r.isIndanger());
        }
    }

}
```

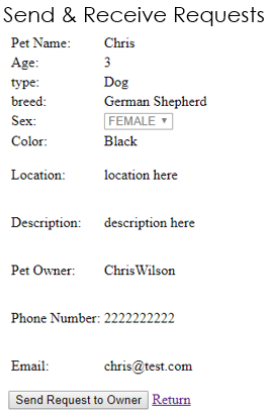
4. Roles

I design this application with the assumption of two clients which are user side and shelter side. So, I did not use a common account class with role type in order to distinguish them. I just put the username and password attributes inside user and employee class for the login function.

User represents two roles in the system which are pet owners and adopters. As an owner, I can list my pets for adoption (shelter employee can manage animals in the same way), browse individual listed pets or animal list made by shelter.



User can also send requests to the pets they want to adopt and owners can process the requests.



User can also report homeless animals to the shelter and the employee can process the report with the option to add new animals to the adoption list.

Homeless Animal Report

User Side

Contact:
Animal Type:
Health Condition: GOOD
Aggressive Behavior:
Food Provided?:
Location:
Description:
This animal is in danger:
Report

Shelter side

Print Task Table

Report Id	Date	Animal Type	Priority	Status	
3	2018-04-23 00:00:00.0	Dog	HIGH	PENDING	View
4	2018-04-23 00:00:00.0	Dog	MEDIUM	PENDING	View
1	2018-04-23 00:00:00.0	dog	HIGH	COMPLETE	View
2	2018-04-23 00:00:00.0	asd	MEDIUM	COMPLETE	View

Return to Portal

Report priority is set automatically depending on report attributes

Rescue team can export all incomplete reports for printing

Process Reports

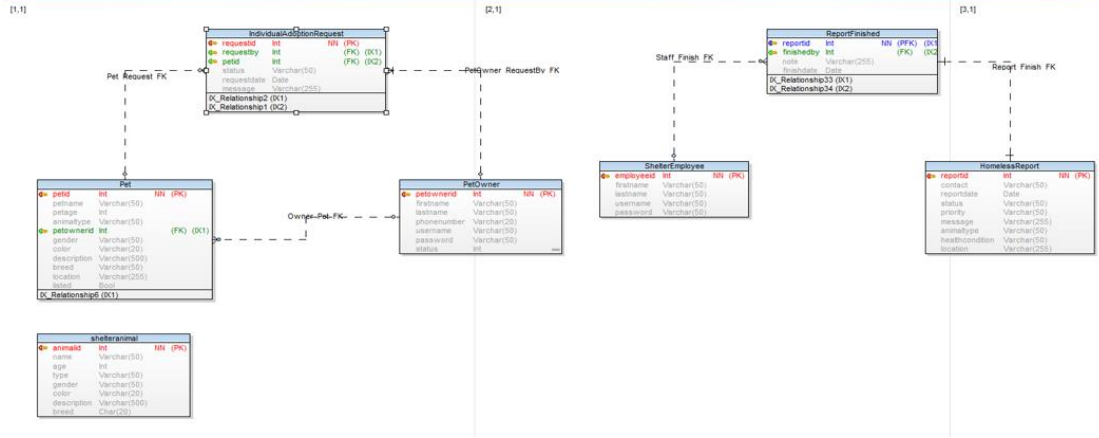
Mark as Complete

ReportID	4
Contact	123456
Report Date	2018-04-23 00:00:00.0
Status	PENDING
Priority	MEDIUM
Animal Type	Dog
Health Condition	GOOD
Location	1111

Option for employee to add a new animal to the list for adoption

Note:
Add new animal to adoption list:
Animal Type: Dog
Age:
breed:
Sex: Male
Color:
Name:
Description:
Submit

ER-Diagram



Appendix

Controllers

Pojos

```
package com.myspring.petfinder.controller;

import java.util.List;
import java.util.Random;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
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 com.myspring.petfinder.dao.AnimalDAO;
import com.myspring.petfinder.dao.PetDAO;
import com.myspring.petfinder.dao.UserDAO;
import com.myspring.petfinder.pojo.Pet;
import com.myspring.petfinder.pojo.ShelterAnimal;
```

```
import com.myspring.petfinder.pojo.User;
import com.myspring.petfinder.pojo.IndividualRequest;
import com.myspring.petfinder.pojo.IndividualRequest.status;

@Controller
public class UserController {

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

    @RequestMapping(value = "/user/login.htm", method =
RequestMethod.GET)
    public String showLoginForm(HttpServletRequest request) {
        HttpSession session = request.getSession();
        if (session.getAttribute("user") != null) {
            return "user-dashboard";
        }
        return "user-login";
    }

