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

## 需求概述

项目背景

对于一门课程，老师觉得对于学生的传统式教学方式显得老套，效率低下和资源浪费。

例如，对于收发作业和实验报告，学习委员难以严格限制时间，总是拖交漏交，使效率低下。

对于写作业和实验报告的作业本和纸张，作业本往往没有只用了几面，实验报告篇幅巨大，造成了纸质的极大浪费。

对纸质作业、实验报告、试卷的批阅，计算课程总成绩需要向电脑录入各个作业的成绩，步骤重复繁琐。

因为学生和老师的见面机会不多，学生少于向老师提问、交流，影响了学习效率。

……

为改善以上问题，决定针对一门课程，开发专门的系统进行教学.

需求说明

作为一个小型的课程教学平台，主要需要以下方面的功能。

* 课程资源管理：发布课件、作业、实验资料
* 课程原理演示：课程典型知识点演示
* 知识考核：试题录入，自动评分
* 课程答疑：可私聊或群聊

业务规则

这里是本系统默认的使用规则

* 本系统作为一个小型系统，只用于一门课程，老师使用后台管理账号。
* 课件、实验资料视为文件资料统一发布
* 典型知识点演示，作为视频播放

## 1.2 用例分析与描述

### 1.2.1 用例分析

老师:

* 学生用户管理
* 发布课件、实验资料
* 发布典型知识点演示
* 发布考试
  + 录入题目和答案
* 作业管理
  + 发布作业
  + 批改作业
* 课程答疑
  + 私聊
  + 群聊

学生:

* 完成作业
* 查看知识点演示
* 查看课件、实验资料
* 进行考试
* 询问问题
  + 私聊
  + 群聊

通过上述分析，可得到系统的用例图如下：



图1.1用例图

### 1.3.2 用例描述

给出每个用例的说明与详细流程

|  |  |
| --- | --- |
| 用例编号 | UC01 |
| 用例名称 | 教师查看学生信息 |
| 简要说明 | 教师登陆后对学生信息进行查看 |
| 执行者 | 教师 |
| 前置条件 | 教师成功登录系统 |
| 后置条件 | 学生信息已存入数据库 |
| 基本事件流 | 1.教师进入课程系统；  2.勾选要查看的学生，点击查看；  3.页面成功跳出学生信息界面；  4.用例结束。 |
| 扩展事件流 | A1：学生信息显示错误  （1）学生信息错误  （2）学生信息界面没有弹出 |

|  |  |
| --- | --- |
| 用例编号 | UC02 |
| 用例名称 | 教师发布课件、资料 |
| 简要说明 | 教师针对课程在系统中发布相应的学习课件和资料 |
| 执行者 | 教师 |
| 前置条件 | 教师成功登录系统 |
| 后置条件 | 资料成功录入数据库 |
| 基本事件流 | 1.教师进入课程系统；  2.点击发布资料按钮，进入上传界面；  3.点击上传按钮，选择文件进行上传；  4.用例结束。 |
| 扩展事件流 | A1：文件上传错误  （1）文件格式错误 |

|  |  |
| --- | --- |
| 用例编号 | UC03 |
| 用例名称 | 教师发布作业 |
| 简要说明 | 教师针对课程在系统中发布相应的作业 |
| 执行者 | 教师 |
| 前置条件 | 教师成功登录系统 |
| 后置条件 | 作业成功录入数据库 |
| 基本事件流 | 1.教师进入课程系统；  2.点击作业管理模块，进入作业内容界面；  3.编辑作业内容，点击发布；  4.用例结束。 |
| 扩展事件流 | A1：作业发布错误  （1）作业内容过长 |

|  |  |
| --- | --- |
| 用例编号 | UC04 |
| 用例名称 | 教师发布演示 |
| 简要说明 | 教师针对课程在系统中发布相应的知识点演示视频 |
| 执行者 | 教师 |
| 前置条件 | 教师成功登录系统 |
| 后置条件 | 视频成功上传至数据库 |
| 基本事件流 | 1.教师进入课程系统；  2.点击课程演示按钮，进行发布；  3.点击上传按钮，选择演示视频进行上传，点击上传；  4.用例结束。 |
| 扩展事件流 | A1：演示发布错误  （1）视频格式错误  （2）演示视频过大 |

