密级：



# NANCHANG UNIVERSITY

**学 士 学 位 论 文**

## THESIS OF BACHELOR

（2008 — 2012 年）



题 目 Android手机端办公自动化OA系统的设计与实现

学 院： 软件学院 系 软件工程

专 业： 软件设计

班 级： 083班

学 号： 8000108383

学生姓名： 张哲

指导教师： 吴金辉

起讫日期： 2012/02/13 – 2012/04/29

目录

NANCHANG UNIVERSITY 1

THESIS OF BACHELOR 1

摘 要 4

Abstract 6

第一章 绪论 8

1.1本课题的来源 8

1.2本课题的目的、内容、要求及作者主要贡献 8

1.2.1目的： 8

1.2.2内容： 9

1.2.3要求： 9

1.2.4作者的主要贡献： 9

第二章 Android 应用程序开发环境与工具的选择 10

2.1本课题开发环境的选择 10

2.2本课题开发工具的选择 11

第三章 需求分析 12

3.1 OA系统企业级需求调研 12

3.2程序需要实现的功能 13

3.2.1 用户安全登陆、退出应用程序 13

3.2.2 流程管理 13

3.2.3 日程管理 13

3.2.4 邮件系统 13

3.2.5 通讯录 13

3.2.6 即时通讯系统 13

3.2.7 信息发布 13

3.2.8 统计报表 13

3.3 程序实现方案选择 13

第四章 概要设计 14

4.1 系统的简单设计 14

4.1.1 登录与身份认证 14

4.1.2 公告发布 14

4.1.3 功能菜单 14

4.2系统的效果说明 15

第五章 系统详细设计 18

5.1 系统功能的详细设计 18

5.1.1 登录与身份认证 18

5.1.2 功能菜单 18

5.1设计应用程序框架 20

5.1.1编写应用程序框架 20

5.1.2为应用程序编写开始界面 25

5.1.3为界面按钮增加事件处理 28

5.2 应用程序代码的包的设计 29

5.2.1 Package org.hexor.oa 29

5.2.2 Package org.hexor.oa.bean 29

5.2.3 Package org.hexor.oa.constant 29

5.2.4 Package org.hexor.oa.network 30

5.2.5 Package org.hexor.oa.service 30

5.2.6 Package org.hexor.oa.util 30

5.2.7 Package org.hexor.oa.adapter 30

5.3应用程序内容的设计 30

5.3.1用户身份验证 30

5.3.2数据库存取操作 37

5.3.3流程对象的设计 43

5.3.4应用程序与服务器之间的通讯 43

5.3.5服务器端接收请求的设计 44

5.3.6应用程序线程的设计 44

5.3.7应用程序后台Service的设计 46

5.4添加应用程序的其它功能 59

5.4.1设计关于对话框 59

5.5系统主要界面效果 61

第六章 系统测试 62

6.1测试举例 62

6.1.1测试程序登录页面 62

6.1.2测试程序主页面 62

6.1.3测试程序退出页面 62

6.2测试项目 63

6.2.1单元测试 63

6.2.2组合测试 63

6.2.3确认测试 63

6.2.4系统测试 64

6.2.5用户验收测试 64

6.3测试方法 64

第七章 总 结 65

参考文献（References） 67

致 谢 68

附件：源代码 69

**OA办公自动化系统在Android手机上的实现**

# 摘 要

OA办公自动化系统自从上个世纪企业级应用的诞生以来，在公司内部作为业务流程的载体，员工沟通的平台，一直以来为保证企业的高效率运作起着十分重要的作用。

OA办公自动化系统一般都拥有固定的业务逻辑，需求以及功能，是学习企业级软件开发或者客户端/服务器软件开发的一个经典实例。随着现代高级编程语言的发展和智能移动终端的普及，也为了提高用户与OA系统的交互能力，手机端的OA系统已经成为企业OA系统不可或缺的一部分。

本毕业设计论文介绍使用Java语言设计一个Android手机端OA系统的过程，整个系统是一个Android手机端APP应用，使用了标准的Android SDK开发接口以及Google官方标准的UI设计。手机应用设计过程中涉及了应用界面的编写，内部业务逻辑的设计，变量的定义，算法设计实现，以及与服务器之间的数据交换，从而完成一整套从前必须在PC端才能完成的业务流程。另外，论文也介绍了如何使用Python编写一个为本Android应用设计的服务器端程序。

通过本次毕业设计可以学到开发一个标准的Android手机应用，如何使用标准的模块设计一款手机应用，如何使用Socket完成Android应用程序与Python服务器之间的数据通讯。

本论文中第一章介绍本课题的来源、目的、内容、要求及作者有关信息；

第二章介绍了开发工具和开发环境的背景；

第三章分析该毕业设计的程序业务流程、使用说明、规则、设计思路及其开发的注意事项；

第四章论述了该毕业设计的思想和总体结构及流程；

第五章并详细阐述系统的开发步骤及有关重点实现细节；

第六章进行系统测试并简要介绍测试方法；

第七章提及了开发过程中心得体会、经验和感想

最后是参考资料

关键词 **：**移动OA；Android； Python网络编程；客户端/服务器；Socket数据通讯

The Implementation of the OA System

on the Android SmartPhone

# Abstract

OA office automation system as the carrier of business processes and employee communication platform within companys, has been played a very important role to ensure the efficient operation of enterprises, since the naissance of enterprise applications in the last century.

OA office automation systems generally have a regular business logic, demand and function, is a classic example of praciticing enterprise level software or client / server software development. With the development of modern high-level programming language development and the popularization of mobile intelligent terminal units, in order to improve the user's ability to interact with the OA system, mobile phone terminal OA system has become an indispensable part of OA system.

This thesis introduced using of Java language to design a Android mobile phone terminal OA system, the whole system is a Android mobile phone APP, using the standard Android SDK interface and Google official standard UI design.Mobile phone application involved the design process of the application interface to prepare, internal business logic design, variable definitions, algorithm design and the implementation of between the server and the clients’ data communication, finally all these completing a set of operation which used must be in the PC terminal to complete the business process.In addition, the paper also introduces how to use Python to write a server end procedure which designed for this Android application.

Through this thesis of bachelor, we can learn how to develop a standard Android mobile phone application, how to use the standard module design a mobile phone application, how to use the Socket to connect a Android application and a server with Python.

Chapter I: Devoted to the subject of the origin, purpose, content, and requests information on the author;

Chapter II: Describes the development tools and development environment background;

Chapter III: Analysis of the graduate design rules of the game, rules, Design and development of the attention;

Chapter IV: Chapter of the graduate design thinking and the overall structure and process;

Chapter V: Detailed steps for the development and realization of the key details;

Chapter VI: System testing and briefed test methods;

Chapter VII: Chapter of the development process center in the experience, experiences and thoughts chap reference

In the end : Some reference sources.

**Keywords :** OA , Android, Python Network Programming, C/S, Socket

# 第一章 绪论

## 1.1本课题的来源

顾名思义，所谓移动OA就是可以在移动终端上进行的办公自动化操作的应用程序。随着现代高级编程语言的发展和智能移动终端的普及，也为了提高用户与OA系统的交互能力，手机端的OA系统已经成为企业OA系统不可或缺的一部分。手机的性能日新月异的增长，目前部分高端手机的CPU运算能力甚至已经超过了十几年前的PC端，而不仅仅是硬件方面的提升，今年来随着iOS与Android智能手机操作系统平台的兴起，激发了移动终端开发者极大的热情，移动终端应用如雨后春笋般被开发，由此许多从前在PC端才能完成的任务现今只要拥有一款智能手机，这些任务都可以随时随地完成了。智能手机用用是一种新型的产品，企业级计算机软件的开发和一般的产品的开发有着明显的不同，程序设计人员不仅需要掌握程序设计技巧和软件工程方法，还需要有坚实的专业知识，才能有效地生成一个比较完整的应用程序。

OA系统历经几十年的发展之后，应用软件已经相当成熟，功能上也有了很大的扩展，从原来的单一的办公消息服务，转变为现在公司内部各种工作流的运作平台。OA的系统的设计目标是不变的，那就是更快捷和方便的实现公司内部各个部门和分支之间的交流协作，节省工作时间，降低工作成本，有效整合公司各处的有效资源，为提供高效的协作工作平台。

移动终端的OA系统对于传统的OA系统而言，更一步的增加了整体的运作灵活性，使员工的工作时间、区域不仅仅局限与公司内网的PC端，只要有智能手机的存在，就有可能实现随时办公，随地办公，这无疑大大增强了OA系统的实用能力，从而也进一步的提高了员工的工作效率。

## 1.2本课题的目的、内容、要求及作者主要贡献

### 1.2.1目的：

获取一次完整的Android项目开发经验，通过对Android SDK文档的阅读和自我学习，加深自我对面向对象概念的理解。这款应用不论是Adnroid客户端还是Python服务器端都有许多细节问题需要解决，比如身份认证时候的非对称密码系统，公钥私钥系统还有防止SQL注入的要求等。除了这些之外作为一款手机应用还必须考虑到系统的流畅度和与用户交互感受等细节问题。

### 1.2.2内容：

一个相对完整的手机手机OA系统。主要包括：身份认证，权限系统，流程管理，邮件系统，日程管理，通讯录，即时通讯，信息发布，统计报表，内部会议申请预约等功能。 要求使用图形用户界面，界面色彩、布局要求美观，操作流畅。

### 1.2.3要求：

主要涉及程序业务逻辑流程的控制，整个手机程序用户界面采用Activity对用户的操作进行响应，主要使用多线程和Service组件来进行相对耗时以及需要在后台运行的耗时操作。在代码的编写上要考虑代码重用的思想，要深入理解抽象类的作用、接口的作用等。

### 1.2.4作者的主要贡献：

由吴金辉导师认真、负责的指导，本人才能完成了此课题的研究与开发，包括调研、分析、设计、编码、测试、文档编写等内容。

# 第二章 Android 应用程序开发环境与工具的选择

## 2.1本课题开发环境的选择

在为开发手机应用时选择编程语言和运行平台时有几个必须考虑的问题：执行平台、开发难易度、语言本身的功能、可使用的外部支持功能与可使用的开发环境、执行速度等。

⒈ 就执行平台而言，考虑到iOS平台必须运行在Apple的设备上，而Apple设备普遍价格较高，对于大多数的中小型企业如果全面实现iOS设备的普及还是有一定难度的。相对而言Android平台大多数针对的是中低端客户，对公司来说实现的成本也较低。而且Android平台上的工具发展也很循序，Google对于自家的平台更新速度也较快，所以目前在Android平台上实现一个相对稳定和安全的企业级应用也是可能的。

⒉ 再者是开发难易程度的问题，C/C++使用较为复杂，设计程序时若不小心将可能导致应用的执行错误，甚至程序终止等情况发生，在测试除错时所花的成本有时并不比开发程序少。Android原生采用Java进行设计，并且Android平台对JVM做了深度的定制和优化，暴露给开发者的接口也都有十分详细的文档来解释。

