**学位论文原创性声明**

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名：

年 月 日

**学位论文版权使用授权书**

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于1、保密囗，在 年解密后适用本授权书

2、不保密囗 。

（请在以上相应方框内打“√”）

作者签名： 年 月 日

导师签名： 年 月 日

**本科生毕业设计（论文）任务书**

**学生姓名** 蔡立维 **专业班级** 电信0801

**指导教师** 陈德军 **工作单位** 信息工程学院

**设计（论文）题目**：基于Android平台的简易即时通信系统的设计与实现

**设计（论文）主要内容：**

熟悉运用Android SDK及其开发工具，在Eclipse集成开发环境中设计实现一个Android下平台的即时通讯系统客户端，它能够使用户在Android手机客户端登录，并解析openfire服务器用户信息，完成在不同平台的客户端即时通信。

**要求完成的主要任务：**

1、熟悉掌握Java的基本知识，学会运用Eclipse进行Android环境搭建，以及windows下的openfire服务器的搭建；

2、掌握用Eclipse进行Android平台的UI基本设计、HTTP网络协议及XML文件解析等关键技术的实现，深刻理解XMPP（可扩展消息处理现场协议）协议；

3、学习使用MVC模式来构建系统的客户端，并掌握这种程序设计模式，分析系统需求，设计出基于Android的简易即时通信系统，并利用Android SDK等予以实现，并对系统进行测试；

4、完成毕业论文的撰写，不少于12000字，阅读并翻译与课题相关的英文资料，不少于20000字符，参考文献不少于15篇，其中英文参考文献不少于2篇，完成的设计图纸不少于12幅。

**必读参考资料：**

[1] (美)埃克尔(Eckel,B.) 著；陈昊鹏译．JAVA编程思想(第四版) ．北京：机械工业出版社，2007.6．

[2] (美)弗里曼(Freeman，E.)等著，O’Reilly Taiwan公司译，UMLChina改编．Head First设计模式(中文版) ．北京：中国电力出版社，2007．

[3] (印)克曼特内尼(Komatinei，S.), (美)麦克莱恩(MacLean，D.) ,(美)哈希米(Hashimi，S.)著；杨越译．精通Android 3．北京：人民邮电出版社，2011.11(2012.2重印)．

[4] 柯元丹著．Android内核剖析．北京：电子工业出版社，2011.9．

**指导教师签名** **系主任签名**

**院长签名（章）**

**武汉理工大学**

**本科学生毕业设计（论文）开题报告**

|  |
| --- |
| 1、目的及意义（含国内外的研究现状分析）  即时通讯是一种终端服务，允许两个人或者多个人使用网路可以跨平台地即时地传递文字信息，文本文件，语音和视频进行交流。自从即时通讯产生以来，这种以网络为基础、与其他在线用户的交互信息的实时方式，已及方便快捷的特点，受到了对消息反馈即时性要求很高的行业和机构的青睐。  随着网络技术的迅猛发展，信息科技水平的不断提高，移动终端技术逐渐增强，各种智能操作平台的出现，为即时通信系统移动化发展提供了硬件基础；移动设备中开放性的Andriod平台，随着3G网络发展迅猛占领了市场，开发Android平台下的软件成为了迎合行业发展的一种趋势。Android 是Google 推出的一个基于Linux 内核的手机操作系统， 包含底层框架， 中间件和开发环境的一套平台系统。它支持流行的组件式程序设计，具有针对性移动设备的Dalvik 虚拟机。浏览器核心是基于开源的WebKit 引擎， 对基于html的各种语言具有良好的支持。图像引擎使用基于openGLES1. 0 的三维图形库。又因为Google 与运营商、设备制造商、开发商和其它第三方结成深层次的合作伙伴关系，希望通过建立标准化、开放式的移动电话软件平台，在移动产业内形成一个开放式的生态系统。Android 平台正在被越来越多的开发者和使用者所接受。  目前，国内外对在Android平台下开发应用程序的热情很高，开发Android平台下的即时通信软件也层出不穷，其中涌现了许多有价值的开源社区以及一些第三方做的框架，例如： beem-project.com，Android Asmack等等，这对于完成Android平台下的即时通信软件有着阶段性的贡献，还有许许多多Android爱好者在一些论坛和博客中，例如最有名的CSDN，共同交流和完善中这个Android即时通信系统，给用户的体验也越来越好；这也吸引了越来越多的人投身到对Android软件开发的研究当中，因为它是开源的，因为它有许许多多的志同道合的人能够进行交流，并且从中获取经验和进步；深深地被吸引后，我也怀着对Android平台学习的热情，看中这个行业的发展，想在这个领域有所发展，于是我选择了开源项目比较丰富，不管是对于企业还是个人来说，实用性比较强的即时通信系统的研究；希望通过对此研究过后，能够完成对Android平台有个整体的学习和掌握，掌握开发Android程序的流程，Android平台下的程序设计模式以及对Android程序框架，内核有个基本认识，为以后更深入的研究打好基础。  我选择的设计项目是“基于Android平台的简易即时通信系统的设计与实现”，目的在于充分运用自己学过的知识，和锻炼自己综合学习的能力，为以后做更深入的研究、发展和就业创造良好的条件。 |
| 2、基本内容和技术方案  设计的构想是利用Eclipse开发工具开发实现在Android平台下的即时通讯系统的客户端，它能够完成获取opnfire服务器上的用户设置信息，好友列表，好友群组等信息，然后实现Android平台下的客户端与Android平台下的客户端，与windows下的客户端，与linux下的客户端，甚至与Web下的客户端的互联互通，进行方便、迅捷地收发即时消息。随后随着技术的完善和改进还可以在该系统上实现图片，语音的传输和接收功能，以及其他一些功能，这样可以使该系统功能更加多样化，并且具有跨平台的特点。  通讯系统架构采用C/S模式，即客户端／服务器的体系结构。客户端是基于Android平台进行开发，通过wifi 网络与Internet建立连接，通过服务器实现Android客户端之间的即时通讯。客户端负责初始化通信过程，进行即时通讯时，由客户端负责向服务器端发起连接请求。服务器端采用开源的Openfire 作为服务器，允许多个客户端同时登录，并且同时连接到一个服务器上。服务器对每个客户端进行认证，对认证通过的客户端创建会话，实现客户端与客户端之间的通讯。  系统的客户端定位于Android平台的手机，采用XMPP 协议作为即时通讯协议，XMPP是Jabber的扩展，它为Jabber协议制定了一整套的架构和扩展方面的规范，用于支持IM。XMPP通过Jabber来传递状态信息和实现分布的信息表示。根据协议规则，采用Android平台提供的XML解析包对XML进行解析。根据Activity模块运行于主线程的特点，采用多线程技术来解决系统通信问题。系统的用户信息和聊天信息存储在Android平台自身所带的SQLite数据库中。  Android平台下即时通信客户端的技术设计方案：（1）、设计出适合系统的UI界面，这个只是系统构建的第一步，UI界面的布局主要采用xml文件来描述实现；（2）、构建系统需要的数据库，用SQLite存放消息记录，用getSharedPreferences（）来保存用户的配置文件，并存放在手机内存卡上。（3）、对用户进行分组，设置讨论群相关功能，可以利用Asmark开源项目的API函数予以实现；（4）、用户实时状态监控，代理服务器配置等等；可以参照开源项目Beem来构建。我的这个项目可能还有很多细节的关键技术没有考虑到，但都会严格按照MVC的设计模式来完成。 |
| 3、进度安排  在综合考虑设计任务之后，设计进度安排如下：  第1-2周：查阅相关文献资料，明确研究内容，了解研究所需掌握的知识范围，完成开题报告。  第3-6周：学习Java、Android等相关基础知识，初步完成系统的设计流程。  第7-8周：分析基于Android平台的即时通信系统的MVC开发模式的过程。  第9-10周：掌握运用该系统开发相关的软件并分析相关源码，编写系统代码。  第10-11周：编写系统代码并予以实现，进行相关性能的测试。  第12-14周：完成并修改毕业论文。  第15周：完成论文答辩，提交资料。 |
| 4、指导教师意见  指导教师签名：  年 月 日 |

目 录

[摘 要 I](#_Toc325810624)

[Abstract II](#_Toc325810625)

[1 绪 论 1](#_Toc325810626)

[1.1选题背景与意义 1](#_Toc325810627)

[1.2 主要内容 2](#_Toc325810628)

[1.3 本章小结 3](#_Toc325810629)

[2 系统相关知识介绍 4](#_Toc325810630)

[2.1 JAVA基础介绍 4](#_Toc325810631)

[2.2面向对象的设计模式 5](#_Toc325810632)

[2.2.1策略模式 5](#_Toc325810633)

[2.2.2观察者模式 6](#_Toc325810634)

[2.2.3工厂模式 7](#_Toc325810635)

[2.2.4 MVC模式 8](#_Toc325810636)

[2.3 Linux简单基础 8](#_Toc325810637)

[2.4 XMPP/Jabber协议 9](#_Toc325810638)

[2.4.1 XMPP协议分析 9](#_Toc325810639)

[2.4.2体系结构 10](#_Toc325810640)

[2.4.3消息格式 10](#_Toc325810641)

[2.4.4 XMPP协议簇 12](#_Toc325810642)