|  |  |
| --- | --- |
| 用例编号 | UC05 |
| 用例名称 | 教师录入试题和答案 |
| 简要说明 | 教师针对课程在系统中录入相应的试题及答案 |
| 执行者 | 教师 |
| 前置条件 | 教师成功登录系统 |
| 后置条件 | 试题和答案成功上传至数据库 |
| 基本事件流 | 1.教师进入课程系统；  2.点击知识考核模块，点击录入；  3.编辑试题和答案；  4.用例结束。 |
| 扩展事件流 | A1：录入试题错误  （1）试题格式不符  （2）答案格式不符 |

|  |  |
| --- | --- |
| 用例编号 | UC06 |
| 用例名称 | 教师批作业 |
| 简要说明 | 教师针对课程在系统中对学生上传的作业进行批阅 |
| 执行者 | 教师 |
| 前置条件 | 教师成功登录系统，学生已上传作业 |
| 后置条件 | 批阅结果上传至数据库 |
| 基本事件流 | 1.教师进入课程系统；  2.点击作业管理模块，点击批改按钮；  3.对于作业进行打分和批注，点击完成按钮；  4.用例结束。 |
| 扩展事件流 | A1：批阅错误  （1）给的分数高于100或低于0 |

|  |  |
| --- | --- |
| 用例编号 | UC07 |
| 用例名称 | 教师对课程进行答疑 |
| 简要说明 | 教师针对课程在系统中对学生的提问进行解答 |
| 执行者 | 教师 |
| 前置条件 | 教师成功登录系统，学生提出并上传问题 |
| 后置条件 | 解答结果上传至数据库 |
| 基本事件流 | 1.教师进入课程系统；  2.点击课程答疑模块，选择群聊或私聊；  3.对于对于问题进行解答编辑，点击发送按钮；  4.用例结束。 |
| 扩展事件流 | A1：发送错误  （1）编辑内容为空或过长 |

（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

|  |  |  |
| --- | --- | --- |
| 层次 | 技术 | 资源 |
| 视图层（VIEW） | Thymeleaf、  Bootstrap、  HTML、jsp、css、js |  |
| 控制层（controller） | RESTful、  jQuery、  Ajax、  jUnit、  AOP |  |
| 数据访问层（DTO） | Mybatis， |  |

