

Android 即时通信

第 1 章 课程简介.....	3
第 2 章 基本概念和原理.....	3
2.1 常用的网络通信协议.....	3
2.2 TCP、UDP 特点对比.....	4
2.3 TCP 三次握手过程.....	4
2.4 即时通讯形式.....	5
第 3 章 ServerSocket 和 Socket.....	5
3.1 使用 Java 完成简单的 Socket 通信.....	5
第 4 章 XMPP 基础.....	8
4.1 XMPP 简介.....	8
4.1.1 节的共通属性.....	8
4.1.2 iq 节点.....	9
4.1.3 message 节点.....	9
4.1.4 presence 节点.....	10
4.2 XMPP 服务器平台.....	10
4.2.1 案例-数组的基本使用 Openfire 的下载和安装.....	11
4.3 XMPP 客户端平台.....	19
4.3.1 Spark 客户端的下载和安装.....	19
4.3.2 Spark 和 Openfire 通信原理.....	25
第 5 章 使用 ASmack 打造即时通信 App.....	25
5.1 项目简介.....	25
5.2 项目搭建.....	28
5.3 项目实施.....	30
5.3.1 Spark 客户端的下载和安装闪屏界面.....	30
5.3.2 登录.....	32
5.3.3 注册.....	43

5.3.4 主界面.....	50
5.3.5 退出登录.....	54
5.3.6 获取联系人功能.....	56
5.3.7 聊天功能.....	66
5.3.8 消息界面.....	75
5.3.9 启动服务监听消息.....	78

Android 即时通信

◆ 了解即时通信的概念

◆ 了解即时通信的开发

第 1 章 课程简介

即时通讯（Instant Messaging）是目前 Internet 上最为流行的通讯方式，各种各样的即时通讯软件也层出不穷；服务提供商也提供了越来越丰富的通讯服务功能。不容置疑，Internet 已经成为真正的信息高速公路。从实际工程应用角度出发,以计算机网络原理为指导,结合当前网络中的一些常用技术,编程实现基于 C/S 架构的网络聊天工具是切实可行的。

目前，中国市场上的企业级即时通信工具主要包括：信鸽、视高科技的视高可视协同办公平台、263EM、群英 CC2010、通软联合的 GoCom、腾讯公司的 RTX、IBM 的 Lotus Sametime、点击科技的 GKE、中国互联网办公室的 imo、中国移动的企业飞信、华夏易联的 e-Link、擎旗的 UcStar 等。相对于个人即时通信工具而言，企业级即时通信工具更加强调安全性、实用性、稳定性和扩展性。

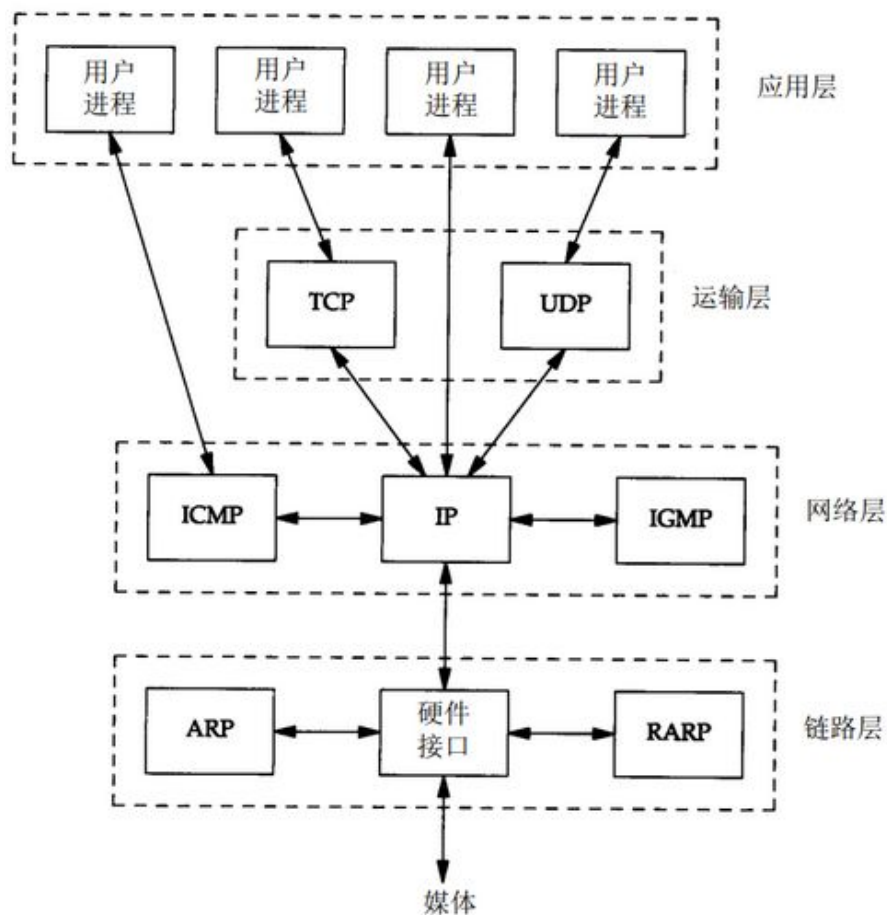
第 2 章 基本概念和原理

2.1 常用的网络通信协议

TCP/IP: Transmission Control Protocol/Internet Protocol 的简写，中译名为传输控制协议/因特网互联协议，又名网络通讯协议，是 Internet 最基本的协议、Internet 国际互联网络的基础，由网络层的 IP 协议和传输层的 TCP 协议组成。TCP/IP 定义了电子设备如何连入因特网，以及数据如何在它们之间传输的标准。协议采用了 4 层的层级结构，每一层都呼叫它的下一层所提供的协议来完成自己的需求。通俗而言：TCP 负责发现传输的问题，一有问题就发出信号，要求重新传输，直到所有数据安全正确地传输到目的地。而 IP 是给因特网的每一台联网设备规定一个地址。

UDP: UDP 协议全称是用户数据报协议，在网络中它与 TCP 协议一样用于处理数据包，是一种无连接的协议。在 OSI 模型中，在第四层——传输层，处于 IP 协议的上一层。UDP 有不提供数据包分组、组装和不能对数据包进行排序的缺点，也就是说，当报文发送之后，是无法得知其是否安全完整到达的。UDP 用来支持那些需要在计算机之间传输数据的网络应用。包括网络视频会议系统在内的众多的客户/服务器模式的网络应用都需要使用 UDP 协议。UDP 协议从问世至今已经被使用了很多年，虽然其最初的光彩已经被一些类似协议所掩盖，但是即使是在今天 UDP 仍然不失为一项非常实用和可行的网络传输层协议。

TCP/IP 协议栈主要分为四层:应用层、传输层、网络层、数据链路层,每层都有相应的协议，如下图：



所谓的协议就是双方进行数据传输的一种格式。

2.2 TCP、UDP 特点对比

TCP 协议是面向连接、保证高可靠性(数据无丢失、数据无失序、数据无错误、数据无重复到达)传输层协议。UDP 协议也是传输层协议，它是无连接，不保证可靠的传输层协议。

java 语言的 8 大基本类型：

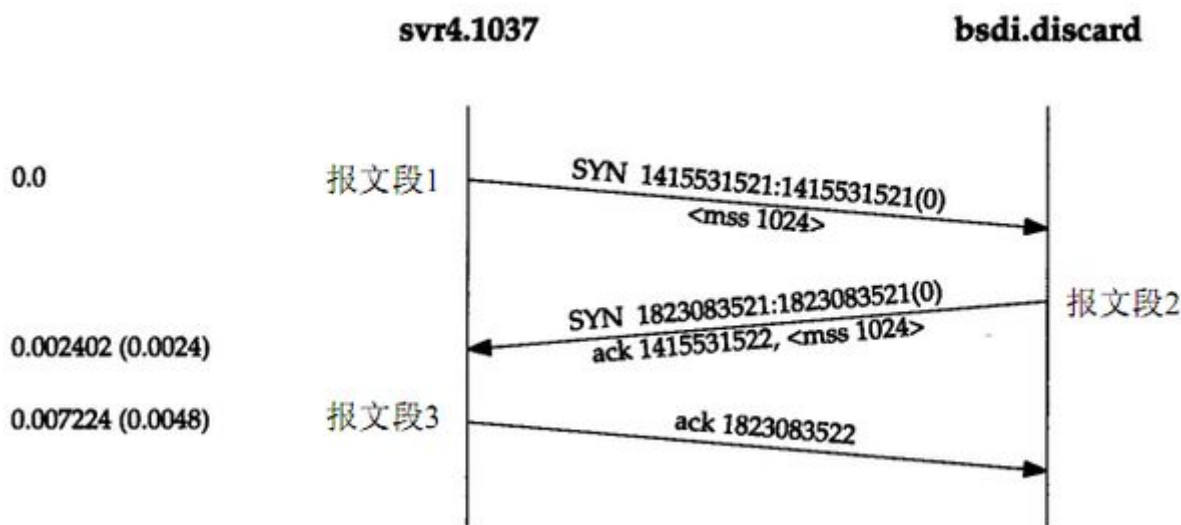
TCP	UDP
面向连接	面向非连接
可靠的连接	不可靠的连接
速度慢	速度快
大文件、重要的数据等	适合小数据、不重要

2.3 TCP 三次握手过程

- 1、请求端(通常称为客户)发送一个 SYN 段指明客户打算连接的服务器的端口，以及初始序号(ISN)
- 2、服务器发回包含服务器的初始序号的 SYN 报文段(报文段 2)作为应答。同时，将确认序号设置为客户的 ISN 加 1 以对客户的 SYN 报文段进行确认。

3、客户必须将确认序号设置为服务器的 ISN 加 1 以对服务器的 SYN 报文段进行确认(报文段 3)
这三个报文段完成连接的建立。这个过程也称为三次握手(three-way handshake)。

上面的过程如下图所示：



2.4 即时通讯形式

1、直接通讯

两个不同客户端之间不经过服务器，直接通过网络进行数据的交互。常用的 p2p 技术就是直接通讯的形式。

2、在线代理通讯

一个客户端发送的消息先发送到服务器，服务器接收到消息后再发送给指定的另外一个客户端。QQ 的消息尤其是离线消息就是同在线代理的方式实现的。

3、离线代理通讯

一个客户端发送消息给服务器，服务器存储在数据库中，当另外一个客户端上线后在发送过去。

4、离线扩展通讯

一个客户端发送消息给服务器，服务器通过邮件、短信等其他形式将消息发送给接收者。

第 3 章 ServerSocket 和 Socket

3.1 使用 Java 完成简单的 Socket 通信

在 Java 中 Socket 可以理解为客户端或者服务器端的一个特殊的对象，这个对象有两个关键的方法，一个是 `getInputStream` 方法，另一个是 `getOutputStream` 方法。`getInputStream` 方法可以得到一个输入流，客户端的 Socket 对象上的 `getInputStream` 方法得到的输入流其实就是从服务器端发回的数据流。`getOutputStream` 方法得到一个输出流，客户端 Socket 对象上的 `getOutputStream` 方法返回的输出流就是将要发送到服务器端的数据流，（其实是一个缓冲区，暂时存储将要发送过去的的数据）。

下面就让我们写一个简单的 Demo 来演示 Socket 是如何使用的。

一、建立服务器类

服务类使用到的核心类的是 `ServerSocket`。这里我们只需要建立一个 Java Project 即可。

```
1. public class IMServer {
2.     private static ServerSocket serverSocket;
3.     private static BufferedReader reader;
4.     public static void main(String[] args) {
5.         try {
6.             serverSocket = new ServerSocket(7788);
7.             /**
8.              * 等待接收客户端连接进来，该方法是线程阻塞的
9.              */
10.            Socket accept = serverSocket.accept();
11.            /**
12.             * 获取输入流，用于接收客户端发来的数据
13.             */
14.            InputStream inputStream = accept.getInputStream();
15.            /**
16.             * 将字节输入流转化为字符输出流
17.             */
18.            reader = new BufferedReader(new InputStreamReader(inputStream));
19.            /**
20.             * 打印数据
21.             */
22.            String tmp = null;
23.            while ((tmp = reader.readLine()) != null) {
24.                System.out.println(tmp);
25.            }
26.        } catch (Exception e) {
27.            e.printStackTrace();
28.        } finally {
29.            try {
30.                if (serverSocket != null) {
31.                    serverSocket.close();
32.                }
33.            } catch (IOException e) {
34.                e.printStackTrace();
35.            }
36.            if (reader != null) {
37.                try {
38.                    reader.close();
39.                } catch (IOException e) {
40.                    e.printStackTrace();
41.                }
            }
        }
    }
}
```

二、建立客户端类

```
1 public class IMClient {
2     private static Socket socket;
```

```
3     private static BufferedWriter writer;
4     /**
5      * @param args
6      */
7     public static void main(String[] args) {
8         try {
9             socket = new Socket("127.0.0.1", 7788);
10            /**
11             * 获取输出流
12             */
13            OutputStream outputStream = socket.getOutputStream();
14            writer = new BufferedWriter(new OutputStreamWriter(outputStream));
15            writer.write("hello wo shi why!" + new Date().getTime());
16            writer.close();
17        } catch (IOException e) {
18            e.printStackTrace();
19        } finally {
20            if (socket != null) {
21                try {
22                    socket.close();
23                } catch (IOException e) {
24                    e.printStackTrace();
25                }
26            }
27            if (writer != null) {
28                try {
29                    writer.close();
30                } catch (IOException e) {
31                    e.printStackTrace();
32                }
33            }
34        }
35    }
36 }
```

在上面的代码中我们仅仅实现了一个最简单的服务器和客户端，服务器启动起来后只能接受到一次消息，然后就关闭了。如果想让服务器一直运行，应该通过死循环来处理不同的发送进来的消息。

第 4 章 XMPP 基础

4.1 XMPP 简介

XMPP（Extensible Messaging and Presence Protocol）是一种基于标准通用标记语言的子集 XML 的协议，它继承了在 XML 环境中灵活的发展性。因此，基于 XMPP 的应用具有超强的可扩展性。经过扩展以后的 XMPP 可以通过发送扩展的信息来处理用户的需求，以及在 XMPP 的顶端建立如内容发布系统和基于地址的服务等应用程序。而且，XMPP 包含了针对服务器端的软件协议，使之能与另一个进行通话，这使得开发者更容易建立客户应用程序或给一个配好系统添加功能。

XMPP（可扩展消息处理现场协议）是基于可扩展标记语言（XML）的协议，它用于即时消息（IM）以及在现场探测。它在促进服务器之间的准即时操作。这个协议可能最终允许因特网用户向因特网上的其他任何人发送即时消息，即使其操作系统和浏览器不同。

XMPP 的前身是 Jabber，一个开源形式组织产生的网络即时通信协议。XMPP 目前被 IETF 国际标准组织完成了标准化工作。标准化的核心结果分为两部分：

XMPP 其实就是用 TCP 传的是 XML 文件流。

下面给大家介绍 XMPP 通信中最核心的三个 XML 节（stanza）。这些节（stanza）有自己的作用和目标，通过组织不同的节（stanza），就能达到我们各种各样的通信目的。

```
1. <stream:stream>
2. <iq type='get' id='roster1'><query xmlns='jabber:iq:roster'/></iq>
3. <message to='william_duan@jabber.org' from='test_account@jabber.org'
4.     type='chat'><body>Hello</body></message>
5. <presence type='unavailable'/>
6. </stream:stream>
```

在上面的 xml 中，我们可以看到一些 XMPP 节（stanza），包括<iq>,<message>以及<presence>。接下来就对这些节（stanza）做一个大致的了解。

4.1.1 节的共通属性

◆ from

表示节（stanza）的发送方，在发送节（stanza）时，一般来说不推荐设定，服务器会自动设定正确的值，如果设定了不正确的值，服务器将会拒收该节（stanza）信息。如果在客户端到服务器端的通信中接收的节（stanza）中没有该属性，会被默认解释为信息是由服务器发出的。如果在服务器到服务器的通信中接收的节（stanza）中没有本属性，则会被解释为一个 error。

◆ to

表示节（stanza）的接收方。如果在客户端到服务器端的通信中没有设置本属性，服务器会默认解释为信息是发给自己的。

◆ type

指定节（stanza）的类型。每种节（stanza）都会有几种可能的设定值。

所有的节（stanza）都会有一个 error 类型，表明这个节（stanza）是一个 error 回应，对这样的节（stanza）信息不需要进行回应。

◆ id

表示一个特定的请求。在<iq>节中，这个属性是必须要指定的，但是在其他两个节（stanza）中是一个可选属性。

4.1.2 iq 节点

iq 节（stanza）主要是用于 Info/Query 模式的消息请求，他和 Http 协议比较相似。可以发出 get 以及 set 请求，就如同 http 中的 GET 以及 POST。iq 节点需要有回应，有 get,set 两种请求以及 result,error 两种回应。

发送查询消息示例：

```
1. <iq from="william_duan@jabber.org/study" type="get" id="roster1">
2. <query xmlns="jabber:iq:roster"/>
3. </iq>
```

上面 xml 的意思是 [william_duan](#) 查询自己的联系人列表。

接收到回应示例：如果请求错误：

```
1. <iq to="william_duan@jabber.org/study" type="error" id="roster1">
2. <query xmlns="jabber:iq:roster"/>
3. <error type="cancel">
4. <feature-not-implemented xmlns="urn:ietf:params:xml:ns:xmpp-stanzas"/>
5. </error>
6. </iq>
```

如果请求成功：

```
1. <iq to="william_duan@jabber.org/study" type="result" id="roster1">
2. <query xmlns="jabber:iq:roster"/>
3. <item jid="account_one@jabber.org" name="one"/>
4. <item jid="account_two@jabber.org" name="two"/>
5. </iq>
```

william 获取到了自己的花名册列表，该列表中有两个好友。

4.1.3 message 节点

正如名字一样，message 节（stanza）用于用户之间传递消息。这消息可以是单纯的聊天信息，也可以某种格式化的信息。message 节点信息是传递之后就被忘记的。当消息被送出之后，发送者是不管这个消息是否已经送出或者什么时候被接收到。通过扩展协议，可以改变这样一种状况。

私人会话示例：

```
1. <message from="william_duan@jabber.org" to="test_account@jabber.org" type="chat">
2. <body>Come on</body>
3. <thread>23sdfewtr234weasdf</thread>
4. </message>
```

群组会话示例：

```
1. <message from="test_account@jabber.org" to="william_duan@jabber.org" type="groupchat">
2. <body>welcome</body>
3. </message>
```

4.1.4 presence 节点

presence 节（stanza）用来控制和表示实体的在线状态，可以展示从离线到在线甚至于离开，不能打扰等复杂状态，另外，还能被用来建立和结束在线状态的订阅。

在线状态示例：

```
1. //设定用户状态为在线
2. <presence/>
3. .....
4. //设定用户状态为离线
5. <presence type="unavailable"/>
6. .....
7. //用于显示用户状态的详细信息。上面的例子表明用户因为 at the ball 在离开状态。
8. <presence>
9. <show>away</show>
10. <status>at the ball</status>
11. </presence>
12. ....
```

<show>标签在 presence 节点中最多出现一次，可以有以下取值：away,chat,dnd,xa。

away: 离线

chat:交谈中

dnd:希望不被打扰

xa:离开一段时间

<status>标签用于显示额外信息。

在线状态预定(presence subscription)：

```
1. <presence
2. from="william_duan@jabber.org"
3. to="test_account@jabber.org"
4. type="subscribe"/>
5.
6. .....
7. <presence
8. from="test_account@jabber.org"
9. to="william_duan@jabber.org"
10. type="subscribed"/>
```

通过上述交互，william_duan 就能看到 test_account 的在线状态，并能接收到 test_account 的在线状态通知了。

4.2 XMPP 服务器平台

我们在学习 JavaWEB 的时候要用到 web 服务器，那么我们就选择了 Tomcat 作为 web 服务器。同样的道理我们要学习即时通信，这整个体系是一个 C/S 架构，Server 端不需要我们编写，那么我们就选择一款市场上免费开源的服务，除了 Openfire 暂无他选，Openfire 是开源免费功能强大的 IM 服务器。为什么选择 Openfire 呢？请往下

看。

Openfire 采用 Java 开发，开源的实时协作（RTC）服务器基于 XMPP（Jabber）协议。Openfire 安装和使用都非常简单，并利用 Web 进行管理。单台服务器可支持上万并发用户。您可以使用它轻易的构建高效率的即时通信服务器。由于是采用开放的 XMPP 协议，您可以使用各种支持 XMPP 协议的 IM 客户端软件登陆服务。

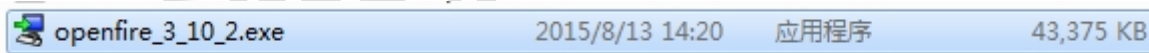
4.2.1 案例-数组的基本使用 Openfire 的下载和安装

下载地址：<http://www.igniterealtime.org/downloads/index.jsp>

The screenshot shows the Openfire 3.10.2 download page. It includes a navigation bar with links like Home, Projects, Downloads, Community, Fans, Support, and About. Below the navigation bar, there's a 'DOWNLOADS' section with a list of links for current releases, source code, and nightly builds. The main content area features the Openfire 3.10.2 release information, including a description of the software as a cross-platform real-time collaboration server based on XMPP. It also provides a 'Choose your platform' section with buttons for Windows, Linux, and Mac. Below this, there's a table listing the download links for the Windows executable (openfire_3_10_2.exe) and the Linux/Mac zip file (openfire_3_10_2.zip), along with their respective sizes and release dates.

