学士学位毕业论文

“八一农大”—移动

校园app的设计与开发

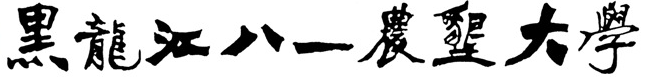
**学生姓名：张龙**

**学 号：20134091122**

**指导教师：朱焕**

**所在学院：理学院**

**专 业：信息与计算科学**



中国·大庆

2017 年 5 月

**黑龙江八一农垦大学**

**本科生毕业论文（设计）任务书**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **论文题目** | 不同品种玉米产量与农艺性状的分析研究 | | | | | | | |
| **学院名称** | 理学院 | | **姓 名** | 张龙 | | **专业班级** | 2013级信息与计算科学 | |
| **指导教师** | 朱焕 | | **课题类型** | 应用研究型 | | | | |
| **毕业论文（设计）基本要求**  1.查阅文献理解不同品种玉米产量与农艺性状分析研究的目的和意义。  2.查阅文献系统了解玉米产量与农艺性状的国内外研究综述，总结研究现状，分析出不足，确定研究方向和将采用的方法。  3.系统分析出影响玉米产量的农艺性状，建立影响不同品种玉米产量的农艺性状体系，利用逐步回归分析对不同品种玉米产量与农艺性状进行分析研究。  4.选取41个不同品种玉米为样本，进行10个农艺性状对产量影响的分析研究，结合实际进行分析，指导实践。 | | | | | | | | |
| **工作量要求**  1．查阅相关文献、资料，文献不少于20篇，其中外文文献不少于4篇，并将至少2篇外文文献的摘要及翻译放到论文附录；  2．撰写论文，正文部分不少于5000字。 | | | | | | | | |
| **毕业论文（设计）的主要阶段计划（分前期、中期、后期）**  前期：2015.9.6- 9.23 查阅资料，准备开题报告；  2015.9.24 开题；  中期：2015.9.27 – 2016.02.29 整理资料，撰写论文；  2016.3 完成初稿，2016.4提交中期报告；  后期：2016.5.1- 5.14 修改完善论文；  2016.5.15正式答辩。 | | | | | | | | |
| **任务下发日期** | | 2015.9 | | | **完成日期** | | | 2016.5 |
| **系主任： 主管教学院长审批（签字）**： | | | | | | | | |

# 摘 要

随着通信技术的发展以及智能手机的普及，移动互联网时代已经到来，Android手机市场占有率达到80%，大学生又是智能手机的主要使用者。因此，基于Android平台，结合本校的实际情况，设计一款服务师生的App必不可少。

本文基于Android 和 Java EE 的移动校园进行设计与开发。首先，简述了校园app的研究背景。然后，分析了整个系统的设计，数据库的设计以及系统的开发环境。其次，分为客户端和服务端两部分对系统的设计和开发进行了详细的论述。接下来，对本系统按照正常流程进行了功能上的测试。最后，进行了总结并对未来工作进行了设想。

本系统设计实现了基于Android平台的移动校园app，包含了三个模块，涉及了服务、社交和学习等方面。本系统界面友好，操作简单，后期可扩展性高，功能设计达到预期，由于时间和本人经验等原因，本系统涉及到的服务较少，将在以后的研究开发中逐步完善。

**关键词**：移动校园；Android；java EE；app。

# Abstract

With the development of communications technologies, as well as the popularity of smartphones, mobile Internet has arrived, Android mobile phone market share of 80%, college students are the main users of smart phones. Therefore, based on the Android platform, combined with the actual situation of the school, teachers and students design a service essential to the App.

Based on Android and mobile campus to design and development of Java EE. First of all, campus app's background is briefly described. Then, the analysis of the overall system design, database design and system development environment. Secondly, it is divided into two parts: client and server of the system design and development are discussed in detail. Next, the system according to the normal procedures of the functional test. Finally, a summary and future work was envisaged.

This system is designed and implemented based on the Android platform mobile app on campus, includes three modules, covered services, such as social networking and learning. Friendly interface, easy operation, high long-term scalability and functionality designed to meet expectations, due to reasons such as time and my experiences, this system involves less service, will gradually improve in future research and development.

Key words：mobile campus，Android，java EE，app。

目录

[摘 要 3](#_Toc481949730)

[Abstract 4](#_Toc481949731)

[第一章 绪论 6](#_Toc481949732)

[1.1 研究背景 6](#_Toc481949733)

[1.2 论文主要工作 6](#_Toc481949734)

[1.3 论文主要结构 7](#_Toc481949735)

[第二章 系统总体设计 7](#_Toc481949736)

[2.1 系统架构设计 7](#_Toc481949737)

[2.2 数据库设计 8](#_Toc481949738)

[2.3 系统开发环境 8](#_Toc481949739)

[第三章 Android客户端设计 8](#_Toc481949740)

[3.1 Android技术点介绍 9](#_Toc481949741)

[3.2 Android应用结构分析 9](#_Toc481949742)

[3.3本系统的界面开发 10](#_Toc481949743)

[3.3.1本系统涉及到的界面元素 10](#_Toc481949744)

[3.3.2 登录界面 11](#_Toc481949745)

[3.3.3 主界面 13](#_Toc481949746)

[3.3.4 主页界面 15](#_Toc481949747)