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

```
    public String handleLoginForm(HttpServletRequest request, UserDAO
userDao, ModelMap map) {
        HttpSession session = request.getSession();

        String username = request.getParameter("username");
        String password = request.getParameter("password");
        try {
            User u = userDao.get(username, password);
            if (u != null && u.getStatus() == 1) {
                session.setAttribute("user", u.getUserName());
                session.setAttribute("userid", u.getPetOwnerId());
                int newRequest = 0;
                for (Pet p : u.getPetList()) {
                    for (IndividualRequest ir : p.getRequestList()) {
                        if (ir.getStatus() == status.PENDING) {
                            newRequest += 1;
                        }
                    }
                }
                map.addAttribute("newRequest", newRequest);
                return "user-dashboard";
            } else if (u != null && u.getStatus() == 0) {
```

```
                    map.addAttribute("errorMessage", "Please activate your account
to login!");
                    return "login-error";
                } else {
                    map.addAttribute("errorMessage", "Invalid
username/password!");
                    return "login-error";
                }
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

            return null;

        }

        @RequestMapping(value = "/user/register.htm", method =
RequestMethod.GET)
        public String showCreateForm() {

            return "user-register-form";
```

```
    }

    @RequestMapping(value = "/user/create.htm", method =
RequestMethod.POST)
    public String handleCreateForm(HttpServletRequest request, UserDAO
userDao, ModelMap map) {
        HttpSession session = request.getSession();
        String username = request.getParameter("username");
        String useremail = request.getParameter("useremail");
        String password = request.getParameter("password");
        User user = new User();
        user.setUserName(username);
        user.setEmail(useremail);
        user.setPassword(password);
        user.setStatus(0);
        User u = null;
        try {
            u = userDao.get(username, password);
        } catch (Exception e) {
            e.printStackTrace();
        }
        if (u != null) {
```

```
            map.addAttribute("errorMessage", "username already exists");
            return "login-error";
        }
        try {
            u = userDao.register(user);
            Random rand = new Random();
            int randomNum1 = rand.nextInt(5000000);
            int randomNum2 = rand.nextInt(5000000);
            try {
                String str =
"http://localhost:8080/petfinder/user/validateemail.htm?email=" + useremail +
"&key1="
                        + randomNum1 + "&key2=" + randomNum2;
                session.setAttribute("key1", randomNum1);
                session.setAttribute("key2", randomNum2);
                sendEmail(useremail, "Click on this link to activate your account :
" + str);
            } catch (Exception e) {
                System.out.println("Email cannot be sent");
            }
        } catch (Exception e) {
            // TODO Auto-generated catch block
```

```
            e.printStackTrace();
        }

        return "user-registered";
    }

    @RequestMapping(value = "/user/forgotpassword.htm", method =
RequestMethod.GET)
    public String getForgotPasswordForm(HttpServletRequest request) {

        return "forgot-password";
    }

    @RequestMapping(value = "/user/forgotpassword.htm", method =
RequestMethod.POST)
    public String handleForgotPasswordForm(HttpServletRequest request,
UserDAO userDao) {

        String useremail = request.getParameter("useremail");

        User user = userDao.get(useremail);
        sendEmail(useremail, "Your password is : " + user.getPassword());
```

```
        return "forgot-password-success";


    }


    @RequestMapping(value = "user/resendemail.htm", method =
RequestMethod.POST)
    public String resendEmail(HttpServletRequest request) {
        HttpSession session = request.getSession();
        String useremail = request.getParameter("username");
        Random rand = new Random();
        int randomNum1 = rand.nextInt(5000000);
        int randomNum2 = rand.nextInt(5000000);
        try {
            String str =
"http://localhost:8080/lab10/user/validateemail.htm?email=" + useremail +
"&key1=" + randomNum1
                    + "&key2=" + randomNum2;
            session.setAttribute("key1", randomNum1);
            session.setAttribute("key2", randomNum2);
            sendEmail(useremail, "Click on this link to activate your account : " +
str);
        } catch (Exception e) {
```

```
                System.out.println("Email cannot be sent");
            }

            return "user-registered";
        }

        public void sendEmail(String username, String message) {
            try {
                Email email = new SimpleEmail();
                email.setHostName("smtp.googlemail.com");
                email.setSmtpPort(465);
                email.setAuthenticator(new
DefaultAuthenticator("info6250final@gmail.com", "syn930310"));
                email.setSSLOnConnect(true);
                email.setFrom("no-reply@msis.neu.edu"); // This user email does
not
                                                        // exist
                email.setSubject("Password Reminder");
                email.setMsg(message); // Retrieve email from the DAO and send
this
                email.addTo(username);
                email.send();
```

```
            } catch (EmailException e) {
                System.out.println("Email cannot be sent");
            }
        }

    @RequestMapping(value = "user/validateemail.htm", method =
RequestMethod.GET)
    public String validateEmail(HttpServletRequest request, UserDAO userDao,
ModelMap map) {

            // The user will be sent the following link when the use registers
            // This is the format of the email
            //
http://hostname:8080/lab10/user/validateemail.htm?email=useremail&key1=<ra
ndom_number>&key2=<body
            // of the email that when user registers>
            HttpSession session = request.getSession();
            String email = request.getParameter("email");
            int key1 = Integer.parseInt(request.getParameter("key1"));
            int key2 = Integer.parseInt(request.getParameter("key2"));
            System.out.println(session.getAttribute("key1"));
            System.out.println(session.getAttribute("key2"));
```

```
            if ((Integer) (session.getAttribute("key1")) == key1 && ((Integer)
session.getAttribute("key2")) == key2) {
                try {
                    System.out.println("HI_______");
                    boolean updateStatus = userDao.updateUser(email);
                    if (updateStatus) {
                        return "user-login";
                    } else {

                        return "login-error";
                    }

                } catch (Exception e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            } else {
                map.addAttribute("errorMessage", "Link expired , generate new
link");
                map.addAttribute("resendLink", true);
                return "login-error";
```

```
        }

        return "user-login";

    }

    @RequestMapping(value = "/user/managepets.htm", method =
RequestMethod.GET)
    public String showManagePets(HttpServletRequest request, ModelMap map,
UserDAO userDao) {
        String un = String.valueOf(request.getSession().getAttribute("user"));
        User u = userDao.get(un);

        // System.out.println(u.getUserName());
        // Set<Pet> petList = u.getPetList();
        map.addAttribute("petList", u.getPetList());
        return "user-managepets";
    }

    @RequestMapping(value = "/user/dashboard.htm", method =
RequestMethod.GET)
```

```
    public String showDashboard(HttpServletRequest request, ModelMap map,
UserDAO userDao) {
        String un = String.valueOf(request.getSession().getAttribute("user"));
        User u = userDao.get(un);
        int newRequest = 0;
        for (Pet p : u.getPetList()) {
            for (IndividualRequest ir : p.getRequestList()) {
                if (ir.getStatus() == status.PENDING) {
                    newRequest += 1;
                }
            }
        }
        map.addAttribute("newRequest", newRequest);
        return "user-dashboard";
    }

    @RequestMapping(value = "/pet/addpet.htm", method =
RequestMethod.GET)
    public String showAddpet(HttpServletRequest request, ModelMap map,
PetDAO petDao) {
        if (request.getParameter("petid") != null) {
            map.addAttribute("option", "edit");
```

```
            map.addAttribute("petid", request.getParameter("petid"));
            map.addAttribute("pet",
petDao.get(Integer.parseInt(request.getParameter("petid"))));
            return "user-addpet";
        }
        map.addAttribute("option", "add");
        return "user-addpet";
    }

    @RequestMapping(value = "/user/logout.htm", method =
RequestMethod.GET)
    public String logOut(HttpServletRequest request) {
        request.getSession().invalidate();
        return "user-login";
    }

    @RequestMapping(value = "/user/editprofile.htm", method =
RequestMethod.GET)
    public String editProfile(HttpServletRequest request, ModelMap map,
UserDAO userDao) {
        String un = String.valueOf(request.getSession().getAttribute("user"));
        User u = userDao.get(un);
```

```
        map.addAttribute("user", u);
        return "user-editprofile";
    }

    @RequestMapping(value = "/user/editprofile.htm", method =
RequestMethod.POST)
    public String updateProfile(HttpServletRequest request, ModelMap map,
UserDAO userDao) {
        String un = String.valueOf(request.getSession().getAttribute("user"));
        User u = userDao.get(un);
        u.setEmail(request.getParameter("email"));
        u.setFirstName(request.getParameter("fname"));
        u.setLastName(request.getParameter("lname"));
        u.setPassword(request.getParameter("password"));
        u.setPhoneNumber(request.getParameter("phone"));
        u.setUserName(request.getParameter("username"));
        try {
            userDao.changeUser(u);
            request.getSession().setAttribute("user",
request.getParameter("username"));
        }catch(Exception e) {
            e.printStackTrace();
```

```
        }
        return "user-dashboard";
    }

    @RequestMapping(value = "/list/browseshelterlist.htm", method =
RequestMethod.GET)
    public String showShelterList(HttpServletRequest request, ModelMap map,
AnimalDAO aDao) {
        map.addAttribute("from", 0);

        try {
            List<ShelterAnimal> reportList = null;

            reportList = aDao.getList();
            map.addAttribute("aList", reportList);

            return "employee-manageanimal";
        } catch (Exception e) {
            e.printStackTrace();
        }

        return null;
```

```
    }

    @RequestMapping(value = "/animal/view.htm", method =
RequestMethod.GET)
    public String handleViewPet(HttpServletRequest request, AnimalDAO aDao,
ModelMap map) {
        int id = Integer.parseInt(request.getParameter("id"));
        try {
            ShelterAnimal a = aDao.get(id);
            map.addAttribute("animal", a);
            map.addAttribute("option", "VIEW");
            return "employee-viewanimal";
        }catch(Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