Platform	File Name	Description	Release Date	Size
Windows	openfire_3_10_2.exe	Includes Java JRE (recommended)	June 22, 2015	42.36 MB
Linux/Mac	openfire_3_10_2.zip	Does not include Java JRE	June 22, 2015	19.53 MB

我们下载 windows 平台的 openfire_3_10_2.exe 文件。



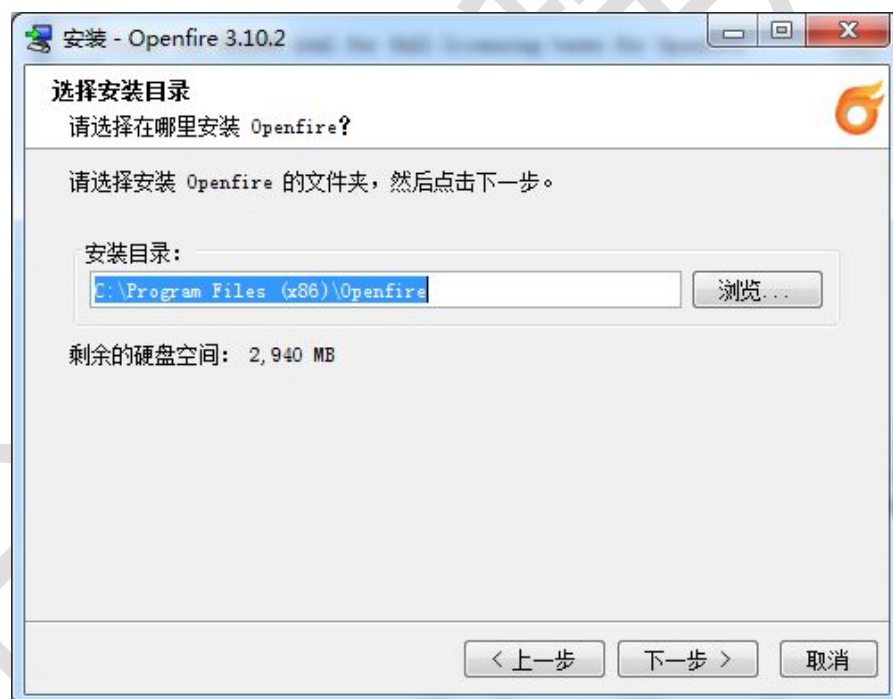
双击开始安装。



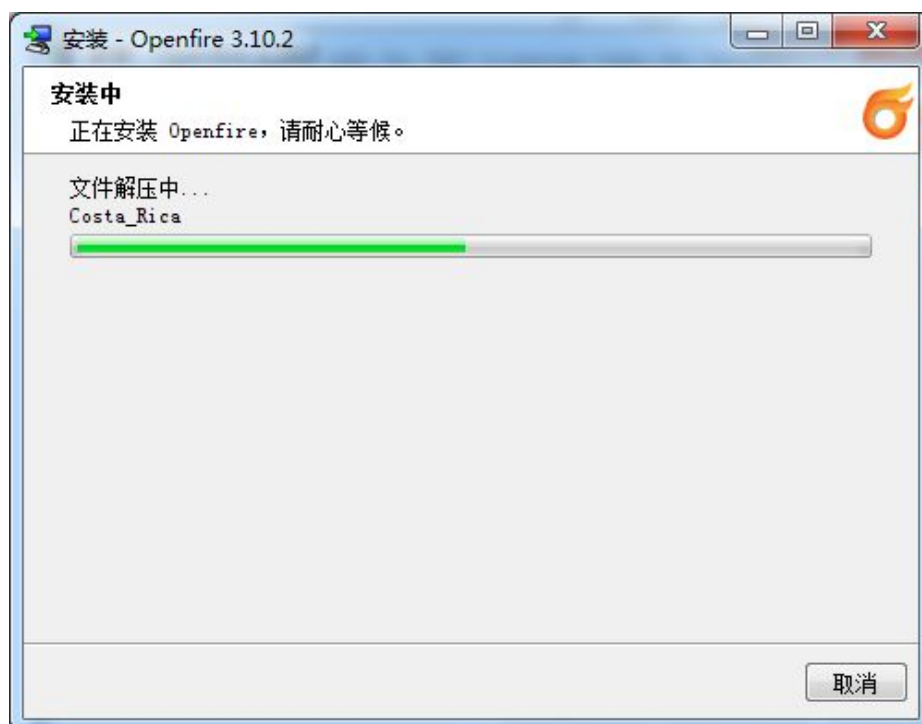
选择中文（简体），然后点击确定。



选择我接受协议，然后点击下一步。



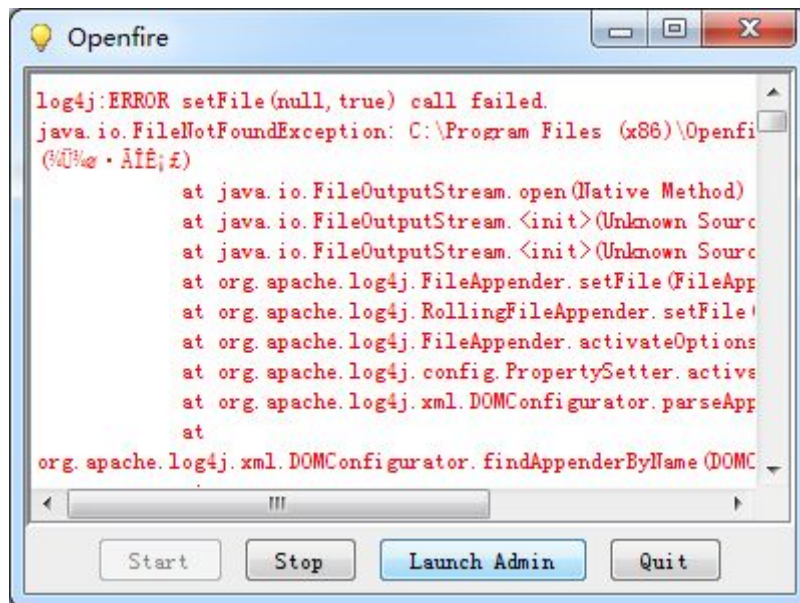
选择好安装路径，然后点击下一步。



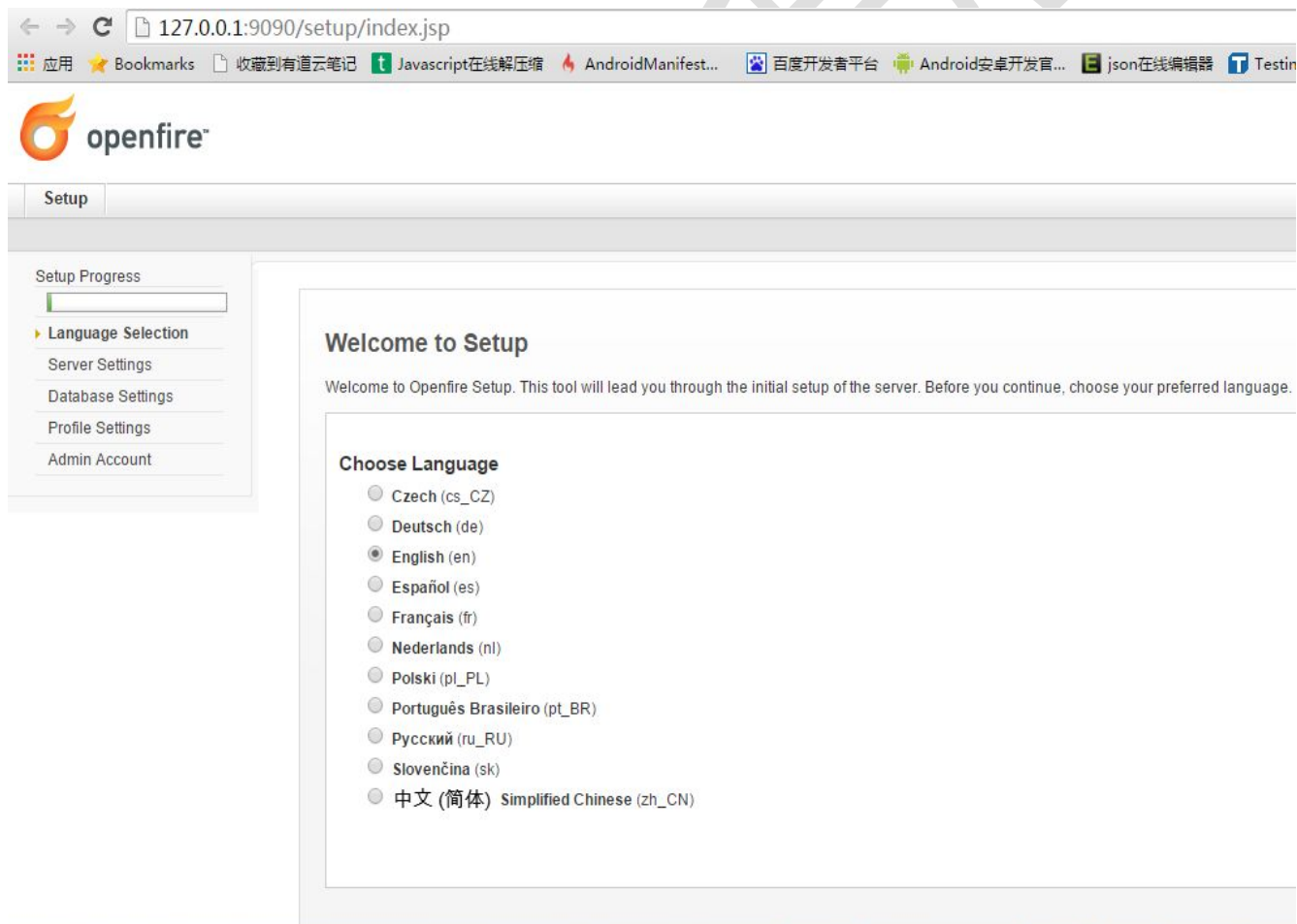
耐心等待，大概 50 秒。



点击完成。

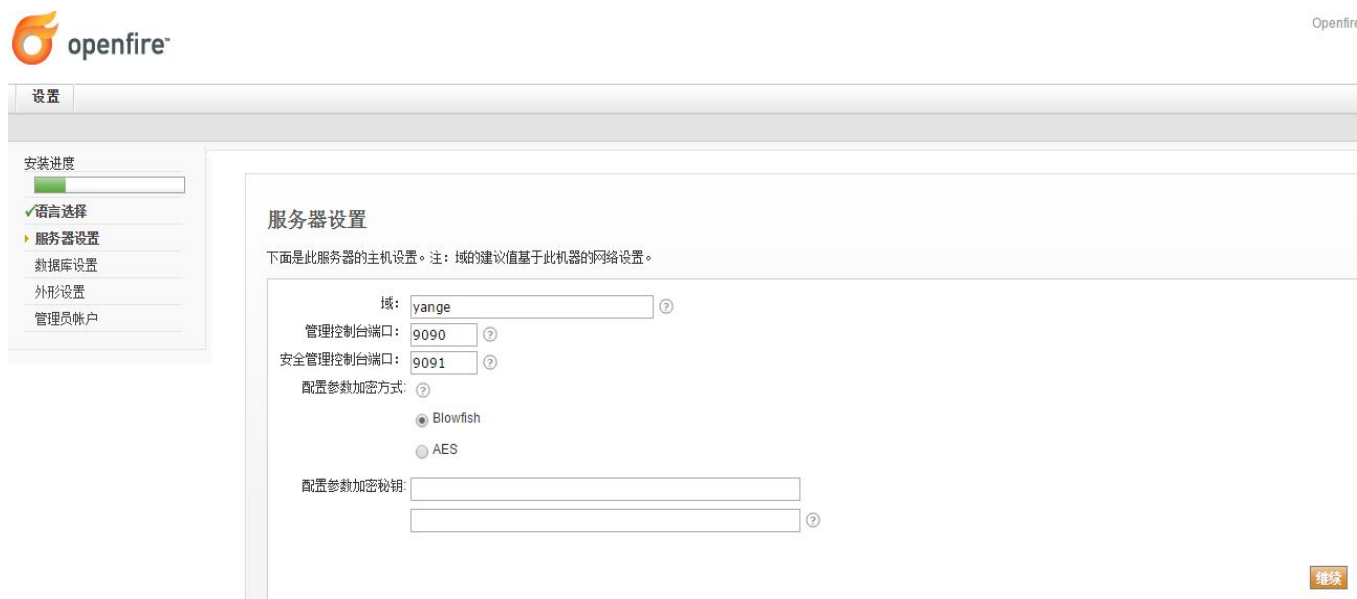


启动之后弹出如上界面，有异常！大概应该是日志相关的文件找不到。我们先不管，点击 Launch Admin 启动管理控制台。



我们注意上图中的地址栏，端口是 9090，这是 openfire 使用的端口，我们有时候喜欢把 tomact 配置层 9090，一定要注意不要占用了该端口。

我们选择中文（简体），然后点击 continue。



Openfire

设置

安装进度

- 语言选择
- 服务器设置
- 数据库设置
- 外形设置
- 管理员帐户

服务器设置

下面是此服务器的主机设置。注：域的建议值基于此机器的网络设置。

域:

管理控制台端口:

安全管理控制台端口:

配置参数加密方式:

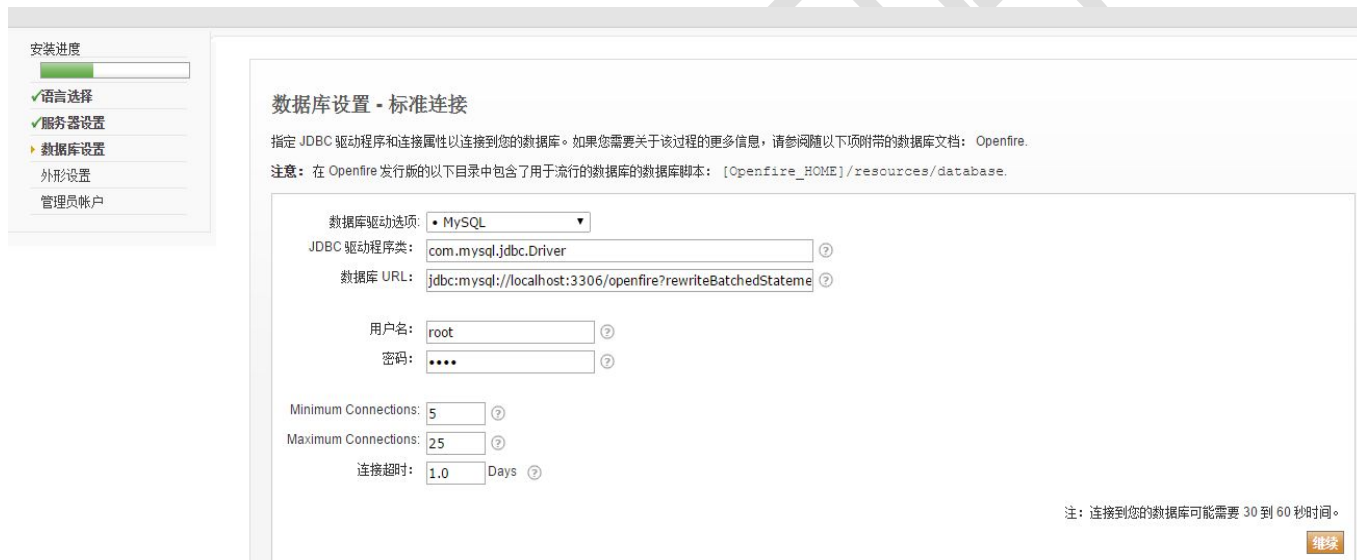
☒ Blowfish

☐ AES

配置参数加密密钥:

继续

设置域，其他设置默认即可。然后点击继续（上一页还是 **continue**，选中过中文后就可成继续了）。



安装进度

- 语言选择
- 服务器设置
- 数据库设置
- 外形设置
- 管理员帐户

数据库设置 - 标准连接

指定 JDBC 驱动程序和连接属性以连接到您的数据库。如果您需要关于该过程的更多信息，请参阅随以下项附带的数据库文档：Openfire。

注意：在 Openfire 发行版的以下目录中包含了用于流行的数据库的数据库脚本：[Openfire_HOME]/resources/database。

数据库驱动选项:

JDBC 驱动程序类:

数据库 URL:

用户名:

密码:

Minimum Connections:

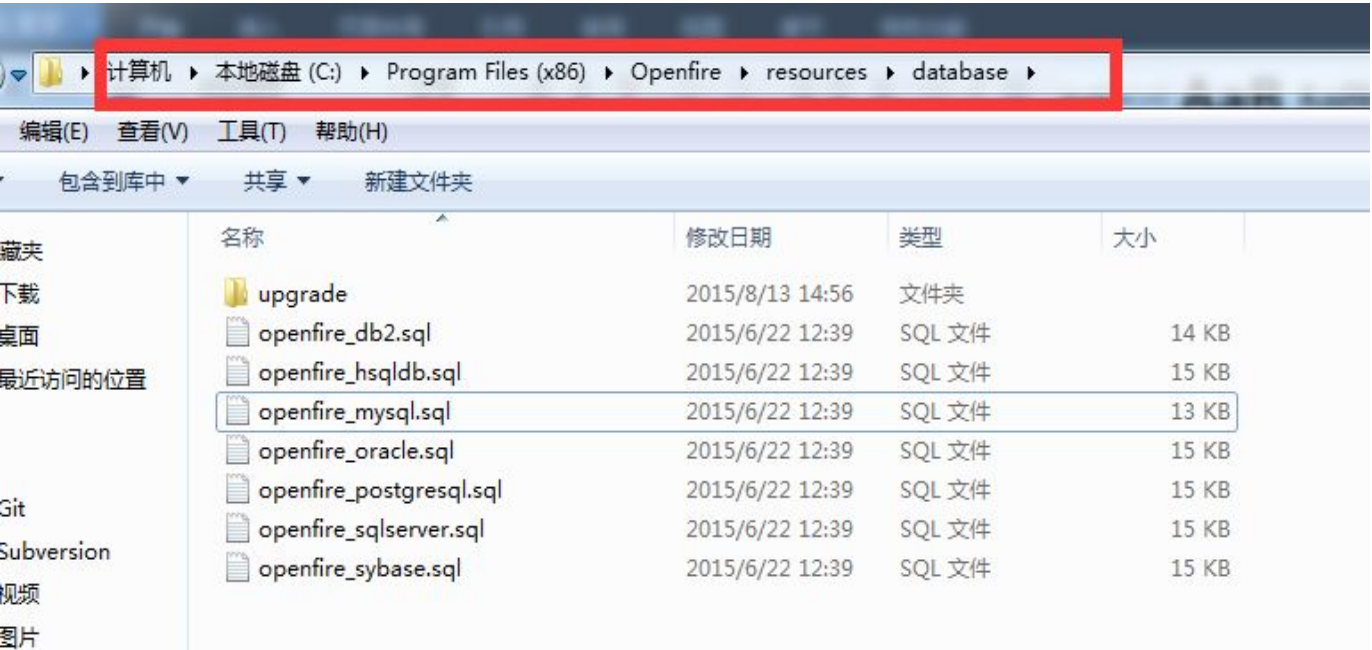
Maximum Connections:

连接超时: Days

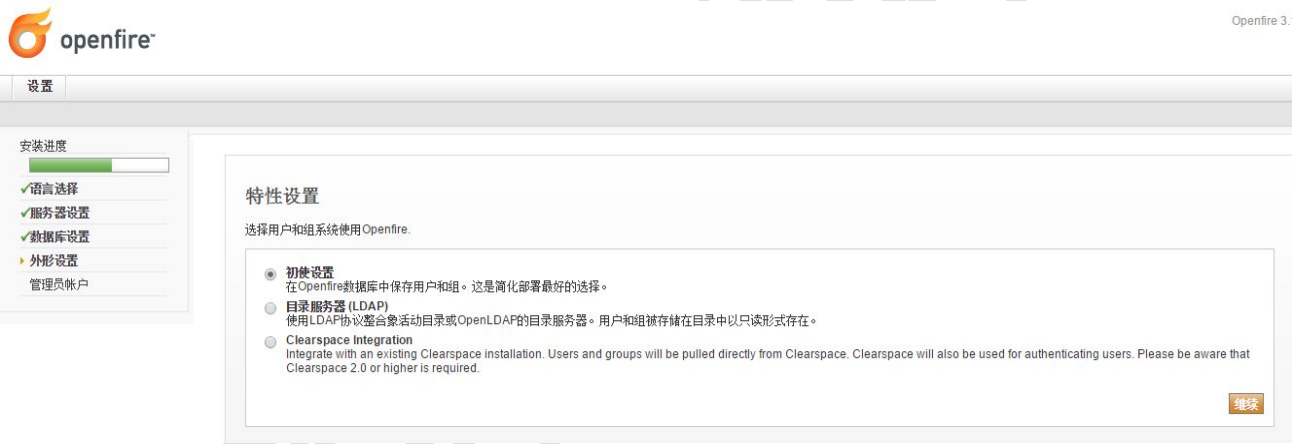
注：连接到您的数据库可能需要 30 到 60 秒时间。

继续

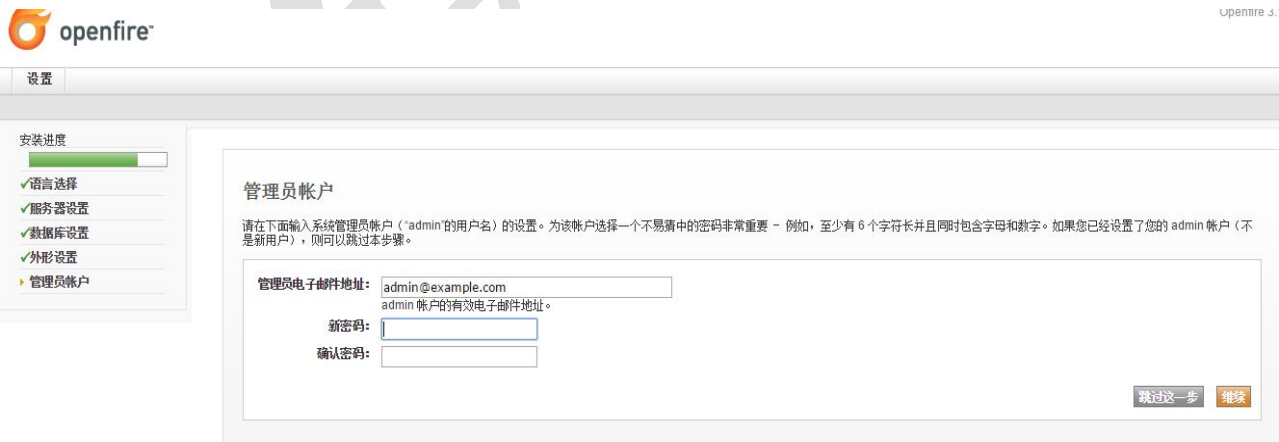
注意：这个时候必须先确保本机已经安装了 MySQL，并创建了数据库，我创建的数据库名字就叫做 **openfire**。然后在该数据库中执行 sql 脚本，该 sql 脚本是 **openfire** 提供的，位置位于：



在数据库中将该脚本内容执行一下，让其初始化数据表结构。
然后点击继续，进入下一个页面。



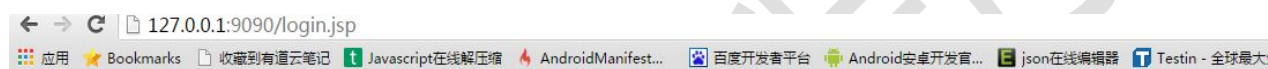
使用默认配置，继续点击继续。进入下一个页面。



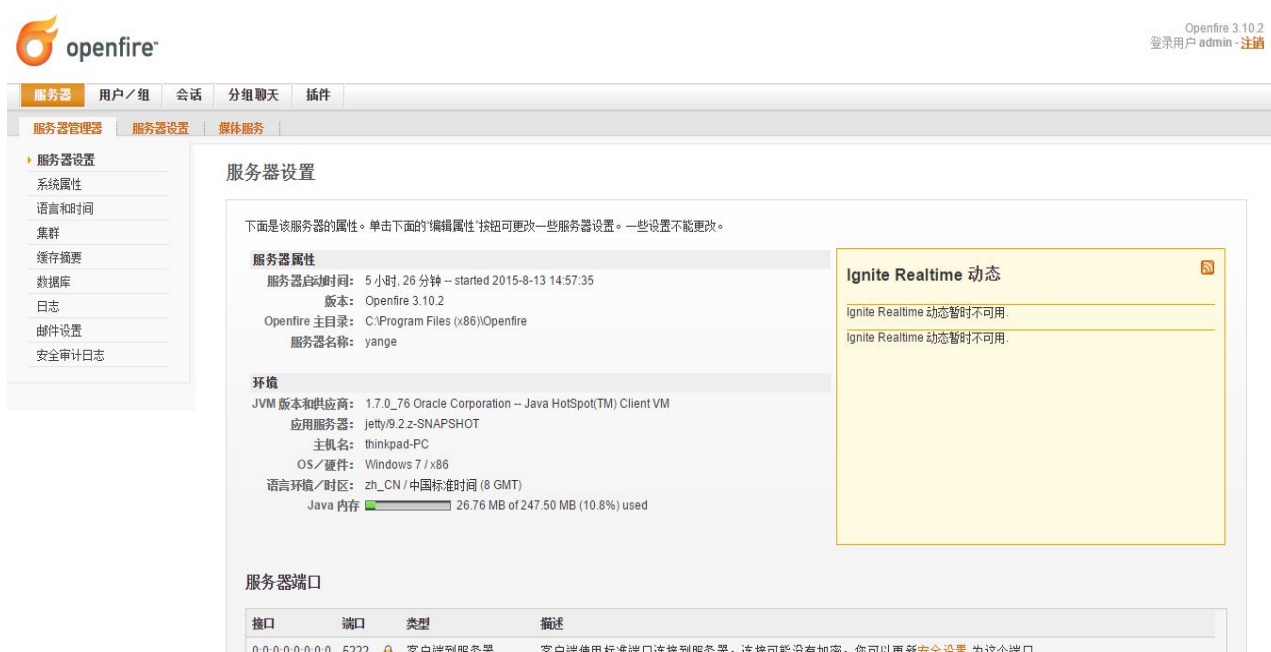
在上面填写邮箱和管理员密码，然后点击继续。



成功啦！点击登录到控制台，进入如下界面。

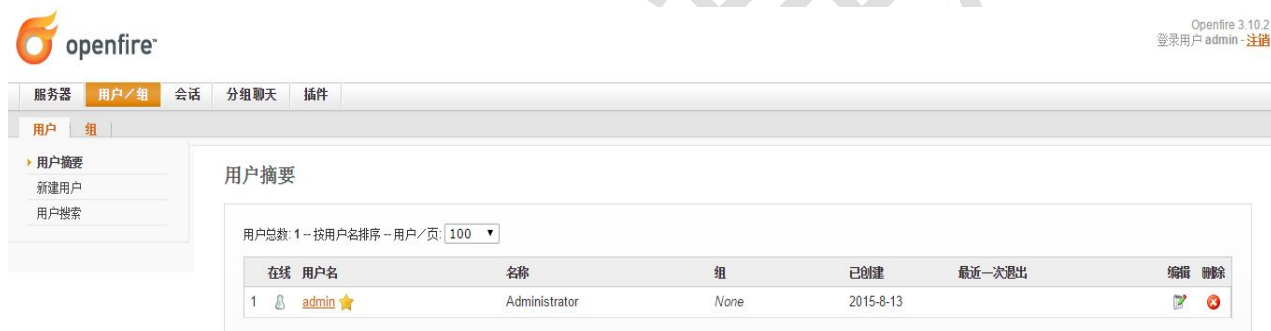


openfire 管理员默认账号为 admin，密码就是我们上一个界面设置的密码。输入账号和密码，然后点击登录。进入下一个界面。



我们就来浏览一下成功后的界面吧。

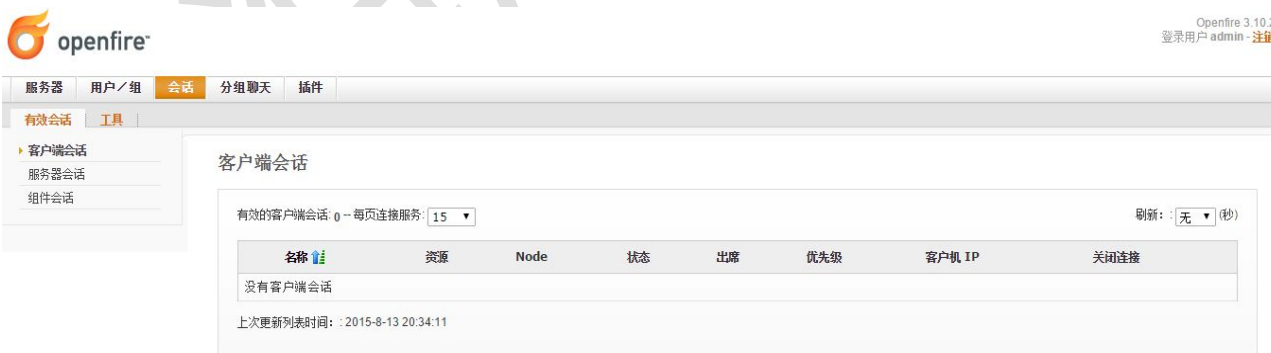
用户列表界面：



服务器 | 用户/组 | 会话 | 分组聊天 | 插件

Built by Jive Software and the IgniteRealtime.org community

会话界面：



服务器 | 用户/组 | 会话 | 分组聊天 | 插件

Built by Jive Software and the IgniteRealtime.org community

分组聊天界面：



The screenshot shows the Openfire 3.10.2 web interface. The top navigation bar includes links for '服务器' (Server), '用户/组' (Users/Groups), '会话' (Sessions), '分组聊天' (Group Chat), and '插件' (Plugins). The '分组聊天' (Group Chat) tab is active, showing a sidebar with '房间管理员' (Room Admin) and '分组聊天设置' (Group Chat Settings). The main content area is titled 'Create New Room' and contains a form for creating a new room. The form includes fields for '房间标识' (Room ID), '房间名称' (Room Name), '描述' (Description), '主题' (Topic), '最大房间占有者人数' (Maximum number of room occupants), '其 Presence 是 Broadcast 的角色' (Role of its Presence is Broadcast), '需要密码才能进入' (Require password to enter), '确认密码' (Confirm password), and '能够发现占有者真实 JID 的角色' (Role of being able to discover the real JID of the occupant). The '房间选项' (Room Options) section on the right includes checkboxes for '列出目录中的房间' (List rooms in directory), '使房间是适度的' (Make room appropriate), '使房间仅对成员开放' (Make room private), '允许占有者邀请其他人' (Allow occupants to invite others), '允许占有者更改主题' (Allow occupants to change topic), '仅允许注册的昵称登录' (Only allow registered nicknames to log in), '允许用户注册房间muc.room.edit.form.log=登录房间对话' (Allow user to register roommuc.room.edit.form.log=login room chat), and '登录房间对话' (Login room chat). The '保存更改' (Save changes) and '取消' (Cancel) buttons are at the bottom right of the form.

4.3 XMPP 客户端平台

4.3.1 Spark 客户端的下载和安装

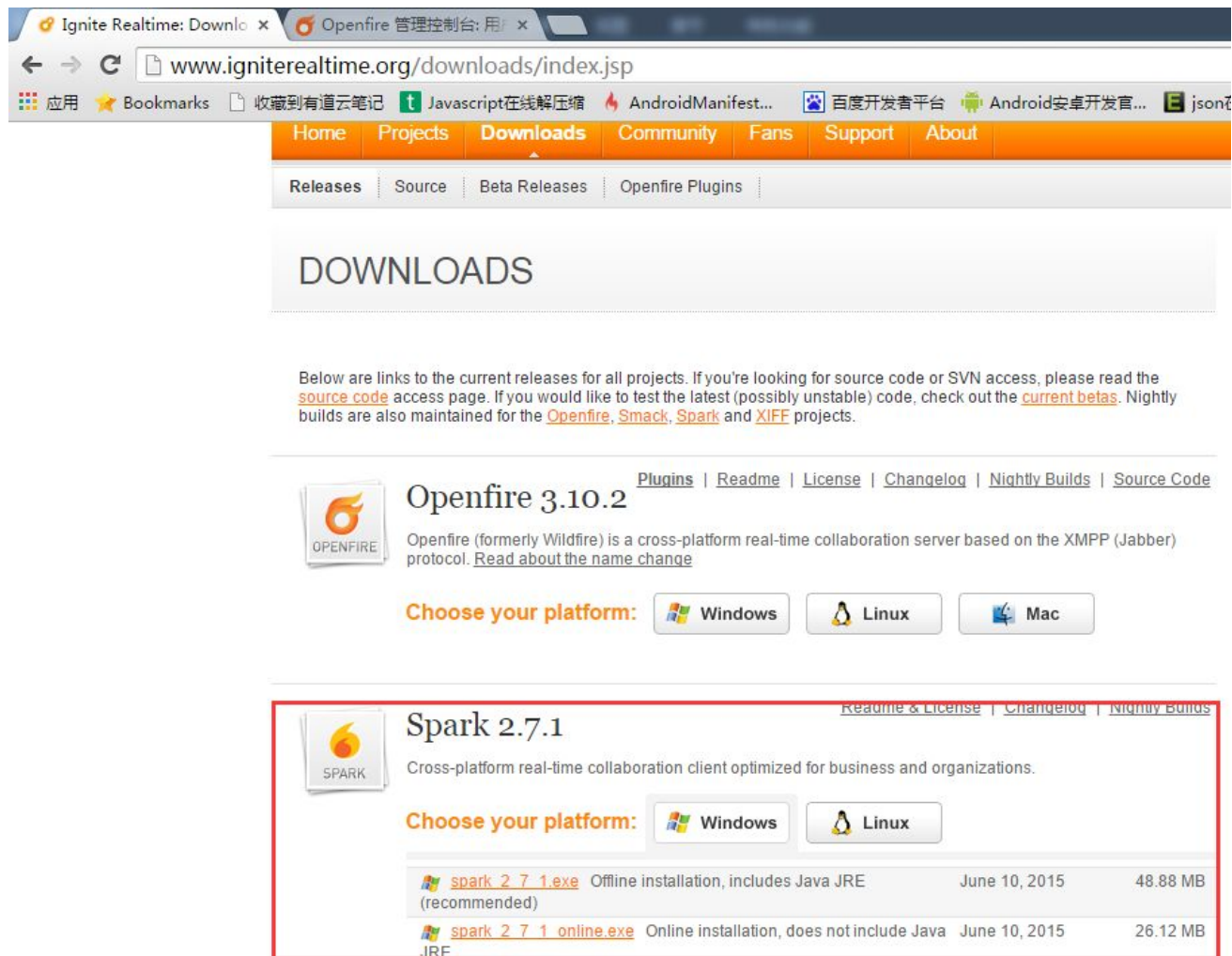
Spark 是一个开源，跨平台 IM 客户端。它的特性支持群组聊天，电话集成和强大安全性能。如果企业内部部署 IM 使用 Openfire+Spark 是最佳的组合。其官网对 Spark 的介绍如下：

Spark is an Open Source, cross-platform IM client optimized for businesses and organizations. It features built-in support for group chat, telephony integration, and strong security. It also offers a great end-user experience with features like in-line spell checking, group chat room bookmarks, and tabbed conversations.

Combined with the Openfire server, Spark is the easiest and best alternative to using un-secure public IM networks。

Spark 的下载地址：

<http://www.igniterealtime.org/downloads/index.jsp>



www.igniterealtime.org/downloads/index.jsp

Home Projects Downloads Community Fans Support About

Releases Source Beta Releases Openfire Plugins

DOWNLOADS

Below are links to the current releases for all projects. If you're looking for source code or SVN access, please read the [source code](#) access page. If you would like to test the latest (possibly unstable) code, check out the [current betas](#). Nightly builds are also maintained for the [Openfire](#), [Smack](#), [Spark](#) and [XIFF](#) projects.

Openfire 3.10.2

Plugins | Readme | License | Changelog | Nightly Builds | Source Code

Openfire (formerly Wildfire) is a cross-platform real-time collaboration server based on the XMPP (Jabber) protocol. [Read about the name change](#)



Choose your platform: Windows Linux Mac

Spark 2.7.1


Readme & License | Changelog | Nightly Builds

Cross-platform real-time collaboration client optimized for business and organizations.

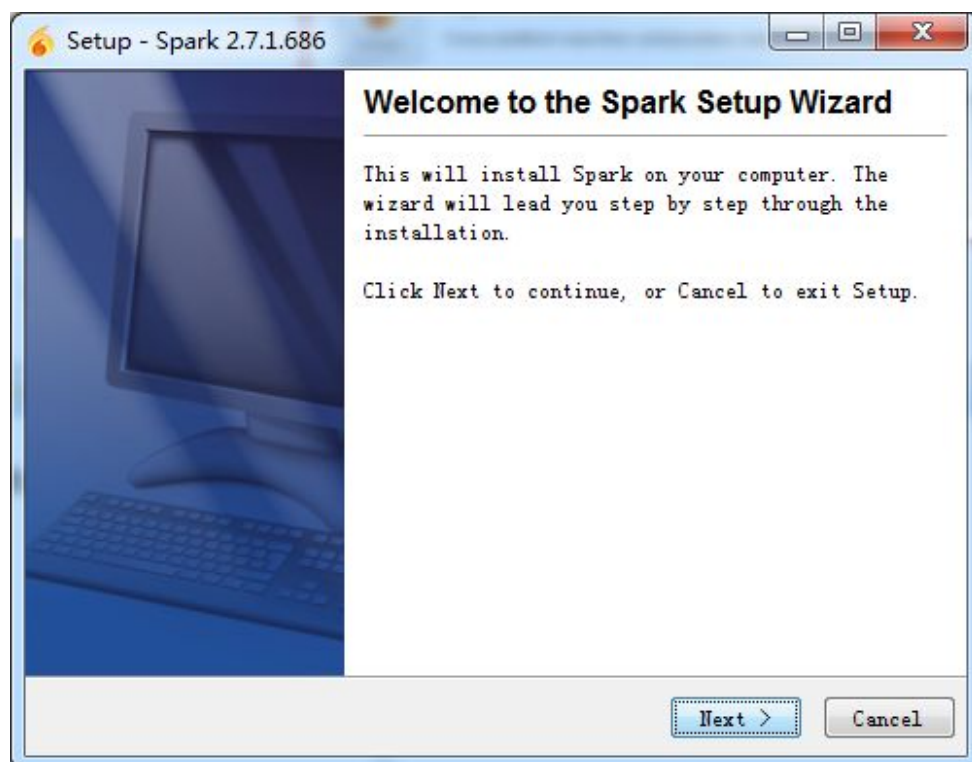
Choose your platform: Windows Linux

 spark_2_7_1.exe (recommended)	Offline installation, includes Java JRE	June 10, 2015	48.88 MB
 spark_2_7_1_online.exe	Online installation, does not include Java JRE	June 10, 2015	26.12 MB

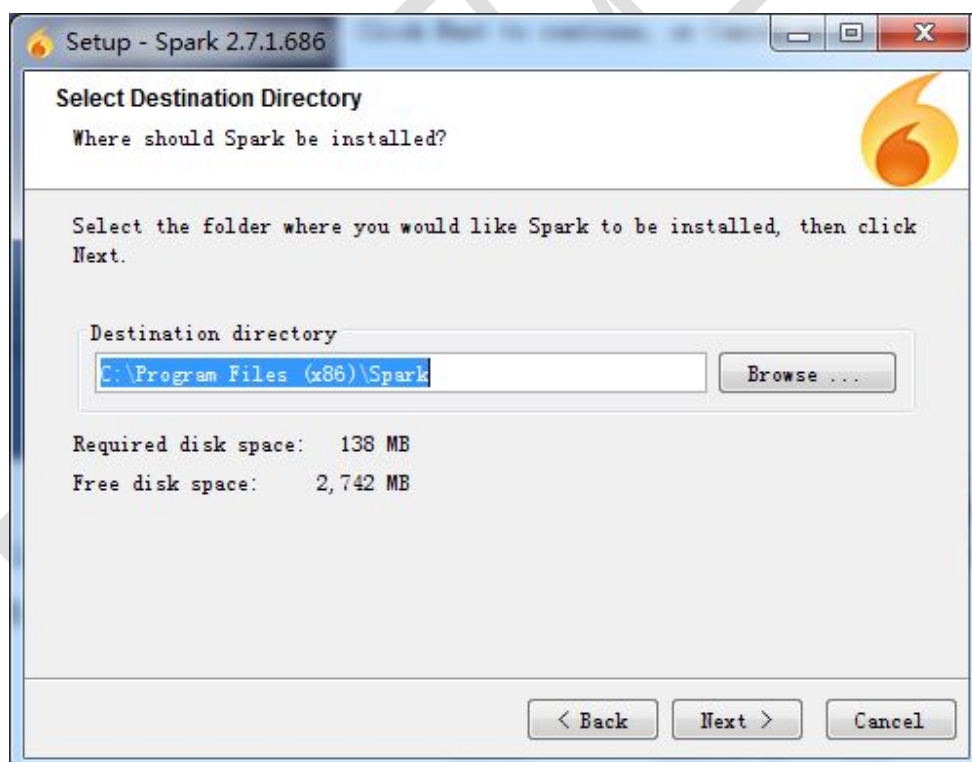
下载好以后：

 spark_2_7_1.exe	2015/8/13 14:24	应用程序	50,049 KB
---	-----------------	------	-----------