csThe 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
* ……

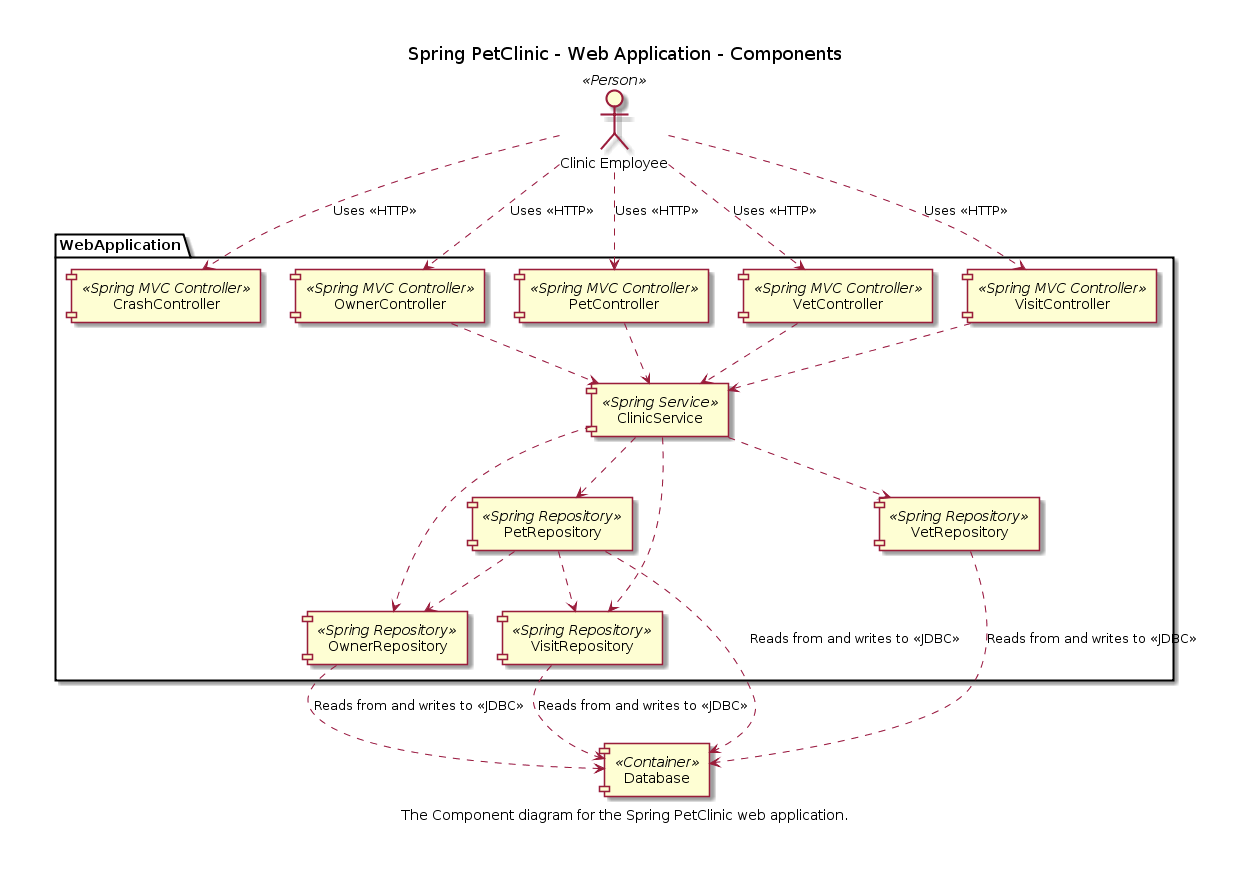


图2.1 宠物医院系统…….