```
package com.myspring.petfinder.controller;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import javax.servlet.http.HttpServletRequest;

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 com.myspring.petfinder.dao.PetDAO;
import com.myspring.petfinder.dao.RequestDAO;
import com.myspring.petfinder.dao.UserDAO;
import com.myspring.petfinder.pojo.IndividualRequest;
import com.myspring.petfinder.pojo.IndividualRequest.status;
import com.myspring.petfinder.pojo.Pet;
import com.myspring.petfinder.pojo.User;

@Controller
```

```
public class RequestController {


    @RequestMapping(value = "/request/getrequests.htm", method =
RequestMethod.GET)
    public String getMessages(HttpServletRequest request, Model model,
UserDAO userDao) {
        // String userid = request.getParameter("userid");
        String un = String.valueOf(request.getSession().getAttribute("user"));
        User u = userDao.get(un);
        int option = Integer.parseInt(request.getParameter("option"));
        model.addAttribute("option", option);


        if (option == 1) {
            List<IndividualRequest> requestList = new
ArrayList<IndividualRequest>();
            for (Pet p : u.getPetList()) {
                for (IndividualRequest ir : p.getRequestList()) {
                    requestList.add(ir);
                }
            }
            model.addAttribute("requestList", requestList);
        } else {
```

```
                model.addAttribute("requestList", u.getRequestList());
            }
            return "user-viewrequests";
        }

        @RequestMapping(value = "/request/createrequest.htm", method =
RequestMethod.GET)
        public String makeRequest(HttpServletRequest request, PetDAO petDao,
Model model) {
            // String userid = request.getParameter("userid");
            model.addAttribute("receiver",

        petDao.get(Integer.parseInt(request.getParameter("petid"))).getOwner().getU
serName());
            model.addAttribute("petid",
Integer.parseInt(request.getParameter("petid")));
            model.addAttribute("action", "create");
            model.addAttribute("message", "");
            model.addAttribute("requestid", "");
            return "request-individualform";
        }
```

```
	@RequestMapping(value = { "/request/sendrequest.htm",
"/request/updaterequest.htm" }, method = RequestMethod.POST)
	public String sendRequest(HttpServletRequest request, PetDAO petDao,
Model model, UserDAO userDao,
			RequestDAO requestDao) {

		if (request.getParameter("action").equals("edit")) {
			try {

	requestDao.updateById(Integer.parseInt(request.getParameter("requestid")),
					request.getParameter("message"));
			} catch (Exception e) {
				e.printStackTrace();
			}

		} else {

			int petid = Integer.parseInt(request.getParameter("petid"));
			String un = String.valueOf(request.getSession().getAttribute("user"));
			User u = userDao.get(un);
			int userid = u.getPetOwnerId();
```

```
            Pet pet = petDao.get(petid);


            IndividualRequest r = new IndividualRequest();
            r.setMessage(request.getParameter("message"));
            r.setStatus(IndividualRequest.status.PENDING);
            r.setRequestdate(new Date());
            r.setFrom(u);
            r.setPet(pet);
            try {
                if(!requestDao.validateRequest(petid, userid)) {
                    model.addAttribute("message", "You have already sent a
request to this pet");
                    return "request-sent";
                }
                u.getRequestList().add(r);
                pet.getRequestList().add(r);
                requestDao.sendRequest(r);
                model.addAttribute("message", r);
                // request.getSession().setAttribute("user", u);
            } catch (Exception e) {
                e.printStackTrace();
            }
```

```
            }
            model.addAttribute("message", "Your request has been sent.");
            return "request-sent";
        }

        @RequestMapping(value = "/request/edit.htm", method =
RequestMethod.GET)
        public String editRequest(HttpServletRequest request, Model model,
RequestDAO requestDao) {
            try {
                IndividualRequest ir =
requestDao.getById(Integer.parseInt(request.getParameter("requestid")));
                model.addAttribute("receiver", request.getParameter("to"));
                model.addAttribute("petid", ir.getPet().getPetName());
                model.addAttribute("action", "edit");
                model.addAttribute("message", ir.getMessage());

        System.out.println(Integer.parseInt(request.getParameter("requestid")));
                model.addAttribute("requestid",
Integer.parseInt(request.getParameter("requestid")));
                return "request-individualform";
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    @RequestMapping(value = "/request/cancel.htm", method =
RequestMethod.GET)
    public String cancelRequest(HttpServletRequest request, PetDAO petDao,
Model model, UserDAO userDao,
            RequestDAO requestDao) {
        try {
            IndividualRequest ir =
requestDao.getById(Integer.parseInt(request.getParameter("requestid")));
            // ir.getPet().setListed(true);
            // ir.getPet().getRequestList().remove(ir);
            String un = String.valueOf(request.getSession().getAttribute("user"));
            User u = userDao.get(un);
            // String un = u.getUserName();
            petDao.list(ir.getPet());
            requestDao.cancel(ir);
            // u = userDao.get(un);
```

```
            // userDao.refreshUser(u);
            int option = 0;
            model.addAttribute("option", option);
            // request.getSession().setAttribute("user", u);
            // outward requests
            if (option == 1) {
                List<IndividualRequest> requestList = new
ArrayList<IndividualRequest>();
                for (Pet p : u.getPetList()) {
                    for (IndividualRequest i : p.getRequestList()) {
                        requestList.add(i);
                    }
                }
                model.addAttribute("requestList", requestList);
            }

            return "user-viewrequests";