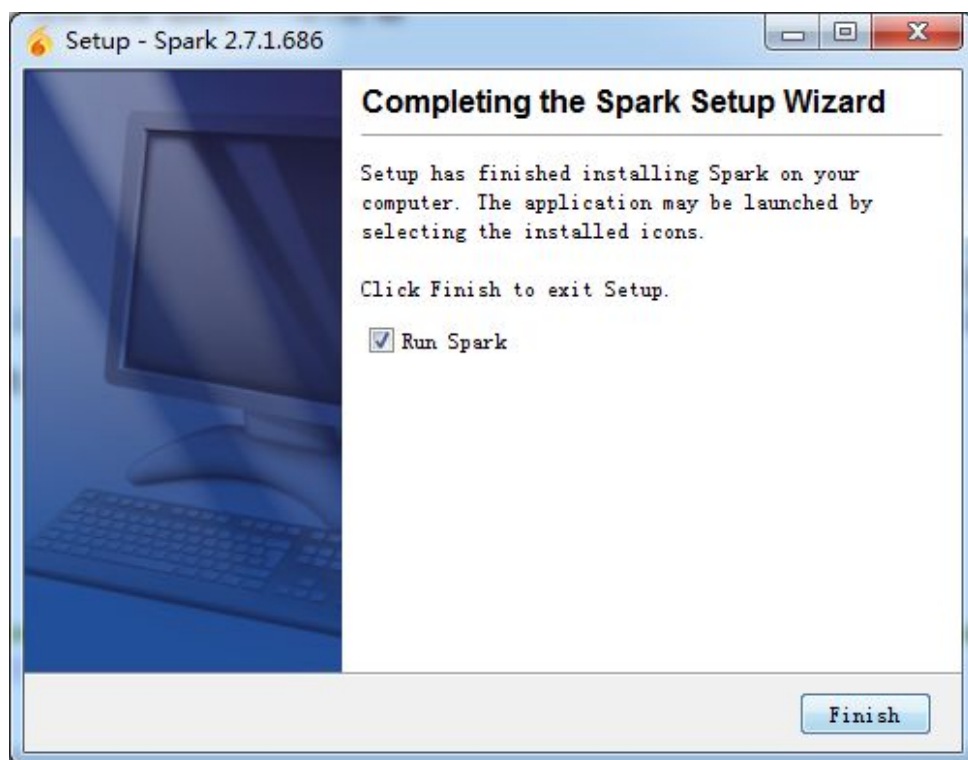
双击安装。安装非常简单，一路点击下一步即可。



点击 Next，进入下一个界面：



点击 Next.....Next 进入下一个界面：



看到如下界面就 OK 了，如果在上一个界面点击 Finish 没有起效，那么可以找到生成的桌面快捷图标，双击。



点击账号，进入下一个界面。



在上图中输入用户、密码和服务地址。



然后创建账号。



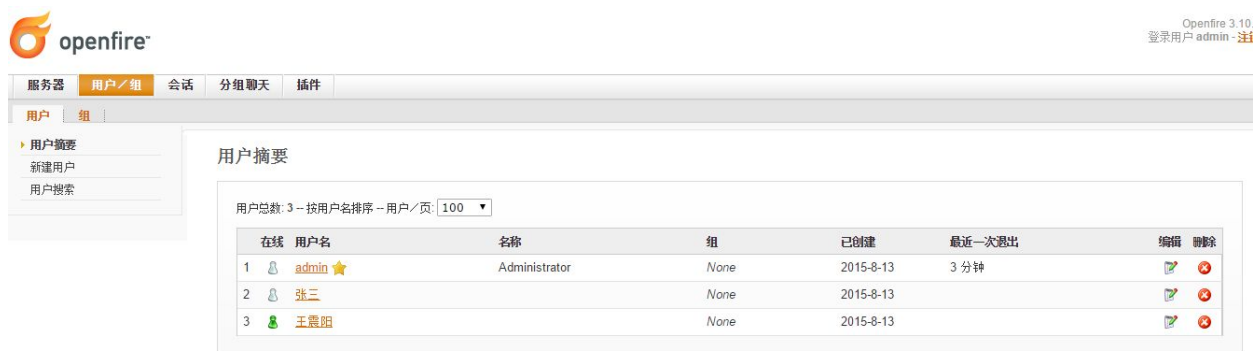
在上图中点击确定，进入如下图界面。



点击 Login 进入如下界面：

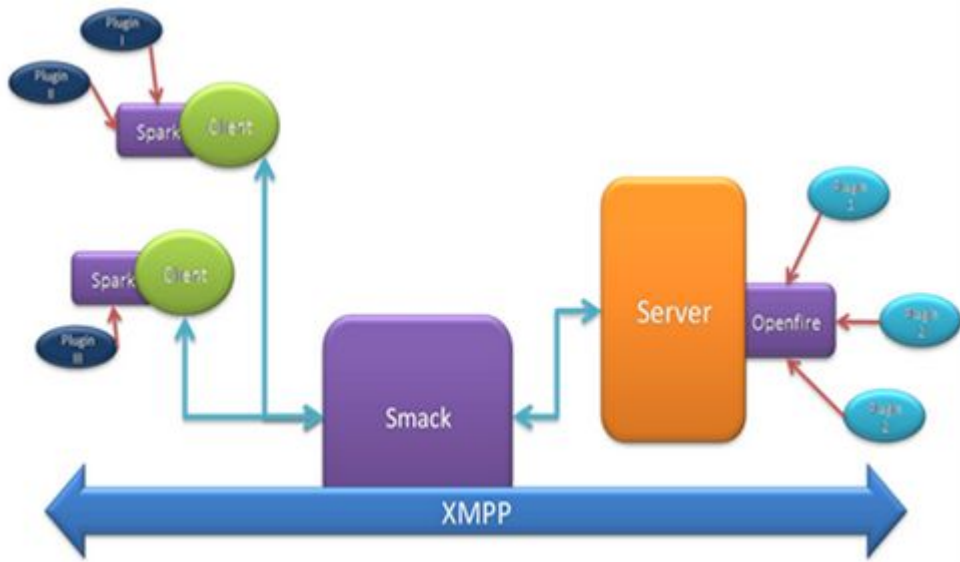


在 openfire 中刷新界面，打开用户/组选项卡，可以看到所有注册的用户列表。



4.3.2 Spark 和 Openfire 通信原理

从网上找到一个 Spark 和 Openfire 直接通讯架构图。看懂这张图就知道他们之间是如何通信的了。



Smack 是对 XMPP 协议的封装库，Smack 是 XMPP 协议的 Java 层的封装。让我们 Java 程序员不用直接跟枯燥无味的 XML 打交道（生成 XML 和解析 XML）。

随着移动互联网的快速发展，尤其是即时通信的发展，几乎 90% 以上的 App 都有这样的功能。因此 asmack 也与 13 年诞生，asmack 其实就是 Android Smack 的简称。asmack 也是我们该门课程用到的主要 API。

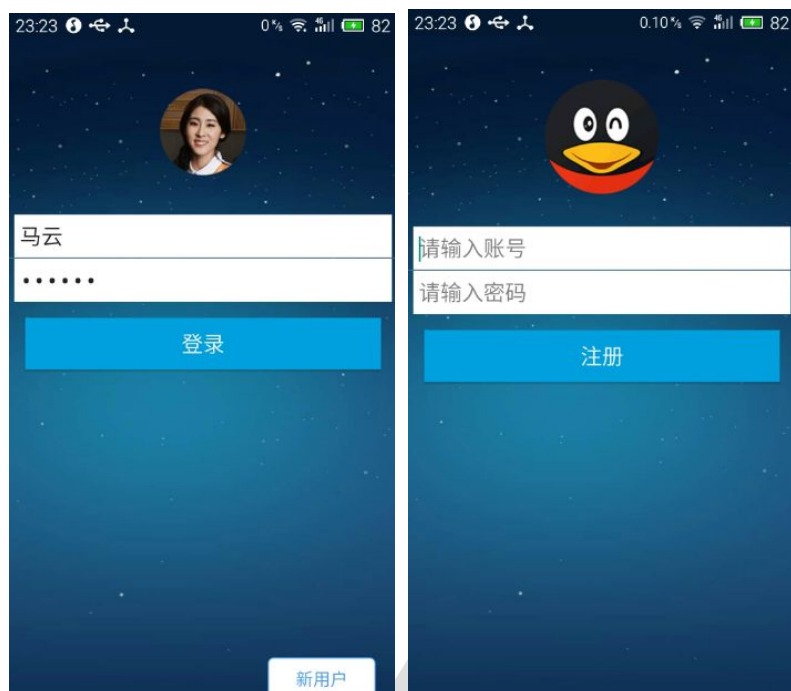
下面请看第 5 章吧，让我们一起学习如何使用 asmack 打造我们的交友神器。

第 5 章 使用 ASmack 打造即时通信 App

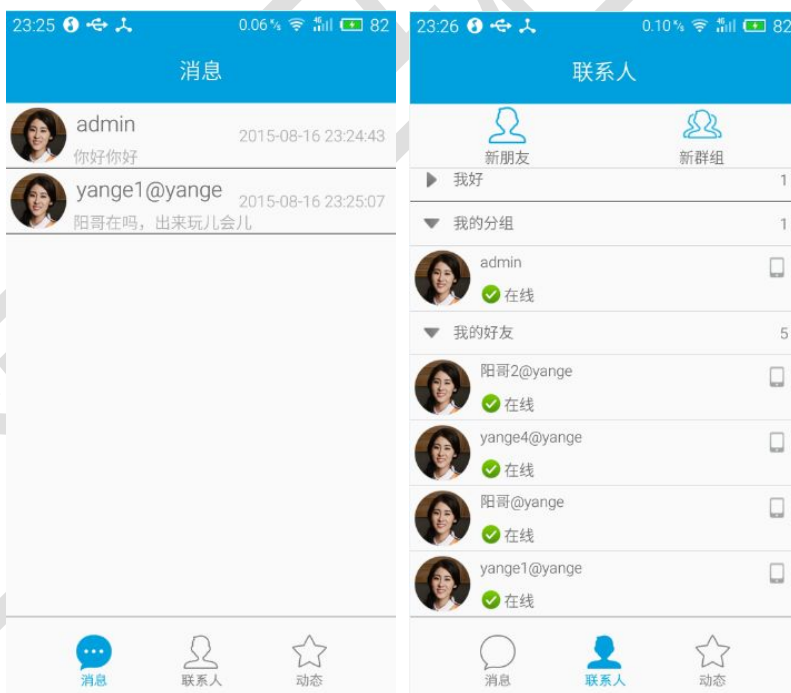
5.1 项目简介

做一个类似 QQ 的通讯工具，要求有注册、登录、添加好友、添加分组、聊天、退出登录等功能。我用 8 张运行效果图来展示我们将要实现的功能。

注意，服务器用的是 Openfire，我们可以用 Spark 作为另外一个客户端进行测试。



闪屏页进来以后是登录界面，要求记住上次登陆的账号和密码，在界面的右下角有新用户按钮，点击后进入注册界面。注册只需要输入账号和密码即可。账号不能和其他人重复，否则注册不成功。



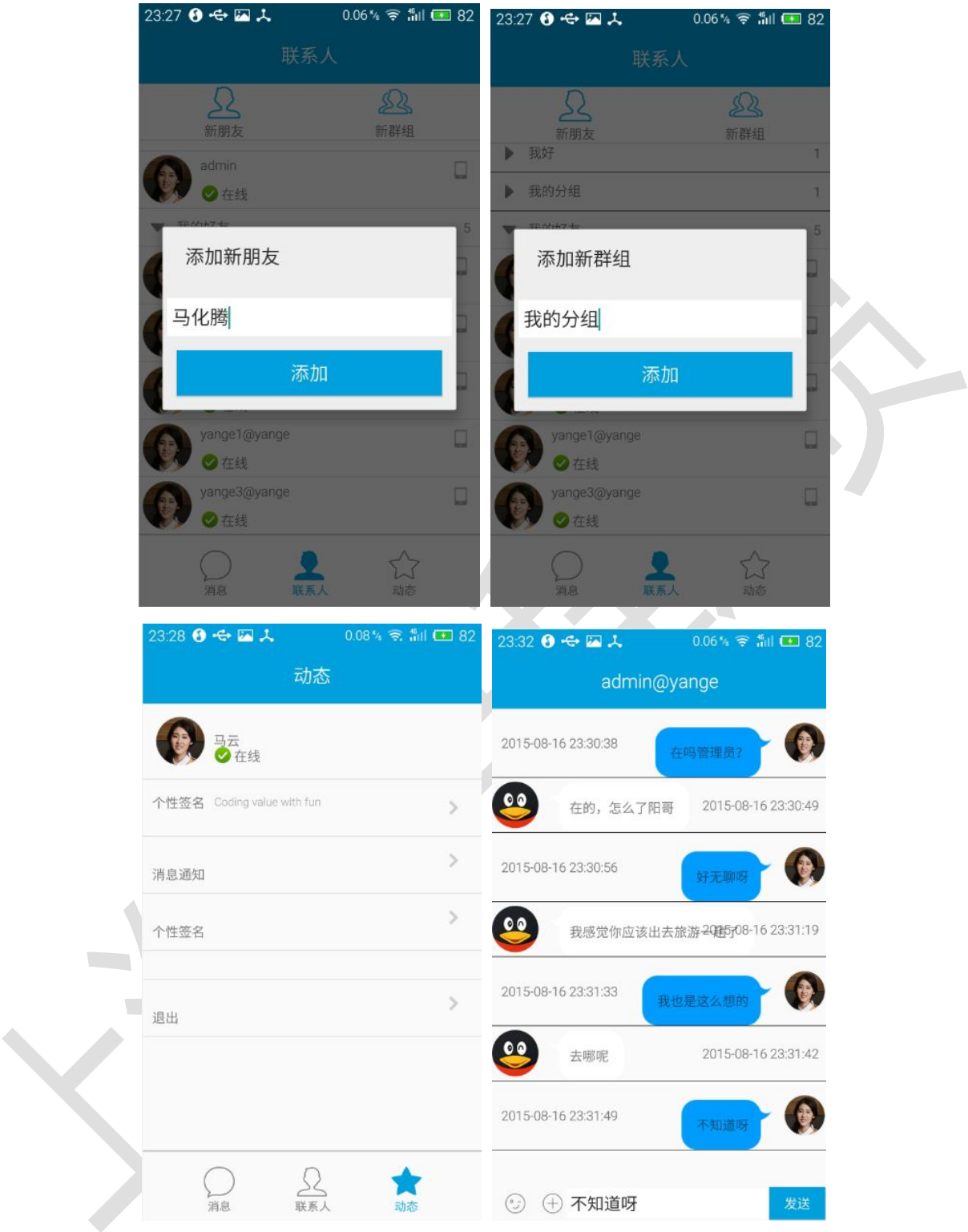
主界面是一个 Activity 绑定 3 个 Fragment 实现，分别是消息、联系人、动态。

其中消息界面是一个 ListView 展出了最近的联系人。点击其中的一个条目可以进入聊天界面。

联系人界面主要是一个 ExpandableListView。ExpandableListView 列出了用户组，以及各个组中的好友。点击任意好友可以进入聊天界面。在 ExpandableListView 上面有两个图标，分别是新朋友、新群组。点击新朋友弹出一个自定义对话框，在该对话框中输入对方好友的名称，等待对方同意了即可添加为好友。

点击新群组也弹出一个自定义对话框，在该对话框中输入分组名称，则可以创建一个分组。

如果好友不存则添加好友失败，如果分组不存在则创建分组失败。



动态界面主要展示了当前用户的信息，最下面有个退出按钮，点击后退出当前登录，并跳转到登录界面。

聊天界面是一个 ListView，该 ListView 的条目有两类布局，分别用来表示好友的消息和自己的消息。在最下方的输入框输入文本内容，然后点击发送可以将该消息发送给好友。好友有消息过来也可以直接显示在该界面。

5.2 项目搭建

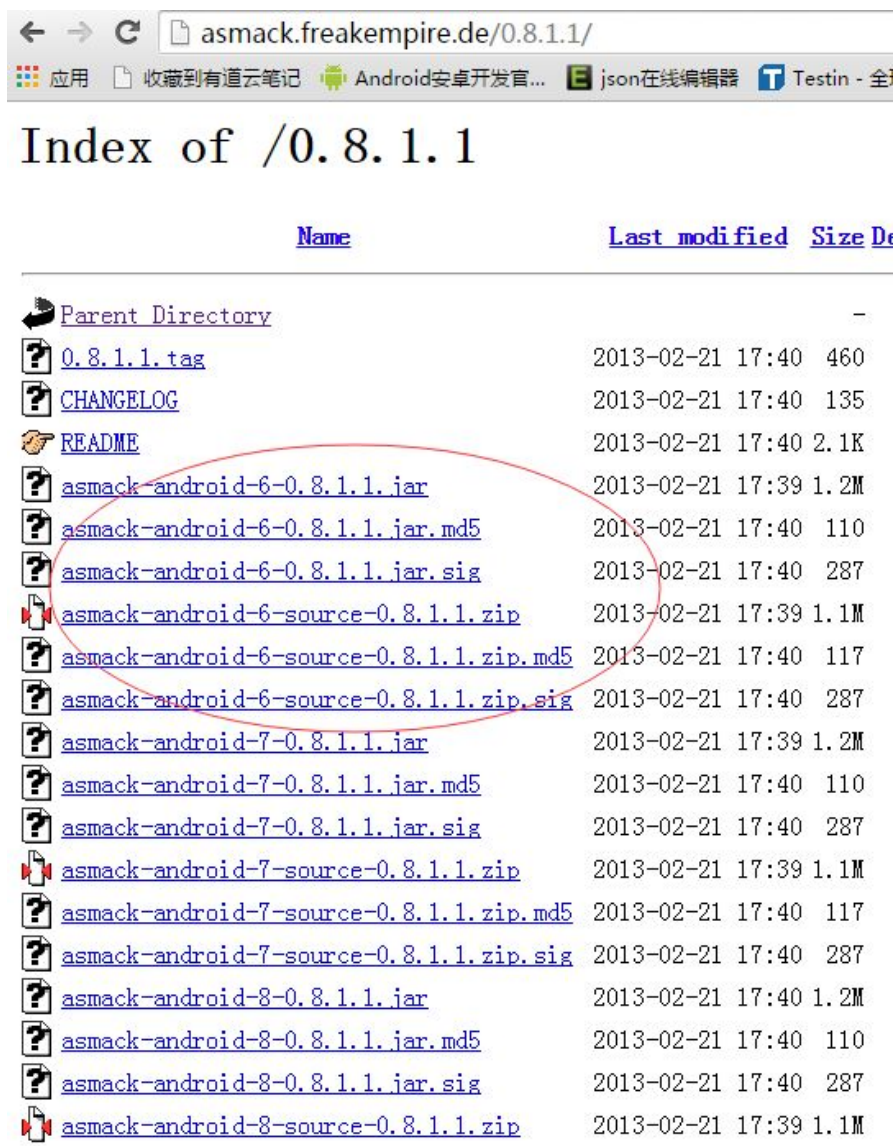
1、下载 asmack.jar

<http://asmack.freakempire.de/>

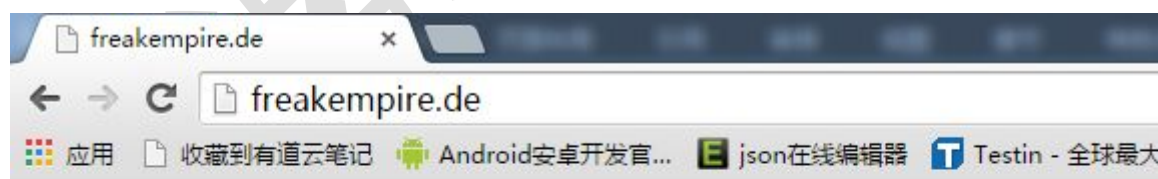
是个德国网站。



asmack 的版本号从 0.x 到现在的 4.x 变化比较大。不通过版本差异也比较大。本次我写项目用的是 0.8.x 的。用的是 13 年的版本。因为该 api 在网上能查到的资料比较多。如果是下一次我再写这个项目我就决定用 15 年发布的最新版本的了。



我想看看 asmack 公司官网，吧 asmack 去掉，想着就是贵公司的网址呢，却得到这样的界面。



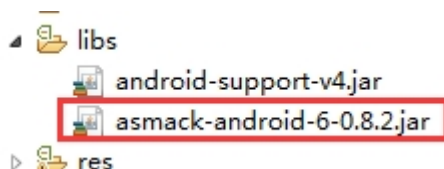
It works!