⒊ 在设计时另一个重要的考虑是程序语言本身的功能，以及其可使用的外部支持。C/C++本身提供有标准函数库，且可调用操作系统本身所提供的一些组件功能。而Java程序语言以C++的语法关键字为基础，由Sun公司所提出，其计划一度面临停止的可能性，然而后来却因为因特网的兴起，这说明了Java程序在因特网平台上拥有极大的优势。

⒋ 执行速度永远是应用运行时的一个重要因素，Android唯一不足的可能就是执行速度方面，不过本应用没有太多十分耗费CPU使用时间的事务，另外相对耗时的事务可以考虑采用Service和多线程来实现。最后，如果速度还是不够理想，Android还可以灵活地使用NDK（Android-Native-Develop-Kit）在Java中嵌入C/C++代码来直接对内存进行操作，从而提高系统的执行速度，不过这样也增加了系统的实现难度，增大了系统出现错误的概率。

在历经数个不同版本的改进与功能加强之后，Android程序本身无论是在绘图、网络、多媒体等方面都提供有相当多的API链接库，甚至包括了3D领域。所以使用Android程序来设计程序可以获得相当多的资源。Android平台的相对平民度，Android开发资源的易获得度，以及实现一个功能可以采用方案的多样性，都是我选择Android平台的重要原因。

## 2.2本课题开发工具的选择

开发工具的给我们的选择似乎不多，Google官方提倡使用Eclipse配合Android开发套件进行开发。Eclipse几乎是目前世界上最强大的IDE，拥有强大的可定制能力，丰富的插件库，跨平台，支持最新的Java标准，配合Android开发套件能够很好的完成开发以及调试的需求。

Eclipse的可视化集成开发环境包括文本编辑器、工程创建工具、对象浏览器和调试器等。用户可以在集成开发环境中创建工程和打开工程，建立、打开和编辑文件，并可以编译、运行和调试应用程序。可以说在Eclipse中开发Android程序，快速而且高效。

# 第三章 需求分析

## 3.1 OA系统企业级需求调研

OA系统就是人体中的神经网络系统，传递领导理念、指令，协调全身肌肉、四肢和谐运行，愉快工作，使企业充满生命力和战斗力，为企业提供一种管理新境界。

总体上讲，它是指一切可满足于企事业单位的、综合型的、能够提高单位内部信息交流、共享、流转处理的和实现办公自动化和提高工作效率的各种信息化设备和应用软件；它不是孤立存在的，而是与企事业单位其它各类管理系统（如电子政务系统、电子商务系统、CRM系统、ERP系统、财务系统）密切相关、有机整合。

OA应该是一个企业除了生产控制之外的一切信息处理与管理的集合。它面向不同层次的使用者，便有不同的功能表现：

对于企业高层领导而言：OA是决策支持系统。OA运用科学的数学模型，结合企业内部／外部的信息为条件，为企业领导提供决策参考和依据；

对于中层管理者而言：OA是信息管理系统，OA利用业务各环节提供的基础“数据”，提炼出有用的管理“信息”，把握业务进程，降低经营风险，提高经营效率；

对于普通员工而言：OA是事务／业务处理系统。OA为办公室人员提供良好的办公手段和环境，使之准确、高效，愉快地工作。

技术范畴：OA是计算机技术在办公业务中的合理应用。计算机技术是OA的前提。如果脱离计算机技术面阔谈OA，无异于痴人说梦。没有计算机技术，OA便成无源之水、无本之木。计算机对信息的存储与处理能力极大地改变了人们的办公方式，提高了工作效率。如：要建立决策支持系统，则需要数据仓库、OLAP等技术；要建立信息管理系统，则要有数据库、程序设计语言等技术；要建立事务／业务处理系统，则离不开数据库、设计良好的人机界面和工作流控制、OLTP等技术。

以上这些探索，都是OA系统需要解决的问题，包括：针对不同的使用者，有不同的管理权限，也有不同的功能需求；对于企业内部业务数据的持久化存储，需要用不同的数据库来实现；而内部工作流的处理，则需要对流程精准的处理方式。

## 3.2程序需要实现的功能

### 3.2.1 用户安全登陆、退出应用程序

### 3.2.2 流程管理

考勤、人事调动、项目申报、发起活动等

### 3.2.3 日程管理

对项目的持续时间以及工作安排进行监控

### 3.2.4 邮件系统

企业内部邮件发送

### 3.2.5 通讯录

企业内部人员的联系方式和通讯信息

### 3.2.6 即时通讯系统

企业内部即时通讯，交流沟通

### 3.2.7 信息发布

发布通知，员工第一时间了解到公司的活动与信息

### 3.2.8 统计报表

定期发送企业内部统计与总结信息

## 3.3 程序实现方案选择

整个应用尽可能的采用MVC的模式进行架构。

用户界面：

使用各类型的View对象为用户呈现实际内容，并配合使用XML写成的Layout配置文件设定设定页面控件的布局。

业务逻辑：

前台业务逻辑使用Activity控制程序流程走向，并且对用户的触控、点击等操作进行响应。

数据存储：

Android应用使用SQLite数据库进行数据的持久化存储，在程序中使用SQLiteDatabseHelper对象封装对数据的操作。Android客户端数据大部分从服务器端同步获得，并且在客户端作为暂时存储。

# 第四章 概要设计

## 4.1 系统的简单设计

### 4.1.1 登录与身份认证

用户输入账户密码，程序向服务器端验证，验证失败提示失败信息，验证成功则进入下一级界面。

### 4.1.2 公告发布

在经过成功的用户验证后，应用程序在后台会去获取最新的公司公告信息，并将公告呈现出来，这个页面用户在查看完了之后就可以进入下一页面，当然用户也可以点击跳过按钮，直接进入功能菜单。关于公告是否显示，用户可以在设置菜单中进行选择是否在程序的开始显示公告信息。

在用户点击跳过之后，程序进入功能菜单界面。

### 4.1.3 功能菜单

经过成功的身份认证后，将进入系统的功能菜单，这里将会看到应用程序所能提供的功能。包括：

* 流程管理
* 日程管理
* 通讯录
* 即时通讯
* 公告信息
* 统计报表
* 系统设置

## 4.2系统的效果说明

本Android应用程序是一个遵循Android开发UI规则的应用程序，程序在各个方面都与当前Android手机ROM的UI风格保持一致。

程序使用Google的标准界面显示，使用GridView、GridLayout来显示块状的界面button分布。

程序使用Fragment来动态显示需要显示的块状数据。

程序使用ListView、ScrollView等来动态生成拥有特定行数的显示列表，比如说如果在数据中查询得到了20行数据，那么ListView会生成一个只有20行的List并显示给用户。

程序不同页面之间的切换，使用StartActivity来进行不同界面的数据传输。

一下是系统基本流程图：

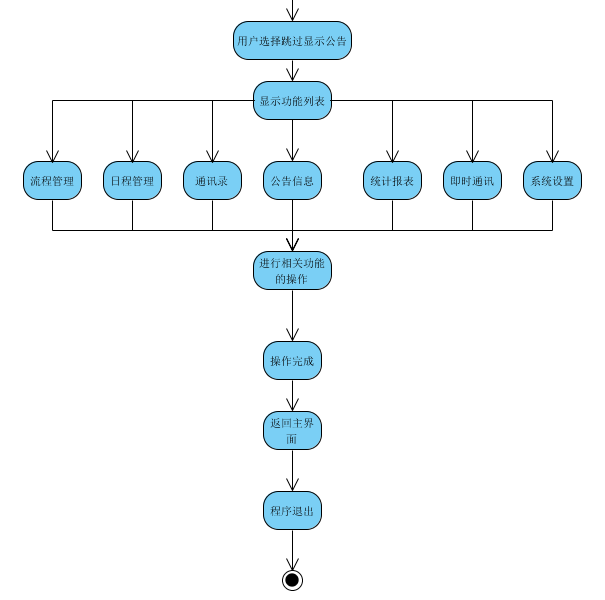
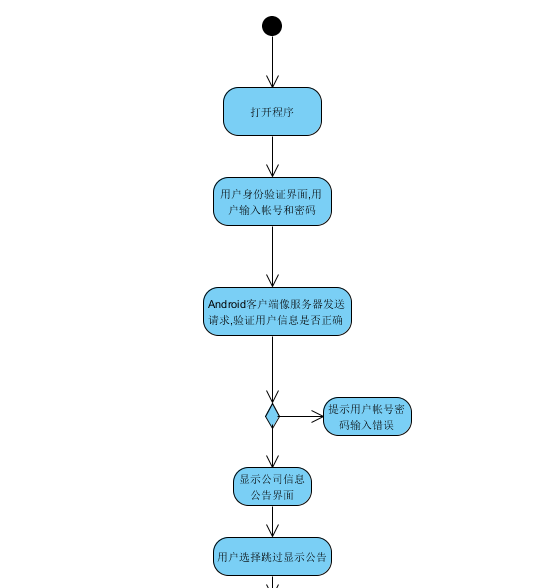
程序的基本结构与流程图如下所示：

图4-1 程序流程图

# 第五章 系统详细设计

## 5.1 系统功能的详细设计

### 5.1.1 登录与身份认证

系统由登陆界面开始，首先要求用户输入登录名和密码，登录名和密码由服务器端预先生成。在这个页面开始的同时，后台Service将会检查网络通讯情况，如果手机此时没有连接互联网，那么就会提示用户：手机此时不能连接互联网，所以不能正常使用。并且提示用户进行连接互联网的操作。

如果网络连接正常，当用户输入帐号密码并且点击提交按钮之后，会将用户名和密码提交给后台运行的Service,Service将要开启一个新的线程，将用户名加密，将用户的身份验证信息打包加入Socket发送到服务器端。

在服务器接收到需要进行用户信息验证的请求之后，服务器会对用户名进行分析，查询出数据库中的密码，然后将密码进行加密，发送到手机应用程序。

在手机端接收到服务器返回的加密数据后，与用户输入的密码进行比对，如果密码匹配，那么验证成功，为用户展示系统公告。

在上面的过程中需要保持网络的连接，如果在上述过程中出现网络错误，那么在隔一定的时间后，再次向服务器发送用户身份验证请求，如果还是有异常发生，那么就提示用户：网络环境出现错误，并将程序退回到用户的输入信息的阶段。

### 5.1.2 功能菜单

经过成功的身份认证后，将进入系统的功能菜单，这里将会看到应用程序所能提供的功能。包括：

* 流程管理
* 日程管理
* 通讯录
* 即时通讯
* 公告信息
* 统计报表
* 系统设置

上述这些功能的图标以九宫格的形式分布在功能菜单界面。在用户点击某一个按钮之后，相应按钮的监听器将会捕获到用户的点击事件，并对此事件进行相应，引导用户进入二级菜单进行更为详细的设定。

这里以流程管理为例进行说明。

在用户点击流程管理的图标后，进入流程管理的界面，并且在顶部的标题中显示的是“我的工作区”。

在这个页面依旧是以九宫格的形式来显示能为用户提供的功能。

这些功能包括：

* 发起流程
* 待处理流程
* 申请记录
* 审批记录
* 待处理任务
* 待参加会议
* 待阅读收文

此处再以发起流程为例，点击发起流程之后，系统会显示出各种流程在ListView中，作为可选项，例如工资业务流程、考勤请假流程、考勤加班流程、资产租用流程等等。这些都可以在服务器端自己定义。