[3.3.5 发现界面 16](#_Toc481949748)

[3.4本系统功能实现 17](#_Toc481949749)

[第四章 JAVA EE服务端设计 17](#_Toc481949750)

[4.1 java EE 框架介绍 17](#_Toc481949751)

[4.1.1 Spring 18](#_Toc481949752)

[4.1.2 Spring MVC 18](#_Toc481949753)

[4.1.3 Hibernate 18](#_Toc481949754)

[4.2 服务器端实现 19](#_Toc481949755)

[4.2.1 数据持久化层设计和实现 19](#_Toc481949756)

[4.2.2 数据访问层设计和实现 22](#_Toc481949757)

[4.2.3 业务逻辑层设计和实现 23](#_Toc481949758)

[4.2.4 数据传输层设计和实现 24](#_Toc481949759)

[第五章 系统测试验证 25](#_Toc481949760)

[5.1 环境测试 25](#_Toc481949761)

[5.2功能测试 26](#_Toc481949762)

[5.2.1登录测试 26](#_Toc481949763)

[5.2.2主页功能测试 28](#_Toc481949764)

[5.2.2动态发布功能测试 30](#_Toc481949765)

[5.2.2评论功能测试 31](#_Toc481949766)

[5.3测试总结 32](#_Toc481949767)

[第六章 总结与展望 32](#_Toc481949768)

[6.1论文总结 32](#_Toc481949769)

[6.2未来工作展望 32](#_Toc481949770)

# 第一章 绪论

## 1.1 研究背景

（一）互联网发展现状

近年来，信息通信技术的发展和互联网的广泛应用，从根本上改变了传统校园信息传播的方式，也改变了校园交流信息的行为。

（二）移动化的可行性

随着智能手机的发展，移动互联网应用向用户的各类生活深入渗透，手机的使用率快速增长。同时，随着高校无线网络覆盖区域的不断扩大，用户可以做到随时随地通过手机进行信息交流。

（三）本地化的必要性

移动校园信息系统不同于互联网系统，具有很大的本地化特征，在校园地域范围内，基于本校的信息交流，比如：校园公告，校园地图等。本系统的设计，用户根据学号和教务网系统密码登录，限制了使用人群，真正实现本地化。

本文基于提供校园服务的基础上，结合大学生渴望与他人进行信息交流以及物品交流的心理，设计本系统的主要功能。

## 1.2 论文主要工作

本论文所述的移动校园app的设计，所涉及到的程序都是由个人独立开发，分为客户端和服务端两部分。

客户端的开发的主要工作在于进行对界面的绘制，运用Android各种组件的布局排列，各个组件的嵌套使用，实现了对各种界面的设计与开发。借助HttpClient与服务端进行数据交互，将接收到的Json格式的数据进行解析并显示。

服务端开发的主要工作就是对SSH框架的使用，总共分为四个部分，结构清晰，代码易维护。数据持久化层使用Hibernate的JPA技术，进行数据库表的创建和维护；数据访问层，对数据库的访问进行了封装，提供了各类持久化对象的增删改查操作；业务逻辑层，主要负责将数据访问层返回的数据进行处理，转化为Dto对象；数据传输层，将业务逻辑层返回的数据，根据对应的访问接口返回给客户端。

## 1.3 论文主要结构

本文主要由六章组成，每一章的具体内容如下：

第一章简述了本课题的研究背景，国内外发展现状，本课题的主要工作和结构。

第二章介绍了本系统的总体设计，包括客户端和服务端的架构设计；本系统采用的数据库产品Mysql；系统的开发环境，客户端采用Gradle 进行项目构建，开发软件为Android Studio，服务端使用Maven进行项目构建，Web容器为Tomact，开发软件为IntelliJ IDEA。

第三章完成Android客户端的设计。介绍了本文使用到的Android的技术点，包括：Activity组件，Intent，Fragment以及Sharepreference存储类的使用。并结合本文，介绍了Android项目的应用结构。最后设计了本项目的各个界面并实现功能。

第四章基于服务端的设计与实现，介绍了本文应用到的三大框架Spring,Spring MVC，Hibernate。服务端设计分为数据持久化层、数据访问层、业务逻辑层和数据传输层四个部分，符合Java EE设计的规范。

第五章是对已完成上述设计与实现的移动校园客户端进行功能测试的部分，主要根据app总体流程，对客户端进行测试。

第六章对本文进行了总结与展望。

# 第二章 系统总体设计

## 2.1 系统架构设计

客户端采用MVC设计模式：

视图层（View）:由xml布局文件进行界面的展示。

控制层（Controller）：由Java文件进行处理，主要在Activity上。

模型层（Model）：业务逻辑的处理，数据交互处理等。

服务端主要分为四个部分：

1. 数据持久化层：实体关系的设计以及数据库表的创建。
2. 数据访问层：封装了各类持久化对象对数据库的操作
3. 业务逻辑层：数据的业务逻辑处理。
4. 数据传输层：根据请求的接口路径，经Service层处理完成之后，通过对应的接口返回给Android端。

## 2.2 数据库设计

本文基于Android 和 Java EE 的移动校园的设计，数据库采用关系型数据库Mysql。

## 2.3 系统开发环境

本文基于Android 和 Java EE 的移动校园的设计，分为客户端和服务端两部分。

客户端采用Java原生开发，基于Android 6.0版本，所用的集成开发工具为Android studio，应用Gradle 进行项目构建。

服务端采用Java语言，基于Spring + Spring MVC + Hibernate三大框架，所用的集成开发工具为IntelliJ IDEA，应用Maven进行项目构建。

