Web高级编程大作业报告模板

封面（使用课程设计报告封面）

纸张（16开）

目录

1. 系统分析

1.1 需求概述

1.2 用例分析与描述

1.2.1 用例分析

1.2.2 用例描述

2. 系统设计

2.1 多层结构设计

2.2 问题域设计

2.3 持久化设计

2.4 界面设计

3. 系统实现

3.1 表达层

3.2 业务层

3.3 持久化层

4. 系统测试

4.1 单元测试

4.2 功能测试

4.3 性能测试（可选）

5. 总结与体会

字体说明：中文宋体、英文Times New Roman

1级标题 四号加粗

2级标题 小四加粗

3级标题 五号加粗

正文 五号

图号：[章节号]-[序号] [说明]，小五，位于图的下方，例如：图 1-5 用例图

表号：[章节号]-[序号] [说明]，小五，位于表的上方，例如：表 1-5 宠物表

# 1 系统分析

## 1.1 需求概述

项目背景：

学校课程主要采取助教配合的管理制度。助教主要职责是通过与任课老师的积极配合来保证教学工作的顺利进行。助教有义务积极主动地解决在教学中出现的有可能影响教学工作正常进行的各项事宜，时刻关注学员的动向。如：学生出勤情况、学生的情绪变化、意外事故的发生及对学生深入有效的了解等等。

需求说明：

要求列表：

* 助教选聘：发布助教岗位信息，进行助教选拔。
* 作业批改：助教将作业、实验等批改成绩录入系统，并汇报作业情况。
* 助教考核：对助教的工作进行考核。
* 报表分析：系统自动生成各类报表与成绩分析结果。

## 1.2 用例分析与描述

### 1.2.1 用例图





### 1.2.2 用例描述

|  |
| --- |
| 用例名称：登录 |
| 描述：用户输入账号密码登录系统 |
| 角色:管理员、助教 |
| 主事件流:  1. 用户点击登录系统，用例开始  2. 用户输入账号密码  3. 输入正确，登录系统；输入错误，返回第二步  4．用例结束 |

|  |
| --- |
| 用例名称：作业批改 |
| 描述：助教对作业进行批改 |
| 角色:助教 |
| 主事件流:  1. 登陆成功，用例开始  2. 助教查看作业信息  3. 助教批改作业  4．批改完成，用例结束 |

（每一章都必须另起一页）

# 2 系统设计

## 2.1 多层结构设计

说明如何进行分层，各层运用哪些技术，各层分别由哪些package、页面以及其他资源构成，画出系统的组件图

我们将助教选聘系统分为三部分:表达层，服务层，仓库层。

从表达层来看：我们准备使用html技术，工厂模式 ，servlet技术，jsp技术等。

我们的表达层有登录页面，注册页面，游客页面，助教管理页面，老师管理页面，在游客页面中，有岗位信息页面，在助教管理页面中，有个人信息页面，批改作业页面，信息报表页面，数据分析页面，作业评价页面。

从服务层来看，我们准备使用jdbc技术，mybatis技术，javabean等。我们用model包存模板，各种bean，用controller包存jdbc的各种逻辑。

从仓库层来看，准备使用jdbc技术，ibatis技术等。

The Presentation Layer is implemented as a Java EE Web Application and provides a very thin and concise Model-View-Controller type user interface to the Business and Persistence Layers. …

Since the PetClinic application is all about database access and there is very little business logic in the application outside of that, there is no separation of the primary Business and Persistence Layer API's. While this design technique should not be used for an application with more complex business logic, it is acceptable here because all of the non-persistence related business rules have been implemented in business objects and have not leaked into the Persistence Layer. The most important facet of the design is that the Business and Persistence Layers are COMPLETELY independent of the Presentation Layer. …

The Persistence Layer can be configured to use either HSQL or MySQL with any one of the following data access technologies aided by infrastructure provided by Spring:

* JDBC
* ……

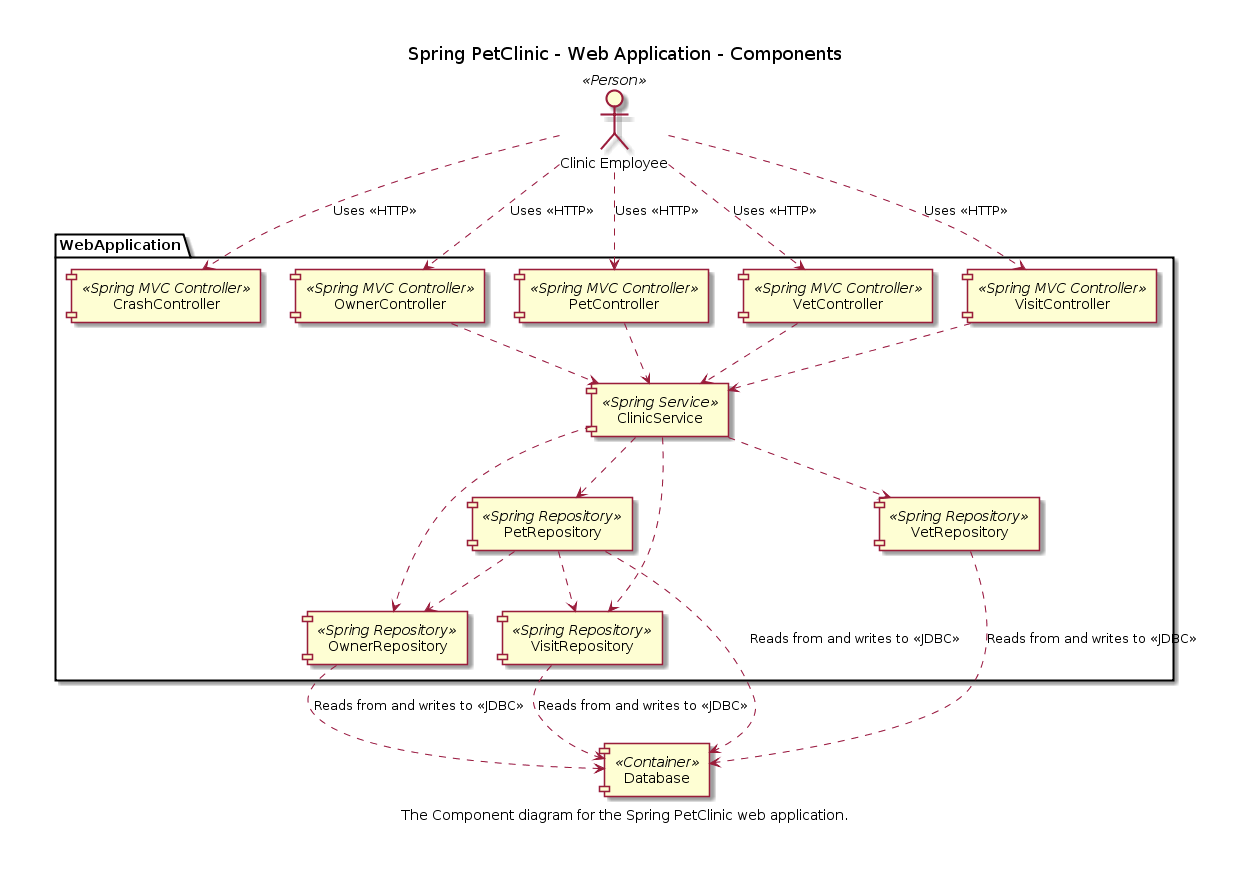


图2.1 宠物医院系统…….