## 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

实体关系建模是系统设计的第一步，寻找出系统中的实体和他们之间的关系，才能进一步进行数据库的物理结构设计。

图2.1.2中的用户包含但不限于图2.1中的老师和学生。

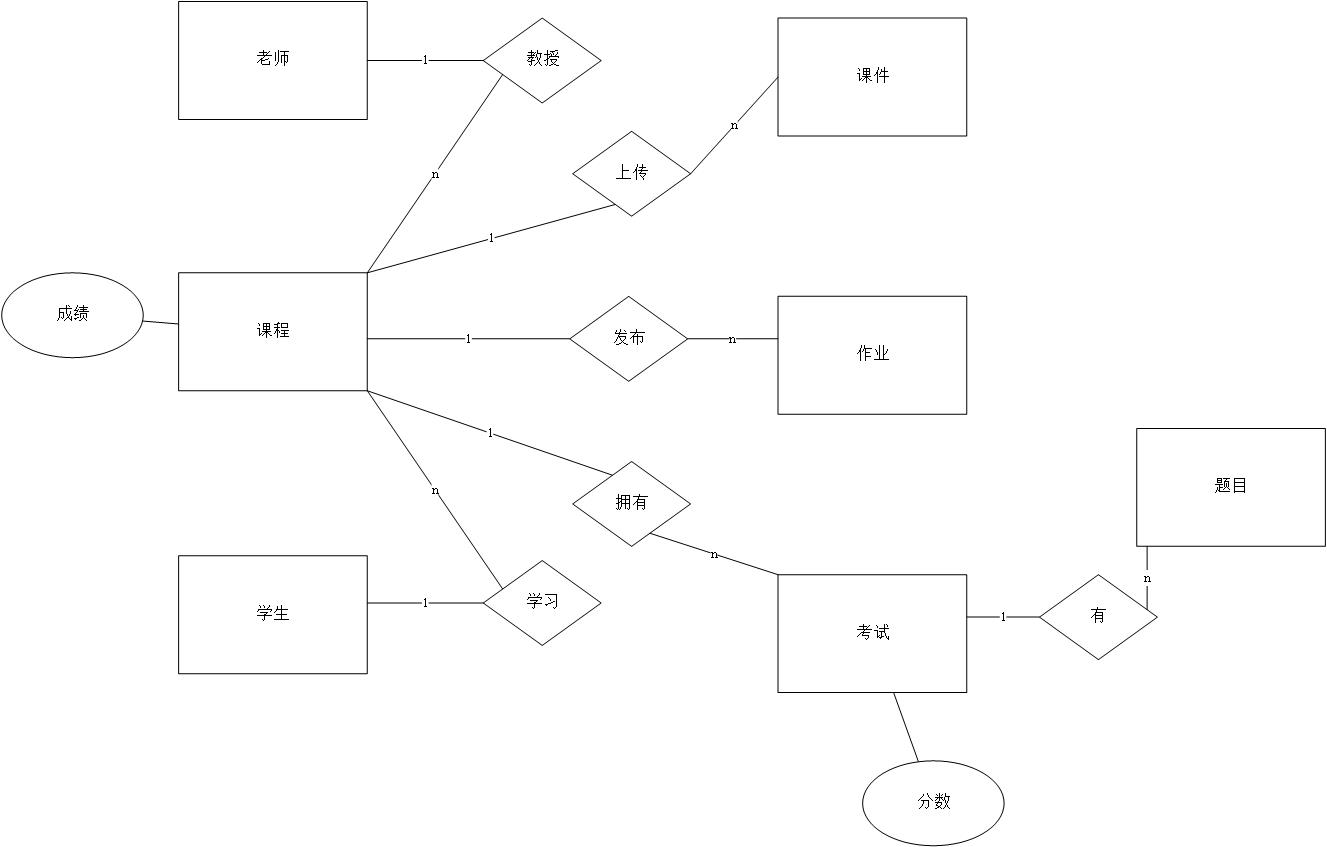


图2.1 XXCEP课程功能系统实体关系模型

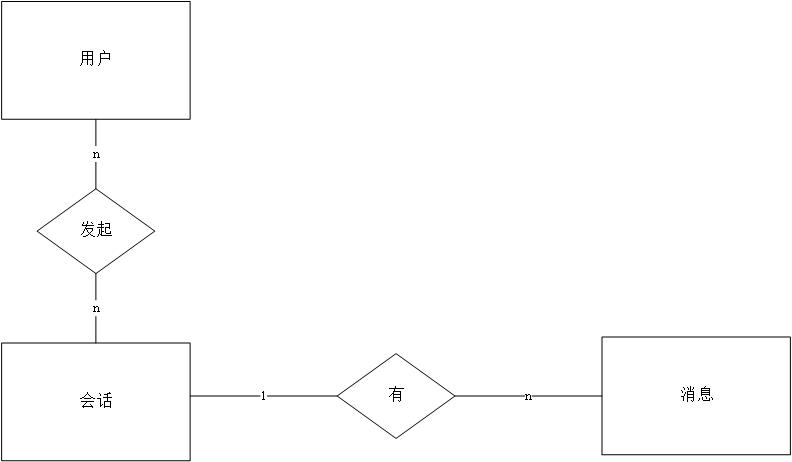


图2.2 XXCEP消息系统实体关系模型

### 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

根据以上给出的实体关系模型可以得到以下数据库物理模型。

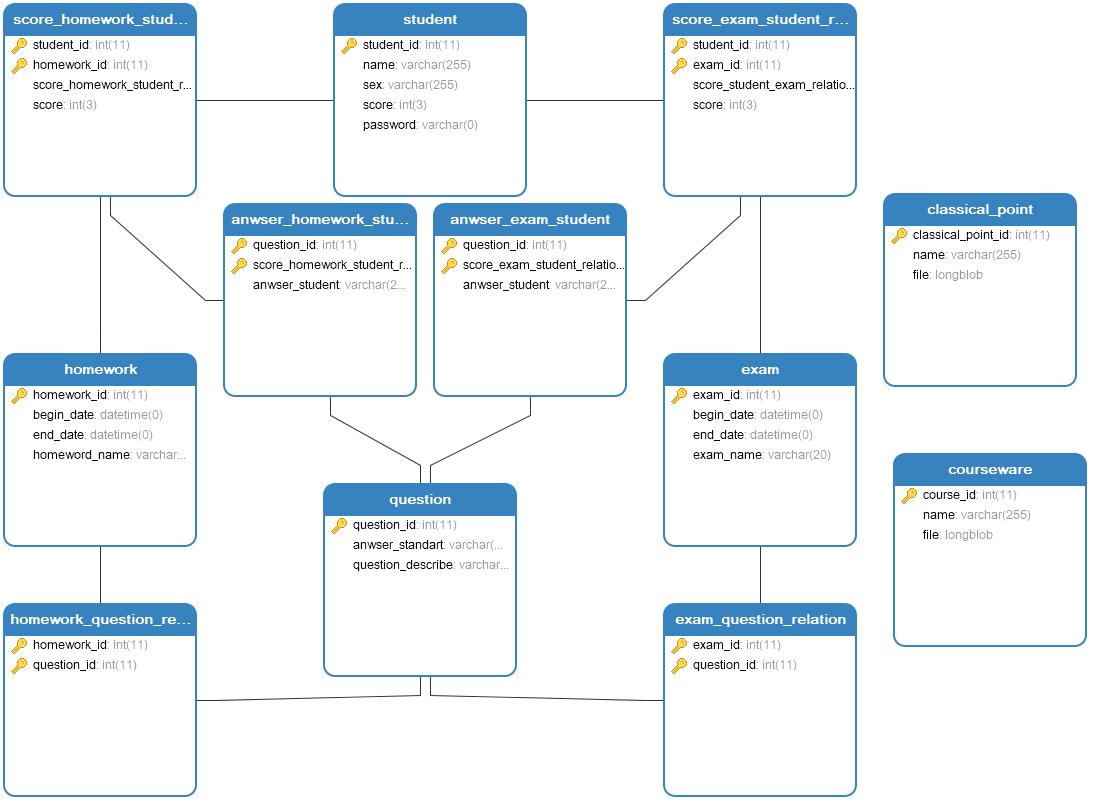