数据库为MySQL Server 5.7。

Jdk版本为Jdk1.8.0\_20。

Web容器为Apache-tomcat-8.0.39。

# 第三章 Android客户端设计

本文基于Android 和 Java EE 的移动校园的设计，客户端主要工作就是用户交互，通过向服务端发送请求，接收数据并展示。Android作为最流行的移动平台之一，其设计思想也是典型的MVC模式， xml文件负责界面布局展现给用户，Java文件负责接收数据和对用户相应的行为进行操作处理。

## 3.1 Android技术点介绍

（一）Activity：作为Android应用与用户交互的窗口，是所有Android应用程序的根本，所有的流程都是运行在Activity之中。在Android程序中，Activity通常是一个单独的屏幕，通过调用方法setContentView(View)来显示指定控件，监听并处理用户的点击事件并作出相应。Activity之间通过Intent进行跳转和传值。

（二）Intent：Android中提供Intent机制来协助应用之间的交互和通讯。比如：调用拨打电话、发短信，以及浏览器等功能。Intent不仅可以应用在应用程序之间，也可以应用于应用程序内部的之间的交互，比如：Activity之间的跳转。

（三）Fragment：Fragment的使用，提供了更加动态和灵活的UI设计，更好的用户体验Fragment必须绑定在一个Activity上，有自己的生命周期，但是受雇主Activity的影响，在Activity中，表示一个行为或者用户交互的一部分。

（四）Sharepreference：轻量级的存储类，以键值对的方式存储文件，以xml文件进行存储。一般用于存储配置文件信息。

## 3.2 Android应用结构分析

本文设计的Android客户端是一个典型的Android应用层程序。主要结构包括：src目录，res目录，R文件和AndroidManifast.xml。（图：3.1）

|  |
| --- |
| 图 1.1 Android应用结构 |

java目录主要存放Java源文件，包括Util工具类和Adapter适配器和Activity等Java文件。

res文件主要存放各种资源文件，包括xml界面布局文件，图标文件以及用户定义的主题文件。

R文件是由aapt（Android Asset Packaging Tool）工具通过资源数据自动生成的。R文件记录着所有资源的id，通过使用id可以很方便地在程序中引用到资源文件。

AndroidManifast.xml位于整个项目的根目录，是Android应用的入口文件，对整个应用就行全局描述，主要包含应用程序包名，应用名，图标，权限，以及程序包含组件（声明Activity、Service等）。

## 3.3本系统的界面开发

### 3.3.1本系统涉及到的界面元素

布局展示主要由xml来定义属性及样式并由java文件调用来进行展示。

1、LinearLayout 线性布局，最基本的界面布局。通过设置布局属性vertical（垂直）和horizontal（水平）两种，

2、RelativeLayout 相对布局，由于Andoroid本身市场的各种分辨率的设备，所以不能采用绝对的像素值来真是界面。相对布局将其内部的各种界面元素按照相对位置来排列。

3、FrameLayout 帧布局，所有添加到这个布局中的视图都以层叠的方式显示。第一个添加的控件被放在最底层，最后一个添加到框架布局中的视图显示在最顶层，上一层的控件会覆盖下一层的控件。这种显示方式有些类似于堆栈。

4、ListView ,recycleView,gridView 都是类似的列表网格视图，以列表或者网格的形式展示内容，通过嵌套可以显示任何复杂格式的信息。

5、Viewpager 是Android SDK v4包中的类，直接继承自ViewGroup类，是一个容器类，可以在其中添加其他的view，使用户可以左右切换当前的view。

6、EditText 输入文本框，也可以设置密码输入类型。

7、Textview 展示普通文本

8、Button 按钮

9、RadioButton 单选按钮，多选一的操作模式。

10、ioGroup RadioButton的容器，可以添加多个RadioButton。

### 3.3.2 登录界面

登录界面为一个大的LineLayout容器，垂直排列，设置了一个背景图片。里面横向排列了用于显示校徽的imageView，两个Edidtext，用于输入用户名和密码，一个按钮用于登录。

总体代码框架为：

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 <LinearLayout  
 <com.example.administrator.superandroid.view.RoundImage/>  
 校徽  
 <EditText/>  
 用户名  
 <EditText/>  
 密码  
 <Button/>  
 </LinearLayout>  
</LinearLayout>

其中圆形校徽是一个继承于Iageview的自定义View，用户名密码和登录按钮都是通过引用shape文件来实现圆角矩形和相应的颜色。

可以看出登录界面的布局比较简洁明了，下面看一下用户名密码的EditText 的一段代码：

<EditText  
 android:id="@+id/username"  
 android:layout\_width="match\_parent"  
 android:layout\_height="50dp"  
 android:layout\_marginLeft="15dp"  
 android:layout\_marginRight="15dp"  
 android:layout\_marginTop="40dp"  
 android:background="@drawable/rect\_gray"  
 android:gravity="center"  
 android:hint="学号" />  
  