        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
```

```
    }

    @RequestMapping(value = "/request/view.htm", method =
RequestMethod.GET)
    public String viewRequest(HttpServletRequest request, Model model,
RequestDAO requestDao) {
        try {
            IndividualRequest ir =
requestDao.getById(Integer.parseInt(request.getParameter("requestid")));
            model.addAttribute("petname", ir.getPet().getPetName());
            model.addAttribute("sender", ir.getFrom().getUserName());
            model.addAttribute("message", ir.getMessage());
            model.addAttribute("status", "noshow");
            if (ir.getStatus() == status.PENDING) {
                model.addAttribute("status", "show");
            }

            model.addAttribute("requestid",
Integer.parseInt(request.getParameter("requestid")));
            return "request-process";

        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
        return null;
    }

    @RequestMapping(value = "/request/process.htm", method =
RequestMethod.GET)
    public String processRequest(HttpServletRequest request, PetDAO petDao,
Model model, RequestDAO requestDao) {
        try {
            IndividualRequest ir =
requestDao.getById(Integer.parseInt(request.getParameter("requestid")));
            String action = request.getParameter("action");
            if (action.equals("reject")) {
                requestDao.reject(ir);
            } else {
                requestDao.approve(ir);
                Pet p = ir.getPet();
                petDao.unlist(p);
                for (IndividualRequest indReq : ir.getPet().getRequestList()) {
                    if (indReq.getRequestid() != ir.getRequestid()) {
                        requestDao.reject(indReq);
```

```
                    }
                }
            }

            return "request-processed";