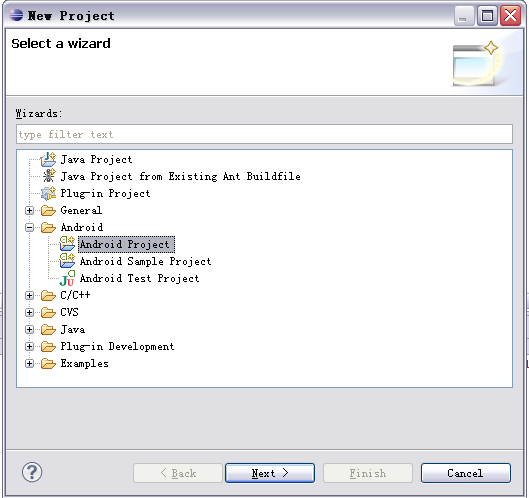
点击某一个流程之后，会显示下一个界面，需要用户填写关于此流程的详细信息，而根据流程的不同，界面中需要填写的详细信息也会有所不同。但大多数的流程都有时间、详细说明等选项需要填写。

## 5.1设计应用程序框架

### 5.1.1编写应用程序框架

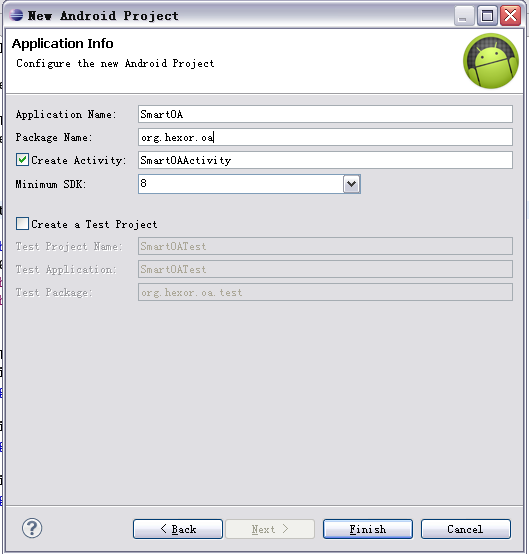
使用Eclipse建立项目，在建立项目之前需要已经在Google下载了最近的Eclipse对Android开发的最新插件，Eclipse只是个开发环境的载体，真正组成Android应用程序的组件都是由Google提供的开发组件来完成的。

#### 5.1.1.1新建项目

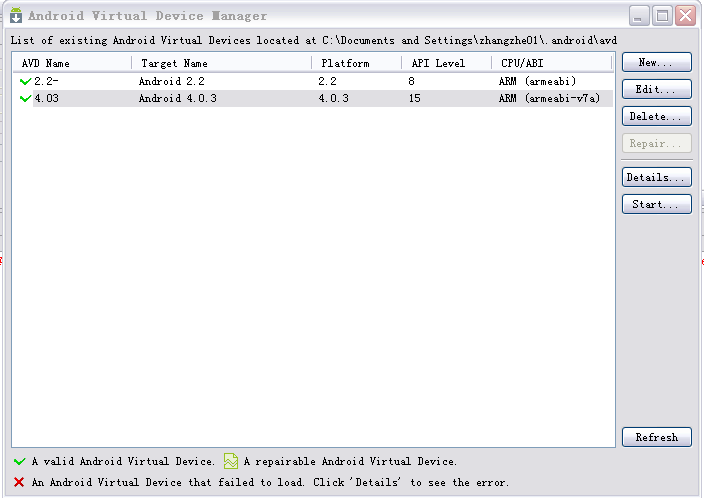


#### 5.1.1.2选择开发环境需要的手机操作系统版本

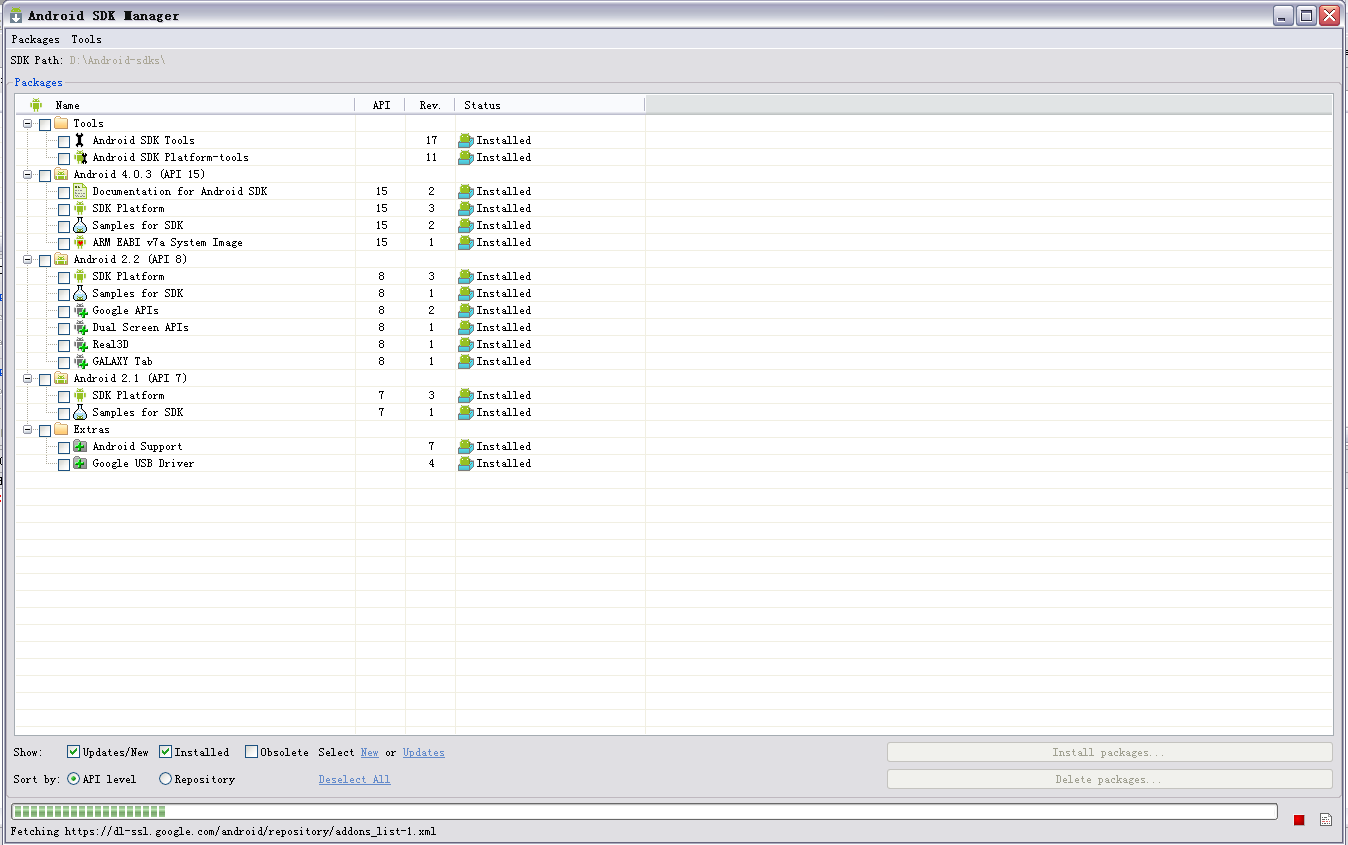
#### 5.1.1.3设定应用程序的名字，包名，以及应用程序运行所需要的操作系统最低的版本号



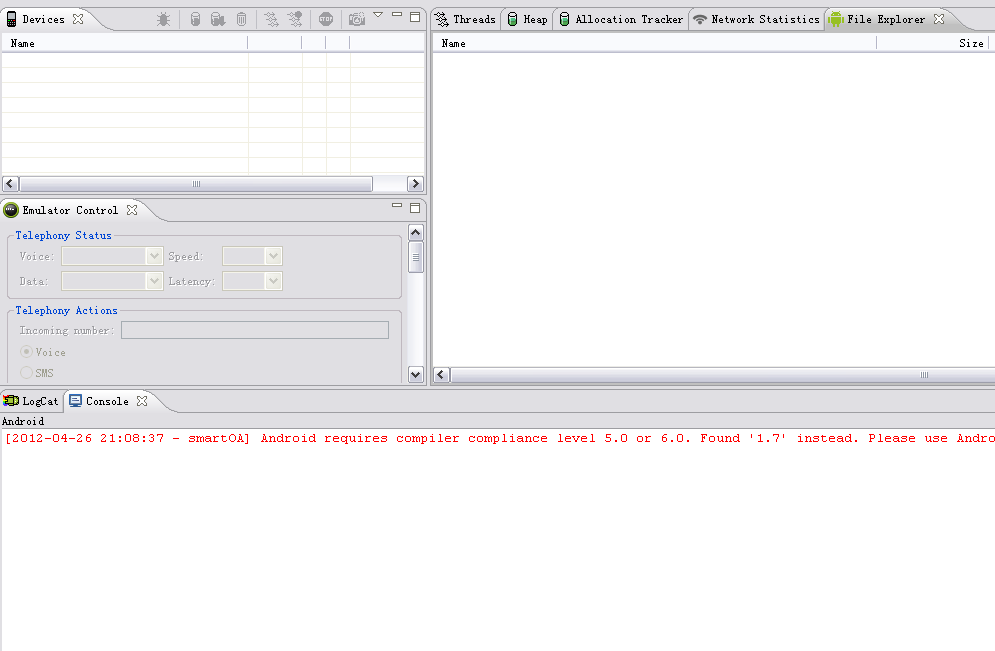
#### 5.1.1.4选择模拟Android手机环境的Android模拟器



#### 5.1.1.5下载开发需要的Android Software Development Kit（SDK）



#### 5.1.1.6准备工作结束，我们看到了Android开发工具之一DDMS



### 5.1.2为应用程序编写开始界面

下图是应用程序的登录界面。



图5-3 程序登录界面

登录界面的菜单包含“登录”、“注册” 2个按钮，以及两个输入框，分别输入帐号和密码。

我们使用XML对菜单进行绘制。

XML的layout：

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:orientation=*"vertical"* >

<TextView

android:id=*"@+id/textView3"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center"*

android:text=*"用户登录"*

android:textAppearance=*"?android:attr/textAppearanceLarge"* />

<LinearLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"* >

<TextView

android:id=*"@+id/textView1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_margin=*"5dp"*

android:layout\_weight=*"1"*

android:text=*"用户帐号"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

<EditText

android:id=*"@+id/editText1"*

android:layout\_width=*"166dp"*

android:layout\_height=*"wrap\_content"*

android:layout\_margin=*"5dp"*

android:layout\_weight=*"0.18"*

android:ems=*"10"*

android:text=*"abc123"* >

<requestFocus />

</EditText>

</LinearLayout>

<LinearLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"* >

<TextView

android:id=*"@+id/textView2"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_margin=*"5dp"*