[2.5 openfire服务器 12](#_Toc325810643)

[3 Android平台结构 13](#_Toc325810644)

[3.1 Android特征 13](#_Toc325810645)

[3.2 Android系统架构 14](#_Toc325810646)

[3.3 Android应用组件 14](#_Toc325810647)

[4 即时通讯系统的设计 17](#_Toc325810648)

[4.1 服务器端处理流程 17](#_Toc325810649)

[4.2 系统客户端的设计 18](#_Toc325810650)

[4.2.1 用户界面布局设计 18](#_Toc325810651)

[4.2.2 数据存储模块设计 19](#_Toc325810652)

[4.2.3 即时通讯模块设计 20](#_Toc325810653)

[5 即时通讯系统的实现 21](#_Toc325810654)

[5. 1 用户界面布局实现 21](#_Toc325810655)

[5.2 数据存储模块实现 23](#_Toc325810656)

[5. 3 即时通讯模块实现 24](#_Toc325810657)

[5. 4 文件结构及函数功能 25](#_Toc325810658)

[6 系统部署与测试 28](#_Toc325810659)

[6.1系统部署 28](#_Toc325810660)

[6.2 系统的功能测试 28](#_Toc325810661)

[6.3 本章小结 33](#_Toc325810662)

[7系统总结与展望 34](#_Toc325810663)

[参考文献 35](#_Toc325810664)

[附 录 36](#_Toc325810665)

[致 谢 38](#_Toc325810666)

摘 要

即时通信IM（Instant Messageing）是目前Internet上最为流行的通讯方式，而各种各样的即时通信软件也层出不穷；服务提供上也提供了越来越丰富的通信服务功能。随着互联网的发展，即时通信的运用日益广泛，即时通信软件业方兴未艾。

本设计利用Jabber/XMPP的体系结构，构建了一个基于XMPP协议的即时通信系统，包括即时通信系统的客户端和服务器。其中服务器采用开源的Jabber服务器Openfire，客户端基于XMPP核心及扩展协议，利用Beem的开发包Asmack进行研究开发。设计了一个与Openfire互联通信的客户端系统，实现与客户端的文字实时数据通信等功能。

本设计采用了常见的MVC设计模式，对系统进行了分层建模，从HTTP传输，XMPP协议解析到客户端的各个模块部分都进行分析设计，提出了系统的体系结构和整体构架设计方案，阐述了系统实现应用的关键技术，建立了可扩展的会话模型，采用松散耦合的方式设计，实现了Android平台上基于XMPP协议的移动即时通信系统。

**关键字**：即时通信系统；Android平台；XMPP协议；JAVA语言；MVC设计模式

Abstract

Instant Messging is currently the most popular way to communicate on the Internet, by the way various Instant Messaging software have been continuously appearing; service provider offers more and more communication services nowadays. As the development of the Internet, Instant Messaging will perform a wider use and is growing up day after day.

This design builds an instant messaging system including the server and client based on XMPP protocol, using the Jabber/XMPP system architecture. It uses the open-source Jabber server Openfire for the server, researches and develops the client system based on the XMPP core and the extensive protocol, using Beem’s developed package Asmack. The client can connect to Openfire , and communicates with others in text and other real time data.

This design uses a common MVC design pattern,  a hierarchical modeling system, from the HTTP transmission, XMPP protocol analysis to each module of client to analyze, design of the proposed architecture of the system and the overall structure design, introduce key technologies of the application establish a scalable model of conversation, use design methods of loosely coupled, realize instant messaging system of Android platform-based using Jabber protocol.

**Key Word:** Instant messaging system；Android platform；XMPP protocol；Java language；MVC design pattern

1 绪 论

这一章主要介绍了选择课题的时代背景，研究课题的目的和意义，以及有关该课题的研究现状和在该课题中的工作侧重点。

# 1.1选题背景与意义

纵观通信行业的发展，我们可以看到，1844年，莫尔斯创造了人类的第一份电报，开启了人类对通信技术探索的大门；虽然在其后的发展历程中，有过短暂的休眠期，直到1973年4月的一天，世界上产生了第一步移动电话。从那以后，人类的通信行业有进入了活跃期，移动终端从大转头变为手掌心大小，功能从单一的通话功能到集成了MP3\MP4\照相 \GPRS\收音机\手机游戏等等各种功能的智能机，通讯技术从模拟手机时代，到GSM时代，到2.5G时代和今天的3G时代，以及未来的4G时代，可谓发展是非常迅猛的。 随着通讯行业的发展和人们现代生活工作节奏的日趋加快，移动通讯产品和服务的品种越来越多，更新换代速度越来越快，手机市场争夺日趋激烈和残酷。自2000年5月，国际电信联盟公布了第三代移动通讯标准，分别是美国CDMA2000，欧洲WCDMA，中国TD-SCDMA，成为3G时代最为主流的三大技术，还有后面加入的WiMAX的第四个3G标准，3G(3ul一generation)是第三代移动通讯技术的简称，是支持高速数据传输的蜂窝移动通讯技术。全球范围内移动通信发展迅速，国内电信市场竞争日益加剧，行业发展面临着新的机遇和挑战，为了以发展3G为契机，合理配置现有电信网络资源，实现全业务经营，形成适度、健康的市场竞争格局， 2008年5月，工信部、国家发改委、财政部联合发布《关于深化电信体制改革的通告》，中国电信收购中国联通CDMA网，中国联通和中国网通合并，中国卫通的电信业务并入中国电信，中国铁通并入中国移动，经过重组合并，国内电信运营商由六家变为三家。并在当年12月，国务院常务委员会开始启动3G牌照发放，中国电信经营CDMA2000网，中国联通经营WCDMA网，中国移动则是TD-SCDMA网，至此，中国的通讯行业进入了全面发展3G网络的时代。

发展普及3G网络需要载体，于是很自然就出现了3G手机；3G手机是指可以使无线通讯技术与全球互联网等多媒体通讯技术相结合的新型移动通讯系统，在当今21世纪，随着移动通讯的高速数据传输能力越来越强、网络和用户对移动终端的需求越来越高，3G手机如雨后春笋，层出不穷，并且衍生了许许多多当今主流的手机操作系统。

首屈一指的就是苹果的iPhone，它是年轻人的挚爱，2007年6月在美国上市，将移动电话、多点触控、电子邮件、搜索、网页浏览和地图导航功能完美的融为一体。但iPhone是闭源的，苹果公司牢牢控制着iPhone的系统软件和硬件标准，一家独大，开发人员只能接触到iPhone的应用层开发，不能深入到系统层面。

诺基亚曾是移动通信的全球领先者，全球手机出货量第一，旗下symbian操作系统在过去的几年内，获得飞速发展，但碍于其内部架构不合适3G时代人们对于移动手机的需求，未来有没落的趋势。诺基亚和英特尔合作强强联合开发了MeeGo开源的手机操作系统，但后来诺基亚和英特尔都放弃了MeeGo，诺基亚和微软合作，携手WindowsMobile，英特尔与三星合作，开发新的Tizen系统。

微软的 windowsMobile是闭源的，对很多开发人员来说，同样束手束脚。

市场迫切需要一个开发性很强的手机操作系统平台。随着第三代数字通讯技术获得普遍接受，Google于2005年收购”Android.Inc”后，继续进行对Android操作系统开发，并于2007年11月推出了专为移动设备设计的软件平台“Android操作系统”。

Android操作系统是完全开源免费的，它包括操作系统内核、中间件、一些非常关键和普遍的平台应用程序。Android的操作系统是将优秀的Linux再进一步优化，使之适应于移动平台应用，应用层采用当今最为流行的Java编程语言，它允许厂商和个人，对Android操作系统源码进行修改、个性化定制开发。Android操作系统平台是开源的，这促进了技术创新，降低了移动开发成木，促进了移动通讯领域的高速发展。

据有关资料显示，我国使用手机上网的人数规模达约三亿。使用移动终端手机上网的所有应用程序中，手机的即时通信软件是使用率最高的应用程序，高达67.7%。究其原因是，在电话、电子邮件、即时通讯等通讯方式中，即时通讯的实时性是最强、最为快捷方便、效果最好的一种方式。即时通讯可以降低交流的成本、提高沟通的效率，避免在打电话上耗费大量时间，使人与人之间跨越时空进行即时交流通讯，真正实现沟通无界。由此可见智能手机终端上即时通讯软件在移动互联网中扮演着重要角色。因此，研究即时通讯软件的设计与实现是非常重要的。

# 1.2 主要内容

在Android手机如此流行的潮流推动下，本文主要介绍了实现一个Android平台的即时通信系统的简易设计，系统主要是基于XMPP协议来完成的，实现各种类型的客户端之间的通讯功能，提供好友的实时状态，实现移动终端设备与PC端的互通，使用户可以通过手机移动终端的即时通信系统的客户端随时随地与他人进行即时消息通信。

Android是基于Linux操作系统和Java面向对象编程语言的智能手机操作系统平台，XMPP是以Jabber协议为基础的，Jabber协议时即时通讯中最常用的开放式协议，于是本文的主要内容包括：

1. 介绍Android平台下的即时通信系统的设计框架；
2. 介绍Linux系统和Android平台的相关知识；
3. 介绍XMPP协议的内容和该协议的解析与运用；
4. 阐述整个系统的设计与实现过程，以及整个系统的部署与测试；