        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

```
package com.myspring.petfinder.controller;

import java.util.Date;
import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

import com.myspring.petfinder.dao.AnimalDAO;
import com.myspring.petfinder.dao.EmployeeDAO;
import com.myspring.petfinder.dao.ReportDAO;
import com.myspring.petfinder.pojo.CompleteReport;
import com.myspring.petfinder.pojo.Employee;
import com.myspring.petfinder.pojo.HomelessReport;
import com.myspring.petfinder.pojo.HomelessReport.healthcondition;
import com.myspring.petfinder.pojo.HomelessReport.priority;
import com.myspring.petfinder.pojo.HomelessReport.status;
```

```
import com.myspring.petfinder.pojo.ShelterAnimal.gender;
import com.myspring.petfinder.pojo.ShelterAnimal;
import com.myspring.petfinder.xls.XlsView;

@Controller
public class ReportController {

	public ReportController() {
	}

	@RequestMapping(value = "/user/report.htm", method =
RequestMethod.GET)
	public String showReportForm(HttpServletRequest request) {
		return "user-homelessreport";
	}

	@RequestMapping(value = "/user/report.htm", method =
RequestMethod.POST)
	public String handleReport(HttpServletRequest request, ReportDAO
reportDao) {
		String location = request.getParameter("location");
		String description = request.getParameter("description");
```

```
String type = request.getParameter("type");
String aggressive = request.getParameter("aggressive");
String food = request.getParameter("food");
String danger = request.getParameter("danger");
String contact = request.getParameter("contact");
String health = request.getParameter("healthcondition");

boolean hasfood = false;
boolean indanger = false;
boolean isaggressive = false;
if (food != null) {
    hasfood = true;
}
if (danger != null) {
    indanger = true;
}
if (aggressive != null) {
    isaggressive = true;

}

HomelessReport report = new HomelessReport();
```

```
if (health.equals("G")) {
        report.setHealthcondition(healthcondition.GOOD);
} else if (health.equals("F")) {
        report.setHealthcondition(healthcondition.FAIR);
} else if (health.equals("P")) {
        report.setHealthcondition(healthcondition.POOR);
} else {
        report.setHealthcondition(healthcondition.CRITICAL);
}

report.setAnimaltype(type);
report.setContact(contact);
report.setLocation(location);
report.setMessage(description);
report.setReportdate(new Date());
report.setPriority(priority.MEDIUM);
report.setStatus(status.PENDING);
report.setIndanger(indanger);
report.setAggressive(isaggressive);
if (!request.getParameter("healthcondition").equals("C") && hasfood) {
        report.setPriority(priority.LOW);
}
```

```
            if (isaggressive || request.getParameter("healthcondition").equals("C") ||
indanger) {
                report.setPriority(priority.HIGH);
            }
            try {
                reportDao.create(report);
            } catch (Exception e) {
                e.printStackTrace();
            }

            return "report-success";
        }

    @RequestMapping(value = "/report/viewreport.htm", method =
RequestMethod.GET)
    public String viewReport(HttpServletRequest request, ReportDAO
reportDao, ModelMap map) {
            try {
                HomelessReport report =
reportDao.getById(Integer.parseInt(request.getParameter("reportid")));
                map.addAttribute("report", report);
                if(report.getStatus()==status.PENDING) {
```

```
                map.addAttribute("editable", true);
            }else {
                CompleteReport subreport =
reportDao.getComplete(report.getReportid());
                map.addAttribute("editable", false);
                //map.addAttribute("subreport", subreport);
                map.addAttribute("emp", subreport.getEmplopyee());
            }
            return "report-view";
        }catch(Exception e) {
            e.printStackTrace();
        }


        return null;
    }

    @RequestMapping(value = "/report/export.xls", method =
RequestMethod.GET)
    public ModelAndView exportReports(HttpServletRequest request,
ReportDAO reportDao) {
        List<HomelessReport> reportList = null;
```

```
        try {
            reportList = reportDao.getList(status.PENDING);
        }catch(Exception e) {
            e.printStackTrace();
        }

        return new ModelAndView(new XlsView(), "list", reportList);
    }

    @RequestMapping(value = "/report/complete.htm", method =
RequestMethod.GET)
    public String completeReport(HttpServletRequest request, ReportDAO
reportDao, ModelMap map) {
        int reportid = Integer.parseInt(request.getParameter("reportid"));
        String animaltype = request.getParameter("type");
        map.addAttribute("reportid", reportid);
        map.addAttribute("type", animaltype);
        return "report-complete";
    }

    @RequestMapping(value = "/report/complete.htm", method =
RequestMethod.POST)
```

```
public String doCompleteReport(HttpServletRequest request, ReportDAO
reportDao, ModelMap map, AnimalDAO aDao, EmployeeDAO eDao) {
        int reportid = Integer.parseInt(request.getParameter("reportid"));
        String username =
String.valueOf(request.getSession().getAttribute("user"));
        try {
            Employee employee = eDao.get(username);
            HomelessReport report = reportDao.getById(reportid);
            CompleteReport subreport = new CompleteReport();
            subreport.setNote(request.getParameter("note"));
            subreport.setEmplopyee(employee);
            subreport.setReport(report);
            subreport.setFinishDate(new Date());
            report.setCompleteReport(subreport);
            report.setStatus(status.COMPLETE);
            reportDao.update(report);
            if(request.getParameter("add")!=null) {
                ShelterAnimal a = new ShelterAnimal();
                a.setType(request.getParameter("animaltype"));
                a.setAge(Integer.parseInt(request.getParameter("age")));
                a.setBreed(request.getParameter("petbreed"));
                a.setGender(gender.MALE);
```

```
                    if(request.getParameter("petgender").equals("F")) {
                        a.setGender(gender.FEMALE);
                    }
                    a.setColor(request.getParameter("petcolor"));
                    a.setName(request.getParameter("name"));
                    a.setDescription(request.getParameter("petdescription"));
                    aDao.register(a);
                }

                return "report-updatesuccess";
            }catch(Exception e) {
                e.printStackTrace();
            }
            return null;
        }

    @RequestMapping(value = "/report/map.htm", method =
RequestMethod.GET)
    public String showMap(HttpServletRequest request, ReportDAO reportDao,
ModelMap map) {

            return "map-test";
```

```
    }

}
```

```
package com.myspring.petfinder.controller;

import javax.servlet.http.HttpServletRequest;


import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.myspring.petfinder.dao.PetDAO;
import com.myspring.petfinder.dao.UserDAO;
import com.myspring.petfinder.pojo.Pet;
import com.myspring.petfinder.pojo.Pet.gender;
import com.myspring.petfinder.pojo.User;

@Controller
public class PetController {

    @RequestMapping(value = "/pet/delete.htm", method =
RequestMethod.GET)
```

```
    public String deletePet(HttpServletRequest request, PetDAO petDao,
UserDAO userDao, ModelMap map) {
        int id = Integer.parseInt(request.getParameter("petid"));

        try {
            Pet p = petDao.get(id);
            petDao.delete(p);
            // boolean isOk = userDao.refreshUser(u);
            // String un = u.getUserName();
            // u = userDao.get(un);
            // request.getSession().setAttribute("user", u);
            String un = String.valueOf(request.getSession().getAttribute("user"));
            User u = userDao.get(un);
            map.addAttribute("petList", u.getPetList());
            return "user-managepets";

        } catch (Exception e) {
            e.printStackTrace();
        }

        return null;
    }
```

```
    @RequestMapping(value = "/pet/addnewpet.htm", method =
RequestMethod.POST)
    public String handleNewPet(HttpServletRequest request, UserDAO userDao,
ModelMap map) {
        String un = String.valueOf(request.getSession().getAttribute("user"));
        User u = userDao.get(un);
        Pet p = new Pet();
        p.setOwner(u);
        p.setAnimalType(request.getParameter("pettype"));
        p.setBreed(request.getParameter("petbreed"));
        p.setColor(request.getParameter("petcolor"));
        p.setDescription(request.getParameter("petdescription"));
        p.setLocation(request.getParameter("petlocation"));
        p.setPetAge(Integer.parseInt(request.getParameter("petage")));
        p.setPetName(request.getParameter("petname"));
        String sex = request.getParameter("petgender");
        p.setListed(true);
        if (sex.equals("MALE")) {
            p.setGender(gender.MALE);
        } else {
            p.setGender(gender.FEMALE);
```

```
        }

        try {