## 2.2 问题域设计

### 2.2.1 域模型设计

分析系统中的实体并给出系统的领域模型

系统中的实体有：老师，学生，游客，助教，成绩单。

以宠物诊所来说，「宠物」（Pet）和「兽医」（Vet）都是经常出现的名词，它们也都是从需求描述中识别出来的「实体」。……

PetClinic is a fairly simple application with only seven models. The following chart shows all the models and relationships among them. …

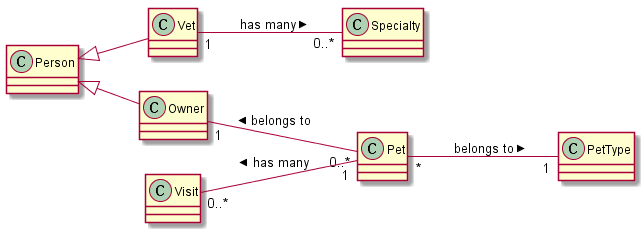


图2.2 宠物医院系统域模型

### 2.2.2 Service接口设计

通过活动图、状态图、通信图或顺序图分析得到Service接口的方法，并画出相关类图。

## 2.3 持久化设计

### 2.3.1 数据库设计

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 助教： | 学号 | 姓名 | 教导班级 | 所属老师 | 老师评分 | 效绩 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 作业表： | 编号 | 作业号 | 学号 | 班级 | 成绩 |

|  |  |  |  |
| --- | --- | --- | --- |
| 学生： | 学号 | 姓名 | 班级 |

|  |  |  |  |
| --- | --- | --- | --- |
| 老师： | 工号 | 姓名 | 班级 |

|  |  |  |
| --- | --- | --- |
| 账号： | 密码 | 级别 |

### 2.3.2 数据访问设计

分析并给出数据库配置以及repository类的类图。

## 2.4 界面设计

### 2.4.1 页面链接关系

（分析页面之间的连接关系以及url所对应的Controller方法。）

The following table shows routes created for resource pets:

表2.1 URL对应功能说明（pets）

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP verb | URL | action（Controller类的方法） | used for |
| GET | /pets/new | add | return an HTML form for creating a new pet |
| POST | /pets/new | create | create a new pet |
| GET | /pets/1/edit | edit | return an HTML form for editing a pet |
| POST | /pets/1/edit | update | update a specific pet |

The models will look like this:

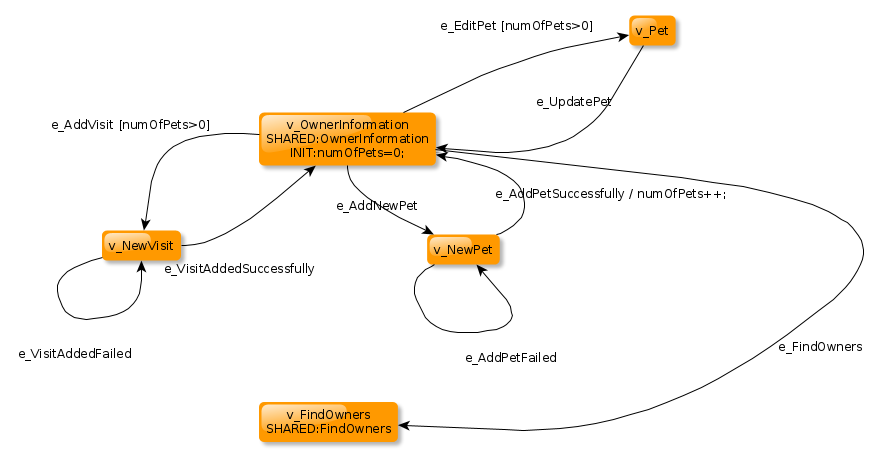


图2.2 宠物医院系统…….