# 1.3 本章小结

绪论部分，主要针对课题的背景和意义，即时通信技术的发展现状，以及论文要阐述的主要观点，进行了简要的介绍。接下来，将对Android内核Linux的特点、系统整体架构、Android应用层，以及系统相关知识进行介绍，然后分别阐述即时通信系统的设计与实现，系统部署与测试等。

2 系统相关知识介绍

这一章主要介绍了在系统中运用到的相关基础知识，包括JAVA相关基础、Linux相关、XMPP协议等基础知识。

# 2.1 JAVA基础介绍

Android上的应用程序大多是通过JAVA语言实现的，因此熟悉JAVA基础知识十分有必要，可以说系统设计能够进行下去的最最基础的条件，这里我将简要介绍JAVA基础的整体知识框架。

1、对象的初始化

（1）非静态对象的初始化 是在在创建对象时，对象所在类的所有数据成员会首先进行初始化， 在所有类成员初始化完成之后，才调用本类的构造方法创建对象， 构造方法的作用就是初始化。

（2） 静态对象的初始化 是在程序中主类的静态变量会在main方法执行前初始化。不仅第一次创建对象时，类中的所有静态变量都初始化，并且第一次访问某类（注意此时 未创建此类对象）的静态对象时，所有的静态变量也要按它们在类中的顺序初始化。

2、继承时，对象的初始化过程

（1）主类的超类由高到低按顺序初始化静态成员，无论静态成员是否为private。

（2）主类静态成员的初始化。

（3）主类的超类由高到低进行默认构造方法的调用。注意，在调用每一个超类的默认构造 方法前，先进行对此超类进行非静态对象的初始化。

（4）主类非静态成员的初始化。

（5）调用主类的构造方法。

3、关于构造方法

（1） 类可以没有构造方法，但如果有多个构造方法，就应该要有默认的构造方法，否则在继承此类时，需要在子类中显式调用父类的某一个非默认的构造方法了。

（2）在一个构造方法中，只能调用一次其他的构造方法，并且调用构造方法的语句必须是 第一条语句。

4、有关public、private和protected

（1） 无public修饰的类，可以被其他类访问的条件是：a、两个类在同一文件中，b、两个类 在同一文件夹中，c、两个类在同一软件包中。

（2）protected：继承类和同一软件包的类可访问。

（3）如果构造方法为private，那么在其他类中不能创建该类的对象。

5、抽象类

（1）抽象类不能创建对象。

（2）如果一个类中一个方法为抽象方法，则这个类必须为abstract抽象类。

（3）继承抽象类的类在类中必须实现抽象类中的抽象方法。

（4）抽象类中可以有抽象方法，也可有非抽象方法。 抽象方法不能为private。

（5）间接继承抽象类的类可以不给出抽象方法的定义。

6、final关键字

（1）一个对象是常量，不代表不能转变对象的成员，仍可以对其成员进行操作。

（2）常量在使用前必须赋值，但除了在声明的同时初始化外，就只能在构造方法中初始化 。

（3）final修饰的方法不能被重置（在子类中不能出现同名方法）。

（4）如果声明一个类为final，则所有的方法均为final，无论其是否被final修饰，但数据 成员可为final也可不是。

7、接口interface （用implements来实现接口）

（1）接口中的所有数据均为 static和final即静态常量。尽管可以不用这两个关键字修饰 ，但必须给常量赋初值。

（2）接口中的方法均为public，在实现接口类中，实现方法必须可public关键字。

（3）如果使用public来修饰接口，则接口必须与文件名相同。

8、多重继承

（1）一个类继承了一个类和接口，那么必须将类写在前面，接口写在后面，接口之间用逗 号分隔。

（2）接口之间可多重继承，注意使用关键字extends。

（3）一个类虽只实现了一个接口，但不仅要实现这个接口的所有方法，还要实现这个接口 继承的接口的方法，接口中的所有方法均须在类中实现。

# 2.2面向对象的设计模式

当熟悉面向对象的语言后，但要想通过这门语言来设计实现一个系统时，我们往往会惊慌失措，因为我们不知道该如何入手，应该遵循什么样的原则，应用什么样的框架来设计比较合理，所以在代码编写之前，我们不得不要做一些思想工作，那就是思考研究系统应该应用哪种设计模式。这里我将简单介绍常用的几种设计模式及规则。

## 2.2.1策略模式

1. 角色组成

抽象策略角色： 策略类，通常由一个接口或者抽象类实现。

具体策略角色：包装了相关的算法和行为。

环境角色：持有一个策略类的引用，最终给[客户端](http://baike.baidu.com/view/930.htm)调用。

1. 应用场景

a、 多个类只区别在表现行为不同，可以使用Strategy模式，在运行时动态选择具体要执行的行为。

b、 需要在不同情况下使用不同的策略(算法)，或者策略还可能在未来用其它方式来实现。

c、 对客户隐藏具体策略(算法)的实现细节，彼此完全独立。

1. 优缺点

优点：①、提供了一种替代继承的方法，而且既保持了继承的优点(代码重用)还比继承更灵活(算法独立，可以任意扩展)。②、避免程序中使用多重条件转移语句，使系统更灵活，并易于扩展。③、遵守大部分GRASP原则和常用设计原则，高内聚、低偶合。

缺点：①、因为每个具体策略类都会产生一个新类，所以会增加系统需要维护的类的数量。

## 2.2.2观察者模式

1. 表现形式

该模式必须包含两个角色：观察者和被观察对象，观察者和被观察者之间存在“观察”的逻辑关联，当被观察者发生改变的时候，观察者就会观察到这样的变化，并且做出相应的响应。如果在用户界面、业务数据之间使用这样的观察过程，可以确保界面和数据之间划清界限，假定应用程序的需求发生变化，需要修改界面的表现，只需要重新构建一个用户界面，业务数据不需要发生变化。

1. 实现过程

实现观察者模式有很多形式，比较直观的一种是使用一种“注册—通知—撤销注册”的形式。下面的三个图详细的描述了这样一种过程：

1. 观察者（Observer）将自己注册到被观察对象（Subject）中，被观察对象将观察者存放在一个容器（Container）里。

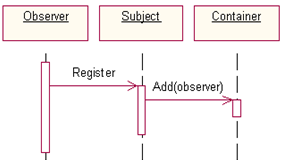


图2-1 观察者模式实现过程UML图（一）

1. 被观察对象发生了某种变化（如图中的SomeChange），从容器中得到所有注册过的观察者，将变化通知观察者。

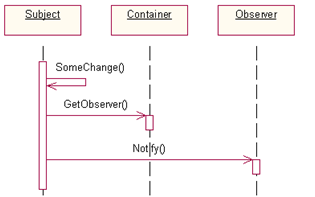


图2-2观察者模式实现过程UML图（二）

1. 观察者告诉被观察者要撤销观察， 被观察者从容器中将观察者去除。

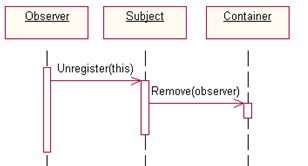


图2-3观察者模式实现过程UML图（三）

观察者将自己注册到被观察者的容器中时，被观察者不应该过问观察者的具体类型，而是应该使用观察者的接口。这样的优点是：假定程序中还有别的观察者，那么只要这个观察者也是相同的接口实现即可。一个被观察者可以对应多个观察者，当被观察者发生变化的时候，他可以将消息一一通知给所有的观察者。基于接口，而不是具体的实现——这一点为程序提供了更大的灵活性。

## 2.2.3工厂模式

其定义为：提供创建对象的接口。而工厂模式的原理可以通俗得比喻：就是具体事情做得越多，越容易范错误；相反,官做得越高，说出的话越抽象越笼统，犯错误可能性就越少。工厂模式在实际的代码编写中运用得很多，主要还是要遵循面向对象编程的原则：封装与分派。

## 2.2.4 MVC模式

MVC是指模型(Model)、视图(View)和控制(Controller)， Android的应用程序通常也有界面（View）的设计，模型（Model）或者服务(Controller)的建立，连接Model与View之间沟通的桥梁的控制器（Controller）。下面将简单介绍MVC的特点。

MVC是一个设计模式，它强制性的使应用程序的输入、处理和输出分开。使用MVC应用程序被分成三个核心部件：模型、视图、控制器。它们各自处理自己的任务。

视图是用户看到并与之交互的界面，在Android中主要体现就是对Layout的XML文件的编写，于是为了使视图美观，得到用户的认可成了一项重要的工程，许多公司还成立了相应的部门招纳相应的工程师：UI设计工程师。但在视图中并没有做一些实质性的处理，只是提供了一些控制接口，供控制器和模型来操作，作为一种输出数据并允许用户操纵的方式。

模型表示企业数据和业务规则，这是Web应用的表现，对于Android应用程序来说，其表现形式也大同小异，模型通常拥有最多的处理任务，被模型返回的数据多是中立的，就是说模型与数据格式无关，这样模型通常得到很好的封装，这样一个模型能为多个视图提供数据。由于应有于模型的代码只需要写一次就可以被多个视图重中，所以减少了代码的重复性。