android:layout\_weight=*"0.78"*

android:text=*"用户密码"*

android:textAppearance=*"?android:attr/textAppearanceMedium"* />

<EditText

android:id=*"@+id/editText3"*

android:layout\_width=*"166dp"*

android:layout\_height=*"wrap\_content"*

android:layout\_margin=*"5dp"*

android:layout\_weight=*"0.18"*

android:ems=*"10"*

android:inputType=*"textPassword"*

android:text=*"abc123"* >

</EditText>

</LinearLayout>

<LinearLayout

android:layout\_width=*"match\_parent"*

android:layout\_height=*"wrap\_content"*

android:baselineAligned=*"true"*

android:orientation=*"horizontal"* >

<Button

android:id=*"@+id/button1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center"*

android:layout\_margin=*"5dp"*

android:text=*"提交"* />

<Button

android:id=*"@+id/button2"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center"*

android:layout\_margin=*"5dp"*

android:text=*"重新输入"* />

<Button

android:id=*"@+id/button3"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_gravity=*"center"*

android:layout\_margin=*"5dp"*

android:text=*"退出"* />

</LinearLayout>

</LinearLayout>

### 5.1.3为界面按钮增加事件处理

上面己经做好了界面，下面要为每个菜单都增加事件响应。

我们用setOnClickListener接口来做菜单的事件响应。setOnClickListener是用来接受一个动作事件的。用来处理这些动作事件的类要实现这个接口里的方法，这个类要被发生动作的事件的类用CreateListener ( )方法来注册。当动作事件发生后，用来处理这些动作事件相应的方法将被调用。

下面添加菜单的事件响应函数代码。部分的代码如下：

下为登录界面的登录按钮所绑定的按钮，点击后会开启新线程loginThread并在线程中进行验证帐号密码之后会跳转至公告界面。

Button loginButton = (Button) findViewById(R.id.button01);

loginButton.setOnClickListener(new loginButtonListener());

**class** loginButtonListener **implements** OnClickListener {

@Override

**public** **void** onClick(View v) {

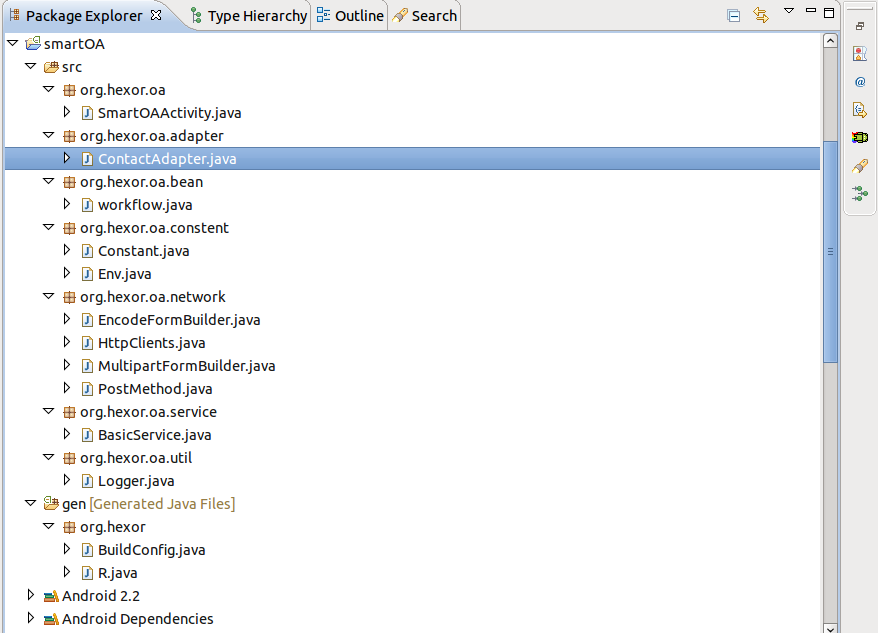
Thread loginThread = **new** Thread(**new** LoginFailureHandler());

loginThread.start();

}

}

## 5.2 应用程序代码的包的设计



整个应用程序的代码分为7个包：

包括Acitivity、Service、Util、Bean、Constent、Network、Adapter

分别介绍一下其中各个包及其中类的作用

### 5.2.1 Package org.hexor.oa

这个包中的类为各种Activity， 负责整个程序用户界面的逻辑流程，负责将设计好的视图显示在用户界面上，并监听用户所有的动作，从而对用户的各个操作进行相应的回馈和响应。

### 5.2.2 Package org.hexor.oa.bean

这个包中的类为各种在程序中使用的对象，将对象封装在Bean中方便这些对象在各个实体类中的使用和数据交换，这与企业级Java开发中的JavaBean是类似的，其中也设置了各种Set/Get、以及重载过数个构造方法，方便对对象的操作，也确保了对象实例内部数据的安全性。

### 5.2.3 Package org.hexor.oa.constant

定义各种常量，服务器地址，应用程序程序安装路径，缓存路径，缓存大小等，在这个包中的类中大多静态常量，一经定义就不再变化。

### 5.2.4 Package org.hexor.oa.network

这个包中的模块负责与服务器的初始化连接，数据的通讯等工具类。

### 5.2.5 Package org.hexor.oa.service

这个包中的类为Service类，以及一些线程的构造等，主要负责后台逻辑与任务的实现，在不打扰用户的情况下完成一系列操作，同时保证了UI线程的顺畅，以实现用户更流畅的操作体验。Service中设计了多个线程以实现不同的任务，线程都是在Service中被初始化去完成特定的任务，比如后台的数据库存储或者调用数据通讯模块在新线程完成数据通讯并动态更新UI线程。

### 5.2.6 Package org.hexor.oa.util

包含对一些主要对象的构建和生成工作。

### 5.2.7 Package org.hexor.oa.adapter

Adapter的存在就是为了将数据库中的内容按照读取出来的行动态的呈现给用户，这个包中的类设计了各种adapter通过不同的样式呈现给用户

## 5.3应用程序内容的设计

### 5.3.1用户身份验证

**package** org.hexor.oa;

**import** java.io.DataInputStream;

**import** java.net.HttpURLConnection;

**import** java.net.URL;

**import** android.app.Activity;

**import** android.app.AlertDialog;

**import** android.app.ProgressDialog;

**import** android.content.DialogInterface;

**import** android.content.Intent;

**import** android.content.SharedPreferences;

**import** android.os.Bundle;

**import** android.os.Handler;

**import** android.os.Message;

**import** android.util.Log;

**import** android.view.Menu;

**import** android.view.MenuItem;

**import** android.view.View;

**import** android.view.View.OnClickListener;

**import** android.widget.Button;

**import** android.widget.CheckBox;

**import** android.widget.CompoundButton;

**import** android.widget.EditText;

**import** android.widget.Toast;

**import** android.widget.CompoundButton.OnCheckedChangeListener;

**public** **class** Login **extends** Activity {

**private** String userName;

**private** String password;

**private** EditText view\_userName;

**private** EditText view\_password;

**private** CheckBox view\_rememberMe;

**private** Button view\_loginSubmit;

**private** Button view\_loginRegister;

**private** **static** **final** **int** *MENU\_EXIT* = Menu.*FIRST* - 1;

**private** **static** **final** **int** *MENU\_ABOUT* = Menu.*FIRST*;

**private** **final** String SHARE\_LOGIN\_TAG = "MAP\_SHARE\_LOGIN\_TAG";

**private** String SHARE\_LOGIN\_USERNAME = "MAP\_LOGIN\_USERNAME";

**private** String SHARE\_LOGIN\_PASSWORD = "MAP\_LOGIN\_PASSWORD";

**private** **boolean** isNetError;

**private** ProgressDialog proDialog;

Handler loginHandler = **new** Handler() {

**public** **void** handleMessage(Message msg) {

isNetError = msg.getData().getBoolean("isNetError");

**if** (proDialog != **null**) {

proDialog.dismiss();

}

**if** (isNetError) {

Toast.*makeText*(Login.**this**, "登陆失败:\n1.请检查您网络连接.\n2.请联系我们.!",

Toast.*LENGTH\_SHORT*).show();

}

**else** {

Toast.*makeText*(Login.**this**, "登陆失败,请输入正确的用户名和密码!",

Toast.*LENGTH\_SHORT*).show();

clearSharePassword();

}

}

};

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*login*);

findViewsById();

initView(**false**);

setListener();

}

**private** **void** findViewsById() {

view\_userName = (EditText) findViewById(R.id.*loginUserNameEdit*);

view\_password = (EditText) findViewById(R.id.*loginPasswordEdit*);

view\_rememberMe = (CheckBox) findViewById(R.id.*loginRememberMeCheckBox*);

view\_loginSubmit = (Button) findViewById(R.id.*loginSubmit*);

view\_loginRegister = (Button) findViewById(R.id.*loginRegister*);

}

**private** **void** initView(**boolean** isRememberMe) {

SharedPreferences share = getSharedPreferences(SHARE\_LOGIN\_TAG, 0);

String userName = share.getString(SHARE\_LOGIN\_USERNAME, "");

String password = share.getString(SHARE\_LOGIN\_PASSWORD, "");

Log

.*d*(**this**.toString(), "userName=" + userName + " password="

+ password);

**if** (!"".equals(userName)) {

view\_userName.setText(userName);

}

**if** (!"".equals(password)) {

view\_password.setText(password);

view\_rememberMe.setChecked(**true**);

}

**if** (view\_password.getText().toString().length() > 0) {

}

share = **null**;

}

**private** **boolean** validateLocalLogin(String userName, String password,

String validateUrl) {

**boolean** loginState = **false**;

HttpURLConnection conn = **null**;

DataInputStream dis = **null**;

**try** {

URL url = **new** URL(validateUrl);

conn = (HttpURLConnection) url.openConnection();

conn.setConnectTimeout(5000);

conn.setRequestMethod("GET");

conn.connect();

dis = **new** DataInputStream(conn.getInputStream());

**if** (conn.getResponseCode() != HttpURLConnection.*HTTP\_OK*) {

Log.*d*(**this**.toString(),

"getResponseCode() not HttpURLConnection.HTTP\_OK");

isNetError = **true**;

**return** **false**;

}

**int** loginStateInt = dis.readInt();

**if** (loginStateInt > 0) {

loginState = **true**;

}

} **catch** (Exception e) {

e.printStackTrace();

isNetError = **true**;

Log.*d*(**this**.toString(), e.getMessage() + " 127 line");

} **finally** {

**if** (conn != **null**) {

conn.disconnect();

}

}

**if** (loginState) {

**if** (isRememberMe()) {

saveSharePreferences(**true**, **true**);

} **else** {

saveSharePreferences(**true**, **false**);

}

} **else** {

**if** (!isNetError) {

clearSharePassword();

}

}

**if** (!view\_rememberMe.isChecked()) {

clearSharePassword();

