# DAY01

1. 展示最终完成项目
2. 项目介绍，引入环信
3. 下载环信SDK，确定ChatDemoUI项目
4. 先修改build.gradle文件

* classpath 'com.android.tools.build:gradle:0.12.+'

+ classpath **'com.android.tools.build:gradle:2.0.0-beta7'**

修改为自己本地已经存在的gradle版本

* buildToolsVersion "23.0.0 rc2"

+ buildToolsVersion **"23.0.2"**

修改为自己本地已经存在的编译版本

1. 修改gradle/wrapper/gradle-wrapper.properties文件

* distributionUrl=https\://services.gradle.org/distributions/gradle-1.12-all.zip

+ **distributionUrl**=**https\://services.gradle.org/distributions/gradle-2.10-all.zip**

修改为自己本地已经存在的gradle版本

1. 将项目文件夹重命名为SuperWeChat，并导入AS中
2. 编译工程会报错，查看log为点九图片问题，修改

AAPT: ERROR: 9-patch image

XXX/SuperWeChat/res/drawable-hdpi/tab\_unread\_bg.9.png malformed.

修改方法为在图片左边画上缺失的黑边

1. 编译工程并运行，对比最终完成项目，说明修改方案
2. 修改包名为cn.ucai.superwechat

在AS中资源管理器区域-设置-Compact Empty Middle Packages去掉勾选，逐个重命名，将applib拖入superwechat包

1. 修改AndroidManifest.xml文件中的包名
2. 修改DemoApplication.java重命名SuperWeChatApplication.java
3. AS的项目唯一标识不是Manifest中的package属性，需要在build.gradle中修改，在android下明确项目的报名

defaultConfig {  
 applicationId **"cn.ucai.superwechat"** minSdkVersion 9  
 targetSdkVersion 23  
}

1. 在Utils类中编写以下方法，获取当前项目的包名：

**public** **static** String getPackageName(Context context){

**return** context.getPackageName();

}

1. 修改ChatActivity. getGridChildView方法，

* Class clz = Class.forName("SmileUtils");

+ Class clz =

Class.forName(Utils.getPackageName(ChatActivity.this)+".utils.SmileUtils");

1. 修改环信appkey为自己申请的key

<!-- 设置环信应用的appkey -->

<meta-data

android:name="EASEMOB\_APPKEY"

android:value="i#superwechat" />

1. 添加接口文件I.java，将服务器端的I文件内容复制过来，修改interface为class，并去掉表名字段，去掉UTF\_8和ISON8859，添加下载头像的接口：

String DOWNLOAD\_AVATAR\_URL=SuperWeChatApplication.SERVER\_ROOT+

"?request="+REQUEST\_DOWNLOAD\_AVATAR+"&avatar=";

添加服务器地址(添加在I文件或者SuperWeChatApplication文件，分别为是否可设置)：SERVER\_ROOT="http://10.0.2.2:8080/SuperQQ4Server/Server";在内部类User中添加字段String UID="uid";

1. 创建实体类包bean包，并从服务器复制实体类到项目，也可以使用GsonFormat插件创建，导入jar包jackson-core-asl-1.9.11.jar和jackson-mapper-asl-1.9.11.jar，并add as library，操作成功后build.gradle文件中会增加如下：

compile files(**'libs/jackson-core-asl-1.9.11.jar'**)  
compile files(**'libs/jackson-mapper-asl-1.9.11.jar'**)

1. ContactBean实体类，为isGetMyLocation和isShowMyLocation方法添加忽略标识@JsonIgnore，修改setter方法名为setIsGetMyLocation和setIsShowMyLocation
2. GroupBean实体类，为isPublic和isExame方法添加忽略标识@JsonIgnore，修改setter方法名为setIsPublic和setIsExame，添加equals和hashCode，以字段name为比较基准
3. MessageBean实体类，字段success没有is前缀，无需json处理
4. UserBean实体类，在构造函数中去掉groups字段，添加equals和hashCode，以字段userName为比较基准