控制器接受用户的输入并调用模型与视图去完成用户的需求，在Android应用中这个过程一般在Activity中完成，它通常是接受请求并决定调用哪个模型国建去处理请求，然后再确定用哪个视图显示返回的数据。

或许有人会问：把MVC设计模式运用在Android应用程序设计中，会不会很牵强，因为它只是在Web应用设计中提炼出来的一种设计模式？我的答案是否定的，因为对于面向对象的应用程序设计而言，MVC的设计模式只是指明了设计中的总体文件布局与任务分配，使在设计过程中会有明确的引导，而且这种设计模式也体现了面向对象设计原则：封装与分派。

通过对设计模式的简单介绍，我们知道了，每一种设计模式都是一种设计思想，也是一种解决方案 ，在设计中起指导作用，并且运用过合适的设计模式编写出来的代码复用性强，可读写也高；模式还为我们提供了观察问题、设计过程和面向对象的更高层次的视角，这将使我们从“过早处理细节”的桎梏中解放出来。

最后，我觉得：好的设计模式有助于提高思考层次。

# 2.3 Linux简单基础

因为我是在基于Linux的Ubuntu系统下进行代码编写，系统部署的，所以有必要简单介绍一下有关的Linux基础知识。

（1）Linux目录结构

/bin是系统基本命令存放目录(/usr/bin)；/boot 是linux的内核及引导系统程序文件存放目录(如：vmlinuz、initrd.img) ；/dev是设备文件存储目录，如声卡、光驱；/ect  存放系统设置文件(如用户账号密码、服务器配置文件等)；/home是普通用户家目录，默认存放目录；/lib  库文件存放目录；/lost+found是在ext2或ext3文件系统中，当系统以外崩溃或机器意外关机，而产生一些文件碎片放在这里；/media是即插即用型存储设备的挂载点自动在这个目录下创建。；/mnt是存放挂载存储设备的挂载目录，如cdrom等目录；/opt是表示可选的意思，有些软件包也会被安装在此，也就是自定义软件包；/proc是操作系统运行时，进程(正在运行的程序)信息及内核信息(比如CPU、硬盘分区、内存信息等)存放在此。/proc目录是伪装的文件系统proc的挂载目录，proc并不是真正的文件系统。；/root是linux超级权限用户root的根目录；/sbin大多是涉及系统管理的命令的存放，只有超级权限用户root才可执行命令存放，普通用户无权限执行此目录下的命令；/tmp是临时文件目录，有时用户运行程序的时候，会产生临时文件。

（2）常用命令及作用

Man：当我们不清楚某个Linux命令的作用和用法时，可以使用man command进行查询。

Find：用于查找某个文件或者文件夹，比如:$find .-name “\*.java” 该命令用于查找当前目录下的扩展名为Java的所有文件，find命令后面的“.”代表当前目录，\*为通配符，代表任何名称。

Grep：为正则表达式的匹配命令，该命令用于字符串匹配。

Xargs：是一个标识，代表了上一个命令的执行结果，并作为下一个命令的参数。

Cat：用于连接文件内容并在Terminal中输入文件内容，该命令后面如果只有一个文件名称，则仅输出该文件内容。

Chmod：用于设置访问者对某文件的访问权限。访问权限分为读（r）、写（w）和执行（x）。

Ps：用于列出当前运行的所有进程，kill 用于杀死某个进程。

Export：用于将某个环境变量值的作用域设为全局范围。

# 2.4 XMPP/Jabber协议

## 2.4.1 XMPP协议分析

目前主流的IM协议有四种，包括：可扩展消息与存在协议(XMPP)、空间和即时信息协议(PRIM)、即时信息和空间协议(lMPP)、针对即时信息和出席扩展的会话发起协议SIP(SIMPLE)。在这四种主流协议中，XMPP协议具有很好的灵活性和扩展性。

由于XMPP协议是高度自由、高度开放的协议，易于掌握和理解，而且在服务器、客户端、插件、源代码库等方面，有多种实现。任何遵循XMPP协议的软件，可以实现相互通讯。加之Google公司开发的Gtalk软件也是基于XMPP协议的及时通讯软件，同时很多软件巨头也对XMPP协议进行支持，促进了XMPP协议的推广和普及。因此，研究XMPP协议及基于此的即时通讯软件的设计与实现，有深远的意义。

## 2.4.2体系结构

XMPP的基木网络结构中定义了三个角色:Client、Server和Gateway。

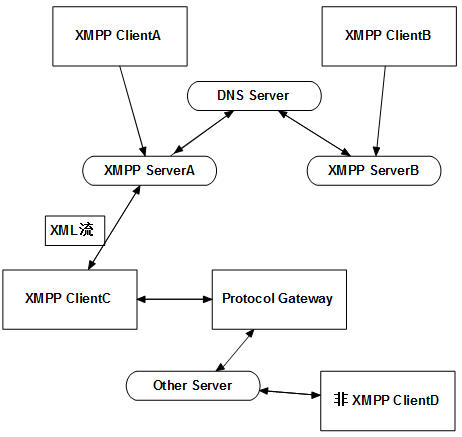


图2-4 XMPP的体系结构

如图2-4所示， 如果XMPP ClientA和XMPP ClientB进行通讯，首先XMPP ClientA登录XMPP ServerA进行身份验证，身份验证通过后，XMPP ServerA搜索本地用户是否存在XMPP ClientB，如果存在，则建立连接，如果不存在，XMPP ServerA则通过DNS Server查询到XMPP ServerB，进而XMPP ServerA与XMPP ServerB交换数据，XMPP ServerB查找木地服务器中是否存在XMPP ClientB，存在，则将XMPP ClientA和XMPP ClientB建立连接，如果不存在，则继续上面的步骤。

如果XMPP ClientC与非XMPP ClientD进行通讯，首先XMPPClientC登录Protocol Gateway进行身份验证，验证成功后，Protocol Gateway与Other Server进行信息交换，由OtherServer查找非XMPP ClientD，查找成功后，则建立XMPP ClientC和非XMPP ClientD之间通讯。

## 2.4.3消息格式

XMPP协议的通信消息是基于XML的，XML数据流中包含了会话过程中传递的所有数据，在XML的数据流中，每个节(Stan-za)都是通过命名空间确定的结构化数据块，开发人员可以通过扩展命名空间来增加功能。

XMPP协议包含三个顶层xml标签。例如：

<stream>

<presence from=‘test1@jabber.com/contaet’

to=’test2@jabber.com/contact’>

<status>online</status>

</presence>

<message to=’test3@jabber.org/contact’ type = ‘chat’>

<body>今天天气很好</body>

</message>

<iq from=’test3@jabber.com/contact’ id=’1364564666’ type=’result’ ></iq>

</stream>

其中，<presence>标签表示用户的状态，即是否在线。当用户的状态发生改

变时，就会在stream中插入相应的presence标签，以表示用户当前的状态。

表 2-1 用户状态<presence>

|  |  |
| --- | --- |
| 标签 | 描述 |
| Presence | 表示包含用户在线状态的信息 |
| To | 接受信息的用户JID |
| From | 发送信息的用户JID |
| Show | 用户的在线状态 |
| Status | 包换用户自定义的状态信息 |
| Type | Type=“unavailable”表示不在线  Type=“ subseribe”发送方定制哪些用户在线  Type=“unsubscribe”为定制用户在线  Type=“prob”查看用户是否在线的请求  Type=“error”出现错误 |

<message>标签表示两个用户之间发送的消息。有一些相关的属性:to、from、

type等表明发送方、接受方、类型，还可以包括一些子元素。

表2-2 消息标签<message>

|  |  |
| --- | --- |
| 标签 | 描述 |
| Message | 表示包含聊天的内容 |
| To | 发送信息的目标JID |
| From | 发送该信息的用户JID |
| Type | 表示聊天的类型  Type=“chat”表示一対一的聊天  Type=“Group chat”表示群聊  Type=“error”表示出现错误 |
| Body | 聊天的详细内容 |
| Error | 错误信息 |

<iq>标签表示一次查询会话。包括发出信息/查询操作的实体和接收查询/设置的实体，以及信息/查询操作的类型。信息/查询主要的用处是获取或设置用户的基本信息，比如用户的名字、电子邮件、地址信息等等。

表2-3 查询标签<iq>

|  |  |
| --- | --- |
| 标签 | 描述 |
| Iq | 表示一次查询会话 |
| To | 接受信息的用户JID |
| From | 发送信息的用户JID |
| Query | 表示要查询的会话内容 |
| Type | 表示查询对话的类型  Type=“get”表示获取信息  Type=“result”表示返回的查询结果  Type=“set”表示用户在服务器设置基本信息 |

## 2.4.4 XMPP协议簇

XMPP协议簇一共分为四层，最底层的是TCP协议，向上依次分别是TLS协议(即传输层安全协议)、SASL协议(简单认证和安全层协议)、XMPP协议。TLS由操作系统提供，用于在两个通信应用程序之间提供保密性和数据完整性。SASL是由应用层提供，而XMPP则是应用层本身。XMPP协议采用SASL协议作为身份认证协议。基于XMPP协议的SASL协议需提供如下三个方面信息:服务名(XMPP)、初始序列、交换序列和安全层协商。其层的顺序为TCP一TLS一SASL一>XMPP。