}

**return** loginState;

}

**private** **void** saveSharePreferences(**boolean** saveUserName, **boolean** savePassword) {

SharedPreferences share = getSharedPreferences(SHARE\_LOGIN\_TAG, 0);

**if** (saveUserName) {

Log.*d*(**this**.toString(), "saveUserName="

+ view\_userName.getText().toString());

share.edit().putString(SHARE\_LOGIN\_USERNAME,

view\_userName.getText().toString()).commit();

}

**if** (savePassword) {

share.edit().putString(SHARE\_LOGIN\_PASSWORD,

view\_password.getText().toString()).commit();

}

share = **null**;

}

**private** **boolean** isRememberMe() {

**if** (view\_rememberMe.isChecked()) {

**return** **true**;

}

**return** **false**;

}

**private** OnClickListener submitListener = **new** OnClickListener() {

@Override

**public** **void** onClick(View v) {

proDialog = ProgressDialog.*show*(Login.**this**, "连接中..",

"连接中..请稍后....", **true**, **true**);

Thread loginThread = **new** Thread(**new** LoginFailureHandler());

loginThread.start();

}

};

**private** OnCheckedChangeListener rememberMeListener = **new** OnCheckedChangeListener() {

@Override

**public** **void** onCheckedChanged(CompoundButton buttonView,

**boolean** isChecked) {

**if** (view\_rememberMe.isChecked()) {

Toast.*makeText*(Login.**this**, "如果登录成功,以后账号和密码会自动输入!",

Toast.*LENGTH\_SHORT*).show();

}

}

};

**private** OnClickListener registerLstener = **new** OnClickListener() {

@Override

**public** **void** onClick(View v) {

Intent intent = **new** Intent();

intent.setClass(Login.**this**, Register.**class**);

startActivity(intent);

}

};

**private** **void** setListener() {

view\_loginSubmit.setOnClickListener(submitListener);

view\_loginRegister.setOnClickListener(registerLstener);

view\_rememberMe.setOnCheckedChangeListener(rememberMeListener);

}

@Override

**public** **boolean** onCreateOptionsMenu(Menu menu) {

**super**.onCreateOptionsMenu(menu);

menu.add(0, *MENU\_EXIT*, 0, getResources().getText(R.string.*MENU\_EXIT*));

menu.add(0, *MENU\_ABOUT*, 0, getResources().getText(R.string.*MENU\_ABOUT*));

**return** **true**;

}

@Override

**public** **boolean** onMenuItemSelected(**int** featureId, MenuItem item) {

**super**.onMenuItemSelected(featureId, item);

**switch** (item.getItemId()) {

**case** *MENU\_EXIT*:

finish();

**break**;

**case** *MENU\_ABOUT*:

alertAbout();

**break**;

}

**return** **true**;

}

**private** **void** alertAbout() {

**new** AlertDialog.Builder(Login.**this**).setTitle(R.string.*MENU\_ABOUT*)

.setMessage(R.string.*aboutInfo*).setPositiveButton(

R.string.*ok\_label*,

**new** DialogInterface.OnClickListener() {

**public** **void** onClick(

DialogInterface dialoginterface, **int** i) {

}

}).show();

}

**private** **void** clearSharePassword() {

SharedPreferences share = getSharedPreferences(SHARE\_LOGIN\_TAG, 0);

share.edit().putString(SHARE\_LOGIN\_PASSWORD, "").commit();

share = **null**;

}

**class** LoginFailureHandler **implements** Runnable {

@Override

**public** **void** run() {

userName = view\_userName.getText().toString();

password = view\_password.getText().toString();

String validateURL="http://192.168.1.229:8080/android\_server/LoginValidate?userName="

+ userName + "&password=" + password;

**boolean** loginState = validateLocalLogin(userName, password,

validateURL);

Log.*d*(**this**.toString(), "validateLogin");

**if** (loginState) {

Intent intent = **new** Intent();

intent.setClass(Login.**this**, IndexPage.**class**);

Bundle bundle = **new** Bundle();

bundle.putString("MAP\_USERNAME", userName);

intent.putExtras(bundle);

startActivity(intent);

proDialog.dismiss();

} **else** {

Message message = **new** Message();

Bundle bundle = **new** Bundle();

bundle.putBoolean("isNetError", isNetError);

message.setData(bundle);

loginHandler.sendMessage(message);

}

}

}

}

### 5.3.2数据库存取操作

package org.hexor.oa.provider;

import java.io.File;

import java.util.UUID;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.os.Environment;

import android.os.Handler;

import android.util.Log;

import android.widget.Toast;

import com.snda.album.constant.Constant;

import com.snda.album.util.StringUtil;

public class DepotOpenHelper {

private final static int VERSION = 3;

private final String DB\_NAME = "face\_depot.db";

private Context mContext;

private SQLiteDatabase mDatabase;

private String mDatabaseName = DB\_NAME;

private boolean mIsInitializing = false;

private Handler mHandler = new Handler();

public DepotOpenHelper(Context context, String databaseName) {

mDatabaseName = databaseName;

mContext = context;

}

public DepotOpenHelper(Context context) {

mContext = context;

}

public synchronized SQLiteDatabase getDatabase() {

if (mDatabase != null && mDatabase.isOpen()) {

return mDatabase;

}

if (mIsInitializing) {

throw new IllegalStateException("getDatabase called recursively");

}

boolean success = false;

SQLiteDatabase db = null;

try {

mIsInitializing = true;

db = SQLiteDatabase.openOrCreateDatabase(getPath(), null);

int version = db.getVersion();

if (version != VERSION) {

db.beginTransaction();

try {

if (version == 0) {

onCreate(db);

} else {

onUpgrade(db, version);

}

db.setVersion(VERSION);

db.setTransactionSuccessful();

} finally {

db.endTransaction();

}

}

success = true;

onOpen(db);

} catch (Exception e) {

e.printStackTrace();

} finally {

mIsInitializing = false;

if (success) {

mDatabase = db;

postOpen();

} else {

if (db != null)

db.close();

}

}

return mDatabase;

}

public SQLiteDatabase explicitGetDatabase() {

return SQLiteDatabase.openOrCreateDatabase(getPath(), null);

}

private File getPath() {

String state = Environment.getExternalStorageState();

if (!Environment.MEDIA\_MOUNTED.equals(state)

|| Environment.MEDIA\_MOUNTED\_READ\_ONLY.equals(state)) {

throw new IllegalStateException(

"External storage not mounted for database operation");

}

File dbdir = new File(Environment.getExternalStorageDirectory()

.getAbsolutePath() + "/Android/smartalbum");

if (!dbdir.exists()) {

dbdir.mkdirs();

}

return new File(dbdir.getAbsolutePath() + "/" + mDatabaseName);

}

public synchronized void close() {

if (mIsInitializing)

throw new IllegalStateException("Closed during initialization");

if (mDatabase != null && mDatabase.isOpen()) {

mDatabase.close();

mDatabase = null;

}

}

private void onUpgrade(SQLiteDatabase db, int version) {

mHandler.post(new Runnable() {

@Override

public void run() {

Toast.makeText(mContext, "数据库正在升级，请稍等..", Toast.LENGTH\_SHORT).show();

}

});

ContentValues cv = new ContentValues();

switch (version) {

case 1:

cv.put(OADepot.Faces.IS\_CONFIRM, 0);

db.update(OADepot.FACE\_TABLE, cv, null, null);

cv.clear();

case 2:

db.execSQL("UPDATE " + OADepot.CONTACT\_TABLE + " SET "

+ OADepot.Contacts.NAME\_ORIGIN + " = "

+ OADepot.Contacts.NAME);

cv.put(OADepot.Contacts.NAME, "");

db.update(OADepot.CONTACT\_TABLE, cv, OADepot.Contacts.\_ID

+ " NOT IN (SELECT DISTINCT " + OADepot.Groups.CONTACT\_ID

+ " FROM " + OADepot.GROUP\_TABLE + ")", null);

}

}

private void onCreate(SQLiteDatabase db) {

db.execSQL(new StringBuilder("CREATE TABLE IF NOT EXISTS ")

.append(OADepot.META\_TABLE).append(" (")

.append(OADepot.Meta.K).append(" TEXT PRIMARY KEY, ")

.append(OADepot.Meta.V).append(" TEXT NOT NULL)").toString());

db.execSQL(new StringBuilder("CREATE TABLE IF NOT EXISTS ")

.append(OADepot.SHADOW\_TABLE).append(" (")

.append(OADepot.Shadows.\_ID).append(" INTEGER PRIMARY KEY, ")

.append(OADepot.Shadows.DATE\_TAKEN).append(" INTEGER, ")

.append(OADepot.Shadows.IS\_HIDDEN)

.append(" INTEGER DEFAULT 0, ").append(OADepot.Shadows.MD5)

.append(" TEXT, ").append(OADepot.Shadows.MIMETYPE)

.append(" TEXT, ").append(OADepot.Shadows.PATH)

.append(" TEXT, ").append(OADepot.Shadows.BUCKET\_ID)

.append(" TEXT, ")

.append(OADepot.Shadows.BUCKET\_DISPLAY\_NAME)

.append(" TEXT, ").append(OADepot.Shadows.LATITUDE)

.append(" REAL, ").append(OADepot.Shadows.LONGITUDE)

.append(" REAL, ").append(OADepot.Shadows.ORIENTATION)

.append(" INTEGER, ").append(OADepot.Shadows.IMAGE\_WIDTH)

.append(" INTEGER, ").append(OADepot.Shadows.IMAGE\_HEIGHT)

.append(" INTEGER, ").append(OADepot.Shadows.WIDTH)

.append(" INTEGER, ").append(OADepot.Shadows.HEIGHT)

.append(" INTEGER, ").append(OADepot.Shadows.DETECTED)

.append(" INTEGER DEFAULT 0, ")

.append(OADepot.Shadows.THUMBNAIL)

.append(" BLOB DEFAULT NULL, ")

.append(OADepot.Shadows.MINI\_THUMBNAIL)

.append(" BLOB DEFAULT NULL, ")

.append(OADepot.Shadows.BACKUP0)

.append(" INTEGER DEFAULT NULL, ")

.append(OADepot.Shadows.BACKUP1)

.append(" INTEGER DEFAULT NULL, ")

.append(OADepot.Shadows.BACKUP2)

.append(" TEXT DEFAULT NULL, ")

.append(OADepot.Shadows.BACKUP3)

.append(" TEXT DEFAULT NULL)").toString());

db.execSQL(new StringBuilder("CREATE TABLE IF NOT EXISTS ")

.append(OADepot.CONTACT\_TABLE).append(" (")

.append(OADepot.Contacts.\_ID)

.append(" INTEGER PRIMARY KEY, ")

.append(OADepot.Contacts.NAME).append(" TEXT DEFAULT '', ")

.append(OADepot.Contacts.PINYIN).append(" TEXT DEFAULT '', ")

.append(OADepot.Contacts.PY).append(" TEXT DEFAULT '', ")

.append(OADepot.Contacts.DATA).append(" TEXT DEFAULT '',")

.append(OADepot.Contacts.PHOTOID).append(" INTEGER ,")