<EditText  
 android:id="@+id/password"  
 android:layout\_width="match\_parent"  
 android:layout\_height="50dp"  
 android:layout\_marginLeft="15dp"  
 android:layout\_marginRight="15dp"  
 android:layout\_marginTop="25dp"  
 android:background="@drawable/rect\_gray"  
 android:gravity="center"  
 android:hint="教务网密码"  
 android:inputType="textPassword" />

android:Id用于在java文件中通过findViewById()方法绑定控件。

android:layout\_width ，为设置控件的宽度，match\_parent即为匹配父布局，由于父布局是整个屏幕，所以宽度为铺满整个屏幕。android:layout\_height设置固定高度，用法和android:layout\_width类似。

andoird:backgroud设置了控件的圆角和背景。Hint为默认显示的值。

android:layout\_marginLeft属性表示该控件和父布局左边的距离，设置一定的距离，使界面更加的美观。android:layout\_marginRight和layout\_marginTop等属性与其类似。

android:InputType设置输入类型为textPassword。即输入类型为text类型，输入时隐藏密码。

最后在activity文件中声明，当activity执行onCreate时，通过setContentView(R.layout. activity\_login)与登录界面绑定。

本系统登录采用教务网学号和密码登录，当用户填写了学号和密码之后，点击登录时，根据后台返回的结果进行处理。登录验证成功之后跳转到主界面。本系统在AndroidManifest.xml文件中，设置应用的启动页为登录页面。

<activity android:name=".activity.LoginActivity">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
</activity>

当用户第一次登录时，验证成功之后使用sharedPreferences数据库将用户的信息保存在本地，之后每次启动经过登录页面都会经过判断用户信息是否存在,不存在则跳转到登录页，存在则直接跳转到主页。

保存用户信息代码：

public void saveUserMeaasge(UserDto userDto){  
 sharedPreferences = getSharedPreferences("UserData", Context.MODE\_PRIVATE);  
 SharedPreferences.Editor editor = sharedPreferences.edit();  
 editor.putString("userId", userDto.getId() + "");  
 editor.putString("username", username);  
 editor.putString("password", password);  
 editor.putString("nickName", userDto.getNickName());  
 editor.putString("avatarUrl",userDto.getAvatarUrl());  
 editor.commit();  
}

### 3.3.3 主界面

登录界面相对比较简单，本系统的重点难点在于主界面的开发。主界面的总体设计为RadioGroup + ViewPager + Fragment件的组。主界面整体布局，主要分为两部分，底部导航栏和内容界面。

底部导航栏采用RadioGroup和RadioButton 实现。RadioButton采用上下布局，上面为图片，下面为文字。

代码框架为：

<RadioGroup>  
 <RadioButton />  
 主页  
 <RadioButton/>  
 发现  
 <RadioButton/>  
 我的

</RadioGroup>

RadioButton的代码片段：

<RadioButton  
 android:id="@+id/btn\_1"  
 style="@style/menu\_button"  
 android:checked="true"  
 android:drawableTop="@drawable/tab\_main\_check"  
 android:text="主页" />

style="@style/menu\_button" 引用一个style，每个RadioButton的风格都统一。

android:checked="true" 初始化主界面的时候默认选中主页

android:drawableTop="@drawable/tab\_main\_check" 设置按钮顶部的图片。

主界面应用viewpager控件， viewpager是一个容器，添加view，用户左右滑动来切换当前显示的view，本系统view用fragment来充当，三个界面主页、发现和我的都是一个fragment，添加到viewpager容器中，实现和底部导航RadioGroup联动效果，用户点击RadioButton实现页面切换。

主要步骤如下：

引用viewpager控件，在对应的xml布局文件中声明。

将三个Fragment添加到viewpager容器中

listFragment = new ArrayList<>();  
listFragment.add(new MainFragment());  
listFragment.add(new FindFragment());  
listFragment.add(new MineFragment());

设置viewpager的适配器。本文的view为fragment所以适配器直接继承自FragmentPagerAdapter。

public class ViewPagerAdapter extends FragmentPagerAdapter {  
 private List<Fragment> listFragment;  
 public ViewPagerAdapter(FragmentManager fm, List<Fragment> listFragment) {  
 super(fm);  
 this.listFragment = listFragment;  
 }  
 @Override  
 public Fragment getItem(int position) {  
 return listFragment.get(position);  
 }  
 @Override  
 public int getCount() {  
 return listFragment.size();  
 }  
}

给viewpager设置适配器，通过构造器传入view。即为listFragment。

（4）给viewpager添加页面切换事件：即为Viewpager添加滑动监听事件，实现OnPageChangeListener接口。实现和底部导航栏RadioGroup的联动。

mViewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {  
 @Override  
 public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {  
 }  
 @Override  
 public void onPageSelected(int position) {  
 switch (position){  
 case 0:  
 mRadioGroup.check(R.id.btn\_1);  
 break;  
 case 1:  
 mRadioGroup.check(R.id.btn\_2);  
 break;  
 case 2:  
 mRadioGroup.check(R.id.btn\_3);  
 break;  
 default:  
 break;  
 }  
 }  
 @Override  
 public void onPageScrollStateChanged(int state) {  
 }  
});

### 3.3.4 主页界面

主页界面相对比较复杂。是一个自定义的view，继承自ListView。可以看做为两部分，实现效果为头部是一个可以下拉放的headview， 剩下一部分为一个listView。本系统使用时，头部为一个背景图片，内容部分为listview嵌套Gridview实现九宫格的效果。包含了本系统的主要功能。