# 2.5 openfire服务器

openfire(原名Wildfire或者JiveMessenger)是由Java语言编写的、基于XMPP协议的服务器，具有跨平台能力，获得了Apache2.0许可证。

openfire是基于XMPP协议的IM的服务器端的一个实现，两个用户想要进行通讯，首先要连接到Openfire。服务器获取一些连接信息，然后两个用户之间才能建立连接。当两个用户之间建立连接后，用户可以采用点对点的方式来发送接收消息。

选择openfire作为即时通讯服务器是因为其自身的存在诸多优点:openfire服务器拥有基于Web的管理界面，方便管理员的管理操作。实现了插件的机制，方便扩展。软件开发前期不用过度考虑用户需求，可以以插件的形式，随时添加功能。Openfire是一个开源的服务器，软件开发人员可按照自己的需求进行定制化更改。提供了对SSL/TLS的支持。安装方便，用户界面有好。支持多种数据库连接，用于存储信息和用户详细内容。openfire是使用纯Java语言开发的、平台独立的服务器，并且都是开源的。

3 Android平台结构

本章将分三个方面对Android平台进行分析阐述，这三个方面包括Android

特征、Android系统架构和Android应用组件。

# 3.1 Android特征

Android应用框架包含很多应用程序组件，这些组件在开发中可以被反复利且软件开发人员可以基于己有的组件开发出属于自己的特色组件，这样可约开发成本，提高软件开发效率。

Dalvik虚拟机是Android所采用的Java虚拟机，区别于甲骨文的Java虚拟它在性能上更加出色，占用资源更少，运行速度快，更适合移动设备平台。

内嵌的浏览器基于WebKit引擎，WebKit在开源界非常流行，它是一个优秀浏览器网页排版引擎。Andrnid的浏览器速度更快，因为Android优化了Java的脚本编辑器，当浏览器调用包含Java的web页面时，与其他浏览器相比，页面速度有明显提升。

Android系统包含OpenGLES的3D图形功能，这意味着Andrnid手机支持程序开发，这无疑对游戏开发商有很大的吸引力，Android上的游戏画面体验会更上一层楼。

多媒体技术是基于Open Core Platform来实现的，它是PacketVideo公司开多媒体软件平台，该软件平台功能非常强大。Android的多媒体能够播放大的音视频格式文件，支持的编码包括MP3、3GPP框架等等。通过Open Core多媒体框架，程序员可以开发出属于自己的多媒体应用，比如多人视频会议应用程序等。

Android操作系统采用SQLite数据库作为标准数据库，SQLite是一款开源据库，轻巧方便，资源占用率低，性能好，速度快，非常适用于移动嵌入式。所有Android应用程序的数据库都存放在/data/package/databases/下面。

Androld系统为传感器提供了丰富的API支持，现在支持的传感器种类包括加速度传感器(aeeelerometer)、距离传感器(Proximity)、陀螺仪(gyroseope)、环境光照传感器(light)、磁力传感器(magnetiofield)、方向传感器(orientation)和温度传感器(temperature)等。这些传感器为应用程序和游戏开发人员提供了创造各种有趣应用的前提条件，现在加速度传感器己经普遍应用于游戏中了。

语音搜索为用户提供了将语音数据转换成文本数据或手机操作命令的功能，如果用户想要使用Google搜索引擎搜索，只需要对着手机说出想要搜索的内容，即可实现手机网页搜索。

# 3.2 Android系统架构

Android系统采用分层的架构实现，从下到上依次是Linux内核(Linux Kernel)、Libraries、C/C++函数库、dalvik虚拟机、应用框架(Application Framework)以及最上层的应用程序(Applications)。如图所示。

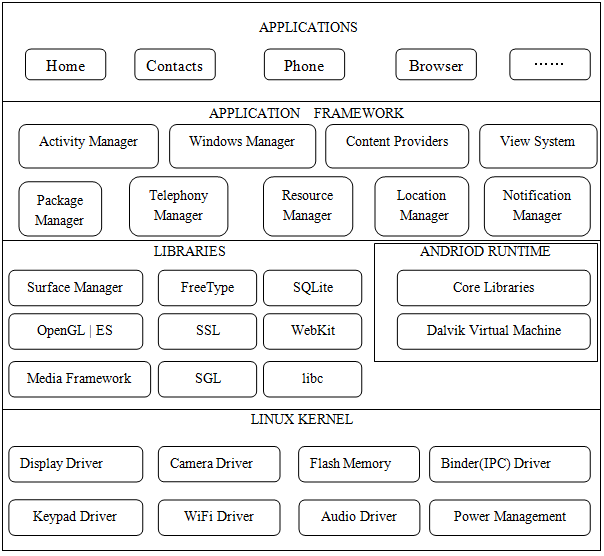


图3-1 Android系统框架图

# 3.3 Android应用组件

Android应用程序由一些零散的有联系的组件组成，通过一个工程manifest绑定在一起。在manifest中，描述了每一个组件以及组件的作用，其中有6个组件，它们是Android应用程序的基石：

（1）Activity 组件

Activity是Android应用程序中最为基本也是最为重要的组件之一，在Android应用程序中，手机屏幕上显示的每个窗口都是一个Activity，用户主要通过Activity组件与应用程序交互。软件开发人员可以在Activity的onCreate()方法中调用setContentView(R.id.main)方法来关联个性化用户界面，R.id.main是main.xml文件的标识符，main.xml是用户界面的配置文件。同时在这个方法中初始化程序中需要的实例对象。关闭窗口时，程序在onDestroyO方法中销毁这个窗口中的实例对象，并释放所使用的资源。



图3-2 Activity的生命周期

一个Activity的整个生命周期是从调用onCreate()方法开始，到调用onDestroy()方法结束。Activity在onCreate()方法中进行所有数据的初始化工作，然后在onDestroy()方法中，清除不再使用的对象实例。比如在onCreate()中使用AysncTask进行异步数据请求或下载数据等操作，在onDestroyO方法中使用AsyneTask的cancle()方法取消异步任务。一个Activity从onStart()方法开始，到onStop()方法之间时，始终处于可见状态，当Activity不可见时，调用onStop()方法，当Activity再次可见时，系统会调用Activity的onRestart()方法，用户可以看到屏幕上显示之前的界面。一个Activity能够在前台显示的生命周期是从onResume()方法开始，直到调用onPauseO方法为止。在这个过程中，Activity始终是在所有Activity的最前面，用户可以与之进行交互。

（2）Intent组件

Intent即“意图”的意思，Android使用Intent这个组件实现Activity与Activity之间的导航功能。描述Intent的结构中，有两个重要的组成部分:动作与动作对应的数据。系统提供的动作包括:MAIN、PICK、VIEW、CALL等，动作对应的数据以URI的形式表示。

通过解析各种Intent，可以实现从一个屏幕跳转到其他屏幕。程序中采用startActivity(Intent intent)方法实现Activity之间的跳转。Android系统会在所有存在的应用程序中定义的InientFilter中进行查找，一旦Intent匹配成功，新的Activity就开始运行了。

（3）Service组件

Service是“服务”的意思，顾名思义，Service在Android中是一种生命周期很长的组件，而且是不存在界面的应用程序组件，它在应用程序后台运行。最常见的例子如:多媒体播放器程序，当用户切换到其他应用程序时，它可以在后台运行并能正常播放声音；或者如文件下载程序，它可以运行在后台执行文件的下载操作，而不会阻塞用户界面。

Activity使用Context.startService()方法来启动一个service，还可以通过Context.bindServiee()方法与其他Serviee相连接，当连接成功时，还可以使用Service提供的接口与之进行通信。

（4）ContentProvider组件

Android提供了ContentProvider组件，这个组件的作用是:凡是实现ContentProvider接口的应用程序，都可以将自己的数据提供给其他应用程序共享。可以说ContentProvider是所有应用程序之间数据共享的纽带。Android系统同时提供了视频、音频、图像、个人基本信息等常用的ContentProvider。不同于普通获取数据的方式，ContentProvider采用URI的形式为外界提供数据。而具体获取哪个数据库中的数据，由Android操作系统的底层程序实现，外界不需要知道存储的具体细节就可以获取想要的数据。

（5）Broadcast Receivers组件

　　Intent广播的“消费者”。通过创建和注册一个Broadcast Receiver，应用程序可以监听符合特定条件的广播的Intent。Broadcast Receiver 会自动的启动你的Android应用程序去响应新来的Intent。Broadcast Receiver是事件驱动程序的理想手段。

　　Broadcase Receiver不执行任何任务，仅仅是接受并响应广播通知的一类组件。大部分广播通知是由系统产生的，例如改变时区，电池电量低，用户选择了一幅图片或者用户改变了语言首选项。应用程序同样也可以发送广播通知，例如通知其他应用程序某些数据已经被下载到设备上可以使用。一个应用程序可以包含任意数量的Broadcase Reveiver来响应它认为很重要的通知。所有的Broadcast Receiver都扩展自类BroadcastReceiver。Broadcast Receiver不包含任何用户界面。然而它们可以启动一个Activity以响应接受到的信息，或者通过NotificationManager通知用户。可以通过多种方式使用户知道有新的通知产生：闪动背景灯、震动设备、发出声音等等。通常程序会在状态栏上放置一个持久的图标，用户可以打开这个图标并读取通知信息。