### 2.4.2 页面设计

（给出页面设计及其操作流程。）

You can then access petclinic here: http://localhost:8080/

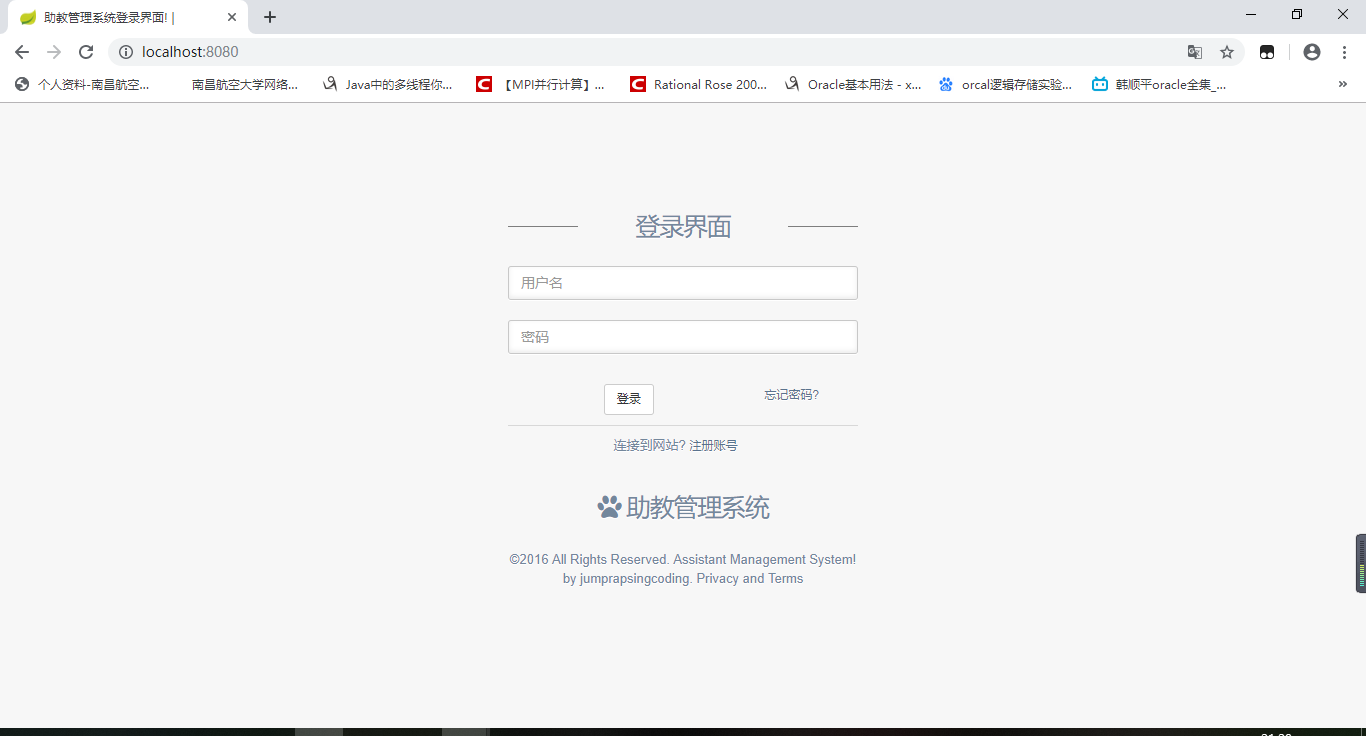


图2.3 登录

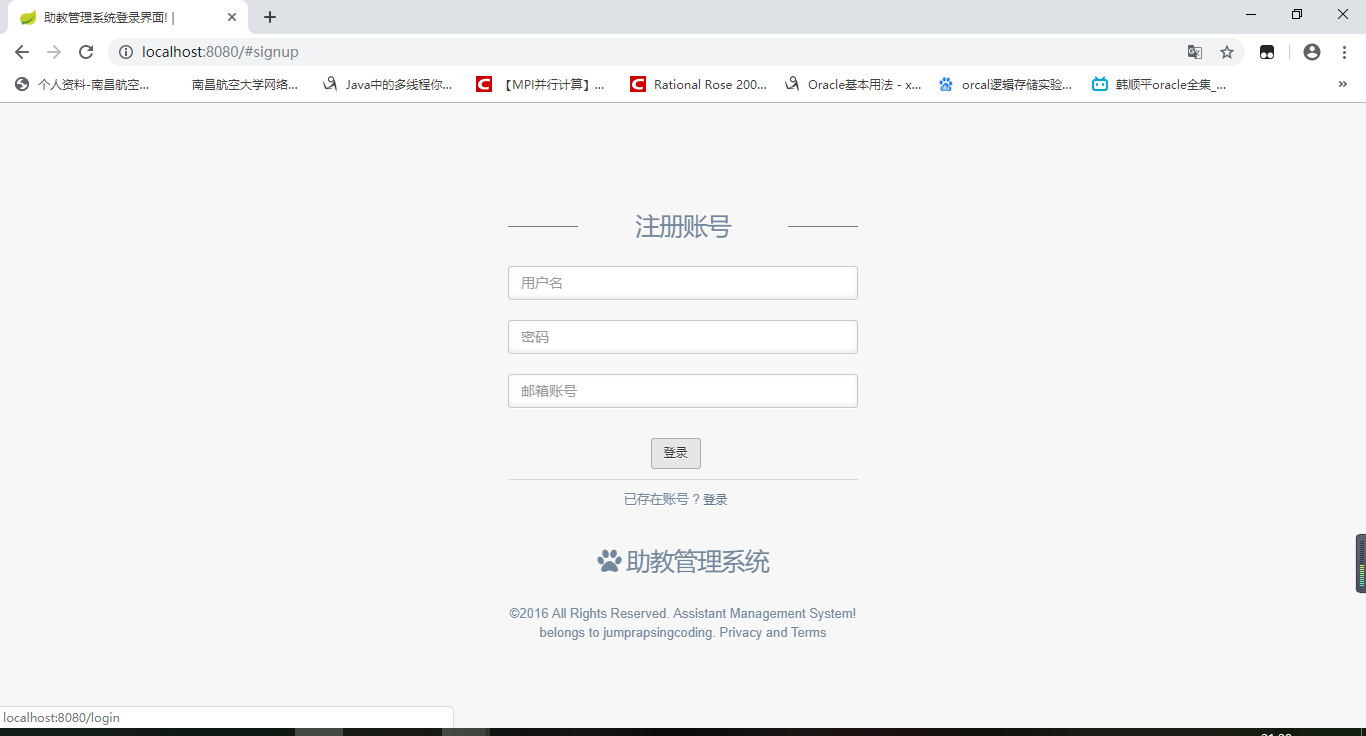


图2.4 注册

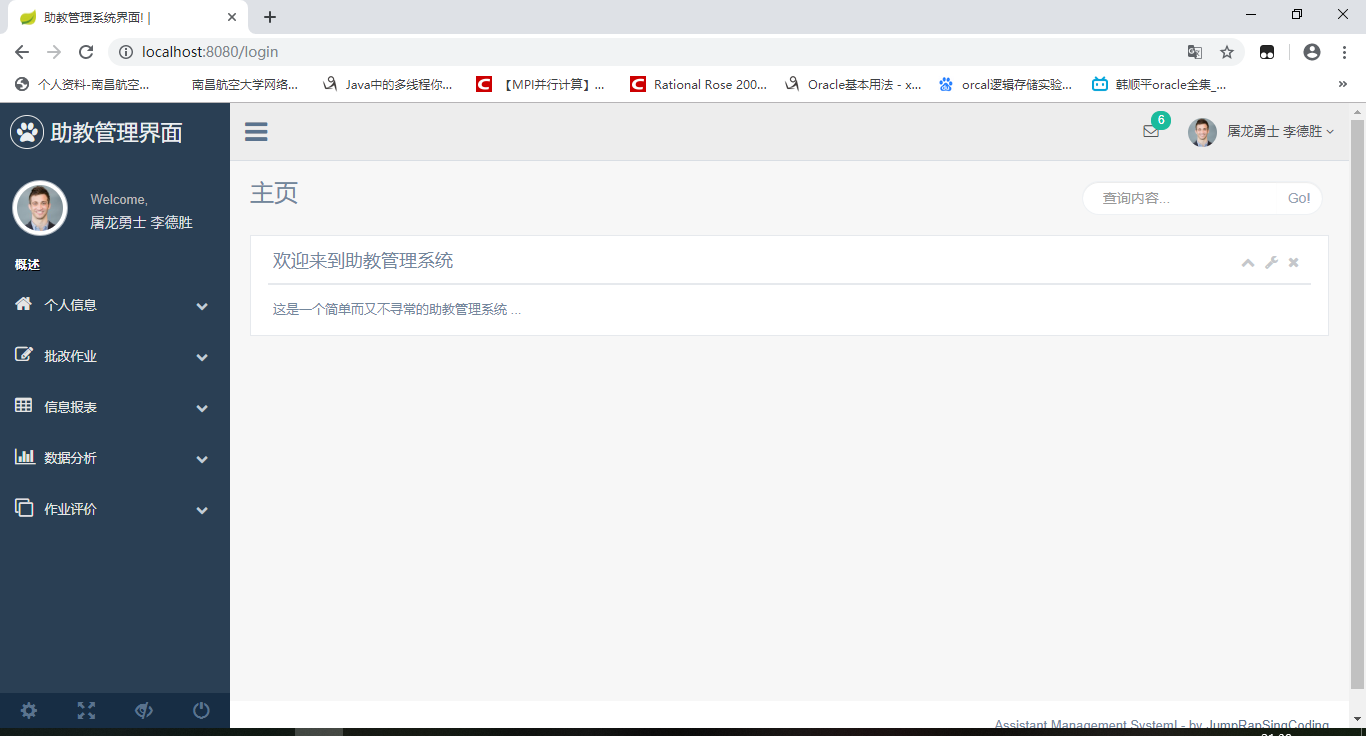


图2.5 界面

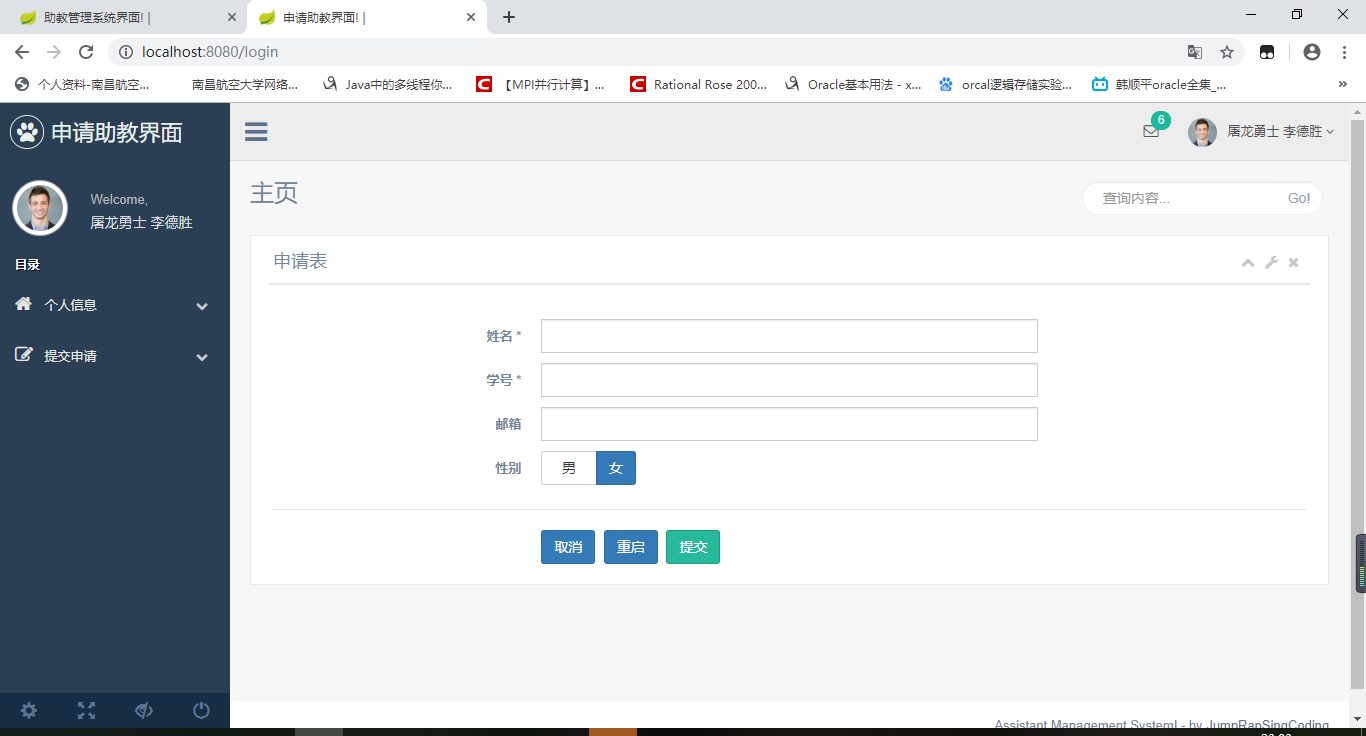


图2.3 界面.

# 3 系统实现

## 3.1 系统开发环境

给出页面设计及其操作流程。

A copy of the Spring runtime library jar file is provided with the sample application along with some of the other required jar files. The developer will need to obtain the following tools externally, all of which are freely available:

* Java SDK 1.5.x
* Ant 1.7.x
* Tomcat 5.x.x, or some other web application container
* JUnit 4.4 - needed to run the tests
* (Optional) MySQL 5.x with MySQL Connector/J 5.x