JSON格式的实体类

{

"id":1001,

"result":"ok",

"userName":"zhangsan",

"nick":"张三",

"password":"1234",

"avatar":"user\_avatar/zhangsan.jpg",

"groups":"2011",

"header":"z",

"latitude":8.34,

"longitude":12.2,

"unreadMsgCount":4

}

{

"result":"ok",

"myuid":1001,

"cuid":1002,

"isGetMyLocation":false,

"isShowMyLocation":false,

}

{

"id":1010,//主键

"name":"Android项目群",//群名

"avatar":" group\_avatar/android.jpg ",//群图标

"intro":"私人讨论小组",//群简介

"owner":"1001",//群主账号

"isPublic":false,//是否公开

"isExame":false,//是否公开

"groupId":"2011",//群ID

"modifiedTime":1010,//群信息修改的时间，单位：毫秒

"members":"1001,1002,1003",//存放群成员账号,格式:账号1,账号2,...

}

{

/\*\* 响应是否成功,true:成功，false：失败\*/

"success":true,

/\*\* 返回的字符串\*/

"msg":"ok",

}

1. 修改db包中的UserDao文件，重命名为EMUserDao，添加UserDao.java文件并继承SQLiteOpenHelper，添加需要重新父类的方法，添加构造函数，只需要context即可，其他参数直接填写，数据库表名为user.db，版本为1
2. 在重新父类的onCreate方法中创建用户表，将I接口文件中的所有字段修改为public static final类型，创建表的sql语句可以从数据库中提取，添加UID和HEADER字段，去除GROUP字段

@Override  
**public void** onCreate(SQLiteDatabase db) {  
 String sql=**"create table if not exists "**+***TABLE\_NAME*** +**"("**+ I.User.***ID***+**" integer primary key autoincrement,"** + I.User.***USER\_NAME***+**" varchar unique,"** + I.User.***UID***+**" int unique,"** + I.User.***NICK***+**" varchar,"** + I.User.***AVATAR***+**" varchar,"** + I.User.***HEADER***+**" varchar,"** + I.User.***LATITUDE***+**" double,"** + I.User.***LONGITUDE***+**" double,"** + I.User.***PASSWORD***+**" varchar,"** + I.User.***UN\_READ\_MSG\_COUNT***+**" int)"**;  
 db.execSQL(sql);  
}

1. 添加本地服务器保存登录用户信息的addUser方法，不添加自增长的ID字段

**public boolean** addUser(UserBean user){  
 ContentValues values = **new** ContentValues();  
 values.put(I.User.***AVATAR***, user.getAvatar());  
 values.put(I.User.***HEADER***, user.getHeader());  
 values.put(I.User.***LATITUDE***, user.getLatitude());  
 values.put(I.User.***LONGITUDE***,user.getLongitude());  
 values.put(I.User.***NICK***, user.getNick());  
 values.put(I.User.***PASSWORD***, user.getPassword());  
 values.put(I.User.***UID***,user.getId());  
 values.put(I.User.***UN\_READ\_MSG\_COUNT***, user.getUnreadMsgCount());  
 values.put(I.User.***USER\_NAME*** , user.getUserName());  
 SQLiteDatabase db = getWritableDatabase();  
 **long** rowId = db.insert(***TABLE\_NAME***, **null**, values);  
 **return** rowId!=0;  
}