This is the default web page for this server.

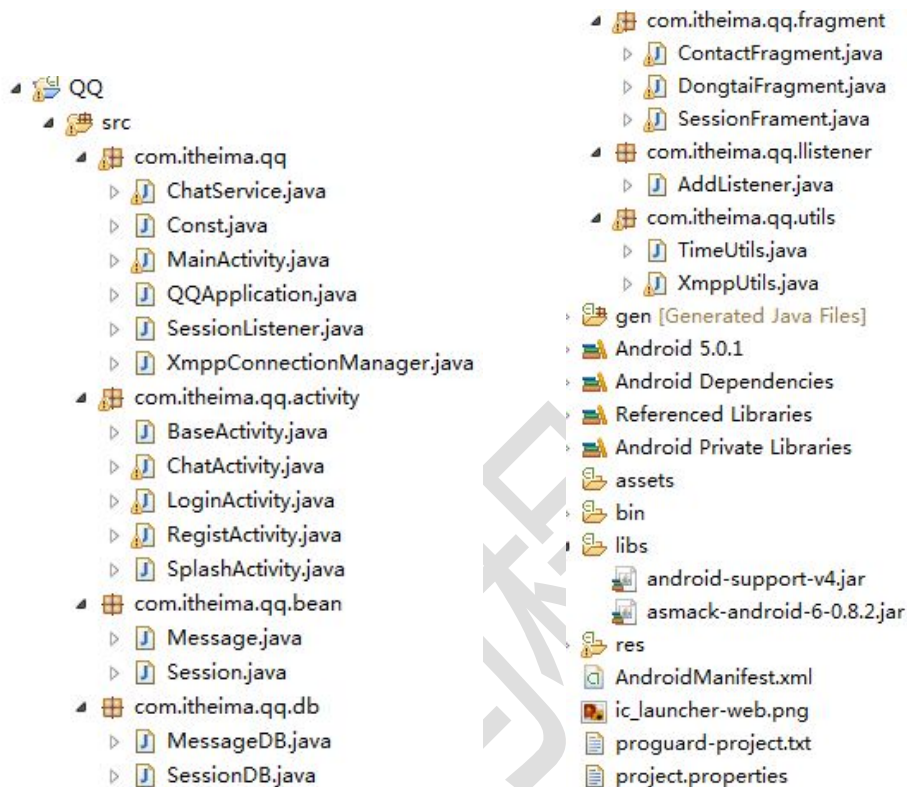
The web server software is running but no content has been added, yet.

2、给工程添加 jar

简单极了，只需把下载好的 jar 包添加到 Android 工程的 libs 目录下即可。注意如果 eclipse 没有自动将该包添加到环境变量中，我们需要手动添加一下。



3、项目目录结构图



5.3 项目实现

5.3.1 Spark 客户端的下载和安装闪屏界面

新颖的闪屏界面，马老师，学习的目的，我借来用一用哈希望您不用太在意。



我要做的效果是闪屏等待 3 秒，然后进入主界面。但是每次都让用户等待 3 秒，对于急性子来讲估计会很抓狂。那怎么办呢，只要是这个界面用户触摸屏幕则立即进入主界面。布局太简单了，不给了。直接给代码。

```
1.  /**
2.   * 闪屏页面 默认等待 3s 触摸时直接进入主界面
3.   *
4.   * @author wzy 2015-8-14
5.   *
6.   */
7. public class SplashActivity extends Activity {
8.
9.     private static final long DURATION = 3000;
10.    /**
11.     * 保证变量的修改是可见的，但是无法保证变量的原子性
12.     */
13.    private volatile boolean isEntered = false;
14.    private Thread splashThread = new Thread(new Runnable() {
15.
16.        @Override
17.        public void run() {
18.            SystemClock.sleep(DURATION);
19.            enter();
20.        }
21.    });
22.
23.    @Override
```

```
24.     protected void onCreate(Bundle savedInstanceState) {
25.         super.onCreate(savedInstanceState);
26.         setContentView(R.layout.activity_splash);
27.         splashThread.start();
28.     }
29.
30.     private synchronized void enter() {
31.         if (!isEntered) {
32.             isEntered = true;
33.             startActivity(new Intent(SplashActivity.this, LoginActivity.class));
34.             finish();
35.         }
36.     }
37.
38.     @Override
39.     public boolean onTouchEvent(MotionEvent event) {
40.         enter();
41.         return true;
42.     }
43. }
```

5.3.2 登录

登录界面布局很简单。如果写不出来就没必要往下看了。登录的核心代码：

```
mXmppConnection.login(name, pwd);
```

activity_login.xml

```
1. <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2.     xmlns:tools="http://schemas.android.com/tools"
3.     android:layout_width="match_parent"
4.     android:layout_height="wrap_content"
5.     android:background="@drawable/login_background" >
6.     <LinearLayout
7.         android:layout_width="match_parent"
8.         android:layout_height="match_parent"
9.         android:background="@drawable/login_background"
10.        android:orientation="vertical" >
11.         <ImageView
12.             android:id="@+id/iv_touxiang"
13.             android:layout_width="wrap_content"
14.             android:layout_height="wrap_content"
15.             android:layout_gravity="center_horizontal"
```



```
16.         android:layout_marginTop="40dp"
17.         android:contentDescription="@null"
18.         android:src="@drawable/login_image" />
19.     <EditText
20.         android:id="@+id/et_name"
21.         android:layout_width="match_parent"
22.         android:layout_height="40dp"
23.         android:layout_marginLeft="5dp"
24.         android:layout_marginRight="5dp"
25.         android:layout_marginTop="30dp"
26.         android:background="#FFFFFF"
27.         android:hint="请输入账号"
28.         android:paddingLeft="5dp"
29.         android:textSize="20sp" />
30.     <View
31.         android:layout_width="match_parent"
32.         android:layout_height="1dp"
33.         android:background="#55AABBCC" />
34.     <EditText
35.         android:id="@+id/et_pwd"
36.         android:layout_width="match_parent"
37.         android:layout_height="40dp"
38.         android:layout_marginLeft="5dp"
39.         android:layout_marginRight="5dp"
40.         android:background="#FFFFFF"
41.         android:hint="请输入密码"
42.         android:inputType="textPassword"
43.         android:paddingLeft="5dp"
44.         android:textSize="20sp" />
45.     <Button
46.         android:id="@+id/btn_login"
47.         android:layout_width="match_parent"
48.         android:layout_height="wrap_content"
49.         android:layout_margin="15dp"
50.         android:background="@color/title_layout"
51.         android:onClick="login"
52.         android:text="登录"
53.         android:textColor="#FFFFFF"
54.         android:textSize="20sp" />
55. </LinearLayout>
56. <Button
57.     android:id="@+id/register"
58.     android:layout_width="100dp"
59.     android:layout_height="40dp"
60.     android:layout_alignParentBottom="true"
```

```
61.         android:layout_alignParentRight="true"
62.         android:layout_marginRight="20dp"
63.         android:background="@drawable/register_user_btn"
64.         android:text="新用户"
65.         android:onClick="gotoRegist"
66.         android:textColor="@color/blue"
67.         android:textSize="16sp" />
68. </RelativeLayout>
```

LoginActivity 和 RegistActivity 都继承自 BaseActivity。因此这里先把 BaseActivity 的代码列出。

BaseActivity.java

```
1. import org.jivesoftware.smack.XMPPConnection;
2. import com.itheima.qq.XmppConnectionManager;
3. import android.app.Activity;
4. public class BaseActivity extends Activity {
5.     //定义一个 XMPPConnection 的全局对象
6.     public XMPPConnection mXmppConnection;
7.     protected void onCreate(android.os.Bundle savedInstanceState) {
8.         super.onCreate(savedInstanceState);
9.         //首先获取 XmppConnectionManager 对象，单例的模式。XmppConnectionManager 是自定义的类
10.        XmppConnectionManager xmppConnectionManager = XmppConnectionManager.getInstance();
11.        //调用 init 方法，返回一个 XMPPConnection 对象
12.        mXmppConnection = xmppConnectionManager.init();
13.    };
14. }
```

XmppConnectionManager.java

```
1. import org.jivesoftware.smack.ConnectionConfiguration;
2. import org.jivesoftware.smack.Roster;
3. import org.jivesoftware.smack.XMPPConnection;
4. import org.jivesoftware.smack.provider.PrivacyProvider;
5. import org.jivesoftware.smack.provider.ProviderManager;
6. import org.jivesoftware.smackx.GroupChatInvitation;
7. import org.jivesoftware.smackx.PrivateDataManager;
8. import org.jivesoftware.smackx.bytestreams.socks5.provider.BytestreamsProvider;
9. import org.jivesoftware.smackx.packet.ChatStateExtension;
10. import org.jivesoftware.smackx.packet.LastActivity;
11. import org.jivesoftware.smackx.packet.OfflineMessageInfo;
12. import org.jivesoftware.smackx.packet.OfflineMessageRequest;
13. import org.jivesoftware.smackx.packet.SharedGroupsInfo;
14. import org.jivesoftware.smackx.provider.AdHocCommandDataProvider;
15. import org.jivesoftware.smackx.provider.DataFormProvider;
```

```
16. import org.jivesoftware.smackx.provider.DelayInformationProvider;
17. import org.jivesoftware.smackx.provider.DiscoverInfoProvider;
18. import org.jivesoftware.smackx.provider.DiscoverItemsProvider;
19. import org.jivesoftware.smackx.provider.MUCAdminProvider;
20. import org.jivesoftware.smackx.provider.MUCOwnerProvider;
21. import org.jivesoftware.smackx.provider.MUCUserProvider;
22. import org.jivesoftware.smackx.provider.MessageEventProvider;
23. import org.jivesoftware.smackx.provider.MultipleAddressesProvider;
24. import org.jivesoftware.smackx.provider.RosterExchangeProvider;
25. import org.jivesoftware.smackx.provider.StreamInitiationProvider;
26. import org.jivesoftware.smackx.provider.VCardProvider;
27. import org.jivesoftware.smackx.provider.XHTMLExtensionProvider;
28. import org.jivesoftware.smackx.search.UserSearch;
29.
30. import android.util.Log;
31.
32. public class XmppConnectionManager {
33.     private static XmppConnectionManager instance = null;
34.     //私有化构造函数
35.     private XmppConnectionManager() {
36.     }
37.     /**
38.      * 获取该对象
39.      * @return
40.      */
41.     public static XmppConnectionManager getInstance() {
42.         if (instance == null) {
43.             instance = new XmppConnectionManager();
44.         }
45.         return instance;
46.     }
47.     /**
48.      * 执行初始化脚本
49.      * @return
50.      */
51.     public XMPPConnection init() {
52.         /**
53.          * 创建连接配置对象<br>
54.          * 第一个参数是 Openfire 服务器地址<br>
55.          * 第二个参数是 Openfire 服务器断号，默认是 5222<br>
56.          * 我们可以把这两个参数配置的清单文件中，也可以写死在代码中
57.          */
58.         ConnectionConfiguration connectionConfig = new
59. ConnectionConfiguration(Const.XMPP_HOST, Const.XMPP_PORT);
60.         /**
```

```
61.         * 不使用 SAL 安全验证
62.         */
63.         connectionConfig.setSASLAuthenticationEnabled(false);
64. /**
65.         * 设置 TLS 安全模式
66.         */
67.         connectionConfig.setSecurityMode(ConnectionConfiguration.SecurityMode.enabled);
68.         // 允许自动连接
69.         connectionConfig.setReconnectionAllowed(true);
70.         // 允许登陆成功后更新在线状态
71.         connectionConfig.setSendPresence(true);
72.         //设置为 debug 模式，该模式可以在控制台看到接收和发送的 xmpp 协议
73.         connectionConfig.setDebuggerEnabled(true);
74.         // 收到好友邀请后同意添加为好友的模式，有三种，manual 表示需要经过同意,accept_all 表示不经同意自动
75. 为好友，reject_all 拒绝加为好友邀请
76.         Roster.setDefaultSubscriptionMode(Roster.SubscriptionMode.accept_all);
77.         /**
78.         * 该配置时为了解决 asmack 的一个 bug 或者说弥补一个不足之处。不用细细追究。
79.         */
80.         configure(ProviderManager.getInstance());
81.         //创建一个连接对象，参数为配置对象
82.         XMPPConnection connection = new XMPPConnection(connectionConfig);
83.         return connection;
84.     }
85.
86.     public void configure(ProviderManager pm) {
87.
88.         // Private Data Storage
89.         pm.addIQProvider("query", "jabber:iq:private", new
90. PrivateDataManager.PrivateDataIQProvider());
91.
92.         // Time
93.         try {
94.             pm.addIQProvider("query", "jabber:iq:time",
95. Class.forName("org.jivesoftware.smackx.packet.Time"));
96.         } catch (ClassNotFoundException e) {
97.             Log.w("TestClient", "Can't load class for org.jivesoftware.smackx.packet.Time");
98.         }
99.
100.        // Roster Exchange
101.        pm.addExtensionProvider("x", "jabber:x:roster", new RosterExchangeProvider());
102.        // Message Events
103.        pm.addExtensionProvider("x", "jabber:x:event", new MessageEventProvider());
104.        // Chat State
```

```
105.         pm.addExtensionProvider("active", "http://jabber.org/protocol/chatstates", new
106.         ChatStateExtension.Provider());
107.         pm.addExtensionProvider("composing", "http://jabber.org/protocol/chatstates", new
108.         ChatStateExtension.Provider());
109.         pm.addExtensionProvider("paused", "http://jabber.org/protocol/chatstates", new
110.         ChatStateExtension.Provider());
111.         pm.addExtensionProvider("inactive", "http://jabber.org/protocol/chatstates", new
112.         ChatStateExtension.Provider());
113.         pm.addExtensionProvider("gone", "http://jabber.org/protocol/chatstates", new
114.         ChatStateExtension.Provider());
115.         // XHTML
116.         pm.addExtensionProvider("html", "http://jabber.org/protocol/xhtml-im", new
117.         XHTMLExtensionProvider());
118.         // Group Chat Invitations
119.         pm.addExtensionProvider("x", "jabber:x:conference", new
120.         GroupChatInvitation.Provider());
121.         // Service Discovery # Items
122.         pm.addIQProvider("query", "http://jabber.org/protocol/disco#items", new
123.         DiscoverItemsProvider());
124.         // Service Discovery # Info
125.         pm.addIQProvider("query", "http://jabber.org/protocol/disco#info", new
126.         DiscoverInfoProvider());
127.         // Data Forms
128.         pm.addExtensionProvider("x", "jabber:x:data", new DataFormProvider());
129.         // MUC User
130.         pm.addExtensionProvider("x", "http://jabber.org/protocol/muc#user", new
131.         MUCUserProvider());
132.         // MUC Admin
133.         pm.addIQProvider("query", "http://jabber.org/protocol/muc#admin", new
134.         MUCAdminProvider());
135.         // MUC Owner
136.         pm.addIQProvider("query", "http://jabber.org/protocol/muc#owner", new
137.         MUCOwnerProvider());
138.         // Delayed Delivery
139.         pm.addExtensionProvider("x", "jabber:x:delay", new DelayInformationProvider());
140.         // Version
141.         try {
142.             pm.addIQProvider("query", "jabber:iq:version",
143.             Class.forName("org.jivesoftware.smackx.packet.Version"));
144.         } catch (ClassNotFoundException e) {
145.             // Not sure what's happening here.
146.         }
147.         // VCard
148.         pm.addIQProvider("vCard", "vcard-temp", new VCardProvider());
149.         // Offline Message Requests
```

```
150.         pm.addIQProvider("offline", "http://jabber.org/protocol/offline", new
151. OfflineMessageRequest.Provider());
152.         // Offline Message Indicator
153.         pm.addExtensionProvider("offline", "http://jabber.org/protocol/offline", new
154. OfflineMessageInfo.Provider());
155.         // Last Activity
156.         pm.addIQProvider("query", "jabber:iq:last", new LastActivity.Provider());
157.         // User Search
158.         pm.addIQProvider("query", "jabber:iq:search", new UserSearch.Provider());
159.         // SharedGroupsInfo
160.         pm.addIQProvider("sharedgroup",
161. "http://www.jivesoftware.org/protocol/sharedgroup", new SharedGroupsInfo.Provider());
162.         // JEP-33: Extended Stanza Addressing
163.         pm.addExtensionProvider("addresses", "http://jabber.org/protocol/address", new
164. MultipleAddressesProvider());
165.         // FileTransfer
166.         pm.addIQProvider("si", "http://jabber.org/protocol/si", new
167. StreamInitiationProvider());
168.         pm.addIQProvider("query", "http://jabber.org/protocol/bytestreams", new
169. BytestreamsProvider());
170.         // Privacy
171.         pm.addIQProvider("query", "jabber:iq:privacy", new Privacy.Provider());
172.         pm.addIQProvider("command", "http://jabber.org/protocol/commands", new
173. AdHocCommandDataProvider());
174.         pm.addExtensionProvider("malformed-action",
175. "http://jabber.org/protocol/commands", new
176. AdHocCommandDataProvider.MalformedActionError());
177.         pm.addExtensionProvider("bad-locale", "http://jabber.org/protocol/commands", new
178. AdHocCommandDataProvider.BadLocaleError());
179.         pm.addExtensionProvider("bad-payload", "http://jabber.org/protocol/commands", new
180. AdHocCommandDataProvider.BadPayloadError());
181.         pm.addExtensionProvider("bad-sessionid", "http://jabber.org/protocol/commands",
182. new AdHocCommandDataProvider.BadSessionIDError());
183.         pm.addExtensionProvider("session-expired", "http://jabber.org/protocol/commands",
184. new AdHocCommandDataProvider.SessionExpiredError());
185.     }
186. }
```