九宫格布局的实现主要为下面几个步骤：

1、GridView的布局文件：

<GridView  
 android:id="@+id/grid\_View"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:numColumns="3"  
 />

android:numColumns="3" 设置GridView为三列。

2、GridView适配器，是一个继承自BaseAdapter的自定义的适配器。实现的方法主要有：

（1）int getCount() 获取GridView的item 的数目。

（2）Object getItem(int position) 初始化position位置的界面

（3）long getItemId(int position) 获取当前界面的位置

（4）View getView(int position, View convertView, ViewGroup parent) 当前界面和item的布局文件进行绑定，设置内容等。

（5）GridAdapter(Contex,List<String>,List<Integer>) 构造器传入上下文和item的文字和icon信息。

3、设置适配器

gridView.setAdapter(new GridAdapter(mContext, listName, listIcon);

4、设置监听事件

通过回调接口OnItemClickListener，实现item的点击事件，跳转到对应的页面。

### 3.3.5 发现界面

发现页面的总体布局为tablayout + ViewPager + Fragment ，实现思路和底部导航类似。实现的效果为左右滑动来切换当前的显示视图。

下面以校园界面的效果为例，说一下具体的实现。

校园界面为一个瀑布流的效果。使用的控件为当下流行的RecyclerView，引入自support-v7。RecyclerView的效果实现主要是通过java代码设置。

瀑布流的实现主要为设置recycleView的LayoutManager(布局管理)。瀑布流的方法为：

StaggeredGridLayoutManager(int spanCount, int orientation)

第一个参数的含义为显示几列，第二个参数含义为recycleView的滑动方向。

所以本系统的设计代码为：

StaggeredGridLayoutManager(2,StaggeredGridLayoutManager.VERTICAL));

然后通过设置适配器将数据填充到recycleView中，还可以设置item之间的间距。

//设置适配器  
adapter = new BeautyRecycleAdapter(beautyDtoList, getContext());  
recyclerView.setAdapter(adapter);  
//设置item之间的间隔  
SpacesItemDecoration decoration = new SpacesItemDecoration(5);  
recyclerView.addItemDecoration(decoration);

## 3.4本系统功能实现

Android客户端功能的实现，主要通过和服务端通信的实现来完成。Android端使用okhttp 3 和 retrofit 2 网络框架，发送Http请求和接受Http响应。接收服务端返回的Json数据，Android端进行解析，并展示。

# 第四章 JAVA EE服务端设计

本文所述的移动校园的设计，服务端主要负责了全部的数据和业务逻辑的处理，其设计符合MVC设计思想。系统架构为当下流行的SSH 的轻量级架构，即为Spring ,Spring MVC , Hibernate，所使用的的数据库为Mysql，web容器为Tomact。

## 4.1 java EE 框架介绍

本设计采用的是SSH整合技术，SSH指Spring，Spring MVC ,Hibernate三大框架的整合。

### 4.1.1 Spring

Spring的基于控制反转（IOC）的核心机制，基于面向切面编程（AOP）事务管理，可以整合多种持久层技术（Hibernate），MVC框架（Spring MVC）等。Spring贯穿于项目中的持久层，业务层和表现层等。是java应用的整体解决方案。有以下优点：

1、用Spring的IOC容器，将对象之间的依赖关系交给Spring，降低组件之间的耦合性，让我们更专注于应用逻辑

2、对第三方主流持久层框架很好的集成支持，简化底层数据库的访问流程。如Hibernate ,JPA, Mybatis等

3、.Spring的高度可开放性，并不强制依赖于Spring，开发者可以自由选择Spring部分或全部。

### 4.1.2 Spring MVC

Spring MVC 是当今最主流的web mvc 框架，提供了MVC(模型 - 视图 - 控制器)架构。 MVC模式使应用程序的不同方面分离，同时提供这些元素之间的松散耦合。

模型（Model）：处理应用数据逻辑部分。

视图（View）：处理应用数据显示部分。由浏览器，手机端等展示。

控制器（Controller）：处理用户交互部分。负责处理用户请求并构建适当的模型，并将其传递给视图进行渲染。

### 4.1.3 Hibernate

目前主流的数据库是关系型的数据库，比如：Mysql,Oracle。二维表结构易于理解；使用sql语言操作方便；丰富的完整性减小数据冗余，易于维护；支持表关联，实现复杂的查询。目前主流的编程语言是面向对象型的编程语言，比如：JAVA。面向对象建模和操作；继承和多态；相对面向过程容易理解，简单易用等。所以当我们使用面向对象的编程语言进行开发时，也是基于对象的分析，设计和开发，但是当实现数据持久化，进行数据库访问时，又不得不采用关系型数据库的访问方式，因此，编程语言和数据库之间具有设计模式上的区别和技术上的差异，发展不协调。ORM（Object Relational Mapping）框架应运而生，可以将关系型数据库转为面向对象模型，方便开发和维护。

Hibernate是典型的java EE持久层的解决方案，也是流行的ORM框架之一。Hibernate是对JDBC的进一步封装，从对象（Object）映射到关系(Relation)，再从关系映射到对象。这样在操作数据库的时候，不需要和复杂的SQL打交道，只需要操作po（persistant object），实现对数据表中的数据进行增，删，改，查等操作。