1. 添加根据用户名查找用户的findUserByUserName方法

**public** UserBean findUserByUserName(String userName){  
 SQLiteDatabase db = getReadableDatabase();  
 String sql = **"select \* from "** + ***TABLE\_NAME***+ **" where "**+ I.User.***USER\_NAME*** +**"=?"**;  
 Cursor c = db.rawQuery(sql, **new** String[]{userName});  
 Log.*e*(**"main"**,**"UserDao.findUserByUserName.sql="**+sql);  
 **if**(c.moveToNext()){  
 **int** uid = c.getInt(c.getColumnIndex(I.User.***UID***));  
 String nick=c.getString(c.getColumnIndex(I.User.***NICK***));  
 String avatar=c.getString(c.getColumnIndex(I.User.***AVATAR***));  
 String header=c.getString(c.getColumnIndex(I.User.***HEADER***));  
 **double** latitude=c.getDouble(c.getColumnIndex(I.User.***LATITUDE***));  
 **double** longitude=c.getDouble(c.getColumnIndex(I.User.***LONGITUDE***));  
 String password=c.getString(c.getColumnIndex(I.User.***PASSWORD***));  
 **int** unreadMsgCount=c.getInt(c.getColumnIndex(I.User.***UN\_READ\_MSG\_COUNT***));  
 UserBean user = **new** UserBean(uid, **"ok"**, userName, nick, password, avatar, latitude, longitude, unreadMsgCount);  
 Log.*e*(**"main"**,**"UserDao.findUserByUserName.user="**+user);  
 **return** user;  
 }  
 **return null**;  
}

1. 添加更新用户的updateUser方法，直接使用addUser的方法，将insert改为update即可

**public boolean** updateUser(UserBean user){  
 ContentValues values = **new** ContentValues();  
 values.put(I.User.***UID***,user.getId());  
 values.put(I.User.***USER\_NAME***,user.getUserName());  
 values.put(I.User.***NICK***,user.getNick());  
 values.put(I.User.***PASSWORD***,user.getPassword());  
 values.put(I.User.***AVATAR***,user.getAvatar());  
 values.put(I.User.***HEADER***,user.getHeader());  
 values.put(I.User.***LATITUDE***,user.getLatitude());  
 values.put(I.User.***LONGITUDE***,user.getLongitude());  
 values.put(I.User.***UN\_READ\_MSG\_COUNT***,user.getUnreadMsgCount());  
 SQLiteDatabase db = getWritableDatabase();  
 **long** rowId = db.update(***TABLE\_NAME***,values,I.User.***USER\_NAME***+ **"=?"**,**new** String[]{user.getUserName()});  
 **return** rowId>0;  
}

1. 添加ImageUtils.java文件，添加 获取头像在sd卡位置的方法

*/\*\*  
 \* 返回头像保存在sd卡的位置:Android/data/cn.ucai.superwechat/files/pictures/user\_avatar  
 \** ***@param context*** *\** ***@param path*** *\** ***@return*** *\*/***public static** String getAvatarPath(Context context, String path) {  
 File dir = context.getExternalFilesDir(Environment.*DIRECTORY\_PICTURES*);  
 File folder = **new** File(dir,path);  
 **if**(!folder.exists()) {  
 folder.mkdir();  
 }  
 **return** folder.getAbsolutePath();  
}

1. 添加Utils六个工具类：

BitmapUtils.java,FileUtils.java,HttpUtils.java,ImageLoader.java,MD5.java,NetUtil.java；添加string.xml资源文件

<string name="Add\_buddy\_success">add buddy success</string>

<string name="Nick\_name\_cannot\_be\_empty">Nick name cannot be empty!</string>

<string name="Group\_name\_existed">Group name existed</string>

<string name="Create\_groups\_Success">Success to create groups</string>

<string name="Create\_groups\_Failed">Failed to create groups</string>

<string name="photo">photo</string>

<string name="near\_people\_hint">See nearby people\'s function will get your location information, your location information will be kept for a period of time. Through the list of the top right corner to remove functions can be manually remove location information at any time.</string>

<string name="Add\_buddy\_success">添加好友成功</string>

<string name="Nick\_name\_cannot\_be\_empty">昵称不能为空！</string>

<string name="Group\_name\_existed">群组名称已经存在</string>

<string name="Create\_groups\_Success">创建群组成功</string>

<string name="Create\_groups\_Failed">创建群组失败</string>

<string name="photo">头像</string>

<string name="near\_people\_hint">

查看附近人的功能将获取你的位置信息，你的位置信息会保留一段时间。通过列表右上角清除功能可随时手动清除位置信息。

</string>

在Utils.java中添加提示方法

public static void showToast(Context context,String text,int time){

Toast.makeText(context,text,time).show();