（6）Notifications组件

　　用户通知的框架。Notification用来在不需要焦点或不中断它们当前Activity的情况下提示用户。它们是Service或 Broadcast Receiver获得用户注意的首选方式。例如，当设备收到文本信息或外部来电时，它通过闪光，发声，显示图标或显示对话框信息来提醒你。

4 即时通讯系统的设计

# 4.1 服务器端处理流程

本系统采用Openfire服务器作为即时通讯服务器，消息报接收处理流程如图4-l所示。



图4-1 服务器端消息包接收处理流程

服务器端的消息处理流程的经过：首先，ConneetionHandler接收Socket数据，然后StandzaHandler对socket数据进行解析，解析为Element对象，最后，根据Element对象的不同，分别分配给相应的路由 (router)去处理。IQRouter、MessageRouter和PresenceRouter分别处理IQ、Message、和Presence。

# 4.2 系统客户端的设计

系统客户端采用了MVC的设计模式，将视图层、控制逻辑层以及数据模型层进行分离，实现客户端程序的高内聚，低祸合，提高代码的重用性，降低系统的维护成本。Google设计的Android手机操作系统，本身也采用Mvc的设计模式，这样一来，为设计和开发Android应用程序提供了良好的基础。在Android程序中，视图层的布局和规划，可以通过XML文件进行配置编码，当然也可以通过Java硬编码的方式进行布局，但这不是Android推荐的布局方式。系统客户端从安全性、稳定性、易用性以及代码复用的角度出发，功能分模块，代码分层设计。考虑到客户端软件的应用人群广泛，系统提供了对于国际化支持，包括中文简体、中文繁体、英文等。

## 4.2.1 用户界面布局设计

Google的Android操作系统中，系统客户端的应用程序界面是通过XML文件或Java硬编码的方式布局的，第一种方式是使用最为广泛的。Android提供了丰富的布局控件，如布局控件LineaarLayout、RelativeLayout、FrameLayout等， Widget在Android中是指应用程序窗口小部件，控件包括Button、EditText、ProgressBar等。窗口布局文件和具体的业务逻辑处理，是在Activity的OnCreate()方法中耦合的，使用setContentView()加载相应的布局XML文件，即可显示用户自定义的交互界面。

系统客户端的界面设计包括用户注册，用户登录，配置设置，联系人列表，个人信息及状态设置等。

用户运行客户端时，首先在内部控制模块判断，是否存在了用户配置好了的信息，如果存在，则直接调用配置过得信息进行网络连接；如果没有，则会进入向导模式，这个时候供用户选择，如果用户已经有账号了，则可以直接到登录界面填写登录信息进行登录，如果用户没有账号，则选择注册，到注册界面进行注册；登录成功后，就会显示联系人列表，不成功时，可以点击手动设置来配置一些登录信息和网络设置信息等；由联系人列表视图可以进行好友管理，分组管理，个人信息管理，状态信息管理等等，如图4-2所示。



图4-2 联系人功能列表视图

用户界面布局设计如图4-3所示。



图4-3 客户端界面设计流程

## 4.2.2 数据存储模块设计

Android系统中，提供了五种数据存储方式，分别是SharedPreferences存储数据、文件存储数据、SQLite数据库存储数据、ContentProvider存储数据和网络存储数据。本系统客户端采用SQLite数据库存储登录用户信息、登录用户的好友列表信息以及登录用户和好友的聊天记录，用户的配置信息用SharedPreferences保存在本地的手机内存卡的/data/data/<package-name>/shared\_prefs中，保存的内容格式为XML文件，例如

<?xml version='1.0' encoding='utf-8' standalone='yes' ?>

<map>

<string name="proxy\_type">HTTP</string>

<string name="connection\_resource">Beem</string>

<boolean name="use\_auto\_away" value="true" />

<string name="settings\_key\_xmpp\_port">5222</string>

<string name="settings\_key\_reco\_delay">10</string>

<string name="account\_username">clw712@t.sina.com.cn</string>

<string name="connection\_priority">0</string>

<string name="account\_password">19880712</string>

</map>

## 4.2.3 即时通讯模块设计

系统客户端通过GPRS、wifi等无线网络到Openfire即时通讯服务器，采用TCP协议，通过Openfire来验证用户身份并与其他客户端连接通讯。虽然TCP是可靠的连接，但是现实中的网络环境异常复杂，所以在连接openfire服务器的过程中，可能存在不同程度的延迟。

用户

登录成功

Openfire服务器

返回好友列表

选择好友发起聊天申请

服务器接受申请，连接到好友

遵循协议，调用Asmack中的API进行消息传送

完成即时通讯过程

结束

图4-4 即时通信模块流程图

5 即时通讯系统的实现

本系统采用C/S(客户端一服务器端)的系统架构，客户端是基于Android手机操作系统开发的即时通讯软件，采用Java编程语言开发和MVC的设计模式。服务器端采用当今流行的开源服务器Openfire和开源数据库MySQL构建。Openfire服务器负责处理客户端软件发送的身份验证请求、连接请求、数据交换等，配合MySQL的数据存储功能，实现Android手机之间，以及其他实现XMPP协议的客户端软件进行即时通讯。

# 5. 1 用户界面布局实现

本系统客户端的用户界面布局文件存放在/res/layout中，Android系统据系统可以根据系统自身的重力感应，进行相应的屏幕适配。当然，如果手机横屏和竖屏的布局存在明显差异，则可以通过在res目录下创建layout-land与layout-port两个文件夹，当手机横屏时，系统会加载res/layout-land目录下的文件，当手机竖屏时竖屏时，系统会自动加载res/layout-port目录下的文件。本系统由于横屏和竖屏的显示差异不大，所以无论手机是横屏还是竖屏，系统都会加载res/layout下的界面布局文件。用户界面布局文件的目录如图所示。

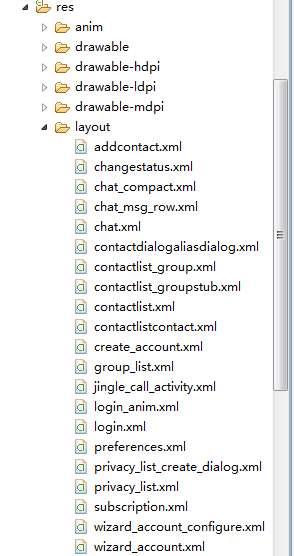


图5-1 布局文件目录



图5-2 用户向导界面

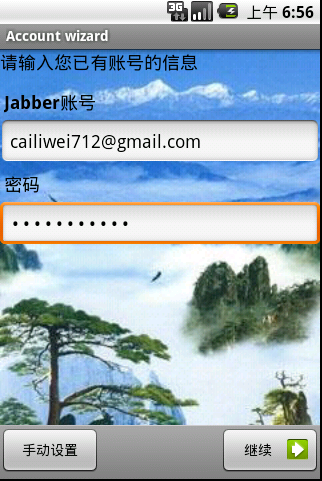


图5-3 用户登录界面

还有许多操作的用户界面，这里就不一一列出了，这些界面的设计都是通过XML文件定义修饰的，并在合适的程序代码中用setContentView(R.layout.*\**)方法来加载定义好了的XML文件，进而实现显示。

# 5.2 数据存储模块实现

本系统客户端需要存储用户登录的基本信息和用户好友的基本信息，以及用

户和好友的聊天记录，分别为保存在本地存储卡的/data/data/包名/shared\_prefs和sdcard的/Android/data/com.beem.project.beem/chat下，用户的聊天记录的目录可以自己设置更改的，如图所示。



图5-4 用户设置界面

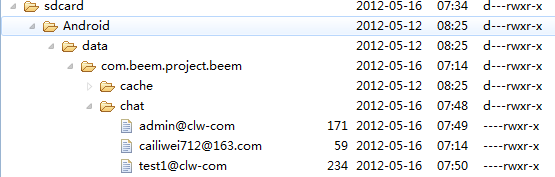


图5-5 聊天记录存放路径

关于所有用户的信息，都是通过openfire服务器存放在MySQL数据库中的，如图所示。



图5-6 openfire服务器上的用户列表信息

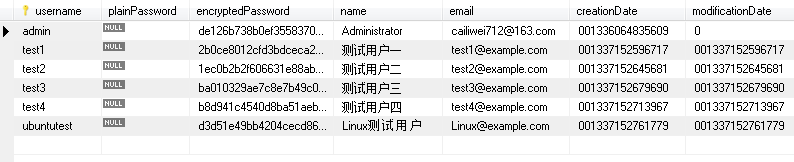


图5-7 MySQL数据库中的用户信息

# 5. 3 即时通讯模块实现

在开发Android应用程序过程中，开发人员必须遵守单线程模型的原则，Android的Ul操作并不是线程安全的，而且这些操作必须在Ul主线程中执行。系统客户端的登录请求、发送数据请求、接收数据的操作，都采用异步任务的方式进行。Android提供了AsyncTask类，实现异步数据请求，不会阻塞ul界面，这样会产生良好的用户体验。