LoginActivity.java

```
1. import org.jivesoftware.smack.XMPPException;
2. import com.itheima.qq.MainActivity;
3. import com.itheima.qq.QQApplication;
4. import com.itheima.qq.R;
```

```
5. import android.content.Intent;
6. import android.content.SharedPreferences;
7. import android.content.SharedPreferences.Editor;
8. import android.os.Bundle;
9. import android.os.Handler;
10. import android.os.Looper;
11. import android.os.Message;
12. import android.text.TextUtils;
13. import android.view.View;
14. import android.widget.EditText;
15. import android.widget.Toast;
16.
17. public class LoginActivity extends BaseActivity {
18.     private EditText et_name;
19.     private EditText et_pwd;
20.     private String name;
21.     private String pwd;
22.     private SharedPreferences sp;
23.     private Handler handler = new Handler() {
24.         public void handleMessage(android.os.Message msg) {
25.             switch (msg.what) {
26.                 case RESULT_OK:
27.                     Toast.makeText(LoginActivity.this, msg.obj + " 登录成功", 0).show();
28.                     //获取自定义的 Application，并将连接对象保存在 Application 中
29.                     QQApplication application = (QQApplication) getApplication();
30.                     application.setXmppConnection(mXmppConnection);
31.                     //进入主界面
32.                     Intent intent = new Intent(LoginActivity.this, MainActivity.class);
33.                     startActivity(intent);
34.                     //关闭登陆页面
35.                     finish();
36.                     break;
37.                 case RESULT_CANCELED:
38.                     Toast.makeText(LoginActivity.this, "登录失败。" + msg.obj, 0).show();
39.                     break;
40.
41.                 default:
42.                     break;
43.             }
44.         };
45.     };
46.
47.     @Override
48.     protected void onCreate(Bundle savedInstanceState) {
49.         super.onCreate(savedInstanceState);
```

```
50.         setContentView(R.layout.activity_login);
51.         sp = getSharedPreferences("config", MODE_PRIVATE);
52.         initView();
53.     }
54.
55.     private void initView() {
56.         et_name = (EditText) findViewById(R.id.et_name);
57.         et_pwd = (EditText) findViewById(R.id.et_pwd);
58.         // 如果 sp 中记录有历史用户名和密码则填充到界面
59.         String name = sp.getString("name", "");
60.         String pwd = sp.getString("pwd", "");
61.         if (!TextUtils.isEmpty(name)) {
62.             et_name.setText(name);
63.         }
64.         if (!TextUtils.isEmpty(pwd)) {
65.             et_pwd.setText(pwd);
66.         }
67.     }
68.
69.     /**
70.      * 登录 Button 绑定的按钮事件
71.      *
72.      * @param view
73.      */
74.     public void login(View view) {
75.         // 首先获取到用户输入的用户名和密码
76.         name = et_name.getText().toString();
77.         pwd = et_pwd.getText().toString();
78.         // 保存到 sp 中
79.         Editor editor = sp.edit();
80.         editor.putString("name", name);
81.         editor.putString("pwd", pwd);
82.         // 一定记得提交
83.         editor.commit();
84.         /**
85.          * 开启一个子线程进行登录，因为登录肯定要连接网络，网络操作必须在子线程中
86.          */
87.         new Thread(new Runnable() {
88.
89.             @Override
90.             public void run() {
91.                 try {
92.                     //连接服务器
93.                     if (!mXmppConnection.isConnected()) {
```



```
94.         mXmppConnection.connect();
95.     }
96. } catch (XMPPException e1) {
97.     e1.printStackTrace();
98.     Looper.prepare();
99.     Toast.makeText(LoginActivity.this, "连接服务器失败。", 0).show();
100.    Looper.loop();
101.    return;
102. }
103. try {
104.     //登录，登录其实也是授权的过程
105.     if (!mXmppConnection.isAuthenticated()) {
106.         mXmppConnection.login(name, pwd); //登录的关键代码
107.     }
108.     //如果授权成功则发送 handler 消息
109.     if (mXmppConnection.isAuthenticated()) {
110.         Message message = Message.obtain();
111.         message.what = RESULT_OK;
112.         //通过连接获取当前登录的用户名
113.         message.obj = mXmppConnection.getUser();
114.         handler.sendMessage(message);
115.     }
116. } catch (XMPPException e) {
117.     e.printStackTrace();
118.     Message message = Message.obtain();
119.     message.what = RESULT_CANCELED;
120.     message.obj = e;
121.     handler.sendMessage(message);
122. }
123. }
124. }).start();
125.
126. }
127. /**
128.  * 绑定的界面中 button 事件<br>
129.  * 跳转到注册界面
130.  *
131.  * @param view
132.  */
133. public void gotoRegist(View view) {
134.     //跳转到注册界面
135.     Intent registIntent = new Intent(this, RegistActivity.class);
136.     startActivity(registIntent);
137. }
138. }
```

在上面的代码中我们将登陆成功后获取的 `connection` 对象设置到了 `Application` 中，这里用到了自定义的 `Application`，并且在清单文件中已经配置。

QQApplication.java

```
1. public class QQApplication extends Application {
2.     private XMPPConnection mXmppConnection;
3.     @Override
4.     public void onCreate() {
5.         super.onCreate();
6.     }
7.     public XMPPConnection getXmppConnection(){
8.         return mXmppConnection;
9.     }
10.    public void setXmppConnection(XMPPConnection xmppConnection){
11.        this.mXmppConnection = xmppConnection;
12.    }
13. }
```

AndroidManifest.xml

```
1. <manifest xmlns:android="http://schemas.android.com/apk/res/android"
2.     package="com.itheima.qq"
3.     android:versionCode="1"
4.     android:versionName="1.0" >
5.     <uses-sdk
6.         android:minSdkVersion="8"
7.         android:targetSdkVersion="21" />
8.     <uses-permission android:name="android.permission.INTERNET" />
9.     <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
10.    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
11.    <application
12.        android:name="com.itheima.qq.QQApplication"
13.        android:allowBackup="true"
14.        android:icon="@drawable/qq"
15.        android:label="@string/app_name"
16.        android:theme="@style/AppTheme" >
17.        <activity
18.            android:name="com.itheima.qq.activity.SplashActivity"
19.            android:label="@string/app_name"
20.            android:theme="@android:style/Theme.NoTitleBar.Fullscreen" >
21.            <intent-filter>
22.                <action android:name="android.intent.action.MAIN" />
23.                <category android:name="android.intent.category.LAUNCHER" />
```

```
24.         </intent-filter>
25.     </activity>
26.     <activity android:name="com.itheima.qq.MainActivity" />
27.     <activity android:name="com.itheima.qq.activity.LoginActivity" />
28.     <activity android:name="com.itheima.qq.activity.RegistActivity" />
29.     <activity android:name="com.itheima.qq.activity.ChatActivity" />
30.     <service android:name="com.itheima.qq.ChatService" />
31. </application>
32.
33. </manifest>
```

清单文件中高亮显示的两个部分，因为需要连接服务器，因此需要联网权限。

5.3.3 注册

注册功能核心代码：

```
1. //创建注册对象，用于封装注册参数
2.         Registration registration = new Registration();
3.         //设置注册属于设置属性，因此这里设置类型为 set
4.         registration.setType(Type.SET);
5.         //设置用户名和密码
6.         registration.setUsername(name);
7.         registration.setPassword(pwd);
8.         /**
9.          * 注册信息封装好之后其实就可以发送了可以直接调用<br>
10.          * mMppConnection.sendPacket(registration);方法<br>
11.          * 但是上面的方法并没有返回值，注册是否成功我们不清楚<br>
12.          * 因此我们需要开启一个数据包收集器来手机服务返回的信息
13.          *
14.          */
15.         //定义一个数据包过滤器
16.         /**
17.          * AndFilter 是一个组合过滤器，形参是可变参数，可以传递多种 PacketFilter 的子类
18.          <br>
19.          * 我们需要要过滤的原则是根据注册数据包的 id 和数据包类型<br>
20.          *
21.          */
22.         PacketFilter filter = new AndFilter(new
23. PacketIDFilter(registration.getPacketID()),new PacketTypeFilter(IQ.class));
24.         //创建一个数据包收集器，形参为数据包过滤器
25.         PacketCollector collector =
26. mMppConnection.createPacketCollector(filter);
27.         /**
28.          * 这个 api 并没有提供最简单的 regist 方法。而是用了很负责的 api，用户体验不佳。
```

```
29.         */
30.         mXmppConnection.sendPacket(registration);
31.     /**
32.      * 上面的代码已经发送过注册数据包了，接下来我们就可以收集服务器的返回值了<br>
33.      * 形参为网络超时时间默认 5s
34.      */
35.         Packet packet =
36. collector.nextResult(SmackConfiguration.getPacketReplyTimeout());
37.     /**
38.      * 将数据包强转为 IQ，因为我们的过滤器已经限定了是 IQ 类型的数据包才收集，因此我们
39. 可以大胆的强转
40.      */
41.         IQ result = (IQ)packet;
42. //收集到数据后就可以将收集器关闭了，不然可能把其他符合条件的数据也收集来了
43.         collector.cancel();
44.         //如果 IQ 为空则代表请求失败 这里代码我写的不太好了，应该先判断 packet 是否为空，
45. 不为空再强转，这样才能避免空指针
46.         if (result==null) {
47.             Message message = Message.obtain();
48.             message.what = RESULT_CANCELED;
49.             handler.sendMessage(message);
50.             return;
51. //如果返回的数据类型为 IQ.Type.Result 则代表成功
52. }else if (result.getType().equals(IQ.Type.RESULT)) {
53.     Message message = Message.obtain();
54.     message.what = RESULT_OK;
55.     handler.sendMessage(message);
56.     return;
57. }else {
58.     Message message = Message.obtain();
59.     //否则代表失败，那么我们就收集失败码
60.     int errorCode = result.getError().getCode();
61.     message.what = RESULT_CANCELED;
62.     //如果失败码是 409，那么代表用户已经被注册
63.     if (409==errorCode) {
64.         message.obj = "该用户名已经被注册，请换一个名字吧";
65.         handler.sendMessage(message);
66.         return;
67.     }else {
68.         message.obj = result.getError();
69.         handler.sendMessage(message);
70.         return;
71.     }
72. }
```

activity_regist.xml

```
1. <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2.     xmlns:tools="http://schemas.android.com/tools"
3.     android:layout_width="match_parent"
4.     android:layout_height="wrap_content"
5.     android:background="@drawable/login_background" >
6.
7.     <LinearLayout
8.         android:layout_width="match_parent"
9.     android:layout_height="match_parent"
10.         android:background="@drawable/login_background"
11.         android:orientation="vertical" >
12.
13.         <ImageView
14.             android:id="@+id/iv_touxiang"
15.             android:layout_width="wrap_content"
16.             android:layout_height="wrap_content"
17.             android:layout_gravity="center_horizontal"
18.             android:layout_marginTop="40dp"
19.             android:contentDescription="@null"
20.             android:src="@drawable/login_default_avatar" />
21.
22.         <EditText
23.             android:id="@+id/et_name"
24.             android:layout_width="match_parent"
25.             android:layout_height="40dp"
26.             android:layout_marginLeft="5dp"
27.             android:layout_marginRight="5dp"
28.             android:layout_marginTop="30dp"
29.             android:background="#FFFFFF"
30.             android:hint="请输入账号"
31.             android:paddingLeft="5dp"
32.             android:textSize="20sp" />
33.
34.         <View
35.             android:layout_width="match_parent"
36.             android:layout_height="1dp"
37.             android:background="#55AABCC" />
38.
39.         <EditText
40.             android:id="@+id/et_pwd"
41.             android:layout_width="match_parent"
42.             android:layout_height="40dp"
43.             android:layout_marginLeft="5dp"
```

```
44.         android:layout_marginRight="5dp"
45.         android:background="#FFFFFF"
46.         android:hint="请输入密码"
47.         android:inputType="textPassword"
48.         android:paddingLeft="5dp"
49.         android:textSize="20sp" />
50.
51.     <Button
52.         android:id="@+id/btn_regist"
53.         android:layout_width="match_parent"
54.         android:layout_height="wrap_content"
55.         android:layout_margin="15dp"
56.         android:background="@color/title_layout"
57.         android:onClick="regist"
58.         android:text="注册"
59.         android:textColor="#FFFFFF"
60.         android:textSize="20sp" />
61. </LinearLayout>
62.
63. </RelativeLayout>
```

RegistActivity.java

```
1. import org.jivesoftware.smack.PacketCollector;
2. import org.jivesoftware.smack.SmackConfiguration;
3. import org.jivesoftware.smack.filter.AndFilter;
4. import org.jivesoftware.smack.filter.PacketFilter;
5. import org.jivesoftware.smack.filter.PacketIDFilter;
6. import org.jivesoftware.smack.filter.PacketTypeFilter;
7. import org.jivesoftware.smack.packet.IQ.Type;
8. import org.jivesoftware.smack.packet.IQ;
9. import org.jivesoftware.smack.packet.Packet;
10. import org.jivesoftware.smack.packet.Registration;
11. import com.itheima.qq.R;
12. import android.content.SharedPreferences;
13. import android.content.SharedPreferences.Editor;
14. import android.os.Bundle;
15. import android.os.Handler;
16. import android.os.Looper;
17. import android.os.Message;
18. import android.view.View;
19. import android.widget.EditText;
20. import android.widget.Toast;
```

```
21.
22. public class RegistActivity extends BaseActivity {
23.     private EditText et_name;
24.     private EditText et_pwd;
25.     private String name;
26.     private String pwd;
27. private SharedPreferences sp;
28.     private static final int RESULT_NO_RESPONSE = 1;
29.     private Handler handler = new Handler() {
30.         public void handleMessage(android.os.Message msg) {
31.             switch (msg.what) {
32.                 case RESULT_NO_RESPONSE:
33.                     Toast.makeText(RegistActivity.this, "服务器未响应，请稍后再试", 0).show();
34.                     break;
35.                 case RESULT_OK:
36.                     Toast.makeText(RegistActivity.this, "注册成功", 0).show();
37.                     finish();
38.                     break;
39.                 case RESULT_CANCELED:
40.                     Toast.makeText(RegistActivity.this, "注册失败。" + msg.obj, 0).show();
41.                     break;
42.
43.                 default:
44.                     break;
45.             }
46.         };
47.     };
48.
49.     @Override
50.     protected void onCreate(Bundle savedInstanceState) {
51.         super.onCreate(savedInstanceState);
52.         setContentView(R.layout.activity_regist);
53.         sp = getSharedPreferences("config", MODE_PRIVATE);
54.         initView();
55.     }
56.
57.     private void initView() {
58.         //获取到注册名和密码
59.         et_name = (EditText) findViewById(R.id.et_name);
60.         et_pwd = (EditText) findViewById(R.id.et_pwd);
61.     }
62.     /**
63.      * 绑定注册按钮事件
64.      * @param view
65.      */
```

```
66.     public void regist(View view) {
67.         //获取用户的数据
68.         name = et_name.getText().toString();
69.     pwd = et_pwd.getText().toString();
70.         //保存到 sp 中
71.         Editor editor = sp.edit();
72.         editor.putString("name", name);
73.         editor.putString("pwd", pwd);
74.         editor.commit();
75.         //因为注册需要联网，因此放在子线程中
76.         new Thread(new Runnable() {
77.
78.             @Override
79.             public void run() {
80.                 try {
81.                     //如果没有连接服务器则连接服务器
82.                     if (!mXmppConnection.isConnected()) {
83.                         mXmppConnection.connect();
84.                     }
85.                 } catch (Exception e1) {
86.                     e1.printStackTrace();
87.                     Looper.prepare();
88.                     Toast.makeText(RegistActivity.this, "连接服务器失败。", 0).show();
89.                     Looper.loop();
90.                     return;
91.                 }
92.                 try {
93.                     //创建注册对象，用于封装注册参数
94.                     Registration registration = new Registration();
95.                     //设置注册属于设置属性，因此这里设置类型为 set
96.                     registration.setType(Type.SET);
97.                     //设置用户名和密码
98.                     registration.setUsername(name);
99.                     registration.setPassword(pwd);
100.                    /**
101.                        * 注册信息封装好之后其实就可以发送了可以直接调用<br>
102.                        * mXmppConnection.sendPacket(registration);方法<br>
103.                        * 但是上面的方法并没有返回值，注册是否成功我们不清楚<br>
104.                        * 因此我们需要开启一个数据包收集器来手机服务返回的信息
105.                        *
106.                        */
107.                    //定义一个数据包过滤器
108.                    /**
109.                        * AndFilter 是一个组合过滤器，形参是可变参数，可以传递多种 PacketFilter 的子类
```



```
110.             * 我们需要要过滤的原则是根据注册数据包的 id 和数据包类型
111.         *
112.             */
113.             PacketFilter filter = new AndFilter(new
114.         PacketIDFilter(registration.getPacketID()),new PacketTypeFilter(IQ.class));
115.             //创建一个数据包收集器，形参为数据包过滤器
116.             PacketCollector collector =
117. mXmppConnection.createPacketCollector(filter);
118.             /*
119.         * 这个 api 并没有提供最简单的 regist 方法。而是用了很负责的 api，用户体验不佳。
120.     */
121.             mXmppConnection.sendPacket(registration);
122.             /**
123.         * 上面的代码已经发送过注册数据包了，接下来我们就可以收集服务器的返回值了
124.         * 形参为网络超时时间默认 5s
125.     */
126.             Packet packet =
127. collector.nextResult(SmackConfiguration.getPacketReplyTimeout());
128.             /**
129.         * 将数据包强转为 IQ，因为我们的过滤器已经限定了是 IQ 类型的数据包才收集，因此我们
130. 可以大胆的强转
131.     */
132.             IQ result = (IQ)packet;
133.             //收集到数据后就可以将收集器关闭了，不然可能把其他符合条件的数据也收集来了
134.             collector.cancel();
135.             //如果 IQ 为空则代表请求失败 这里代码我写的不太好了，应该先判断 packet 是否为空，
136. 不为空再强转，这样才能避免空指针
137.             if (result==null) {
138.                 Message message = Message.obtain();
139.                 message.what = RESULT_CANCELED;
140.                 handler.sendMessage(message);
141.                 return;
142.             //如果返回的数据类型为 IQ.Type.Result 则代表成功
143.             }else if (result.getType().equals(IQ.Type.RESULT)) {
144.                 Message message = Message.obtain();
145.                 message.what = RESULT_OK;
146.                 handler.sendMessage(message);
147.                 return;
148.             }else {
149.                 Message message = Message.obtain();
150.                 //否则代表失败，那么我们就收集失败码
151.                 int errorCode = result.getError().getCode();
152.                 message.what = RESULT_CANCELED;
153.                 //如果失败码是 409，那么代表用户已经被注册
154.                 if (409==errorCode) {
```

```
155.             message.obj = "该用户名已经被注册，请换一个名字吧";
156.             handler.sendMessage(message);
157.             return;
158.         }else {
159.             message.obj = result.getError();
160.             handler.sendMessage(message);
161.             return;
162.         }
163.     }
164.     } catch (Exception e) {
165.         e.printStackTrace();
166.         Message message = Message.obtain();
167.         message.what = RESULT_CANCELED;
168.         message.obj = e;
169.         handler.sendMessage(message);
170.     }
171. }
172. }).start();
173. }
174. }
```

5.3.4 主界面

登录之后就进入主界面了。

activity_main.xml

```
1. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2.     xmlns:tools="http://schemas.android.com/tools"
3.     android:layout_width="match_parent"
4.     android:layout_height="match_parent"
5.     android:orientation="vertical" >
6.     <LinearLayout
7.         android:layout_width="match_parent"
8.         android:layout_height="60dp"
9.         android:background="@color/title_layout"
10.        android:gravity="center"
11.        android:orientation="horizontal" >
12.     <TextView
13.         android:id="@+id/tv_title"
14.         style="@style/TitleStyle"
15.         android:padding="5dp"
16.         android:text="@string/title_message" />
```

```
17.     </LinearLayout>
18.
19.     <FrameLayout
20.         android:id="@+id/fl_content"
21.         android:layout_width="match_parent"
22.         android:layout_height="0dp"
23.         android:layout_weight="1" >
24. </FrameLayout>
25.
26. <RelativeLayout
27.     android:layout_width="match_parent"
28.     android:layout_height="@dimen/activity_main_tab_height"
29.     android:gravity="center_horizontal"
30.     android:orientation="horizontal" >
31.
32.     <View
33.         android:layout_width="match_parent"
34.         android:layout_height="1dp"
35.         android:layout_alignParentTop="true"
36.         android:background="#55000000" />
37.
38.     <RadioGroup
39.         android:layout_alignParentBottom="true"
40.         android:id="@+id/rg_group"
41.         android:layout_width="match_parent"
42.         android:layout_height="wrap_content"
43.         android:orientation="horizontal"
44.         android:gravity="center_horizontal"
45.     >
46.
47.         <RadioButton
48.             android:id="@+id/session_rbn"
49.             android:layout_width="@dimen/message_icon_width"
50.             android:layout_height="@dimen/message_icon_height"
51.             android:layout_marginBottom="@dimen/activity_main_tab_margin_top_and_bottom"
52.             android:layout_marginTop="@dimen/activity_main_tab_margin_top_and_bottom"
53.             android:layout_marginRight="30dp"
54.             android:background="@drawable/message_icon"
55.             android:button="@null" />
56.         <RadioButton
57.             android:id="@+id/contact_rbn"
58.             android:layout_width="@dimen/contact_icon_width"
59.             android:layout_height="@dimen/contact_icon_height"
60.             android:layout_marginBottom="@dimen/activity_main_tab_margin_top_and_bottom"
61.             android:layout_marginTop="@dimen/activity_main_tab_margin_top_and_bottom"
```

```
62.         android:background="@drawable/contact_icon"
63.         android:button="@null" />
64.     <RadioButton
65.         android:id="@+id/dongtai_rbn"
66.         android:layout_width="@dimen/dongtai_icon_width"
67.         android:layout_height="@dimen/dongtai_icon_height"
68.         android:layout_marginBottom="@dimen/activity_main_tab_margin_top_and_bottom"
69.         android:layout_marginTop="@dimen/activity_main_tab_margin_top_and_bottom"
70.         android:background="@drawable/dongtai_icon"
71.         android:layout_marginLeft="30dp"
72.         android:button="@null" />
73.     </RadioGroup>
74. </RelativeLayout>
75.
76. </LinearLayout>
```