## 4.2 服务器端实现

### 4.2.1 数据持久化层设计和实现

本应用总体的数据库结构设计如图（部分）：

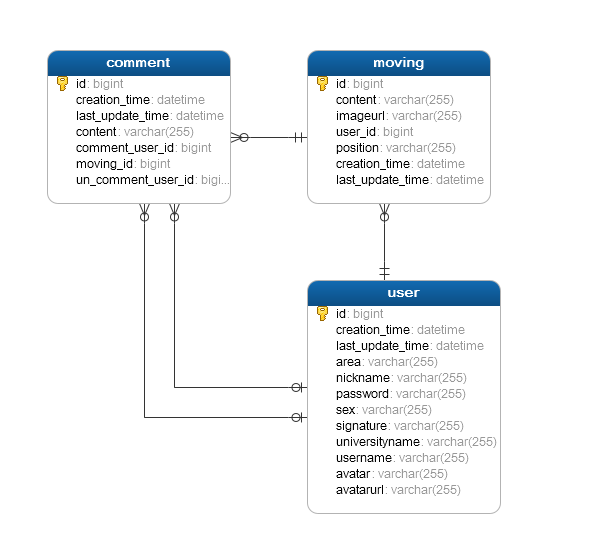


图 4.1 数据库结构设计图

在持久化层，应用到了JPA（java持久化API）技术，数据库中的每张表对应一个Entity（实体）。总的来说JPA包含以下三方面的技术：

1、ORM映射元数据，元数据描述对象和表之间的映射关系，框架将此实体对象持久化到数据库表中。

2、JPA的API，用来操作实体对象，执行CRUP（Create，Retrieve，Update，Delete）操作，框架在后台完成所有的事情，开发者从繁琐的JDBC和SQL代码中解脱出来。

3、查询语言，这是持久化操作中很重要的一个方面，通过面向对象而非面向数据库的查询语句查询数据，避免程序的SQL语句紧密耦合。

本文以动态表为例，来说明持久层实现步骤：

@Entity

@Table(name = "MOVING")

public class Moving extends AggregateRoot {

@Id

@GeneratedValue()

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

private long id;

@Column(name = "CONTENT")

private String content;

@Column(name = "IMAGEURL")

private String imageUrl;

@Column(name = "POSITION")

private String position;

@ManyToOne(optional = false, fetch = FetchType.LAZY)

@JoinColumn(name = "USER\_ID", referencedColumnName = "ID", nullable = false)

private User user;

@OneToMany(fetch = FetchType.LAZY, mappedBy = "moving",cascade = CascadeType.ALL, orphanRemoval = true)

private List<Comment> commentList = Lists.newArrayList();

}

该例子只展现了属性值，省略了相应的getter/setter方法,由IDE自动生成。下面解释相应Annotation（注解)的含义。

@Entity标记该bean为实体bean

@Table(name = "MOVING")，映射到数据库表名为MOVING。

@Id标识主键，@GeneratedValue()标识主键的生成策略，默认为自动增长。

@Column，标识映射到数据库表中的字段名称。

@ManyToOne，描述多对一的关系，一个用户可以发表多条动态，所以多条动态可以对应一个用户。

@OneToMany，描述一对多的关系，一条动态对应了多条评论内容。

对于数据库表，比如，user表会记录用户注册时间，moving表记录动态发表时间，activity表记录活动创建时间。他们都有一个公共的属性createdTime。所以建立一个entity的基类BaseEntity，有两个属性creationTime和lastUpdateTime。所有实体都继承自BaseEntity。

如图所示：

@MappedSuperclass

@EntityListeners({EntityListener.class})

public class BaseEntity {

@Column(name = "CREATION\_TIME")

private Date creationTime;

@Column(name = "LAST\_UPDATE\_TIME")

private Date lastUpdateTime;

public Date getCreationTime() {

return creationTime;

}

public void setCreationTime(Date creationTime) {

this.creationTime = creationTime;

}

public Date getLastUpdateTime() {

return lastUpdateTime;

}

public void setLastUpdateTime(Date lastUpdateTime) {

this.lastUpdateTime = lastUpdateTime;

}

}

### 4.2.2 数据访问层设计和实现

在Hibernate持久层的上层，一般应用DAO（Data Access Object）组件将一些数据库的操作进行进一步的封装，这就是JAVA EE中常用的DAO模式。

DAO 模式实现需要三部分组成：DAO 工厂类（应用工厂模式）、DAO 接口类及其对应的接口实现类。应用 DAO 模式则每个 DAO 组件可以完成对某个数据库表的增删改查操作，且包含数据库访问逻辑。

DAO组件提供各类持久化对象基本增删改查操作，有以下的基本的用法：persist（插入和更新），remove（删除）, findById（根据id单个查找），createQuery（jpql语句查询）。通过实现EntityManager接口，在业务逻辑层调用相应的CRUP。方法实现对数据库的操作。

同样的，对于一些共用的方法我们抽离出来一个接口类，如下：

public interface IBaseDao {

void persist(AggregateRoot entity);

void remove(AggregateRoot entity);

<T extends AggregateRoot> T findById(Class<T> entityClass, long primaryKey);

<T extends AggregateRoot> T findById(Class<T> entityClass,String primaryKey);

<T extends AggregateRoot> T find(Class<T> entityClass, String field, Object value);

<T extends AggregateRoot> List<T> finds(Class<T> entityClass, String field,Object value);