本客户端采用Asmack库提供的接口与openfire服务器进行通讯。Asmack是Smack在Android上的实现，Smack是一个基于Java编程语言的XMPP客户端库。库中的org.jivesoftware.smack.packet包封装了message，presence，和iq的处理类。当然你也可以直接创建和发送信息包。Smack提供简单的机制可以将任意的属性添加到信息包中。每一个属性property存储的是一个键值对(key-value)，开发人员可以将值存入，然后根据key获取value。这个值可以是Java原始数据类型(byte，int，short，long，float，double，boolean)，或者是其他可以序列化的对象(当java对象实现了Serializable接口时，它就是可序列化的)。

# 5. 4 文件结构及函数功能

虽然系统的功能比较简单，但为了使系统更完善，对系统函数和方法分类和打包，为以后的功能拓展做好基础，文件结构如图所示。

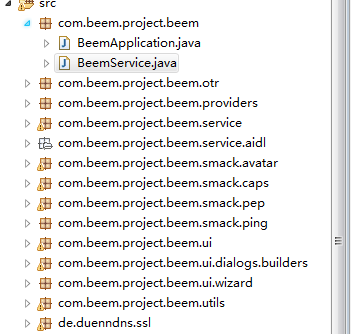


图5-8 项目源码包结构

（1）AIDL接口：com.beem.project.beem.service.aidl

AIDL接口的定义把系统的模块一层一层封装使结构更加清晰；在AIDL中封装Contact、Message、UserInfo等类是因为在Service中如果要传递对象必须是要继承自Parcelable的对象的；在多个类中都用了RomoteCallbackList主要是用于回调在Activity中实现的一些监听器；Avater是指头像信息，PrivacyList是用来定义更多用户状态信息（如定义离开消息：“我吃饭去了”等等）。一些AIDL的实现及描述见下表。

表5-1 主要aidl接口功能说明表

|  |  |  |
| --- | --- | --- |
| Service.aidl包 | aidl对应的实现 | 描述 |
| IBeemConnectionListener.aidl | IXmppConnection中调用 | 添加连接监听的回调接口 |
| IXmppConnection.aidl | XmppConnectionAdapter | 用于xmpp连接处理接口 |
| IXmppFacade.aidl | XmppFacade | 用于xmpp连接管理提供外部调用接口 |
| IChatManager.aidl | BeemChatManager | 用于聊天管理 |
| IChatManagerListener.aidl | IChatManger中调用 | 为聊天加入回调接口 |
| IChat.aidl | ChatAdapter | 封装聊天处理方法 |
| IMessageListener.aidl | IChatManger中调用  IChat中调用 | 提供消息处理回调接口 |

（2）service的主要类：XmppConnectionAdapter连接XMPP协议客户端的适配器，此类提供了一些公共方法，见下表：

表5-2 xmpp连接适配器功能说明表

|  |  |
| --- | --- |
| 方法名 | 描述 |
| Connect | Xmpp连接方法 |
| addConnectionListener | 添加xmpp连接监听器 |
| Login | Xmpp登录方法 |
| ConnectAsync | 异步连接 |
| getChatManager | 得到聊天管理器 |
| getRoster | 得到用户花名册 |

XmppFacade：xmpp连接管理外部接口。

表5-3 XmppFacade功能说明表

|  |  |
| --- | --- |
| 方法名 | 描述 |
| Viod changeStatus(int status,String msg) | 修改状态信息 |
| xmppConnectionAdapter createConnection() | 创建xmpp连接 |
| Void disconnect() | 销毁xmpp连接 |
| IChatManager getChatManeger() | 获得聊天管理器 |
| Void sendPresencePacket(PresenceAdapter presence) | 发送状态包（离线unavailable，上线available） |
| Boolean publishAvatar（Uri avatarUri） | 发布自己的头像 |
| void disableAvatarPublishing() | 隐藏自己的头像 |
| UserInfo getUserInfo() | 获取当前用户的信息 |

BeemChatManager：聊天管理，用于管理聊天对象的创建，消息的监听。

表5-4 聊天管理说明表

|  |  |
| --- | --- |
| 方法名 | 描述 |
| IChat createChat(Contact contact, IMessageListener listener) | 通过联系人创建对话 |
| IChat createChat(String jid, IMessageListener listener) | 通过JID创建对话 |
| void destroyChat(IChat chat) | 销毁对话 |
| void deleteChatNotification | 删除消息 |
| private ChatAdapter getChat(Chat chat) | 得到会话适配器 |
| addChatCreationListener | 添加会话监听 |
| ChatAdapter getChat(Contact contact) | 通过联系人得到会话适配器 |
| List<Contact> getOpenedChatList() | 得到联系人列表 |
| void removeChatCreationListener(  IChatManagerListener listener) | 删除会话监听器 |

ChatAdapter：主要处理对话的内容，状态等信息。

表5-5 聊天适配器功能说明表

|  |  |
| --- | --- |
| void addMessageListener(  IMessageListener listen) | 当创建新聊天对象创建时加入消息监听 |
| void removeMessageListener(  IMessageListener listen ) | 移除消息监听器，用于聊天对象注销 |
| String getState() | 获取当前聊天对象的状态 |
| void setOpen(boolean isOpen) | 如果打开为当前聊天窗口，消息条数清空 |
| void addNewMsgNum() | 只要聊天对象存在计数器都要+1 |
| int getNewMsgNum() | 计算上次窗体调用setOpen(true)到当前时间所到的新消息条数 |

6 系统部署与测试

# 6.1系统部署

首先，运行openfire服务器，在openfire服务器上新增几个测试用户，在windows系统下运行Spark，并登陆一个用户等待测试。

其次，把在eclipse中编写好的代码，编译成Apk文件，启动Android模拟器，Android模拟器的版本用的是Android2.1的，将编译好的Akp文件安装进模拟器中，等待下一步的功能测试。

最后，做个系统部署小结：整个系统部署算比较简单的，但是如果你是首次下载或安装eclipse，Android SDK eclipse插件ADT，还有openfire服务器的话，会碰见很多问题，不过网上大多应该有安装配置教程供参考的；我遇见身边许多同学或者朋友用eclipse配置Android开发环境都花了两三天都没有搞定，如果有经验的就会知道，配置环境的时候，要注意路径，版本匹配等等。配置openfire服务器是要注意一些插件包文件是否存在或者匹配，或者放置的路径是否正确等。如果详细介绍应该又会占很大一篇幅了，所以就不一一介绍了。

# 6.2 系统的功能测试

首先，这是模拟器进行简单测试，登录测试如图。



图6-1 登录账号输入

登录成功后会出现联系人界面，如图所示。



图6-2 联系人列表

可以浏览本地图库中的图片来更改用户头像，以及更新用户状态等，如下图所示。



图6-3 头像设置和状态修改

可以通过设置添加联系人，并且对联系取别名和分组，如图所示。



图6-4 添加联系人

长按联系人列表中的联系人会出现管理该联系人的对话框，如图所示。



图6-5 好友管理列表

在手机或者模拟器上手触“列表键”，选择“设置”，会出现如下图所示的画面，这里提供了“好友列表”、“历史记录”、“通知设置”、“用户设置”、“网络选项”等的相关功能设置。



图6-6 设置相关功能

即时通信的消息发送接收测试如图所示。



图6-7 模拟器的即时消息测试

将软件安装分别在真机斐讯810和华为C8650手机上，登录gmail账号，即时消息测试如图所示。

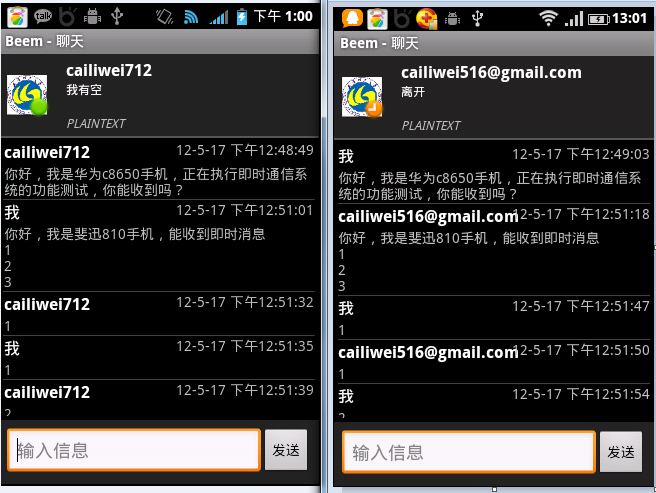


图6-8 用360手机助手看到的效果图



图6-9 用手机照相的效果图

# 6.3 本章小结

本章主要简单讲述了系统部署及测试的相关过程，经过测试验证，基于Android的即时通信系统的客户端设计的功能有好友管理，状态修改，头像设置（部分XMPP协议支持的系统暂不支持，如新浪博客，其JID为t.sina.com.cn）,好友分组，即时聊天，图片浏览等。经过测试，此客户端支持任意支持XMPP协议的即时聊天系统的登录，例如谷歌talk系统，其JID格式为example@gmail.com，新浪博客即时聊天系统，其JID格式为example@t.sina.com.cn等等支持XMPP协议的即时通信系统。经过测试和修正也发现了不少问题，聊天场景的变换，历史消息记录的查询等功能还存在BUG，偶尔显示正常，偶尔不正常，还有就是用户注册也存在问题，有待进一步改进。