## 3.2 表达层

给出表达层文件结构，配置说明，并阐述个人模块内表达层的实现过程（含html、css、js、controller等，如有拦截器、配置类、校验类也一并说明）。

The PetClinic web application is configured via the following files:

* war/WEB-INF/web.xml: the web application configuration file.
* war/WEB-INF/petclinic-servlet.xml: configures the petclinic dispatcher servlet and the other controllers and forms that it uses. The beans defined in this file reference the Business/Persistence Layer beans defined in applicationContext-\*.xml.
* war/WEB-INF/classes/messages\*.properties: configures the definition of internationalizable message resources.

Presentation Layer classes

* org.springframework.samples.petclinic.web.ClinicController

is an annotation-driven, POJO MultiActionController that is used to handle simple display-oriented URLs.

* org.springframework.samples.petclinic.web.FindOwnersForm

is an annotation-driven, POJO Form controller that is used to search for Owners by last name.

* org.springframework.samples.petclinic.web.AddOwnerForm

is an annotation-driven, POJO Form controller that is used to add a new Owner to the system.

* org.springframework.samples.petclinic.web.EditOwnerForm

 is an annotation-driven, POJO Form controller that is used to edit an existing Owner. A copy of the existing Owner is used for editing.

* org.springframework.samples.petclinic.web.AddPetForm

is an annotation-driven, POJO Form controller that is used to add a new Pet to an existing Owner.

* org.springframework.samples.petclinic.web.EditPetForm

is an annotation-driven, POJO Form controller that is used to edit an existing Pet. A copy of the existing Pet is used for editing.

* org.springframework.samples.petclinic.web.AddVisitForm

is an annotation-driven, POJO Form controller that is used to add a new Visit to an existing Pet.

## 3.3 业务层

给出业务层文件结构，配置说明，并阐述个人模块内业务层的实现过程（含Service接口及其实现类等，如有Aspect类、日志类、自添加功能类也一并说明）。

The Business Layer consists of a number of basic JavaBean classes representing the application domain objects and associated validation objects that are used by the Presentation Layer. The validation objects used in PetClinic are all implementations of the org.springframework.validation.Validator interface.