<T extends AggregateRoot> List<T> findLike(Class<T> entityClass,String field, Object value);

<T extends AggregateRoot> List<T> findByIds(Class<T> entityClass,List primaryKeys);

<T extends AggregateRoot> List<T> findAll(Class<T> clazz);

}

再写一个BaseDao的类实现IBaseDao，实现具体的逻辑。以persist和findById方法为例。

@Repository

public class BaseDao implements IBaseDao {

@PersistenceContext

public EntityManager em;

public void persist(AggregateRoot entity) {

em.persist(entity);

}

@Override

public <T extends AggregateRoot> T findById(Class<T> entityClass,

long primaryKey) {

try {

return em.find(entityClass, primaryKey);

} catch (NoResultException e) {

return null;

}

}

}

对于复杂的查询，使用jpql语句，进行查询。比如查询所有动态：

@Repository

public class MovingDao extends BaseDao {

public List<Moving> listMoving(int start,int size){

String jpql="from Moving m order by m.creationTime desc ";

Query query = em.createQuery(jpql);

query.setMaxResults(size);

query.setFirstResult(start);

return query.getResultList();

}

}

数据访问层的实现，是通过结合BaseDao和jpql实现对数据的操作。

### 4.2.3 业务逻辑层设计和实现

业务逻辑层是数据访问层的门面，也就是业务逻辑层需要DAO层组件的支持。本层和DAO层的分离，保证了底层数据和业务逻辑相互透明，有利于复用性和可扩展性。业务逻辑层通过@Service注解，标注业务层组件。下面，以获取所有动态为例：

@Service

public class MovingService {

public ListResponseDto<MovingDto> listMoving(int offset, int size) {

ListResponseDto<MovingDto> listResponseDto = new ListResponseDto<>();

List<MovingDto> listMovingDto = Lists.newArrayList();

List<Moving> listMoving = movingDao.listMoving((offset - 1) \* size, size);

listMoving.stream().forEach(p -> listMovingDto.add(formatMoving(p)));

listResponseDto.setObjs(listMovingDto);

return listResponseDto;

}

}

通过调用MovingDao类中的listMoving(int start,int size)方法，返回一个Moving实体的集合。再根据具体的业务需要，进行逻辑处理。

### 4.2.4 数据传输层设计和实现

在控制层暴露接口，供Android端调用，@Controller注解，标注为控制层组件。@ResponseBody注解，标注返回数据格式为json数据。@RequestMapping("/moving")注解，标注url路径。还是以获取所有动态为例：

@Controller

@ResponseBody

@RequestMapping("/moving")

public class MovingWebService {

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

public ListResponseDto<MovingDto> listMoving(

@RequestParam("size") int size,

@RequestParam("offset") int offset) {

return movingService.listMoving(offset, size);

}

}

通过url地址（ip 地址+ 端口号 + /moving/allMoving）,通过post请求，传入两个参数，offset和size，调用MovingService的方法，返回MovingDto的集合。在业务逻辑层将实体的集合转为DTO(Data Transfer Object)的集合，返回给Android端进行数据展示。

在数据传输层，通过DTO来进行服务器端和客户端的数据交互，通过比较moving的实体和moving的dto，不难发现，MovingDto展现的数据都是view层直接显示的数据格式，不需要在进行额外的处理。使用DTO的好处就是，按需组织DTO对象，view层需要的字段我才组织，不需要的我不组织，可以避免传输整个表的字段，一定程度上提高了安全性。下面是MovingDto的设计：

public class MovingDto {

private String content;

private String position;

private String imageUrl;

private String publishTime;

private String userId;

private String userName;

private String avatarUrl;//头像

private List<CommentDto> listComment = Lists.newArrayList();

}

# 第五章 系统测试验证

## 5.1 环境测试

本系统服务端采用Tomact作为服务器，端口为8090，使用url地址，通过servlet访问WebService层，返回Json数据。通过浏览器模拟访问，返回正确的Json 数据，表示服务端的Spring， Hibernate框架以及数据库都能正常运行。

Android端使用Android Studio开发，本地模拟器调试成功后，文件打包生成apk文件，使用真机测试。本系统客户端名为“八一农大”，图标为黑龙江八一农垦大学校徽。在手机上安装此应用后，在系统中看到相应软件。如下图：

|  |
| --- |
| C:\Users\lixi2000\Desktop\Screenshot_2017-04-28-09-27-54-186_com.miui.home.png  图 5.1 app启动图标 |

点击启动图标，系统跳转到登录界面，表明安卓端运行环境正常。

## 5.2功能测试

### 5.2.1登录测试

用户输入正确的学号和教务网密码之后，进入到主界面。目前数据库采用的都是测试数据。

|  |  |
| --- | --- |
| C:\Users\lixi2000\Desktop\Screenshot_2017-04-28-09-38-25-985_com.example.ad.png | C:\Users\lixi2000\Desktop\Screenshot_2017-04-28-09-39-13-913_com.example.ad.png |
| 图 5.2 登陆界面 | |

当用户将应用退出到后台，再次进入应用时，直接跳转到主界面，说明自动登录功能实现。

|  |
| --- |
| C:\Users\lixi2000\Desktop\QQ截图20170425154620.png  图 5.3 主页界面 |

### 5.2.2主页功能测试

进入主界面之后，首先是主页界面，主页界面主要是九宫格，已经实现的功能有学校简介，公告展示，校历查询，图书查询，学校公众号文章的展示，校园街景展示，快递查询。下面就以公告展示、图书查询和快递查询测试功能。