            userDao.registerPet(p);
            // User user = userDao.get(u.getUserName());
            // request.getSession().setAttribute("user", user);
            return "pet-addsuccess";
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        return null;
    }

    @RequestMapping(value = "/pet/editpet.htm", method =
RequestMethod.POST)
    public String handleEditPet(HttpServletRequest request, UserDAO userDao,
PetDAO petDao, ModelMap map) {
        String un = String.valueOf(request.getSession().getAttribute("user"));
        User u = userDao.get(un);
        Pet pet = null;
```

```
for (Pet p : u.getPetList()) {
    if (p.getPetId() == Integer.parseInt(request.getParameter("petid"))) {
        pet = p;
    }
}
// Pet p = new Pet();
// pet.setOwner(u);
pet.setAnimalType(request.getParameter("pettype"));
pet.setBreed(request.getParameter("petbreed"));
pet.setColor(request.getParameter("petcolor"));
pet.setDescription(request.getParameter("petdescription"));
pet.setLocation(request.getParameter("petlocation"));
pet.setPetAge(Integer.parseInt(request.getParameter("petage")));
pet.setPetName(request.getParameter("petname"));
String sex = request.getParameter("petgender");
if (sex.equals("MALE")) {
    pet.setGender(gender.MALE);
} else {
    pet.setGender(gender.FEMALE);
}

try {
```

```
            boolean isOk = petDao.updatePet(pet);
            if (isOk) {
                // request.getSession().setAttribute("user", u);
                u = userDao.get(un);
                map.addAttribute("petList", u.getPetList());
                return "user-managepets";
            }
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        return "null";
    }

    @RequestMapping(value = "/pet/view.htm", method =
RequestMethod.GET)
    public String handleViewPet(HttpServletRequest request, PetDAO petDao,
ModelMap map) {
        int petid = Integer.parseInt(request.getParameter("petid"));
        Pet pet = petDao.get(petid);
        map.addAttribute("pet", pet);
```

```
            map.addAttribute("option", "view");

            return "user-viewpet";

        }

    }
```

```
package com.myspring.petfinder.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.myspring.petfinder.dao.PetDAO;
import com.myspring.petfinder.dao.ReportDAO;
import com.myspring.petfinder.dao.UserDAO;
import com.myspring.petfinder.pojo.User;

@Controller
public class ListController {

    @RequestMapping(value = "/list/browseadoptionlist.htm", method =
RequestMethod.GET)
    public String getMessages(HttpServletRequest request, Model model,
PetDAO petDao, UserDAO userDao) {
```

```
        //String userid = request.getParameter("userid");
        String un = String.valueOf(request.getSession().getAttribute("user"));
        User u = userDao.get(un);
        int option = Integer.parseInt(request.getParameter("option"));
        int page = Integer.parseInt(request.getParameter("page"));
        //outward requests
        if(option==0) {
            //individual list
            model.addAttribute("petList", petDao.getByPageI(page,
u.getPetOwnerId()));
        }
        model.addAttribute("option", option);
        model.addAttribute("pages", petDao.getTotalPage(u.getPetOwnerId()));
        return "user-viewlist";
    }
    @RequestMapping(value = "/list/map.htm", method = RequestMethod.GET)
    public String showMap(HttpServletRequest request, ReportDAO reportDao,
ModelMap map) {

        return "map-test2";
    }
}
```

```
package com.myspring.petfinder.controller;

import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;


import com.myspring.petfinder.dao.AnimalDAO;
import com.myspring.petfinder.dao.EmployeeDAO;

import com.myspring.petfinder.dao.ReportDAO;

import com.myspring.petfinder.pojo.Employee;
import com.myspring.petfinder.pojo.HomelessReport;

import com.myspring.petfinder.pojo.HomelessReport.status;
```

```
import com.myspring.petfinder.pojo.ShelterAnimal.gender;
import com.myspring.petfinder.pojo.ShelterAnimal;



@Controller
public class EmployeeController {


    @RequestMapping(value = "/employee/login.htm", method =
RequestMethod.GET)
    public String showLogin(HttpServletRequest request) {
        HttpSession session = request.getSession();
        if (session.getAttribute("user") != null) {
            return "employee-portal";
        }
        return "employee-login";
    }


    @RequestMapping(value = "/animal/add.htm", method =
RequestMethod.GET)
    public String showAdd(ModelMap map) {
        map.addAttribute("option", "ADD");
        return "employee-viewanimal";
```

```
    }

    @RequestMapping(value = "/animal/add.htm", method =
RequestMethod.POST)
    public String handleAdd(HttpServletRequest request, ModelMap map,
AnimalDAO aDao) {
        try {
            map.addAttribute("from", 1);
            ShelterAnimal a = new ShelterAnimal();
            a.setType(request.getParameter("type"));
            a.setAge(Integer.parseInt(request.getParameter("age")));
            a.setBreed(request.getParameter("breed"));
            a.setGender(gender.MALE);
            if(request.getParameter("gender").equals("FEMALE")) {
                a.setGender(gender.FEMALE);
            }
            a.setColor(request.getParameter("color"));
            a.setName(request.getParameter("name"));
            a.setDescription(request.getParameter("description"));
            aDao.register(a);
            List<ShelterAnimal> reportList = null;
            reportList = aDao.getList();
```

```
            map.addAttribute("aList", reportList);
            return "employee-manageanimal";
        }catch(Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    @RequestMapping(value = "/employee/login.htm", method =
RequestMethod.POST)
    public String doLogin(HttpServletRequest request, EmployeeDAO empDao,
ReportDAO reportDao, ModelMap map) {
        HttpSession session = request.getSession();

        String username = request.getParameter("username");
        String password = request.getParameter("password");
        try {
            Employee u = empDao.get(username, password);
            if (u != null) {
                session.setAttribute("user", u.getUsername());
                // session.setAttribute("userid", u.getPetOwnerId());
                int newRequest = 0;
```

```
                List<HomelessReport> list = reportDao.getList();
                for (HomelessReport hr : list) {

                    if (hr.getStatus() == status.PENDING) {
                        newRequest += 1;
                    }

                }
                map.addAttribute("newRequest", newRequest);
                return "employee-portal";
            } else {
                map.addAttribute("errorMessage", "Invalid
username/password!");
                return "login-error";
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
```

```
    @RequestMapping(value = "/employee/viewreports.htm", method =
RequestMethod.GET)
    public String showReports(HttpServletRequest request, ReportDAO
reportDao, ModelMap map) {
        int option = Integer.parseInt(request.getParameter("option"));

        try {
            List<HomelessReport> reportList = null;
            if (option == 0) {
                reportList = reportDao.getList();
                map.addAttribute("reportList", reportList);
            }
            return "employee-viewreport";
        } catch (Exception e) {
            e.printStackTrace();
        }

        return null;
    }

    @RequestMapping(value = "/employee/manageanimals.htm", method =
RequestMethod.GET)
```

```
    public String manageAnimal(HttpServletRequest request, AnimalDAO aDao,
ModelMap map) {
        map.addAttribute("from", 1);

        try {
            List<ShelterAnimal> reportList = null;

            reportList = aDao.getList();
            map.addAttribute("aList", reportList);

            return "employee-manageanimal";
        } catch (Exception e) {
            e.printStackTrace();
        }

        return null;
    }

    @RequestMapping(value = "/animal/edit.htm", method =
RequestMethod.GET)
    public String handleViewPet(HttpServletRequest request, AnimalDAO aDao,
ModelMap map) {
```

```
        int id = Integer.parseInt(request.getParameter("id"));
        try {
            ShelterAnimal a = aDao.get(id);
            map.addAttribute("animal", a);
            map.addAttribute("option", "EDIT");
            return "employee-viewanimal";
        }catch(Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    @RequestMapping(value = "/animal/delete.htm", method =
RequestMethod.GET)
    public String handleDeletePet(HttpServletRequest request, AnimalDAO
aDao, ModelMap map) {
        int id = Integer.parseInt(request.getParameter("id"));
        try {
            ShelterAnimal a = aDao.get(id);
            aDao.delete(a);

            List<ShelterAnimal> reportList = null;
```

```
            map.addAttribute("from", 1);
            reportList = aDao.getList();
            map.addAttribute("aList", reportList);
            return "employee-manageanimal";
        }catch(Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    @RequestMapping(value = "/animal/edit.htm", method =