}

public static void showToast(Context context,int strId,int time){

Toast.makeText(context, strId, time).show();

}

在Utils.java中添加数组集合操作方法

/\*\*

\* 将数组转换为ArrayList集合

\* @param ary

\* @return

\*/

public static <T> ArrayList<T> array2List(T[] ary){

List<T> list = Arrays.asList(ary);

ArrayList<T> arrayList=new ArrayList<T>(list);

return arrayList;

}

/\*\*

\* 添加新的数组元素：数组扩容

\* @param array：数组

\* @param t：添加的数组元素

\* @return：返回添加后的数组

\*/

public static <T> T[] add(T[] array,T t){

array=Arrays.copyOf(array, array.length+1);

array[array.length-1]=t;

return array;

}

1. 在BaseActivity.java中添加getViewById的方法

**public** <T> T getViewById(**int** id){  
 **return** (T)findViewById(id);  
}

1. 在SuperWeChatApplication.java中添加全局的当前用户的操作

UserBean **userBean**;  
**public** UserBean getUserBean() {  
 **return userBean**;  
}  
**public void** setUserBean(UserBean userBean) {  
 **this**.**userBean** = userBean;  
}  
**public void** logout(**final** EMCallBack emCallBack) {  
 *hxSDKHelper*.logout(**true**, emCallBack);  
}

1. 在SuperWeChatApplication.java中添加全局的好友用户的操作

**private** ArrayList<UserBean> **contactList** = **new** ArrayList<UserBean>();  
**public** ArrayList<UserBean> getContactList() {  
 **return contactList**;  
}  
**public void** setContactList(ArrayList<UserBean> contactList) {  
 **this**.**contactList** = contactList;  
}

1. 在SuperWeChatApplication.java中添加全局的联系人集合的操作

**private** HashMap<Integer, ContactBean> **contacts** = **new** HashMap<Integer, ContactBean>();  
**public** HashMap<Integer, ContactBean> getContacts() {  
 **return contacts**;  
}  
**public void** setContacts(HashMap<Integer, ContactBean> contacts) {  
 **this**.**contacts** = contacts;  
}

1. 在SuperWeChatApplication.java中添加全局的群组集合的操作

**private** ArrayList<GroupBean> **mGroupList** = **new** ArrayList<GroupBean>();  
**public** ArrayList<GroupBean> getGroupList() {  
 **return mGroupList**;  
}  
**public void** setGroupList(ArrayList<GroupBean> mGroupList) {  
 **this**.**mGroupList** = mGroupList;  
}

1. 在SuperWeChatApplication.java中添加全局的公开群组集合的操作

**private** ArrayList<GroupBean> **mPublicGroupList** = **new** ArrayList<GroupBean>();  
**public** ArrayList<GroupBean> getPublicGroupList() {  
 **return mPublicGroupList**;  
}  
**public void** setPublicGroupList(ArrayList<GroupBean> mPublicGroupList) {  
 **this**.**mPublicGroupList** = mPublicGroupList;  
}

1. 在SuperWeChatApplication.java中添加全局的缓存指定群成员集合的操作

*/\*\*  
 \* 缓存指定群成员的集合  
 \*/***private** HashMap<String, ArrayList<UserBean>> **mGroupMembers** = **new** HashMap<String, ArrayList<UserBean>>();  
**public** HashMap<String, ArrayList<UserBean>> getGroupMembers() {  
 **return mGroupMembers**;  
}  
**public void** setGroupMembers(HashMap<String, ArrayList<UserBean>> mGroupMembers) {  
 **this**.**mGroupMembers** = mGroupMembers;  
}