.append(OADepot.Contacts.TYPE).append(" INTEGER, ")

.append(OADepot.Contacts.BACKUP0)

.append(" INTEGER DEFAULT NULL, ")

.append(OADepot.Contacts.INSTERT\_TIME)

.append(" INTEGER DEFAULT NULL, ")

.append(OADepot.Contacts.NAME\_ORIGIN)

.append(" TEXT DEFAULT NULL, ")

.append(OADepot.Contacts.BACKUP3)

.append(" TEXT DEFAULT NULL)").toString());

}

private void onOpen(SQLiteDatabase db) {

db.execSQL("PRAGMA foreign\_keys = ON");

db.execSQL("PRAGMA synchronous = OFF");

db.execSQL("PRAGMA recursive\_triggers = ON");

db.execSQL("PRAGMA cache\_size = 4096");

db.rawQuery("PRAGMA journal\_mode = MEMORY", null).close();

}

private void postOpen() {

if (getMeta(Constant.K\_REVISION) == null) {

setMeta(Constant.K\_REVISION,

StringUtil.md5(UUID.randomUUID().toString().getBytes()));

}

}

public String getMeta(String key) {

SQLiteDatabase db = getDatabase();

Cursor c = db.query(OADepot.META\_TABLE,

new String[] { OADepot.Meta.V }, OADepot.Meta.K + " = ?",

new String[] { key }, null, null, null);

try {

if (c.moveToFirst()) {

return c.getString(0);

} else {

return null;

}

} catch (Exception e) {

Log.e(getClass().getSimpleName(), "getMeta", e);

return null;

} finally {

if (c != null) {

c.close();

}

}

}

public void setMeta(String key, String value) {

SQLiteDatabase db = getDatabase();

db.execSQL("INSERT OR REPLACE INTO " + OADepot.META\_TABLE + " ("

+ OADepot.Meta.K + ", " + OADepot.Meta.V + ") VALUES ('"

+ key + "', '" + value + "')");

}

@Override

protected void finalize() throws Throwable {

close();

super.finalize();

}

}

### 5.3.3流程对象的设计

### 5.3.4应用程序与服务器之间的通讯

Package org.hexor.oa.network;

import java.io.InputStream;

import java.io.OutputStream;

import java.net.URLConnection;

public class HttpUtils {

/\*\*关闭URLConnection的InputStreame和OutputStream\*/

public static void close(URLConnection conn) {

try {

if (conn != null) {

InputStream is = conn.getInputStream();

OutputStream out = conn.getOutputStream();

if (is != null) {

is.close();

}

if (out != null) {

out.close();

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*关闭InputStreame\*/

public static void close(InputStream is) {

try {

if(is!=null){

is.close();

}

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*关闭OutputStream\*/

public static void close(OutputStream out) {

try {

if(out!=null){

out.close();

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

### 5.3.5服务器端接收请求的设计

### 5.3.6应用程序线程的设计

package org.hexor.service;

import java.util.Comparator;

import android.content.ContentResolver;

import android.content.Context;

import android.os.Process;

import android.text.GetChars;

import android.util.Log;

public abstract class Task implements Runnable, Comparable<Task> {

private static final boolean VERBOSE = true;

public final static int LOW\_PRIORITY = -1;

public final static int NORM\_PRIORITY = 0;

public final static int HIGH\_PRIORITY = 1;

public final static int EMER\_PRIORITY = 2;

private int mPriority = NORM\_PRIORITY;

private Context mContext;

private OAService.Spawn mSpawn;

protected ContentResolver mResolver;

public Task(int priority, Context context, OAService.Spawn spawn) {

mPriority = priority;

mContext = context;

mSpawn = spawn;

mResolver = context.getContentResolver();

}

public static class PriorityComparator implements Comparator<Runnable> {

@Override

public int compare(Runnable left, Runnable right) {

return ((Task) left).compareTo((Task) right);

}

}

@Override

public int compareTo(Task another) {

return getPriority() - another.getPriority();

}

public int getPriority() {

return mPriority;

}

protected Context getContext() {

return mContext;

}

protected OAService.Spawn spawn() {

return mSpawn;

}

@Override

public void run() {

Process.setThreadPriority(Process.THREAD\_PRIORITY\_LOWEST);

try {

if (VERBOSE) {

Log.v(getClass().getSimpleName(), "executing in "

+ Thread.currentThread().getName());

}

execute();

if (VERBOSE) {

Log.v(getClass().getSimpleName(), "exit from "

+ Thread.currentThread().getName());

}

} catch (Exception e) {

Log.w(getClass().getSimpleName(), Thread.currentThread().getName(),

e);

} finally {

mSpawn.sendMessage(OAService.MSG\_END\_OF\_TASK, 500);

}

}

public void d(String message) {

Log.d(getClass().getSimpleName(), "["

+ Thread.currentThread().getName() + "] " + message);

}

public void w(String message) {

Log.w(getClass().getSimpleName(), "["

+ Thread.currentThread().getName() + "] " + message);

}

public void e(String message, Throwable tr) {

Log.e(getClass().getSimpleName(), "["

+ Thread.currentThread().getName() + "] " + message, tr);

}

protected abstract void execute();

}

### 5.3.7应用程序后台Service的设计

package org.hexor.service;

import java.io.File;

import java.util.concurrent.LinkedBlockingQueue;

import java.util.concurrent.RejectedExecutionException;

import java.util.concurrent.ThreadPoolExecutor;

import java.util.concurrent.TimeUnit;

import java.util.concurrent.atomic.AtomicInteger;

import android.app.ActivityManager;

import android.app.Notification;

import android.app.NotificationManager;

import android.app.PendingIntent;

import android.app.Service;

import android.content.ComponentName;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences;

import android.database.ContentObserver;

import android.database.Cursor;

import android.net.Uri;

import android.os.Environment;

import android.os.FileObserver;

import android.os.Handler;

import android.os.IBinder;

import android.os.Message;

import android.os.RemoteCallbackList;

import android.os.RemoteException;

import android.provider.ContactsContract;

import android.provider.MediaStore.Images;

import android.provider.MediaStore.Images.Media;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.RemoteViews;

import com.snda.album.BaseGalleryPicker;

import com.snda.album.R;

import com.snda.album.SettingActivity;

import com.snda.album.SplashActivity;

import com.snda.album.bean.Image;

import com.snda.album.constant.Constant;

import com.snda.album.provider.FaceDepot;

import com.snda.album.util.Common;

import com.snda.album.util.Logger;

public class OAService extends Service {

private static final String TAG = "OAService";

private static final boolean VERBOSE = false;

public static final int MSG\_SYNC\_MEDIA\_FINISH = 0;

public static final int MSG\_SYNC\_CONTACT\_FINISH = 1;

public static final int MSG\_CONSTRUCT\_PROGRESS = 2;

public static final int MSG\_START\_SYNC\_NETWORK = 3;

public static final int MSG\_MEDIA\_CHANGED = 4;

public static final int MSG\_START\_TASK = 10;

public static final int MSG\_END\_OF\_TASK = 11;

public static final int MSG\_DOWNLOAD\_PROGRESS = 20;

public static final int MSG\_UPDATE\_PROGRESS = 22;

public static final int MSG\_DOWNLOAD\_FINISH = 23;

public static final int MSG\_DOWNLOAD\_START = 21;

public static final int MSG\_DOWNLOAD\_FAILED = 24;

private int downloadProgress = 0;

private String mSessionId = null;

private ThreadPoolExecutor mPool;

private LinkedBlockingQueue<Runnable> mQueue;

private AtomicInteger mCounter = new AtomicInteger(0);

private Spawn mSpawn;

ContentObserver mObserver;

ContentObserver mContactObserver;

Notification notification;

NotificationManager nm;

PendingIntent pi;

private Handler mHandler = new Handler() {

@Override

public void handleMessage(Message msg) {

Intent intent;

switch (msg.what) {

case MSG\_SYNC\_MEDIA\_FINISH:

Common.completeIndex(getApplicationContext());

intent = new Intent(SYNC\_MEDIA\_FINISH);

sendBroadcast(intent);

break;

case MSG\_SYNC\_CONTACT\_FINISH:

intent = new Intent(SYNC\_CONTACT\_FINISH);

sendBroadcast(intent);

break;

case MSG\_CONSTRUCT\_PROGRESS:

int num = mCallbacks.beginBroadcast();

for (int i = 0; i < num; i++) {

try {

mCallbacks.getBroadcastItem(i).notifyMediaScan(

msg.arg1, msg.arg2);

} catch (RemoteException e) {

Log.e(TAG, "", e);

}

}

mCallbacks.finishBroadcast();

break;

case MSG\_MEDIA\_CHANGED:

if (!Common.isIndexComplete(getApplicationContext())) {

mSpawn.sendMessage(MSG\_MEDIA\_CHANGED, 2000);

} else {

mSpawn.refreshShadow();

}

break;

case MSG\_START\_TASK:

startTask((Task) msg.obj);

break;

case MSG\_END\_OF\_TASK:

if (decrementCount() == 0) {

Common.Assert(!hasMessages(MSG\_START\_TASK));

try {

Common.Assert(mPool.getActiveCount() == 0);

} catch (AssertionError e) {

Logger.e(

TAG,

"Thread Pool active count = "

+ mPool.getActiveCount(), e);

}

// stopSelf();

}

case MSG\_DOWNLOAD\_PROGRESS:

if (msg.arg1 == MSG\_UPDATE\_PROGRESS) {

downloadProgress = msg.arg2;

notification.contentView.setProgressBar(R.id.progressBar,

100, downloadProgress, false);

notification.contentView.setTextViewText(

R.id.textViewProgress, " " + downloadProgress

+ "%");

nm.notify(MSG\_DOWNLOAD\_START, notification);

if (downloadProgress == 100) {

nm.cancel(MSG\_DOWNLOAD\_START);

}

} else if (msg.arg1 == MSG\_DOWNLOAD\_FINISH) {

// 发通知 下载完成

Log.d(TAG, "get MSG\_DOWNLOAD\_FINISH ");

sendNotification(OAService.this, "新版本下载完成,点击安装",

msg.arg1);

} else if (msg.arg1 == MSG\_DOWNLOAD\_START) {

// 发通知 下载开始s

Log.d(TAG, "get MSG\_DOWNLOAD\_START ");

// Intent it = new Intent(OAService.this,

// SettingActivity.class);

// it.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

// ActivityManager am = (ActivityManager) getSystemService(ACTIVITY\_SERVICE);

// ComponentName cn = am.getRunningTasks(2).get(1).topActivity;

// System.out.println(cn);

//// startActivity(it);

sendNotification(OAService.this, "新版本下载开始", msg.arg1);

} else if (msg.arg1 == MSG\_DOWNLOAD\_FAILED) {

// 下载异常, 点击通知返回SettingActivity

Log.d(TAG, "get MSG\_DOWNLOAD\_FAILED ");

nm.cancel(1);

sendNotification(OAService.this, "新版本下载失败,点击重新下载",

msg.arg1);

}

break;

}

};