MainActivity.java

```
1. import com.itheima.qq.fragment.ContactFragment;
2. import com.itheima.qq.fragment.DongtaiFragment;
3. import com.itheima.qq.fragment.SessionFrament;
4. import android.content.Intent;
5. import android.os.Bundle;
6. import android.support.v4.app.FragmentActivity;
7. import android.support.v4.app.FragmentManager;
8. import android.support.v4.app.FragmentTransaction;
9. import android.widget.RadioButton;
10. import android.widget.RadioGroup;
11. import android.widget.RadioGroup.OnCheckedChangeListener;
12. import android.widget.TextView;
13. public class MainActivity extends FragmentActivity implements OnCheckedChangeListener {
14. public static final int COUNT = 3;
15.     private RadioButton session_rbn;
16.     private RadioButton contact_rbn;
17.     private RadioButton dongtai_rbn;
18.     private TextView title_tv;
19.     private RadioGroup radioGroup;
20.     private FragmentManager fragmentManager;
21.     private SessionFrament sessionFrament;
22.     private ContactFragment contactFragment;
23.     private static final String TAG_SESSION_FRAGMENT = "SessionFragment";
24.     private static final String TAG_CONTACT_FRAGMENT = "ContactFragment";
25.     private static final String TAG_DONGTAI_FRAGMENT = "DongtaiFragment";
26.     private DongtaiFragment dongtaiFragment;
```

```
27.
28.     @Override
29.     protected void onCreate(Bundle savedInstanceState) {
30.         super.onCreate(savedInstanceState);
31.         setContentView(R.layout.activity_main);
32.     }
33.
34.     @Override
35.     protected void onResume() {
36.         super.onResume();
37.         //初始化控件
38.         initView();
39.         //初始化监听器监听切换 Fragment 事件
40.         initListener();
41.         //初始化 Fragment
42.         initData();
43.         //开启聊天服务进程，监听聊天消息
44.         startChatService();
45.     }
46.
47.     private void initListener() {
48.         radioGroup.setOnCheckedChangeListener(this);
49.     }
50.
51.     private void startChatService() {
52.         Intent intent = new Intent(this, ChatService.class);
53.         startService(intent);
54.     }
55.
56.     private void initData() {
57. fragmentManager = getSupportFragmentManager();
58.         sessionFragment = new SessionFrament();
59.         contactFragment = new ContactFragment();
60.         dongtaiFragment = new DongtaiFragment();
61.         //默认选中消息 Fragment
62.         radioGroup.check(R.id.session_rbn);
63.
64.     }
65.
66.     private void initView() {
67.         title_tv = (TextView) findViewById(R.id.tv_title);
68.         session_rbn = (RadioButton) findViewById(R.id.session_rbn);
69.         contact_rbn = (RadioButton) findViewById(R.id.contact_rbn);
70.         dongtai_rbn = (RadioButton) findViewById(R.id.dongtai_rbn);
71.         radioGroup = (RadioGroup) findViewById(R.id.rg_group);
```

```
72.     }
73.
74.     @Override
75.     public void onCheckedChanged(RadioGroup group, int checkedId) {
76.         // 把其他选中状态取消
77.         session_rbn.setChecked(checkedId == R.id.session_rbn);
78.         contact_rbn.setChecked(checkedId == R.id.contact_rbn);
79.         dongtai_rbn.setChecked(checkedId == R.id.dongtai_rbn);
80.         if (checkedId == R.id.session_rbn) {
81.             title_tv.setText(getResources().getString(R.string.title_message));
82.             FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
83.             fragmentTransaction.replace(R.id.fl_content, sessionFragment,
84. TAG_SESSION_FRAGMENT);
85.             fragmentTransaction.commit();
86.         } else if (checkedId == R.id.contact_rbn) {
87.             title_tv.setText(getResources().getString(R.string.title_contact));
88.             FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
89.             fragmentTransaction.replace(R.id.fl_content, contactFragment,
90. TAG_CONTACT_FRAGMENT);
91.             fragmentTransaction.commit();
92.         } else if (checkedId == R.id.dongtai_rbn) {
93.             title_tv.setText(getResources().getString(R.string.title_dongtai));
94.             FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
95.             fragmentTransaction.replace(R.id.fl_content, dongtaiFragment,
96. TAG_DONGTAI_FRAGMENT);
97.             fragmentTransaction.commit();
98.         }
99.     }
100. }
```

5.3.5 退出登录

退出登录的功能是在动态中实现的。消息、联系人、动态都是 **Fragment** 实现的。核心代码：

```
1. if (xmppConnection.isConnected()) {
2.     if (xmppConnection.isAuthenticated()) {
3.         try {
4.             xmppConnection.disconnect();
5.         } catch (Exception e) {
6.             e.printStackTrace();
7.             Looper.prepare();
8.             Toast.makeText(application, "注销失败"+e, 0).show();
9.         }
10.    }
```

```
9.             Looper.loop();
10.             getActivity().finish();
11.             return ;
12.         }
13.     }
14. }
```

DongtaiFragment.java

```
1. public class DongtaiFragment extends Fragment {
2.     private TextView tv_name;
3.     private RelativeLayout rl_logout;
4.
5.     @Override
6.     public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
7. savedInstanceState) {
8.         View view = initView();
9.         return view;
10.    }
11.
12.    private View initView() {
13.        View view = View.inflate(getActivity(), R.layout.fragment_dongtai2, null);
14.        SharedPreferences sp = getActivity().getSharedPreferences("config",
15. Context.MODE_PRIVATE);
16.        String name = sp.getString("name", "");
17.        tv_name = (TextView) view.findViewById(R.id.tv_name);
18.        rl_logout = (RelativeLayout) view.findViewById(R.id.rl_logout);
19.        if (!TextUtils.isEmpty(name)) {
20.            tv_name.setText(name);
21.        }
22.        rl_logout.setOnClickListener(new OnClickListener() {
23.            @Override
24.            public void onClick(View v) {
25.                final QQApplication application = (QQApplication)
26. getActivity().getApplication();
27.                final XMPPConnection xmppConnection = application.getXmppConnection();
28.                new Thread(new Runnable() {
29.                    @Override
30.                    public void run() {
31.                        if (xmppConnection.isConnected()) {
32.                            if (xmppConnection.isAuthenticated()) {
33.                                try {
34.                                    xmppConnection.disconnect();
35.                                } catch (Exception e) {
36.                                    e.printStackTrace();

```

```
37.         Looper.prepare();
38.         Toast.makeText(application, "注销失败"+e, 0).show();
39.         Looper.loop();
40.         getActivity().finish();
41.         return ;
42.     }
43.     Looper.prepare();
44.     Toast.makeText(application, "注销成功", 0).show();
45.     getActivity().startActivity(new Intent(application,
46. LoginActivity.class));
47.     getActivity().finish();
48.     Looper.loop();
49.     }
50.     }
51.     }
52.     }).start();
53.     }
54.     });
55.     return view;
56.     }
57. }
```

5.3.6 获取联系人功能

联系人界面对应的是 **ContactFragment**，这个界面包含了获取联系人，添加新朋友，添加新群组等三个功能。我先将上面三个功能的核心代码列出来，然后在把 **ContactFragment.java** 代码给列出来。

获取到分组，然后每个分组里面有联系人。界面用的是一个 **ExpendableListView**。

获取联系人核心代码：

```
1.  thread = new Thread(new Runnable() {
2.      @Override
3.      public void run() {
4.          FragmentActivity activity = getActivity();
5.          if (activity==null) {
6.              return ;
7.          }
8.          QQApplication application = (QQApplication) activity.getApplication();
9.          xmppConnection = application.getXmppConnection();
10.         Roster roster = xmppConnection.getRoster();
11.         Collection<RosterGroup> groups = roster.getGroups();
12.         Iterator<RosterGroup> iterator = groups.iterator();
13.         rosterGroups.clear();
```



```
14.         while (iterator.hasNext()) {
15.             RosterGroup rosterGroup = (RosterGroup) iterator.next();
16.             rosterGroups.add(rosterGroup);
17.         }
18.         handler.sendMessage(1);
19.     }
20. });
```

添加新朋友核心代码：

```
1. public static boolean addUsers(Roster roster, String userName, String name, String groupName)
2. {
3.     try {
4.         roster.createEntry(userName, name, new String[] { groupName });
5.         return true;
6.     } catch (Exception e) {
7.         e.printStackTrace();
8.         Log.e("XmppUtils", "添加好友异常: " + e.getMessage());
9.     }
```

添加新群组核心代码：

```
1. public static RosterGroup addGroup(Roster roster, String groupName) {
2.     try {
3.         return roster.createGroup(groupName);
4.     } catch (Exception e) {
5.         e.printStackTrace();
6.         Log.e("XmppUtils", "创建分组异常: " + e.getMessage());
7.         return null;
8.     }
```

联系人布局 ragment_contact.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:layout_width="match_parent"
4.     android:layout_height="match_parent"
5.     android:orientation="vertical" >
6.     <LinearLayout
7.         android:id="@+id/rl_roster"
8.         android:layout_width="match_parent"
9.         android:layout_height="wrap_content"
10.        android:orientation="horizontal"
11.    >
12.     <LinearLayout
```

```
13.         android:id="@+id/ll_new_friend"
14.         android:layout_width="0dp"
15.         android:layout_weight="1"
16.         android:layout_height="wrap_content"
17.         android:orientation="vertical"
18.         android:gravity="center_horizontal">
19.     <ImageView
20.         android:layout_width="40dp"
21.         android:layout_height="40dp"
22.         android:src="@drawable/new_friend"
23.     />
24.     <TextView
25.         android:layout_width="wrap_content"
26.         android:layout_height="wrap_content"
27.         android:text="新朋友"
28.     />
29. </LinearLayout>
30. <LinearLayout
31.     android:id="@+id/ll_new_group"
32.     android:layout_width="0dp"
33.     android:layout_weight="1"
34.     android:layout_height="wrap_content"
35.     android:orientation="vertical"
36.     android:gravity="center_horizontal"
37. >
38.     <ImageView
39.         android:layout_width="40dp"
40.         android:layout_height="40dp"
41.         android:src="@drawable/new_group"
42.     />
43.     <TextView
44.         android:layout_width="wrap_content"
45.         android:layout_height="wrap_content"
46.         android:text="新群组"
47.     />
48. </LinearLayout>
49. </LinearLayout>
50. <View
51.     android:layout_width="match_parent"
52.     android:layout_height="0.5dp"
53.     android:background="@color/devide_line"
54. />
55. <ExpandableListView
56.     android:id="@+id/lv_roster"
```

```
57.         android:childDivider="@color/devide_line"
58.         android:layout_width="match_parent"
59.         android:cacheColorHint="#00000000"
60.         android:listSelector="#00000000"
61.         android:groupIndicator="@null"
62.         android:layout_height="0dp"
63.         android:layout_weight="1"
64.     ></ExpandableListView>
65. </LinearLayout>
```

联系人代码 ContactFragment.java

```
1. public class ContactFragment extends Fragment {
2.     private ExpandableListView lv_roster;
3.     private MyAdapter adapter;
4.     private XMPPConnection xmppConnection;
5.     private ArrayList<RosterGroup> rosterGroups = new ArrayList<RosterGroup>();
6.     private LinearLayout ll_new_friend;
7.     private LinearLayout ll_new_group;
8.     private Thread thread;
9.     private Handler handler = new Handler(){
10.         @Override
11.         public void handleMessage(android.os.Message msg) {
12.             if (msg.what==0) {
13.                 initData();
14.             }else if (msg.what==1) {
15.                 adapter.notifyDataSetChanged();
16.             }
17.         };
18.     };
19.
20.     @Override
21.     public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
22. savedInstanceState) {
23.         View view = initView();
24.         initData();
25.         return view;
26.     }
27.
28.     private View initView() {
29.         View view = View.inflate(getActivity(), R.layout.fragment_contact, null);
30.         ll_new_friend = (LinearLayout) view.findViewById(R.id.ll_new_friend);
31.         ll_new_group = (LinearLayout) view.findViewById(R.id.ll_new_group);
32.         ll_new_friend.setOnClickListener(new OnClickListener() {
```

```
33.         @Override
34.         public void onClick(View v) {
35.             //创建一个自定义布局的 Dialog
36.             AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
37.             View view = View.inflate(getActivity(), R.layout.dialog_new_friend, null);
38.             builder.setView(view);
39.             builder.setTitle("添加新朋友");
40.             Button button = (Button) view.findViewById(R.id.btn_add);
41.             final EditText et_name = (EditText) view.findViewById(R.id.et_name);
42.             final AlertDialog dialog = builder.create();
43.             dialog.setCanceledOnTouchOutside(false);
44.             button.setOnClickListener(new OnClickListener() {
45.
46.                 @Override
47.                 public void onClick(View v) {
48.                     String name = et_name.getText().toString();
49.                     addNewFriend(name, new AddListener() {
50.
51.                         @Override
52.                         public void onAddSuccess() {
53.                             handler.sendMessage(0);
54.                             //添加成功后取消 Dialog
55.                             dialog.dismiss();
56.
57.                             @Override
58.                             public void onAddFailure(String msg) {
59.                                 Looper.prepare();
60.                                 Toast.makeText(getActivity(), "添加新朋友失败。"+msg,
61.                                     0).show();
62.                                 Looper.loop();
63.                             }
64.                         });
65.                     }
66.
67.                 });
68.             dialog.show();
69.         }
70.     });
71.     ll_new_group.setOnClickListener(new OnClickListener() {
72.
73.         @Override
74.         public void onClick(View v) {
75.             AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
76.             View view = View.inflate(getActivity(), R.layout.dialog_new_group, null);
77.             builder.setView(view);
```

```
77.         builder.setTitle("添加新群组");
78.         final AlertDialog dialog = builder.create();
79.         Button button = (Button) view.findViewById(R.id.btn_add);
80.         final EditText et_name = (EditText) view.findViewById(R.id.et_name);
81.         dialog.setCanceledOnTouchOutside(false);
82.         button.setOnClickListener(new OnClickListener() {
83.
84.             @Override
85.             public void onClick(View v) {
86.                 String name = et_name.getText().toString();
87.                 addNewGroup(name, new AddListener() {
88.
89.                     @Override
90.                     public void onAddSuccess() {
91.                         dialog.dismiss();
92.                         handler.sendMessage(0);
93.                     }
94.
95.                     @Override
96.                     public void onAddFailure(String message) {
97.                         Looper.prepare();
98.                         Toast.makeText(getActivity(), "添加新分组失败。"+message,
99. 0).show();
100.
101.                         Looper.loop();
102.                     }
103.                 });
104.             }
105.         });
106.         dialog.show();
107.     }
108. });
109. lv_roster = (ExpandableListView) view.findViewById(R.id.lv_roster);
110. lv_roster.setSmoothScrollbarEnabled(false);
111. //点击子 ListView 的条目，其实也就是点击联系人的时候跳转到聊天界面
112. lv_roster.setOnChildClickListener(new OnChildClickListener() {
113.     @Override
114.     public boolean onChildClick(ExpandableListView parent, View v, int groupPosition,
115.     Int childPosition, long id) {
116.         RosterGroup rosterGroup = rosterGroups.get(groupPosition);
117.         String user = new
118.         ArrayList<>(rosterGroup.getEntries()).get(childPosition).getUser();
119.         Intent chatIntent = new Intent(getActivity(), ChatActivity.class);
120.         chatIntent.putExtra("user", user);
121.         startActivity(chatIntent);
```

```
122.         return true;
123.     }
124. });
125.     return view;
126. }
127.     public void initData() {
128.         adapter = new MyAdapter();
129.         lv_roster.setAdapter(adapter);
130.         //在子线程中请求联系人
131.         thread = new Thread(new Runnable() {
132.             @Override
133.             public void run() {
134.                 FragmentActivity activity = getActivity();
135.                 if (activity==null) {
136.                     return ;
137.                 }
138.                 QQApplication application = (QQApplication) activity.getApplication();
139.                 xmppConnection = application.getXmppConnection();
140.                 //获取到花名册对象
141.                 Roster roster = xmppConnection.getRoster();
142.                 //获取到所有的分组
143.                 Collection<RosterGroup> groups = roster.getGroups();
144.                 Iterator<RosterGroup> iterator = groups.iterator();
145.                 rosterGroups.clear();
146.                 while (iterator.hasNext()) {
147.                     RosterGroup rosterGroup = (RosterGroup) iterator.next();
148.                     rosterGroups.add(rosterGroup);
149.                 }
150.                 handler.sendMessage(1);
151.             }
152.         });
153.         if (thread.isAlive()) {
154.             return ;
155.         }else {
156.             thread.start();
157.         }
158.     }
159.     //添加新朋友
160.     private void addNewFriend(final String name,final AddListener listener) {
161.         final Roster roster = xmppConnection.getRoster();
162.         new Thread(new Runnable() {
163.             @Override
164.             public void run() {
165.                 try {
```

```
166.         if(!XmppUtils.searchUsers(xmppConnection, name)){
167.             if (listener!=null) {
168.                 listener.onAddFailure(name+"不存在");
169.             }
170.             return ;
171.         }
172.         //先判断该用户是否存在
173.         XmppUtils.addGroup(roster, "我的好友");//先默认创建一个分组
174.         XmppUtils.addUsers(roster,name+"@"+xmppConnection.getServiceName(),
175. name,"我的好友");
176.         if (listener!=null) {
177.             listener.onAddSuccess();
178.         }
179.     } catch (Exception e) {
180.         e.printStackTrace();
181.         if(listener!=null){
182.             listener.onAddFailure(e.toString());
183.         }
184.     }
185. }
186. }).start();
187. }
188. //添加新群组
189. private void addNewGroup(final String name,final AddListener listener) {
190.     final Roster roster = xmppConnection.getRoster();
191.     new Thread(new Runnable() {
192.         @Override
193.         public void run() {
194.             RosterGroup group = XmppUtils.addGroup(roster, name);
195.             if(group==null){
196.                 if (listener!=null) {
197.                     listener.onAddFailure("创建分组失败。");
198.                 }
199.             }
200.             try {
201.                 /**
202.                  * 书写格式，注意书写格式!!!
203.                  */
204.                 RosterEntry rosterEntry = roster.getEntry("admin");
205.                 if (rosterEntry==null) {
206.                     rosterEntry =
207. roster.getEntry("admin@"+xmppConnection.getServiceName());
208.                 }
209.                 if (rosterEntry!=null) {
210.                     group.addEntry(rosterEntry);
```

```
211.         if (listener!=null) {
212.             listener.onAddSuccess();
213.         }
214.     }else {
215.         if (listener!=null) {
216.             listener.onAddFailure("创建分组失败。");
217.         }
218.     }
219.
220.     } catch (Exception e) {
221.         e.printStackTrace();
222.         if (listener!=null) {
223.             listener.onAddFailure(e.toString());
224.         }
225.     }
226. }
227. }).start();
228. }
229.
230. class MyAdapter extends BaseExpandableListAdapter {
231.
232.     @Override
233.     public int getGroupCount() {
234.         return rosterGroups.size();
235.     }
236.
237.     @Override
238.     public int getChildrenCount(int groupPosition) {
239.         RosterGroup rosterGroup = rosterGroups.get(groupPosition);
240.         //根据组群获取该组群下的联系人数量
241.         return rosterGroup.getEntryCount();
242.     }
243.     @Override
244.     public Object getGroup(int groupPosition) {
245.         return rosterGroups.get(groupPosition);
246.     }
247.
248.     @Override
249.     public Object getChild(int groupPosition, int childPosition) {
250.
251.         return rosterGroups.get(groupPosition).getEntries();
252.     }
253.
254.     @Override
```



```
255.         public long getGroupId(int groupPosition) {
256.             return groupPosition;
257.         }
258.
259.         @Override
260.         public long getChildId(int groupPosition, int childPosition) {
261.             return groupPosition * 100000 + childPosition;
262.         }
263.
264.         @Override
265.         public boolean hasStableIds() {
266.             return false;
267.         }
268.
269.         @Override
270.         public View getGroupView(int groupPosition, boolean isExpanded, View convertView,
271. ViewGroup parent) {
272.             if (convertView == null) {
273.                 convertView = View.inflate(getActivity(),
274. R.layout.list_item_roaster_group, null);
275.             }
276.             ImageView iv_indicator = (ImageView)
277. convertView.findViewById(R.id.iv_indicator);
278.             TextView tv_groupName = (TextView)
279. convertView.findViewById(R.id.roast_group_name);
280.             TextView tv_roaster_count = (TextView)
281. convertView.findViewById(R.id.tv_roaster_count);
282.             if (isExpanded) {
283.                 iv_indicator.setBackgroundResource(R.drawable.indicator_expanded);
284.             } else {
285.                 iv_indicator.setBackgroundResource(R.drawable.indicator_unexpanded);
286.             }
287.             RosterGroup rosterGroup = rosterGroups.get(groupPosition);
288.             String name = rosterGroup.getName();
289.             tv_groupName.setText(name);
290.             int entryCount = rosterGroup.getEntryCount();
291.             tv_roaster_count.setText(entryCount + "");
292.             return convertView;
293.         }
294.
295.         @Override
296.         public View getChildView(int groupPosition, int childPosition, boolean isLastChild,
297. ViewGroup convertView, ViewGroup parent) {
298.             if (convertView == null) {
299.                 convertView = View.inflate(getActivity(), R.layout.list_item_roaster,
```