公告展示：公告为管理员发布的学校公告，根据时间的有效期而展示。点击公告进入公告展示界面（图12），点击具体的公告进入公告详情，为一个网页展示具体内容（图23）。

|  |  |
| --- | --- |
| C:\Users\lixi2000\Desktop\Screenshot_2017-04-28-10-23-38-306_com.example.ad.png | D:\hcy\install\QQ\qqfile\1679937491\FileRecv\MobileFile\Screenshot_2017-04-28-10-23-45-042_com.example.ad.png |
| 图 5.4 公告界面 | |

图书查询：图书查询功能根据，一些关键词，比如：书名、作者、ISBN（国际标准书号）以及ISSN（国际刊号）搜索之后，显示图书馆是否存在此图书。测试结果如下：

|  |  |
| --- | --- |
| D:\hcy\install\QQ\qqfile\1679937491\FileRecv\MobileFile\Screenshot_2017-04-28-10-09-23-011_com.example.ad.png | D:\hcy\install\QQ\qqfile\1679937491\FileRecv\MobileFile\Screenshot_2017-04-28-10-09-30-495_com.example.ad.png |
| 图 5.5 图书查询界面 | |

快递查询：用户输入提单号，可以选择或者手动输入快递公司，查询物流信息。具体测试功能如下：

|  |  |
| --- | --- |
| C:\Users\lixi2000\Desktop\Screenshot_2017-04-28-10-34-54-390_com.example.ad.png | C:\Users\lixi2000\Desktop\Screenshot_2017-04-28-10-35-16-779_com.example.ad.png |

|  |  |
| --- | --- |
| C:\Users\lixi2000\Desktop\Screenshot_2017-04-28-10-35-10-011_com.example.ad.png | 1C:\Users\lixi2000\Desktop\Screenshot_2017-04-28-10-35-30-412_com.example.ad.png |
| 图 5.6 快递界面 | |

### 5.2.2动态发布功能测试

发现模块有校园、动态和二手三个功能组成。下面主要以动态功能测试系统。用户点击发布按钮进入到发布界面（图23），填写内容和选择图片之后，点击发送，即成功发表一条动态，返回到动态展示界面，即可看见刚刚发表的动态。

|  |  |
| --- | --- |
| D:\hcy\install\QQ\qqfile\1679937491\FileRecv\MobileFile\Screenshot_2017-04-28-10-55-25-673_com.example.ad.png | C:\Users\lixi2000\Desktop\Screenshot_2017-04-28-10-56-27-935_com.example.ad.png |
| 图 5.7 动态界面 | |

### 5.2.2评论功能测试

评论功能使用的是一个盖楼的效果，根据评论时间倒叙排序，最新评论展现在最高楼。

|  |  |
| --- | --- |
| D:\hcy\install\QQ\qqfile\1679937491\FileRecv\MobileFile\Screenshot_2017-04-28-11-15-52-653_com.example.ad.png | D:\hcy\install\QQ\qqfile\1679937491\FileRecv\MobileFile\Screenshot_2017-04-28-11-15-59-089_com.example.ad.png |

## 5.3测试总结

本章，应用一个测试账号，首先验证了测试环境，然后介绍和分析了Android端软件的测试过程，执行了本系统的主要流程，验证了本文设计的基于java EE 和Android 的“掌上农大”移动校园app的设计和开发，正常运行并顺利实现预期效果。

# 第六章 总结与展望

## 6.1论文总结

随着移动互联网技术的发展，基于Android平台的高校APP的应用会越来越多。许多高校都有自己的官方App，相比较来说，APP比微信公众号有更强的交互性，比百度贴吧、知乎等有更好的易维护性以及权威性。通过对Android 和 Java 技术的深入研究和分析，简单设计了基于本校app的设计。本文主要完成了一下工作：

学习了当下流行的ORM框架——Hibernate，设计了系统的数据表结构，搭建了web开发以及运行环境。

研究了Android开发技术，一些框架的使用，包括：网络请求框架，图片加载框架等。以及Android控件的使用，UI的设计。

查阅了大量的关于校园app设计的论文，参考了一些app的设计，结合本校现状，设计的软件能满足基本的需求。

## 6.2未来工作展望

Android作为校园信息交流的方式，有很大的发展前景。特别是大学生作为智能手机的主要使用人群，针对校园app的设计大有可为。由于时间有限以及本人经验不足，还有需要完善的地方。

用户需要更好的体验，更方便的交流，聊天功能必不可少。在动态发表和二手市场更方便的进行交流。因此，在以后工作中，实现聊天的功能，完善本系统。

校园app的设计，使用人群针对的就是在校大学生。针对现在，找兼职、找工作苦于找不到正确招聘信息的现状，可以设计一个找工作的模块，学校对于兼职、招聘信息进行筛选，提供一个好的平台，很大程度解决就业问题。

# 参考文献

[1] 姜远航，孙继禹，韩宇霏，王尊 基于Java EE 和Android 的校园信息系统的设计

[2] 张言林，李博，王文博，张财学，张刘通 基于数字化校园的手机app客户端设计初探

[3]