7系统总结与展望

本系统是在使用开源Openfire作为即时通讯服务器的基础上，使用Eclipse开发环境和Java软件开发语言，开发即时通信手机客户端软件。软件基于XMPP协议，通过openfire即时通信服务器通信，在Android2.1模拟器、华为C8650和斐讯810手机上安装测试，成功传输文字消息。

在系统设计和开发过程中，主要经历了以下几个阶段：

（1）对系统相关基础知识的学习过程，有java语言程序设计的学习，面向对象的设计模式的学习，UML画图的相关学习，linux系统的相关知识的补充学习，最后是Android系统内核，Android应用框架以及即时通讯XMPP协议的学习等等。这是一个漫长的阶段。

（2）对系统进行试探性的分析与整体架构的设计，画出整体框架图，用Eclipse进行相关页面的UI设计。

（3）在已有系统框架的基础上，细化系统相关的服务程序，进行分层设计与实现；并在UI界面的基础上将服务程序，多线程处理程序等等加入到系统中。

（4）完成系统的代码编写，进行系统的相关功能测试。

经过这个毕业设计的磨练，对移动通讯行业有了更深的了解，也有了更多的知识来充实自己的大脑。知道了即时消息已经成为语音及文本的在线即时通信的主要技术，它的特点决定了其在未来移动商务、在线协作及Internet应用中将扮演更为重要的角色。而随着移动通信技术的迅猛发展，中国3G网络投入商业运营，为移动即时通信提供了更强有力的支撑平台，实时多媒体技术也会移植到移动即时通信应用中来。集成多媒体的应用有着更强的吸引力，为用户提供更多个性化的服务，将成为未来移动即时通信发展的一个必然趋势。

即时通信系统能够实现其基本功能，这里要感谢陈德军老师对项目的进展的合理管控，基本思路的指点以及方向的把握，以使我在学习过程少走一些不必要的弯路，陈老师的思维方式和项目开发经验很值得敬仰和学习。这里也要感谢网络这个平台，让我在对系统关键技术的进展愁眉苦脸的时候，找到了解决问题的方案，这里有许多论坛，论坛里有许多有智慧的朋友，都是值得学习和借鉴的。

参考文献

[1] (美)埃克尔(Eckel,B.) 著；陈昊鹏译．JAVA编程思想(第四版) ．北京：机械工业出版社，2007．

[2] (美)弗里曼(Freeman，E.)等著，O’Reilly Taiwan公司译，UMLChina改编．Head First设计模式(中文版) ．北京：中国电力出版社，2007．

[3] (印)克曼特内尼(Komatinei，S.), (美)麦克莱恩(MacLean，D.) ,(美)哈希米(Hashimi，S.)著；杨越译．精通Android 3．北京：人民邮电出版社，2011．

[4] Suetterlin P,Thiele O,Knapp H. An OSGi based mobile development overview.ISI.2008.4. 22-26.

[5] Jongbok Byun．Instant Messaging and Presence Technologies for College Campuses．IEEE.

2005.6. 4-13．

[6] Peter Saint-Andre．Streaming XML with Jabber/XMPP.IEEE. 2006.9. 82-89．

[7] Peter S A.XMPP Instant messaging and presence.RFC3921.2004.

[8] 柯元丹著．Android内核剖析．北京：电子工业出版社，2011．

[9] 丁永明．基于Android平台移动学习软件的研究与实现．数字通信世界，2011．

[10] 樊燕红，谭香．基于XMPP即时通信网关应用研究．电子技术应用，2007．

[11] 付莎．基于XMPP即协议企业级IM的研究与实现．西南交通大学，2009．

[12] 郭春雷．基于Jabber协议的手机即时通讯系统的研究与实现．贵州大学，2007．

[13] 马志强．基于Android平台即使通信系统的设计与实现．北京交通大学，2009．

[14] 顾书宽．基于XMPP协议的即使通讯系统的设计与实现．中国地质大学（北京），2011.

[15] 招俏春．基于XMPP协议的即时通讯系统的研究．华南师范大学，2008．

[16] http://www.igniterealtime.org/builds/smack/docs/latest/javadoc/

[17] http://android-ui-utils.googlecode.com/hg/asset-studio/dist/index.html

[18] http://beem-project.com/

[19] http://www.csdn.net/

[20] http://www.elecfans.com/tongxin/119/20111011220978\_2.html

[21] http://ishare.iask.sina.com.cn/f/18101903.html?retcode=0

[22] http://developer.android.com/guide/index.html

[23] http://www.oschina.net/android

附 录

程序部分代码：

配置文件AndroidManifest.xml:

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

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

package=*"com.beem.project.beem"* android:versionCode=*"12"*

android:versionName=*"0.1"* android:installLocation=*"auto"*>

<application android:label=*"@string/app\_name"*

android:icon=*"@drawable/beem\_launcher\_icon\_silver"* android:theme=*"@style/Theme.BEEM.Default"*

android:name=*".BeemApplication"*>

<activity android:name=*".ui.Login"* android:label=*"@string/app\_name"*

android:launchMode=*"standard"*>

<intent-filter>

<action android:name=*"android.intent.action.MAIN"* />

<category android:name=*"android.intent.category.LAUNCHER"* />

</intent-filter>

</activity>

<activity android:name=*".ui.wizard.Account"* android:label=*"Account wizard"* />

<activity android:name=*".ui.wizard.AccountConfigure"* android:label=*"Account wizard"* />

<activity android:name=*".ui.LoginAnim"* android:label=*"@string/login\_login\_progress"*

android:launchMode=*"singleTop"* android:screenOrientation=*"portrait"* />

<activity android:name=*".ui.Settings"* android:label=*"@string/edit\_settings\_name"* />

<activity android:name=*".ui.Chat"* android:label=*"@string/chat\_name"*

android:launchMode=*"singleTop"* />

<activity android:name=*".ui.ChangeStatus"* android:label=*"@string/ChangeStatusActTitle"*

android:launchMode=*"singleTop"*

android:windowSoftInputMode=*"stateHidden"* />

<activity android:name=*".ui.AddContact"* android:label=*"@string/AddCActTitle"* />

<activity android:name=*".ui.Subscription"* android:label=*"@string/app\_name"* />

<activity android:name=*".ui.CreateAccount"* android:label=*"@string/create\_account\_name"* />

<activity android:name=*".ui.ContactList"* android:label=*"@string/contact\_list\_name"*

android:launchMode=*"singleTask"* />

<activity android:name=*".ui.GroupList"* android:label=*"GroupList"* />

<activity android:name=*".ui.PrivacyList"* android:label=*"@string/privacy\_list\_name"* />

<!-- MemorizingTrustManagerActivity -->

<activity android:name=*"de.duenndns.ssl.MemorizingActivity"* />

<provider android:name=*".providers.AvatarProvider"*

android:authorities=*"com.beem.project.beem.providers.avatarprovider"*

/>

<service android:name=*"BeemService"* android:enabled=*"true"*

android:label=*"Beem Service"* android:permission=*"com.beem.project.beem.BEEM\_SERVICE"*>

<intent-filter>

<action android:name=*"com.beem.project.beem.BeemService"*></action>

</intent-filter>

</service>

</application>

<permission android:permissionGroup=*"android.permission-group.NETWORK"*

android:label=*"BeemService"* android:description=*"@string/BeemServiceDescription"*

android:name=*"com.beem.project.beem.BEEM\_SERVICE"*/>

<uses-permission android:name=*"android.permission.INTERNET"*/>

<uses-permission android:name=*"android.permission.VIBRATE"*/>

<uses-permission android:name=*"android.permission.WRITE\_EXTERNAL\_STORAGE"* />

<uses-permission android:name=*"android.permission.ACCESS\_NETWORK\_STATE"*/>

<uses-permission android:name=*"com.beem.project.beem.BEEM\_SERVICE"*/>

<uses-feature name=*"android.hardware.touchscreen"* required=*"false"* />

<uses-sdk android:minSdkVersion=*"5"* android:targetSdkVersion=*"7"* />

<supports-screens android:largeScreens=*"true"*

android:normalScreens=*"true"* android:smallScreens=*"true"* android:anyDensity=*"true"* />

</manifest>

致 谢

毕业论文即将完成，我的大学生生涯也要告一段落了。大学生四年匆匆就过去了，在学习和工作上走了一些弯路，留下了生活上的一些幸酸，也取得了一些成绩，收获了不少快乐。借这个论文致谢的机会，我想对在四年中帮助过我的人，给我带来快乐的人表示我对他们真诚的感谢。

首先衷心感谢我的所有任课老师们，在四年的学习生涯中给予我丰富的知识和技能，在他们的丰富的文化底蕴的熏陶下，我相信在以后的工作学习中，仍然会受益匪浅。

其次要深深地感谢陈德军老师对项目进展的合理管控，基本思路的指点以及方向的把握，以使我在学习过程少走一些不必要的弯路，陈老师的思维方式和项目开发经验很值得敬仰和学习。

感谢电信0801班的所有同学和我的三位室友，我很高兴认识你们，感谢你们平时对我的帮助、关心，与你们一起渡过的三四年的时光让我终生难忘。

感谢我的家人，他们对我无微不至的关心和支持。

感谢那些在生活中和在网络上的不知道名字的朋友们，谢谢你们在这四年的大学生涯中，给予我的丰富的生活感悟，使我能慢慢成为一个合格的社会人。

蔡立维

2012年5月26日