1. 导入nodpi图片资源，修改闪屏界面布局文件
2. 修改登录界面布局文件，去掉布局文件中的登录和注册方法声明，为登录和注册按钮添加id
3. 修改LoginActivity.java文件，添加setListener()方法整理登录注册和清空密码方法，添加setUserNameTextChangeListener();添加setLoginClickListener();方法整理登录代码；添加setRegisterClickListener();方法整理注册代码；
4. 将登录方法中的ProgressDialog放到线程中

runOnUiThread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 pd.show();  
 }  
});

1. 添加登录本地服务器的方法

**private boolean** loginAppServer(String userName, String password){  
 UserDao dao = **new** UserDao(**mContext**);  
 UserBean user = dao.findUserByUserName(userName);  
 **if**(user != **null**){  
 **if**(user.getPassword().equals(MD5.*getData*(password))){  
 saveUser(user);  
 **return true**;  
 }  
 }**else**{  
 **try**{  
 user = NetUtil.*login*(userName, password);  
 **if**(**"ok"**.equals(user.getResult())){  
 saveUser(user);  
 user.setPassword(MD5.*getData*(password));  
 dao.addUser(user);  
 **return true**;  
 }  
 }**catch**(Exception e){  
 e.printStackTrace();  
 }  
 }  
 **return false**;  
}  
  
**private void** saveUser(UserBean user) {  
 SuperWeChatApplication instanceApplication = SuperWeChatApplication.*getInstance*();  
 instanceApplication.setUserBean(user);  
 instanceApplication.setUserName(user.getUserName());  
 instanceApplication.setPassword(user.getPassword());  
 instanceApplication.*currentUserNick* = user.getNick();  
}

1. 在setLoginClickListener方法中，如果登录环信服务器成功后，继续登录本地服务器，如果服务器验证失败则返回

*//添加到本地的数据库中***boolean** isSuccess = loginAppServer(**currentUsername**, **currentPassword**);  
Log.*i*(***TAG***,**"login...name="**+**currentUsername**+**",pas="**+**currentPassword**+**",isSuccess="**+isSuccess);  
**if**(!isSuccess){  
 runOnUiThread(**new** Runnable() {  
 **public void** run() {  
 Utils.*showToast*(**mContext**, **"登陆失败"**, Toast.***LENGTH\_LONG***);  
 }  
 });  
 **return**;  
}

1. 在libs中添加volley.jar，gson-2.2.4.jar，并添加为库文件

compile files(**'libs/volley.jar'**)   
compile files(**'libs/gson-2.2.4.jar'**)

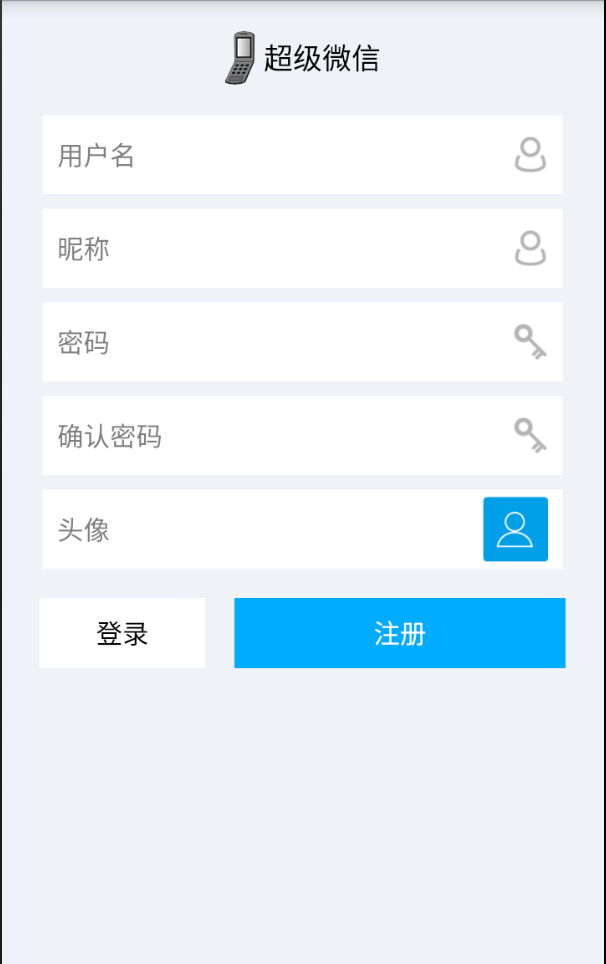
1. 导入data包中的4个工具类：ApiParams.java，BitmapLruCache.java，GsonRequest.java，RequestManager.java
2. 在SuperWeChatApplication.java的onCreate中添加初始化volley的代码：