};

public class Spawn {

public void spawn(Task task) {

spawn(task, 0);

}

public void spawn(Task task, long delay) {

incrementCount();

Message msg = mHandler.obtainMessage(MSG\_START\_TASK, task);

mHandler.sendMessageDelayed(msg, delay);

}

public void spawnNetworkTask() {

if (sessionReady()) {

spawn(new SyncNetworkTask(getApplicationContext(), mSessionId,

mSpawn));

}

}

public void spawnDownloadApkTask(String apkUrl) {

if (sessionReady()) {

Log.d(TAG, "spawnDownloadApkTask ");

spawn(new DownloadApkTask(getApplicationContext(), mSessionId,

mSpawn, apkUrl));

}

}

public void spawnResultTask() {

if (sessionReady()) {

spawn(new GetGroupResultTask(getApplicationContext(),

mSessionId, mSpawn));

}

}

public void spawnResultTask(long delay) {

if (sessionReady()) {

spawn(new GetGroupResultTask(getApplicationContext(),

mSessionId, mSpawn), delay);

}

}

public void spawnUploadTask(int[] ids) {

if (sessionReady()) {

spawn(new UploadTask(getApplicationContext(), mSessionId, ids,

mSpawn));

}

}

public void refreshShadow() {

Cursor c = getContentResolver().query(FaceDepot.SHADOW\_NATIVE\_URI,

new String[] { "MAX(" + FaceDepot.Shadows.\_ID + ")" },

null, null, null);

try {

c.moveToFirst();

long maxId = c.getInt(0);

c.close();

c = getContentResolver().query(

Images.Media.EXTERNAL\_CONTENT\_URI,

MediaStoreScanTask.FIT\_PROJECTIONS,

Images.ImageColumns.\_ID + " > " + maxId + " AND "

+ MediaStoreScanTask.FIT\_MIME\_TYPE\_SEL,

MediaStoreScanTask.FIT\_MIME\_TYPE, null);

if(c == null) {

return;

}

if (c.getCount() > 0) {

Common.refreshIndex(getApplicationContext());

while (c.moveToNext()) {

Image image = new Image(getContentResolver());

image.\_id = c.getInt(0);

if (c.getString(1).equalsIgnoreCase(".jpg")) {

image.mimetype = "image/jpeg";

} else {

image.mimetype = c.getString(1);

}

image.date\_taken = c.getLong(2);

image.path = c.getString(3);

image.bucket\_id = c.getLong(4);

image.bucket\_name = c.getString(5);

image.orientation = c.getInt(6);

image.latitude = c.getInt(7);

image.longitude = c.getInt(8);

ShadowBuilder.put(getApplicationContext(), mSpawn,

image);

}

} else {

spawn(new MediaStoreScanTask(getApplicationContext(), this,

false));

}

} finally {

if (c != null && !c.isClosed()) {

c.close();

}

}

}

public void disableMessage(int what) {

mHandler.removeMessages(what);

}

public void sendMessage(int what) {

mHandler.sendEmptyMessage(what);

}

public void sendMessage(Message msg) {

mHandler.sendMessage(msg);

}

public void sendMessage(int what, long delay) {

mHandler.sendEmptyMessageDelayed(what, delay);

}

public void progress(int cur, int total) {

mHandler.obtainMessage(MSG\_CONSTRUCT\_PROGRESS, cur, total)

.sendToTarget();

}

public Message obtainMessage() {

return mHandler.obtainMessage();

}

}

private void incrementCount() {

if (VERBOSE) {

Log.d(TAG, "count increment to " + mCounter.incrementAndGet());

} else {

mCounter.incrementAndGet();

}

}

private int decrementCount() {

if (VERBOSE) {

int count = mCounter.decrementAndGet();

Log.d(TAG, "count decrement to " + count);

return count;

} else {

return mCounter.decrementAndGet();

}

}

private boolean sessionReady() {

if (mSessionId != null) {

return true;

}

SharedPreferences sp = getSharedPreferences(Constant.PREF\_NAME\_AUTH,

MODE\_PRIVATE);

mSessionId = sp.getString(Constant.PREF\_SESSIONID, null);

if (mSessionId != null) {

return true;

}

return false;

}

private RemoteCallbackList<ICallback> mCallbacks = new RemoteCallbackList<ICallback>();

private IService.Stub mBinder = new IService.Stub() {

@Override

public boolean registerServiceCallback(ICallback callback)

throws RemoteException {

if (callback != null) {

return mCallbacks.register(callback);

}

return false;

}

@Override

public void unregisterServiceCallback(ICallback callback)

throws RemoteException {

if (callback != null) {

mCallbacks.unregister(callback);

}

}

@Override

public void removeNonFace(int[] ids) throws RemoteException {

Log.d(TAG, "removeNonFace " + ids.length);

if (sessionReady()) {

mSpawn.spawn(new RemoveNonFaceTask(getApplicationContext(),

mSessionId, ids, mSpawn));

}

}

@Override

public void removeFromGroup(int[] ids, int[] faceIds)

throws RemoteException {

if (sessionReady()) {

mSpawn.spawn(new RemoveFromGroupTask(getApplicationContext(),

mSessionId, ids, faceIds, mSpawn));

}

}

@Override

public void assignToContact(int groupId, int contactId, int[] faceIds)

throws RemoteException {

if (sessionReady()) {

mSpawn.spawn(new AssignToContactTask(getApplicationContext(),

mSessionId, groupId, contactId, faceIds, "", mSpawn));

}

}

@Override

public void upload(int[] ids) throws RemoteException {

mSpawn.spawnUploadTask(ids);

}

@Override

public void group() throws RemoteException {

mSpawn.spawnResultTask();

}

@Override

public void confirmContact(int[] ids) throws RemoteException {

if (sessionReady()) {

mSpawn.spawn(new ConfirmContactTask(getApplicationContext(),

mSessionId, ids, mSpawn));

}

}

@Override

public void assignToInputName(int groupId, int contactId,

int[] faceIds, String inputName) throws RemoteException {

mSpawn.spawn(new AssignToContactTask(getApplicationContext(),

mSessionId, groupId, contactId, faceIds, inputName, mSpawn));

}

};

@Override

public void onCreate() {

super.onCreate();

mQueue = new LinkedBlockingQueue<Runnable>();

mPool = new ThreadPoolExecutor(8, 8, 10, TimeUnit.SECONDS, mQueue);

mSpawn = new Spawn();

mObserver = new ContentObserver(mHandler) {

@Override

public void onChange(boolean selfChange) {

startService(new Intent(MEDIA\_CHANGED));

}

};

mContactObserver = new ContentObserver(mHandler) {

@Override

public void onChange(boolean selfChange) {

startService(new Intent(SYNC\_CONTACT));

}

};

getContentResolver().registerContentObserver(

Media.EXTERNAL\_CONTENT\_URI, true, mObserver);

getContentResolver().registerContentObserver(

ContactsContract.Contacts.CONTENT\_URI, true,

mContactObserver);

}

@Override

public void onStart(Intent intent, int startId) {

// do scan and re-pack here.

if (!sdcardAvailable()) {

stopSelf();

return;

}

if (intent.getAction().equals(SYNC\_CONTACT)) {

mSpawn.spawn(new ContactScanTask(getApplicationContext(), mSpawn));

} else if (intent.getAction().equals(SYNC\_MEDIA)) {

mSpawn.spawn(new MediaStoreScanTask(getApplicationContext(), mSpawn));

} else if (intent.getAction().equals(SYNC\_NETWORK)) {

mSpawn.spawnNetworkTask();

} else if (intent.getAction().equals(MEDIA\_CHANGED)) {

mSpawn.disableMessage(MSG\_MEDIA\_CHANGED);

mSpawn.sendMessage(MSG\_MEDIA\_CHANGED, 1000);

} else if (intent.getAction().equals(APK\_DOWNLOAD)) {

String apkUrl = intent.getExtras().getString("url");

Log.d(TAG, "Service Get apk Url");

mSpawn.spawnDownloadApkTask(apkUrl);

}

}

@Override

public int onStartCommand(Intent intent, int flags, int startId) {

onStart(intent, startId);

return Service.START\_NOT\_STICKY;

}

@Override

public IBinder onBind(Intent intent) {

return mBinder;

}

@Override

public boolean onUnbind(Intent intent) {

return true;

}

@Override

public void onRebind(Intent intent) {

super.onRebind(intent);

}

@Override

public void onDestroy() {

super.onDestroy();

if (VERBOSE)

Log.d(TAG, "onDestroy");

mPool.shutdown();

mQueue = null;

getContentResolver().unregisterContentObserver(mObserver);

getContentResolver().unregisterContentObserver(mContactObserver);

}

private boolean sdcardAvailable() {

return Environment.MEDIA\_MOUNTED.equals(Environment

.getExternalStorageState())

&& !Environment.MEDIA\_MOUNTED\_READ\_ONLY.equals(Environment

.getExternalStorageState());

}

private void startTask(Task t) {

try {

mPool.execute(t);

} catch (RejectedExecutionException e) {

if (mPool.isShutdown()) {

Logger.e(TAG, "task[" + t.getClass().getSimpleName()

+ "] posted after shutdown.", e);

} else {

Log.e(TAG, t.getClass().getSimpleName(), e);

}

}

}

private void sendNotification(Context context, String message,

int notificationId) {

String ns = Context.NOTIFICATION\_SERVICE;

nm = (NotificationManager) context.getSystemService(ns);

notification = new Notification();

notification.icon = R.drawable.icon;

notification.tickerText = message;

notification.when = System.currentTimeMillis();

if (notificationId == MSG\_DOWNLOAD\_FINISH) {

Intent intent = new Intent(Intent.ACTION\_VIEW);

File apkFile = new File(SV\_FILENAME);

intent.setDataAndType(Uri.fromFile(apkFile),

"application/vnd.android.package-archive");

notification.flags = Notification.FLAG\_AUTO\_CANCEL;

pi = PendingIntent.getActivity(context, 0, intent, 0);

notification.setLatestEventInfo(context, "智能相册", message, pi);

notification.contentIntent = pi;

nm.notify(notificationId, notification);

} else if (notificationId == MSG\_DOWNLOAD\_START) {

notification.contentView = new RemoteViews(getPackageName(),

R.layout.notifi\_progressbar);

notification.contentView.setProgressBar(R.id.progressBar, 100, 0,

false);

Intent intent = new Intent(this, SettingActivity.class);

PendingIntent pi = PendingIntent.getActivity(context, 0, intent, 0);

notification.contentIntent = pi;

nm.notify(notificationId, notification);

} else if (notificationId == MSG\_DOWNLOAD\_FAILED) {

Intent intent = new Intent(this, SettingActivity.class);

notification.flags = Notification.FLAG\_AUTO\_CANCEL;

pi = PendingIntent.getActivity(context, 0, intent, 0);

notification.setLatestEventInfo(context, "智能相册", message, pi);

notification.contentIntent = pi;

nm.notify(notificationId, notification);

}

}

}

## 5.4添加应用程序的其它功能

### 5.4.1设计关于对话框

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<resources>

<string name=*"loginUserNameLabel"*>用户名</string>

<string name=*"loginPasswordLabel"*>密码</string>

<string name=*"loginSubmit"*>登陆</string>

<string name=*"loginRegister"*>注册</string>

<string name=*"loginTitle"*>登陆</string>

<string name=*"loginRememberMe"*>记住我的账号和密码</string>

<string name=*"MENU\_EXIT"*>退出</string>

<string name=*"MENU\_ABOUT"*>关于</string>

<string name=*"aboutInfo"*>

制作人:张哲 \n邮 箱: cupsoda@gmail.com \n本应用为毕业设计测试应用!