RequestMethod.POST)
    public String handleEditPet(HttpServletRequest request, AnimalDAO aDao,
ModelMap map) {
        int id = Integer.parseInt(request.getParameter("id"));
        try {
            map.addAttribute("from", 1);
            ShelterAnimal a = aDao.get(id);
            a.setAge(Integer.parseInt(request.getParameter("age")));
            a.setBreed(request.getParameter("breed"));
            a.setColor(request.getParameter("color"));
            a.setDescription(request.getParameter("description"));
```

```
            a.setName(request.getParameter("name"));
            a.setType(request.getParameter("type"));
            a.setGender(gender.MALE);
            if(request.getParameter("gender").equals("FEMALE")) {
                a.setGender(gender.FEMALE);
            }
            aDao.update(a);
            List<ShelterAnimal> reportList = null;

            reportList = aDao.getList();
            map.addAttribute("aList", reportList);
            return "employee-manageanimal";
        }catch(Exception e) {
            e.printStackTrace();
        }
        return null;
    }

}
```

```
package com.myspring.petfinder.controller;

import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;

import org.json.JSONArray;
import org.json.JSONObject;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import com.myspring.petfinder.dao.PetDAO;
import com.myspring.petfinder.dao.ReportDAO;
import com.myspring.petfinder.dao.UserDAO;
import com.myspring.petfinder.pojo.HomelessReport;
import com.myspring.petfinder.pojo.HomelessReport.status;
import com.myspring.petfinder.pojo.Pet;

@Controller
```

```
public class AjaxController {


    @RequestMapping(value = "/ajaxservice.htm", method =
RequestMethod.GET)
    @ResponseBody
    public String ajaxService(HttpServletRequest request, PetDAO petDao) {
        String type = request.getParameter("type");
        String breed = request.getParameter("breed");
        String color = request.getParameter("color");
        String agefrom = request.getParameter("agefrom");
        String ageto = request.getParameter("ageto");
        String sex = request.getParameter("sex");


        String uid = String.valueOf(request.getSession().getAttribute("userid"));
        int userid = Integer.parseInt(uid);
        List<Pet> petlist = new ArrayList<Pet>();
        try {
            petlist = petDao.query(userid, type, breed, color, agefrom, ageto,
sex);
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
        JSONArray jarray = new JSONArray();
        for (Pet p : petlist) {
            JSONObject j = new JSONObject();
            j.put("petid", p.getPetId());
            j.put("petname", p.getPetName());
            j.put("petage", p.getPetAge());
            j.put("pettype", p.getAnimalType());
            j.put("sex", p.getGender());
            j.put("breed", p.getBreed());
            j.put("color", p.getColor());
            jarray.put(j);
        }
        // System.out.println(petlist.size());
        return jarray.toString();
    }

    @RequestMapping(value = "/validateusernameajax.htm", method =
RequestMethod.GET)
    @ResponseBody
    public String validateUn(HttpServletRequest request, UserDAO userDao) {
        String username = request.getParameter("username");
        String userid = request.getParameter("userid");
```

```
        int id = 0;
        if (userid != null) {
            id = Integer.parseInt(userid);
        }
        String result = "";
        if (!userDao.validateUsername(id, username)) {
            result += "Username already inuse";
        }
        return result;
    }

    @RequestMapping(value = "/mapservice.htm", method =
RequestMethod.GET)
    @ResponseBody
    public String gmap(HttpServletRequest request, ReportDAO rDao) {
        try {
            List<HomelessReport> list = rDao.getList(status.PENDING);
            JSONArray jarray = new JSONArray();
            for (HomelessReport r : list) {
                JSONObject j = new JSONObject();
                j.put("type", r.getAnimaltype());
                j.put("location", r.getLocation());
```

```
                j.put("description", r.getMessage());
                j.put("priority", r.getPriority().toString());
                j.put("contact", r.getContact());
                jarray.put(j);
            }
            // System.out.println(petlist.size());
            return jarray.toString();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    @RequestMapping(value = "/petmapservice.htm", method =
RequestMethod.GET)
    @ResponseBody
    public String pmap(HttpServletRequest request, PetDAO pDao) {
        try {
            int uid =
Integer.parseInt(String.valueOf(request.getSession().getAttribute("userid")));
            List<Pet> list = pDao.getOthers(uid);
            JSONArray jarray = new JSONArray();
```

```
            for (Pet p : list) {
                JSONObject j = new JSONObject();
                j.put("type", p.getAnimalType());
                j.put("location", p.getLocation());
                j.put("breed", p.getBreed());
                j.put("description", p.getDescription());
                j.put("contact", p.getOwner().getPhoneNumber());
                jarray.put(j);
            }
            // System.out.println(petlist.size());
            return jarray.toString();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

}
```

```
package com.myspring.petfinder.pojo;

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.Table;

@Entity
@Table(name = "petowner")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "petownerid", unique = true, nullable = false)
    private int petOwnerId;
```

```
@Column(name = "username")
private String userName;

@Column(name = "password")
private String password;

@Column(name = "firstname")
private String firstName;

@Column(name = "lastname")
private String lastName;

@Column(name = "phonenumber")
private String phoneNumber;

@Column(name = "email")
private String email;

@Column(name = "status")
private int status;
```

```
    @OneToMany(fetch = FetchType.EAGER, mappedBy = "owner",
orphanRemoval=true, cascade = CascadeType.ALL)
    private Set<Pet> petList;

    @OneToMany(fetch = FetchType.EAGER, mappedBy = "from", cascade =
CascadeType.ALL)
    private Set<IndividualRequest> requestList;

    public User() {
    }

    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 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 getPhoneNumber() {
```

```
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public int getStatus() {
        return status;
    }

    public void setStatus(int status) {
        this.status = status;
    }
```

```
    public Set<Pet> getPetList() {
        return petList;
    }

    public void setPetList(Set<Pet> petList) {
        this.petList = petList;
    }

    public int getPetOwnerId() {
        return petOwnerId;
    }

    public Set<IndividualRequest> getRequestList() {
        return requestList;
    }

    public void setRequestList(Set<IndividualRequest> requestList) {
        this.requestList = requestList;
    }

}
```

```
package com.myspring.petfinder.pojo;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="shelteranimal")
public class ShelterAnimal {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "animalid", unique = true, nullable = false)
    private    int animalid;

    @Column(name="name")
    private String name;
```

```
@Column(name="age")

private int age;


@Column(name="type")

private String type;


@Enumerated(EnumType.STRING)

@Column(name="gender")

private gender gender;


@Column(name = "color")

private String color;


@Column(name = "description")

private String description;


@Column(name = "breed")

private String breed;


public ShelterAnimal() {}
```

```
public enum gender {
    MALE, FEMALE
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public int getAge() {
    return age;
}

public void setAge(int age) {
    this.age = age;
}

public String getType() {
    return type;
```

```
}

public void setType(String type) {
    this.type = type;
}

public gender getGender() {
    return gender;
}

public void setGender(gender gender) {
    this.gender = gender;
}

public String getColor() {
    return color;
}

public void setColor(String color) {
    this.color = color;
}
```

```
    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getBreed() {
        return breed;
    }

    public void setBreed(String breed) {
        this.breed = breed;
    }

    public int getAnimalid() {
        return animalid;
    }

}
```

```
package com.myspring.petfinder.pojo;


import java.util.Set;


import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.EnumType;

import javax.persistence.Enumerated;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.OneToMany;

import javax.persistence.Table;


@Entity

@Table(name = "pet")

public class Pet {
```

```
@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

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

private int petId;


@Column(name = "petname")

private String petName;


@Column(name = "petage")