* org.springframework.samples.petclinic.Entity is a simple JavaBean superclass used for all persistable objects.
* org.springframework.samples.petclinic.NamedEntity is an extension of Entity that adds a name property.
* org.springframework.samples.petclinic.Specialty is an extension of NamedEntity.
* org.springframework.samples.petclinic.PetType is an extension of NamedEntity.
* org.springframework.samples.petclinic.Person is an extension of Entity that provides a superclass for all objects that implement the notion of a person.
* org.springframework.samples.petclinic.Vet is an extension of Person that implements a veterinarian. It holds a List of specialties that the Vet is capable of.
* org.springframework.samples.petclinic.Owner is an extension of Person that implements a pet owner. It holds a List of pets owned.
* org.springframework.samples.petclinic.Pet is an extension of NamedEntity that implements a pet. It holds a List of visits made concerning the pet.
* org.springframework.samples.petclinic.Visit is a simple JavaBean that implements the notion of a clinic visit for a pet.
* org.springframework.samples.petclinic.util.EntityUtils provides utility methods for handling entities.
* org.springframework.samples.petclinic.validation.OwnerValidator is a Spring Validator that verifies correct data entry for the Add and Edit Owner forms.
* org.springframework.samples.petclinic.validation.PetValidator is a Spring Validator that verifies correct data entry for the Add and Edit Pet forms.
* org.springframework.samples.petclinic.validation.VisitValidator is a Spring Validator that verifies correct data entry for the AddVisit form.

## 3.4 持久化层

给出业务层文件结构，数据库配置说明，并阐述个人模块内业务层的实现过程（含repository、类等，如有网络、文件访问类也一并说明）。

A Spring org.springframework.context.ApplicationContext object provides a map of user-defined JavaBeans that specify either a singleton object or the initial construction of prototype instances. These beans constitute the Business/Persistence Layer of PetClinic. The following beans are defined in all 3 versions (1 per access strategy) of the PetClinic war/WEB-INF/applicationContext-\*.xml file:

* A PropertyPlaceholderConfigurer, which is configured via <context:property-placeholder … /> and is a singleton bean that replaces ${…} placeholders with values from a properties file, in this case, JDBC-related settings for the dataSource bean described below (see src/jdbc.properties).
* dataSource, which is a singleton bean that defines the implementation of the source of database connections used by the application.
* transactionManager, which is a singleton bean that defines the implementation of the transaction management strategy for the application.
* clinic, which is a singleton bean that defines the implementation of the Clinic interface that provides the primary Business Layer API of the application.

# 4 系统测试

给出测试用例，说明测试方法与测试过程，并阐述个人模块内单元测试与功能测试的完成过程（测试过程需截图，同时并给出测试结果）。

AbstractClinicTests and its subclasses benefit from the following services provided by the Spring TestContext Framework

* org.springframework.samples.petclinic.OwnerTests

is a simple JUnit 4 based TestCase that supports Business Rule #1.

* org.springframework.samples.petclinic.AbstractClinicTests

is a JUnit 4 based TestCase requiring a live database connection that is used to confirm correct operation of the database access objects in the various implementations of the Clinic interface. "AbstractClinicTests-context.xml" declares a common javax.sql.DataSource. Subclasses specify additional context locations which declare a org.springframework.transaction.PlatformTransactionManager and a concrete implementation of Clinic.

* AbstractClinicTests extends AbstractTransactionalJUnit4SpringContextTests, one of the valuable testing support classes provided by the Spring TestContext Framework found in the org.springframework.test.context package. The annotation-driven configuration used here represents best practice for integration tests with Spring. Note, however, that AbstractTransactionalJUnit4SpringContextTests serves only as a convenience for extension. For example, if you do not wish for your test classes to be tied to a Spring-specific class hierarchy, you may configure your tests with annotations such as @ContextConfiguration, @TestExecutionListeners, @Transactional, etc.

# 5 总结与体会

不少于1500字，说明系统的特色，分析系统的不足，给出今后的改进方向，最后谈谈自己的体会。