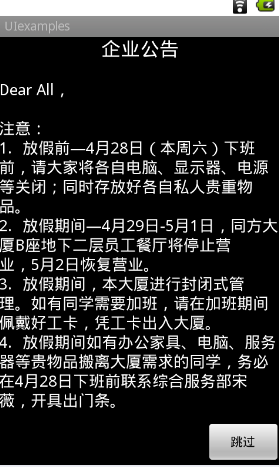
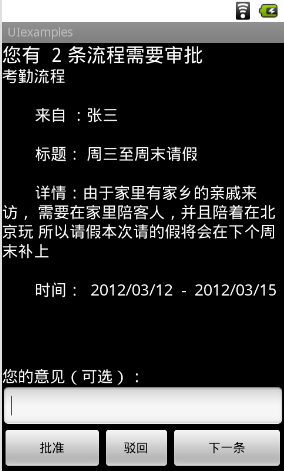
</string>

<string name=*"ok\_label"*>确认</string>

</resources>



## 5.5系统主要界面效果

# 第六章 系统测试

为了确保本应用能够正常运行，需要在发布之后做一次较全面的测试。现将具体操作及过程举例说明如下：

## 6.1测试举例

### 6.1.1测试程序登录页面

打开程序后，首先出现的是登录页面，如果没有提示说网络连接失败或者其他异常情况，证明登录界面成功

### 6.1.2测试程序主页面

登录成功后进入公告界面，点击跳过按钮，会进入主页面，主页面会显示系统的功能列表，包括流程管理、日程管理、邮件系统、通讯录、即时通讯、信息发布、统计报表。当点击某一个功能选项之后，这个界面就会切换到相应功能的界面，应用程序撤销功能列表页面并平滑过度到下一个页面，等待用户进行详细的操作。在操作过程中会有需要用户进行确认的动作，如果用户在操作过程中点击了后退按钮且有为保存的数据，则会跳出提示用户是否保存当前未完成的数据，如果用户点击了保存，则当用户下一次使用此功能时，应用程序会自动为用户填充上次的未填写完成的数据。

### 6.1.3测试程序退出页面

当用户结束某一项功能之后，则此项功能完成，系统会再次跳入功能列表的界面，用户可以继续点击其他的功能来继续使用本应用程序。

如果用户再次点击后退按钮，那么会提示用户是否退出本应用程序。点击确定则退出应用，否则继续停留在本界面。

经过以上各项的测试，证明本系统完全可以正常运行，至此测试成功！

## 6.2测试项目

由于实际开发过程中还是不可避免地会产生差错，应用中通常可能隐藏着错误和缺陷，不经周密测试的系统投入运行，将会造成难以想象的后果，因此系统测试是软件开发过程中为保证软件质量必须进行的工作。

从测试本身看，分为单元测试，组合测试，确认测试等。测试对象是随不同阶段而异的，最基本、最初的测试是单元测试，后面的组合测试、确认测试都是以被测过的模块作为测试对象的。

### 6.2.1单元测试

单元测试也称模块测试或程序测试，单元测试是对每个模块单独进行的，验证模块接口与设计说明书是否一致，对模块的所有主要处理路径进行测试且与预期的结构进行对照，还要对所有错误处理路径进行测试。对源码进行审查，对照设计说明书，表态地检查源程序是否符合功能的逻辑要求，是进行单元测试前的重要工作工。单元测试一般是由程序员完成，也称程序调试。

### 6.2.2组合测试

组合测试也称集成测试或子系统测试，通常采用自顶向下测试和自底向上测试两种测试方法。组合测试的对象是指已经通过单元测试的模块，不是对零散模块进行单个测试，而是用系统化的方法装配和测试软件系统，是一个严格的过程，必须认真地进行，其计划的产生和单元模块测试的完成日期要协调起来，这种测试应在系统目标机上进行，造成系统应用的环境条件，除了开发部分项目负责人参加以外，还应该有相应系统的用户参加，给评审员进行演示。

### 6.2.3确认测试

确认测试是对通过组合测试的软件进行的，这些软件已经存于系统目标设备的介质上，确认测试的目的是表明软件是可以工作的，并且符合“软件需求说明书”中规定的全部功能和性能要求。确认测试是按照这些要求定出的“确认测试计划”进行的。测试工作是由一个独立的组织进行，而且测试要从用户的角度出发。

### 6.2.4系统测试

系统测试是对整体性能的测试，主要解决各子系统之间的数据通信和数据共享问题以及检测系统是否达到用户的实际要求，系统测试的依据是系统分析报告。系统测试应在系统的整个范围内进行，这种测试不只对软件进行，而是对构成系统的硬、软件一起进行。系统测试与建构同时进行或略慢。系统测试需要确认从头到尾的功能正常才算完成，应当尽量避免系统测试延到项目末尾进行。

### 6.2.5用户验收测试

在系统测试完成后，进行用户的验收测试，它是用户在实际应用环境中所进行的真实数据测试。

## 6.3测试方法

一般来说，对程序测试有两种测试方法：如果已经知道了软件系统应具有的功能，可通过测试来对每个功能是否都能正常使用；如果知道程序的内部工作过程，可以通过测试来检测程序内部是否按照规格说明书的规定正常进行。前一种方法称为黑盒测试，后一种方法称为白盒子测试。黑盒测试又称为功能测试，白盒子测试又称为结构测试。

这次毕业设计并不是一个具有实际意义的大系统，所以没有完全按照上述的测试方法进行系统的测试，但经过一些简单的步骤的测试，证明本应用具有一定的稳定性。

# 第七章 总 结

首先，我选择了本课题是因为这是一个我没有接触过的领域，在我们的毕业设计可选题目中，多数是图书管理系统，酒店管理系统之类的课题，而这些多多少少在之前的校内实习和大作业中使用过，而对于OA系统的设计从来都没有接触过。而我又想能通过毕业设计增长更多的对软件技术的了解，抱着对OA系统设计的好奇心，所以选择了OA系统的设计作为我毕业论文的课题。

在设计过程中，我并没有采用J2EE或者ASP.NET等企业级WEB开发框架作为主要实现方式。在最近一些年来，随着iPhone手机、Android手机的兴起，人们对移动互联网的需求也越来越大，认识也越来越广泛，而近些年来的手机硬件的发展更是日新月异。而这些原因加在一起，就形成了一股企业级应用移动平台话的趋势。

在这股趋势中，iOS平台与Android平台上的开发则是中流砥柱般的推进力量。相对与iOS平台较高的开发门槛（只能使用Mac OS X平台作为研发操作系统，需要购买应用程序开发包的使用权限等)，Android平台则更加大众化，较便宜的价格，较易配置的开发环境，这些都是我选择Android平台进行本应用开发的重要原因。

除了上面这些原因之外，作为Android平台主要的开发语言，Java这门语言也是我所掌握的编程语言，之前在大作业中，例如中间件课程的Java 远程方法调用 （RMC），还有JavaWeb编程课程中的J2EE项目设计，都使用过Java这门语言的一些方面。Java是一门应用广泛的语言，在不同的领域有不同的应用，

不论是服务器，客户端，WebApplet，还是分布式计算，都有他的身影。但是万变不离其宗，Java的基本语法和一些基本的数据结构是不会变的，再者像Android SDK 这种Google为开发者封装的开发包，都会有非常详细的文档来说明各个函数以及对象、模块的使用方法以及继承关系。综上，了解JavaSE的基本构造，再加上对Android开发者文档的仔细阅读，就拥有了Android开发的基础。

在之前的程序设计或者大作业中，大多时候都是设计一个程序的一个小模块，或者通过团队的合作来完成一个小项目的建立。但是作为毕业设计，是一个单人进行开发的过程，而代码量相对与以往的项目来讲有了很大的增长。而在开发的时候，如果面对问题，也是只能通过互联网和开发者文档来寻找答案，虽然对于学习速度来说，是一个比较慢的学习过程，但是参考文档来学习一门编程语言或者是技术无疑是最好的，也是最扎实的。

通过这次的设计，我对Android的设计有了更大的兴趣，我已经初步尝试了一些简单的开发，使用了一些Android基础的模块，例如Activity、Service、Provider、Broadcast Receiver、Content Provider，也对Java进行了部分的深入探索，例如使用了JNI(Java Native Interface)来加快程序的执行速度，用多线程来保证应用程序界面的流畅以及对用户操作的响应速度。

总之，我在本次毕业设计中学到了很多的东西，它使我更加熟悉了一门有用的编程语言，也使我了解了开发的主要步骤和流程，了解怎样去分析问题、解决问题，这对我将来的工作和生活都有很大帮助！

# 参考文献（References）

[1] 杜春华 《什么是OA系统》 价值中国 (http://www.chinavalue.net/Biz/Article/2009-3-2/162334.html)

[2] 童虎科技 《OA系统的发展方向》 （http://www.toohuu.com/release/article1317.htm）

[3] 陈立伟等. 精通Java手机应用与应用程序设计[M]. 北京：中国青年电子出版社，2005

[4] 米川英树. J2ME MIDP手机应用程序设计[M]. 北京：中国铁道出版社，2005

[5] Martin J. Wells(美). J2ME应用编程[M]. 北京：清华大学出版社，2005

[6] 李振鹏等. J2ME手机应用开发技术详解 [M]. 北京：清华大学出版社，2006

[7] 黄聪明. Java移动通信程序设计--J2ME MIDP[M]. 北京：清华大学出版社，2002

# 致 谢

这几个月我收获颇多，但也有不少教训。计算机科学是一门比较宽广的学科，涉及许多领域，应该大量翻阅书籍资料，借鉴别人的心得体会做到事半功倍。程序的开发要养成良好的习惯，将工程分为几个功能明确的模块，逐层分解。遇到难题要虚心请教，做到取长补短，避免钻牛角尖。

毕业设计是对我们四年所学知识的一次全面的检查，经过这段时间的设计，是我们对以前所学的知识的巩固和理解，让我学会怎样把理论联系到实践中去，更使我明白了一个道理，做项目要多看、多想、多做，使自己能把所学的知识灵活运用到计算机中去，更好的完成工作。

我要衷心感谢学校吴金辉导师在我的设计过程中给予我的极大帮助，使我能够及时、顺利地完成此次的毕业设计；他认真负责的工作态度，一直激励着我的学习和生活。同时，也感谢所有在学校帮助过我的老师和同学；感谢父母对我求学的支持；最后感谢各位专家导师对本论文的审阅！

附件：源代码