图2.3 XXCEP的PDM图

表格详细情况如下：

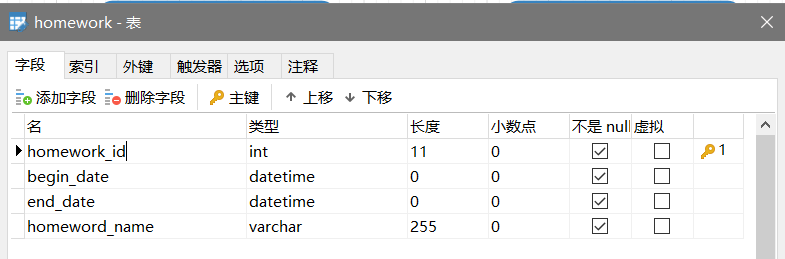


图2.4 homework表

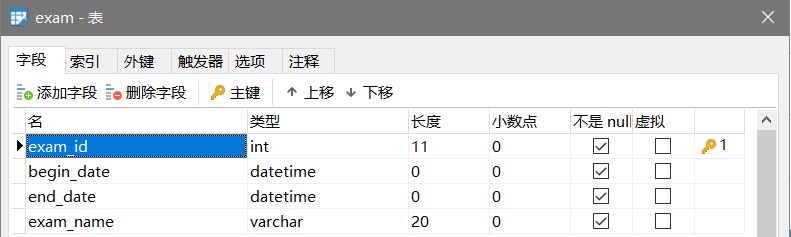


图2.5 exam表

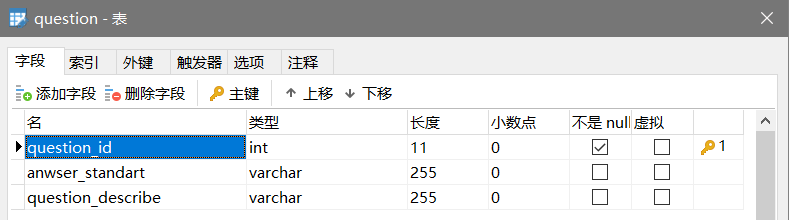


图2.6 question表

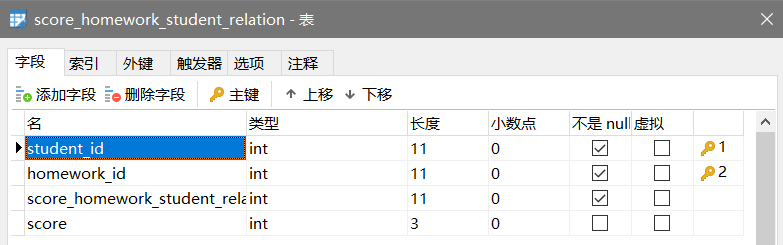


图2.7 表结构

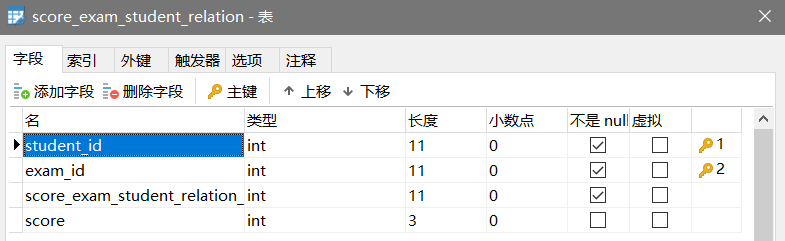


图2.8 表结构

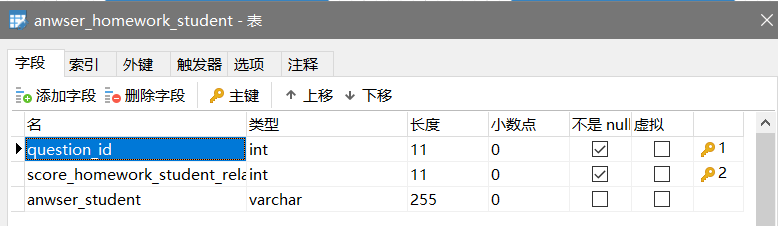


图2.9 表结构