private int petAge;


@Column(name = "animaltype")

private String animalType;


@ManyToOne(fetch = FetchType.EAGER)

@JoinColumn(name = "petownerid", nullable = false)

private User owner;


@Enumerated(EnumType.STRING)

@Column(name="gender")

private gender gender;
```

```
    @Column(name = "color")
    private String color;

    @Column(name = "breed")
    private String breed;

    @Column(name = "location")
    private String location;

    @Column(name = "description")
    private String description;

    @OneToMany(fetch = FetchType.EAGER, mappedBy = "pet",
orphanRemoval=true, cascade = CascadeType.ALL)
    private Set<IndividualRequest> requestList;

    @Column(name = "listed")
    private boolean listed;

    public Pet() {
    }
```

```
public enum gender {
    MALE, FEMALE
}

public String getPetName() {
    return petName;
}

public void setPetName(String petName) {
    this.petName = petName;
}

public int getPetAge() {
    return petAge;
}

public void setPetAge(int petAge) {
    this.petAge = petAge;
}

public String getAnimalType() {
    return animalType;
```

```
}

public void setAnimalType(String animalType) {
    this.animalType = animalType;
}

public User getOwner() {
    return owner;
}

public void setOwner(User owner) {
    this.owner = owner;
}

public String getColor() {
    return color;
}

public void setColor(String color) {
    this.color = color;
}
```

```
public gender getGender() {
    return gender;
}

public void setGender(gender gender) {
    this.gender = gender;
}

public String getBreed() {
    return breed;
}

public void setBreed(String breed) {
    this.breed = breed;
}

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 int getPetId() {
    return petId;
}

public Set<IndividualRequest> getRequestList() {
    return requestList;
}

public void setRequestList(Set<IndividualRequest> requestList) {
    this.requestList = requestList;
}
```

```
    public boolean isListed() {
        return listed;
    }

    public void setListed(boolean listed) {
        this.listed = listed;
    }

}
```

```
package com.myspring.petfinder.pojo;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

@Entity
@Table(name = "individualadoptionrequest")
public class IndividualRequest {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

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

private int requestid;


@ManyToOne(fetch = FetchType.EAGER)

@JoinColumn(name = "requestby", nullable = false)

private User from;


@Enumerated(EnumType.STRING)

@Column(name="status")

private status status;


@Column(name="requestdate")

private Date requestdate;


@Column(name="message")

private String message;


@ManyToOne(fetch = FetchType.EAGER)

@JoinColumn(name = "petid", nullable = false)

private Pet pet;


public IndividualRequest() {}
```

```
public enum status{
	PENDING, REJECTED, APPROVED
}

public User getFrom() {
	return from;
}

public void setFrom(User from) {
	this.from = from;
}

public status getStatus() {
	return status;
}

public void setStatus(status status) {
	this.status = status;
}

public Date getRequestdate() {
```

```
        return requestdate;
    }

    public void setRequestdate(Date requestdate) {
        this.requestdate = requestdate;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public int getRequestid() {
        return requestid;
    }

    public Pet getPet() {
        return pet;
    }
```

```
    public void setPet(Pet pet) {
        this.pet = pet;
    }

}
```

```
package com.myspring.petfinder.pojo;

import java.util.Date;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;

@Entity
@Table(name="homelessreport")
public class HomelessReport {

    @Id
```

```
@GeneratedValue(strategy = GenerationType.IDENTITY)

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

private int reportid;


@Column(name="contact")

private String contact;


@Column(name="reportdate")

private Date reportdate;


@Enumerated(EnumType.ORDINAL)

@Column(name="status")

private status status;


@Enumerated(EnumType.ORDINAL)

@Column(name="priority")

private priority priority;


@Column(name="message")

private String message;


@Column(name="animaltype")
```

```
    private String animaltype;

    @Enumerated(EnumType.STRING)
    @Column(name="healthcondition")
    private healthcondition healthcondition;

    @Column(name="location")
    private String location;

    @OneToOne(mappedBy="report", cascade=CascadeType.ALL, fetch =
FetchType.EAGER)
    private CompleteReport completeReport;

    @Column(name="aggressive")
    private boolean aggressive;

    @Column(name="indanger")
    private boolean indanger;

    public HomelessReport() {
```

```
}


public enum status {
    PENDING, COMPLETE
}
public enum priority {
    HIGH, MEDIUM, LOW
}
public enum healthcondition {
    GOOD, FAIR, POOR, CRITICAL
}
public String getContact() {
    return contact;
}
public void setContact(String contact) {
    this.contact = contact;
}
public Date getReportdate() {
    return reportdate;
}
public void setReportdate(Date reportdate) {
    this.reportdate = reportdate;
```

```
}
public status getStatus() {
    return status;
}
public void setStatus(status status) {
    this.status = status;
}
public priority getPriority() {
    return priority;
}
public void setPriority(priority priority) {
    this.priority = priority;
}
public String getMessage() {
    return message;
}
public void setMessage(String message) {
    this.message = message;
}
public String getAnimaltype() {
    return animaltype;
}
```

```
public void setAnimaltype(String animaltype) {
    this.animaltype = animaltype;
}
public healthcondition getHealthcondition() {
    return healthcondition;
}
public void setHealthcondition(healthcondition healthcondition) {
    this.healthcondition = healthcondition;
}
public String getLocation() {
    return location;
}
public void setLocation(String location) {
    this.location = location;
}
public int getReportid() {
    return reportid;
}
public CompleteReport getCompleteReport() {
    return completeReport;
}
public void setCompleteReport(CompleteReport completeReport) {
```

```
        this.completeReport = completeReport;
    }
    public boolean isAggressive() {
        return aggressive;
    }
    public void setAggressive(boolean aggressive) {
        this.aggressive = aggressive;
    }
    public boolean isIndanger() {
        return indanger;
    }
    public void setIndanger(boolean indanger) {
        this.indanger = indanger;
    }


}
```

```
package com.myspring.petfinder.pojo;



import javax.persistence.Column;

import javax.persistence.Entity;


import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;


@Entity

@Table(name="shelteremployee")

public class Employee {


    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

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

    private int id;


    @Column(name="firstname")

    private String fname;
```

```
@Column(name="lastname")
private String lname;

@Column(name="username")
private String username;

@Column(name="password")
private String password;

public Employee() {}

public String getFname() {
    return fname;
}

public void setFname(String fname) {
    this.fname = fname;
}

public String getLname() {
    return lname;
```

```
}

public void setLname(String lname) {
    this.lname = lname;
}

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 int getId() {
        return id;
    }

}
```

```
package com.myspring.petfinder.pojo;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToOne;
import javax.persistence.PrimaryKeyJoinColumn;
import javax.persistence.Table;

import org.hibernate.annotations.Fetch;
import org.hibernate.annotations.FetchMode;
import org.hibernate.annotations.GenericGenerator;
import org.hibernate.annotations.Parameter;

@Entity
@Table(name="reportfinished")
```

```
public class CompleteReport {

    @Id
    @Column(name="reportid", unique=true, nullable=false)
    @GeneratedValue(generator="gen")
    @GenericGenerator(name="gen", strategy="foreign",
parameters=@Parameter(name="property", value="report"))
    private int reportid;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name="finishedby")
    @Fetch(FetchMode.SELECT)
    private Employee emplopyee;

    @Column(name="note")
    private String note;

    @Column(name="finishdate")
    private Date finishDate;

    @OneToOne
    @PrimaryKeyJoinColumn
```

```
private HomelessReport report;

public CompleteReport() {}

public Employee getEmplopyee() {
    return emplopyee;
}

public void setEmplopyee(Employee emplopyee) {
    this.emplopyee = emplopyee;
}

public String getNote() {
    return note;
}

public void setNote(String note) {
    this.note = note;
}

public Date getFinishDate() {
    return finishDate;
```

```
    }

    public void setFinishDate(Date finishDate) {
        this.finishDate = finishDate;
    }

    public int getReportid() {
        return reportid;
    }

    public HomelessReport getReport() {
        return report;
    }

    public void setReport(HomelessReport report) {
        this.report = report;
    }

}
```