```
300.     null);
301.         }
302.         TextView tv_name = (TextView) convertView.findViewById(R.id.tv_name);
303.         RosterGroup rosterGroup = rosterGroups.get(groupPosition);
304.         List<RosterEntry> list = new ArrayList<RosterEntry>(rosterGroup.getEntries());
305.         RosterEntry rosterEntry = list.get(childPosition);
306.         String name = rosterEntry.getUser();
307.         tv_name.setText(name);
308.         return convertView;
309.     }
310.
311.     @Override
312.     public boolean isChildSelectable(int groupPosition, int childPosition) {
313.         return true;
314.     }
315.
316.     }
317. }
```

5.3.7 聊天功能

聊天界面是一个 ListView，这个 ListView 有两个布局一个是我发送的消息，另外一个为好友发送的消息。

聊天功能核心代码：

```
1.  if (chatManager==null) {
2.      //获取聊天管理器
3.          chatManager = xmppConnection.getChatManager();
4.      }
5.      if (chat==null) {
6.          //创建聊天，并制定消息监听器用于监听好友发送的消息
7.          chat = chatManager.createChat(user, messageListener);
8.      }
9.
10. chat.sendMessage(msg);
11.
12. private MessageListener messageListener = new MessageListener() {
13.
14.     @Override
15.     public void processMessage(Chat chat, Message message) {
16.         String body = message.getBody();
17.         if (TextUtils.isEmpty(body)) {
18.             return ;
```

```
19.        }
20.        com.itheima.qq.bean.Message message2 =new com.itheima.qq.bean.Message();
21.        message2.setBody(body);
22.        message2.setTime(TimeUtils.getNowTimeString());
23.        message2.setFrom(user);
24.        message2.setTo(xmppConnection.getUser());
25.        dataList.add(message2);
26.        MessageDB.putMessage(message2);
27.        android.os.Message message3 = android.os.Message.obtain();
28.        message3.what=2;
29.        handler.sendMessage(message3);
30.    }
31.};
```

activity_chat.xml

```
1. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2.     xmlns:tools="http://schemas.android.com/tools"
3.     android:layout_width="match_parent"
4.     android:layout_height="match_parent"
5.     android:orientation="vertical" >
6.
7.     <LinearLayout
8.         android:layout_width="match_parent"
9.         android:layout_height="60dp"
10.        android:background="@color/title_layout"
11.        android:gravity="center"
12.        android:orientation="horizontal" >
13.
14.        <TextView
15.            android:id="@+id/tv_title"
16.            style="@style/TitleStyle"
17.            android:padding="5dp"
18.            android:text="马化腾" />
19.    </LinearLayout>
20.
21.    <ListView
22.        android:id="@+id/lv_chat"
23.        android:layout_width="match_parent"
24.        android:layout_height="0dp"
25.        android:layout_weight="1" >
26.    </ListView>
27.
28.    <LinearLayout
29.        android:layout_width="match_parent"
```

```
30.         android:layout_height="50dp"
31.         android:gravity="center_vertical"
32.         android:orientation="horizontal" >
33.
34.         <ImageView
35.             android:layout_width="30dp"
36.             android:layout_height="30dp"
37.             android:layout_marginLeft="10dp"
38.             android:src="@drawable/chat_emo_normal" />
39.
40.         <ImageView
41.             android:layout_width="30dp"
42.             android:layout_height="30dp"
43.             android:layout_marginLeft="10dp"
44.             android:src="@drawable/chat_add_normal" />
45.
46.         <EditText
47.             android:id="@+id/et_msg"
48.             android:layout_width="0dp"
49.             android:layout_height="wrap_content"
50.             android:layout_gravity="center_vertical"
51.             android:layout_weight="1"
52.             android:background="#FFFFFF"
53.             android:padding="4dp"
54.             android:text="你好" />
55.
56.         <Button
57.             android:id="@+id/btn_send"
58.             android:layout_width="60dp"
59.             android:layout_height="35dp"
60.             android:layout_marginBottom="3dp"
61.             android:layout_marginLeft="3dp"
62.             android:layout_marginRight="3dp"
63.             android:layout_marginTop="3dp"
64.             android:background="@color/title_layout"
65.             android:text="发送"
66.             android:textColor="#FFFFFF" />
67.     </LinearLayout>
68.
69. </LinearLayout>
```

list_item_chat_me.xml

该布局是 ListView 中“自己”发送消息时使用的布局。

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.      android:layout_width="match_parent"
4.      android:layout_height="wrap_content"
5.      android:orientation="horizontal" >
6.
7.      <TextView
8.          android:id="@+id/tv_time"
9.          android:layout_width="wrap_content"
10.         android:layout_height="wrap_content"
11.         android:layout_centerVertical="true"
12.         android:layout_marginLeft="10dp"
13.         android:text="12:32" />
14.
15.     <TextView
16.         android:id="@+id/tv_me_msg"
17.         android:layout_width="wrap_content"
18.         android:layout_height="wrap_content"
19.         android:layout_marginRight="10dp"
20.         android:layout_marginTop="20dp"
21.         android:layout_toLeftOf="@id/iv_touxiang"
22.         android:background="@drawable/fv_chat_content_r_normal"
23.         android:gravity="center_vertical"
24.         android:paddingRight="25dp"
25.         android:text="这是我发送的消息" />
26.
27.     <ImageView
28.         android:id="@+id/iv_touxiang"
29.         android:layout_width="50dp"
30.         android:layout_height="50dp"
31.         android:layout_alignParentRight="true"
32.         android:layout_centerVertical="true"
33.         android:src="@drawable/kkj" />
34.
35. </RelativeLayout>
```

list_item_chat_you.xml

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.      android:layout_width="match_parent"
4.      android:layout_height="wrap_content"
5.      android:orientation="horizontal" >
6.
7.     <ImageView
```

```
8.         android:id="@+id/iv_touxiang"
9.         android:layout_width="50dp"
10.        android:layout_height="50dp"
11.        android:layout_centerVertical="true"
12.        android:src="@drawable/login_default_avatar" />
13.    <TextView
14.        android:id="@+id/tv_you_msg"
15.        android:layout_width="wrap_content"
16.        android:layout_height="wrap_content"
17.        android:layout_marginLeft="10dp"
18.        android:layout_marginTop="5dp"
19.        android:layout_toRightOf="@id/iv_touxiang"
20.        android:background="@drawable/fv_chat_content_l_normal"
21.        android:gravity="center_vertical"
22.        android:paddingLeft="23dp"
23.        android:text="这是我发送的消息" />
24.
25.    <TextView
26.        android:id="@+id/tv_time"
27.        android:layout_width="wrap_content"
28.        android:layout_height="wrap_content"
29.        android:layout_alignParentRight="true"
30.        android:layout_centerVertical="true"
31.        android:layout_marginRight="10dp"
32.        android:text="12:32" />
33.
34. </RelativeLayout>
```

ChatActivity.java

```
1. public class ChatActivity extends Activity implements OnClickListener {
2.     private ListView lv_chat;
3.     private TextView tv_title;
4.     private EditText et_msg;
5.     private Button btn_send;
6.     private XMPPConnection xmppConnection;
7.     private String user;
8.     private ChatManager chatManager;
9.     private Chat chat;
10.    private MyAdapter adapter;
11.    private List<com.itheima.qq.bean.Message> dataList = new
12.    ArrayList<com.itheima.qq.bean.Message>();
13.    private MessageListener messageListener = new MessageListener() {
14.
```

```
15.         @Override
16.     public void processMessage(Chat chat, Message message) {
17.         String body = message.getBody();
18.         if (TextUtils.isEmpty(body)) {
19.             return ;
20.         }
21.         com.itheima.qq.bean.Message message2 =new com.itheima.qq.bean.Message();
22.         message2.setBody(body);
23.         message2.setTime(TimeUtils.getNowTimeString());
24.         message2.setFrom(user);
25.         message2.setTo(xmppConnection.getUser());
26.         dataList.add(message2);
27.         MessageDB.putMessage(message2);
28.         android.os.Message message3 = android.os.Message.obtain();
29.         message3.what=2;
30.         handler.sendMessage(message3);
31.     }
32. };
33.
34. private Handler handler = new Handler(){
35.     public void handleMessage(android.os.Message msg) {
36.         switch (msg.what) {
37.             case 0:
38.                 Toast.makeText(ChatActivity.this, "消息发送失败"+msg.obj, 0).show();
39.                 break;
40.             case 1:
41.                 adapter.notifyDataSetChanged();
42.                 Toast.makeText(ChatActivity.this, "发送成功", 0).show();
43.                 break;
44.             case 2:
45.                 //接收到新消息
46.                 adapter.notifyDataSetChanged();
47.                 break;
48.
49.             default:
50.                 break;
51.         }
52.     };
53. };
54. @Override
55. protected void onCreate(Bundle savedInstanceState) {
56.     super.onCreate(savedInstanceState);
57.     setContentView(R.layout.activity_chat);
58.     initView();
59.     initData();
```

```
60.     }
61.
62.     private void initData() {
63.         Intent intent = getIntent();
64.         user = intent.getStringExtra("user");
65.         if (TextUtils.isEmpty(user)) {
66.             Toast.makeText(this, "聊天对象为空", 0).show();
67.             finish();
68.         }
69.         tv_title.setText(user);
70.         QQApplication application = (QQApplication)getApplication();
71.         xmppConnection = application.getXmppConnection();
72.         ArrayList<com.itheima.qq.bean.Message> messages =
73. MessageDB.getMessagesIgnoreFromAndTo(xmppConnection.getUser(), user);
74.         if (messages!=null) {
75.             dataList = messages;
76.         }
77.     }
78.
79.     private void initView() {
80.         lv_chat = (ListView) findViewById(R.id.lv_chat);
81.         tv_title = (TextView) findViewById(R.id.tv_title);
82.         et_msg = (EditText) findViewById(R.id.et_msg);
83.         btn_send = (Button) findViewById(R.id.btn_send);
84.         btn_send.setOnClickListener(this);
85.         adapter = new MyAdapter();
86.         lv_chat.setAdapter(adapter);
87.     }
88.
89.     @Override
90.     public void onClick(View v) {
91.         if(v.getId()==R.id.btn_send){
92.             String msg = et_msg.getText().toString();
93.             sendMsg(msg);
94.         }
95.     }
96.     @Override
97.     protected void onDestroy() {
98.         super.onDestroy();
99.         if (chat!=null) {
100.             chat.removeMessageListener(messageListener);
101.         }
102.         startService(new Intent(this, ChatService.class));
103.     }
```



```
104.         @Override
105.         protected void onPause() {
106.             super.onPause();
107.             if (TextUtils.isEmpty(user)) {
108.                 return;
109.             }
110.             Session session = new Session();
111.             session.setTime(TimeUtils.getNowTimeString());
112.             com.itheima.qq.bean.Message message = dataList.get(dataList.size()-1);
113.             session.setFrom(message.getFrom());
114.             session.setTo(message.getTo());
115.             session.setMsg(message.getBody());
116.             session.setUsr(user);
117.             SessionDB.updateSession(session );
118.         }
119.         @Override
120.         protected void onResume() {
121.             super.onResume();
122.             if (chatManager==null) {
123.                 chatManager = xmppConnection.getChatManager();
124.             }
125.             if (chat==null) {
126.                 chat = chatManager.createChat(user, messageListener);
127.             }
128.         }
129.         private void sendMsg(final String msg) {
130.             if (TextUtils.isEmpty(msg)) {
131.                 Toast.makeText(this, "不能发送空消息", 0).show();
132.                 return ;
133.             }
134.             new Thread(new Runnable() {
135.
136.                 @Override
137.                 public void run() {
138.
139.                     try {
140.                         chat.sendMessage(msg);
141.                         com.itheima.qq.bean.Message message = new
142. com.itheima.qq.bean.Message();
143.                         message.setBody(msg);
144.                         message.setTime(TimeUtils.getNowTimeString());
145. message.setTo(user);
146.                         message.setFrom(xmppConnection.getUser());
147.                         dataList.add(message);
148.                         MessageDB.putMessage(message);
```

```
149.                //发送成功
150.                android.os.Message message2 = android.os.Message.obtain();
151.                message2.what = 1;
152.                handler.sendMessage(message2);
153.            } catch (XMPPEException e) {
154.                e.printStackTrace();
155.                android.os.Message message = android.os.Message.obtain();
156.                message.what = 0;
157.                message.obj = e.toString();
158.                handler.sendMessage(message);
159.            }
160.        }
161.    }).start();
162.}
163.
164.    class MyAdapter extends BaseAdapter{
165.
166.        @Override
167.        public int getCount() {
168.            return dataList.size();
169.        }
170.
171.        @Override
172.        public Object getItem(int position) {
173.            return dataList.get(position);
174.        }
175.
176.        @Override
177.        public long getItemId(int position) {
178.            return position;
179.        }
180.
181.        @Override
182.        public int getViewTypeCount() {
183.            return 2;
184.        }
185.
186.        @Override
187.        public View getView(int position, View convertView, ViewGroup parent) {
188.            com.itheima.qq.bean.Message message = dataList.get(position);
189.            if (message.getFrom().equals(xmppConnection.getUser())) {
190.                //自己的消息
191.                if (convertView==null) {
192.                    convertView = View.inflate(ChatActivity.this,
```

```
193.     R.layout.list_item_chat_me, null);
194.         }
195.         TextView tv_me_msg = (TextView) convertView.findViewById(R.id.tv_me_msg);
196.         TextView tv_time = (TextView) convertView.findViewById(R.id.tv_time);
197.         tv_me_msg.setText(message.getBody());
198.         tv_time.setText(message.getTime());
199.     }else {
200.         //别人的消息
201.         if (convertView==null) {
202.             convertView = View.inflate(ChatActivity.this,
203. R.layout.list_item_chat_you, null);
204.         }
205.         TextView tv_you_msg = (TextView)
206. convertView.findViewById(R.id.tv_you_msg);
207.         TextView tv_time = (TextView) convertView.findViewById(R.id.tv_time);
208.         tv_you_msg.setText(message.getBody());
209.         tv_time.setText(message.getTime());
210.     }
211.     return convertView;
212. }
213. }
214. }
```

5.3.8 消息界面

SessionFragment 布局就是一个 ListView 十分的简单，因此就不给出了。

SessionFragment.java

```
1. public class SessionFrament extends Fragment {
2.
3.     private ListView lv_session;
4.     private ArrayList<Session> dataList = new ArrayList<Session>();
5.     private MyAdapter adapter = new MyAdapter();
6.
7.     public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
8.         savedInstanceState) {
9.         View view = initView();
10.        initData();
11.        return view;
12.    }
13.
14.    private View initView() {
```

```
15.     View view = View.inflate(getActivity(), R.layout.fragment_session, null);
16.     lv_session = (ListView) view.findViewById(R.id.lv_session);
17.     return view;
18. }
19.
20. public void initData() {
21.     lv_session.setAdapter(adapter);
22.     ArrayList<Session> sessions = SessionDB.getSessions();
23.     if (sessions != null && sessions.size() > 0) {
24.         dataList = sessions;
25.         adapter.notifyDataSetChanged();
26.     }
27.     lv_session.setOnItemClickListener(new OnItemClickListener() {
28.
29.         @Override
30.         public void onItemClick(AdapterView<?> parent, View view, int position, long id)
31.     {
32.         String user = dataList.get(position).getUshr();
33.         Toast.makeText(getActivity(), user, 0).show();
34.         Intent chatIntent = new Intent(getActivity(), ChatActivity.class);
35.         chatIntent.putExtra("user", user);
36.         startActivity(chatIntent);
37.     }
38.     });
39. };
40.
41. class MyAdapter extends BaseAdapter {
42.     @Override
43.     public int getCount() {
44.         return dataList.size();
45.     }
46.
47.     @Override
48.     public Object getItem(int position) {
49.         return dataList.get(position);
50.     }
51.
52.     @Override
53.     public long getItemId(int position) {
54.         return position;
55.     }
56.
57.     @Override
58.     public View getView(int position, View convertView, ViewGroup parent) {
```

```
59.         ViewHolder viewHolder = null;
60.         if (convertView == null) {
61.             viewHolder = new ViewHolder();
62.             convertView = View.inflate(getActivity(), R.layout.list_item_session, null);
63.             viewHolder.tv_msg = (TextView) convertView.findViewById(R.id.tv_msg);
64.             viewHolder.tv_name = (TextView) convertView.findViewById(R.id.tv_name);
65.             viewHolder.tv_time = (TextView) convertView.findViewById(R.id.tv_time);
66.             convertView.setTag(viewHolder);
67.         } else {
68.             viewHolder = (ViewHolder) convertView.getTag();
69.         }
70.         Session session = dataList.get(position);
71.         viewHolder.tv_msg.setText(session.getMsg());
72.         viewHolder.tv_time.setText(session.getTime());
73.         // 到底是 from 还是 to
74.         viewHolder.tv_name.setText(session.getUsr());
75.         return convertView;
76.     }
77.
78. }
79. @Override
80. public void onResume() {
81.     super.onResume();
82.     if(adapter!=null){
83.         adapter.notifyDataSetChanged();
84.     }
85.     SessionDB.setListener(new SessionListener() {
86.
87.         @Override
88.         public void onChange() {
89.             if (adapter!=null) {
90.                 adapter.notifyDataSetChanged();
91.             }
92.         }
93.     });
94. }
95. static class ViewHolder {
96.     TextView tv_name;
97.     TextView tv_time;
98.     TextView tv_msg;
99. }
100. }
```

5.3.9 启动服务监听消息

当用户登录成功的时候在后台开启一个服务，用于监听主动发过来的消息。

ChatService.java

```
1. public class ChatService extends Service {
2.     private static QQApplication application;
3.     private static NotificationManager notificationManager;
4.     private ChatManager chatManager;
5.     private MessageListener messageListener = new MessageListener() {
6.         public void processMessage(Chat chat, Message message) {
7.             if (TextUtils.isEmpty(message.getBody())) {
8.                 return ;
9.             }
10.            Session session = new Session();
11.            session.setMsg(message.getBody());
12.            String from = message.getFrom();
13.            if (from.endsWith("/Spark")) {
14.                from = from.substring(0, from.length()-"/Spark".length());
15.            }
16.            session.setFrom(from);
17.            session.setTo(message.getTo());
18.            session.setUsr(from);
19.            session.setTime(TimeUtils.getNowTimeString());
20.            SessionDB.updateSession(session);
21.            com.itheima.qq.bean.Message message2 = new com.itheima.qq.bean.Message();
22.            message2.setBody(message.getBody());
23.            message2.setFrom(from);
24.            message2.setTo(message.getTo());
25.            message2.setTime(TimeUtils.getNowTimeString());
26.            MessageDB.putMessage(message2);
27.            android.os.Message msg = android.os.Message.obtain();
28.            msg.obj = message;
29.            handler.sendMessage(msg);
30.        }
31.    };
32.    private ChatManagerListener chatManagerListener = new ChatManagerListener() {
33.        @Override
34.        public void chatCreated(Chat chat, boolean createdLocally) {
35.            if (!createdLocally) {
36.                chat.addMessageListener(messageListener);
37.            }
38.        }
39.    };
40.}
```

```
38.     }
39. };
40.     private Handler handler = new Handler() {
41.         public void handleMessage(android.os.Message msg) {
42.             Notification notification = new Notification(R.drawable.kkj, "收到新消息",
43.                 SystemClock.uptimeMillis());
44.             notification.flags = Notification.FLAG_AUTO_CANCEL;
45.             Intent intent = new Intent(ChatService.this, ChatActivity.class);
46.             Message message = (Message)msg.obj;
47.             String from = message.getFrom();
48.             if (from.endsWith("/Spark")) {
49.                 from = from.substring(0, from.length()-"/Spark".length());
50.             }
51.             intent.putExtra("user",from);
52.             intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
53.             PendingIntent pendingIntent = PendingIntent.getActivity(ChatService.this, 1,intent,
54.                 PendingIntent.FLAG_UPDATE_CURRENT);
55.             notification.setLatestEventInfo(ChatService.this, "新消息", "请注意查收",
56.                 pendingIntent);
57.             notificationManager.notify(2, notification);
58.         };
59.     };
60.     public IBinder onBind(Intent intent) {
61.         return null;
62.     }
63.     public void onCreate() {
64.         super.onCreate();
65.         application = (QQApplication) getApplication();
66.         notificationManager = (NotificationManager)
67.             ChatService.this.getSystemService(Context.NOTIFICATION_SERVICE);
68.     }
69.     public int onStartCommand(Intent intent, int flags, int startId) {
70.         initListener();
71.         return super.onStartCommand(intent, flags, startId);
72.     }
73.     private void initListener() {
74.         XMPPConnection xmppConnection = application.getXmppConnection();
75.         if (xmppConnection==null) {
76.             return;
77.         }
78.         chatManager = xmppConnection.getChatManager();
79.         Toast.makeText(this, "服务已经开启", 0).show();
80.         chatManager.addChatListener(chatManagerListener);
81.     }
82. }
```