图2.10 表结构



图2.11 表结构

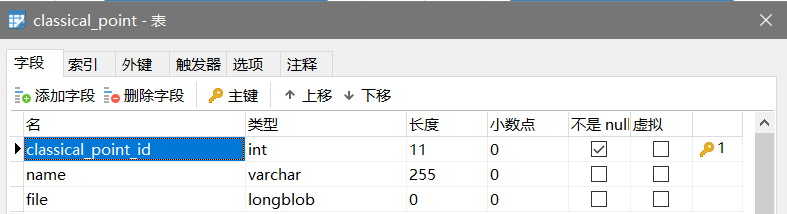


图2.12 表结构

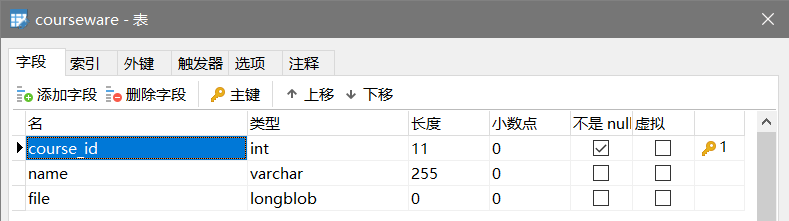


图2.13 表结构



图2.14 表结构

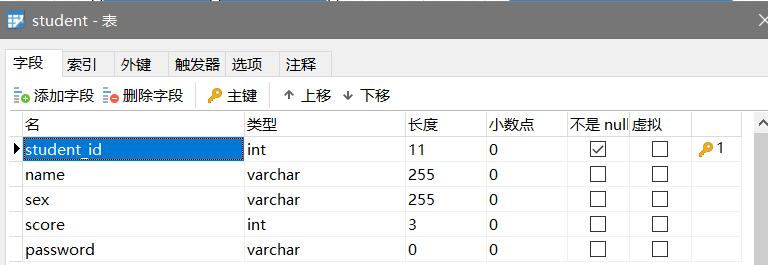


图2.15 student表

### 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:

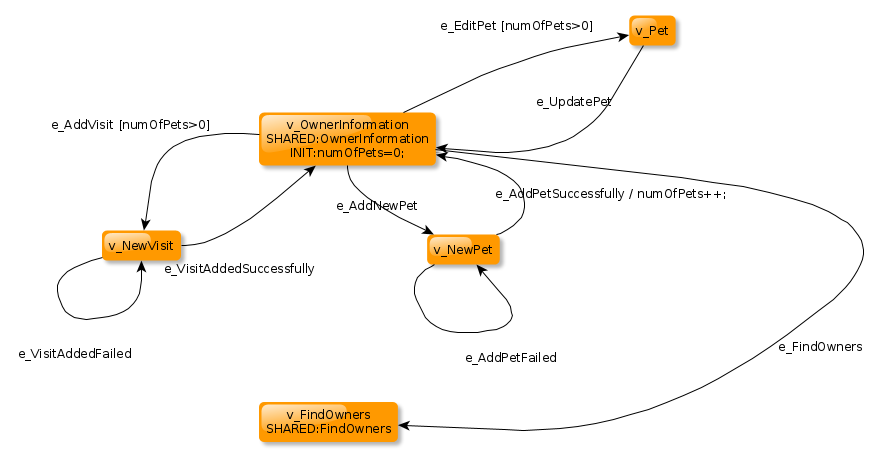


图2.2 宠物医院系统…….

### 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/

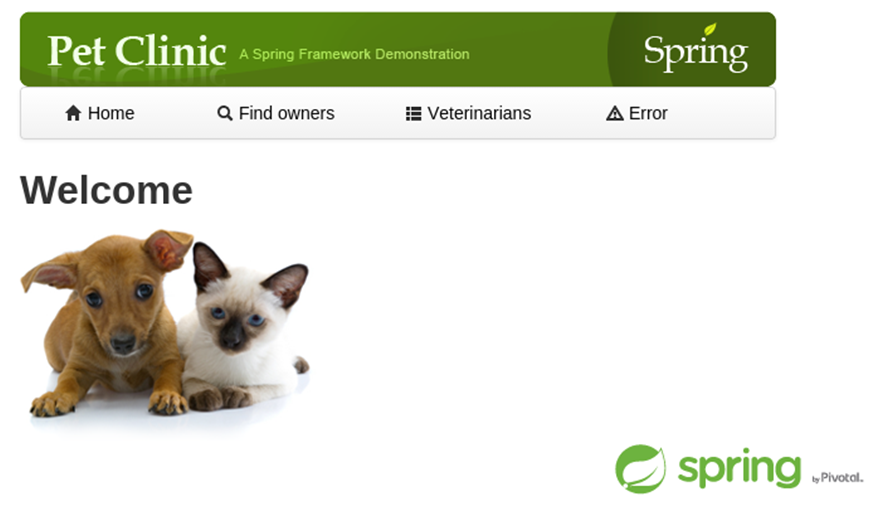


图2.3 宠物医院系统…….

# 3 系统实现

## 3.1 系统开发环境

开发环境主要如下：

* Java SDK 1.8.x
* Gradle 1.7.x
* JUnit 4.4 - needed to run the tests
* git 2.20 版本控制
* github
* MySQL 5.x 数据库
* SpringToolSuite4 作为IDE工具

## 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）。

AOP

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。

# 参考资料

[1] 王珊,萨师煊，《数据库概论》高等教育出版