RequestManager.*init*(**this**);

1. 在BaseActivity.java中添加以下代码

**protected** Activity **activity**;  
  
@Override  
**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 **activity** = **this**;  
}  
  
@Override  
**public void** onStop() {  
 **super**.onStop();  
 RequestManager.*cancelAll*(**this**);  
}  
  
**protected void** executeRequest(Request<?> request) {  
 RequestManager.*addRequest*(request, **this**);  
}  
  
**protected** Response.ErrorListener errorListener() {  
 **return new** Response.ErrorListener() {  
 @Override  
 **public void** onErrorResponse(VolleyError error) {  
 Toast.*makeText*(**activity**, error.getMessage(), Toast.***LENGTH\_LONG***).show();  
 }  
 };  
}

1. 修改注册界面的布局文件，效果图如下



1. 整理RegisterActivity，添加initView和setListener方法
2. 在setRegisterClickListener方法中添加文本非空的处理操作

添加用户名的非法字符判断

} **else if**(!username.matches(**"[\\w][\\w\\d\_]+"**)){  
 **metUserName**.requestFocus();  
 **metUserName**.setError(**"账号只能包含英文、数字和下划线"**);  
 **return**;

将错误提示Toast.makeText修改为setError方法

1. 添加拍照的框架文件OnSetAvatarListener到listener包，向res/layout中添加popu\_show\_avatar.xml文件：弹出拍照和从相册获取图片的popu\_window窗口；向res/drawable中添加selector\_pop\_avatar.xml文件；向res/drawable-mdpi中添加pop\_bg\_press.9.png和pop\_bg.9.png；添加pop\_enter.xml和pop\_exit.xml至res/anim；在res/values/styles.xml中编写以下样式：

<**style name="styles\_pop\_window"**>  
 <**item name="@android:windowEnterAnimation"**>@anim/pop\_enter</**item**>  
 <**item name="@android:windowExitAnimation"**>@anim/pop\_exit</**item**>  
</**style**>

1. 添加调用拍照的方法setUserAvatarClickListener

**private void** setUserAvatarClickListener() {  
 findViewById(R.id.***layout\_user\_avatar***).setOnClickListener(**new** OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **try** {  
 **mOnSetAvatarListener**=**new** OnSetAvatarListener(RegisterActivity.**this**,R.id.***layout\_register***,getUserName(),**"user\_avatar"**);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 });  
}

1. 重写父类的onActivityResult的方法，接收获取的头像

@Override  
**protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  
 **super**.onActivityResult(requestCode, resultCode, data);  
 Log.*i*(***TAG***, **"onActivityResult-resultCode="**+resultCode);  
 **if**(resultCode != ***RESULT\_OK***){  
 **return**;  
 }  
 **try** {  
 String userName = getUserName();  
 */\*  
 \* 设置拍照或从相册获取图片后返回的结果  
 \* @param requestCode:请求码  
 \* @param data:返回的intent  
 \* @param ivAvatar：显示头像的ImageView  
 \* @param userName：注册窗口输入的账号  
 \*/* **mOnSetAvatarListener**.setAvatar(requestCode, data, **mivAvatar**);  
 } **catch** (Exception e) {  
 Log.*i*(***TAG***, **"onActivityResult-e="**+e.getMessage());  
 e.printStackTrace();  
 }  
}

1. 将注册环信的代码整合到RegisterTask中，并添加向服务器注册的方法

1、首先向应用服务器注册。

2、注册成功后，上传头像。

3、向环信服务器注册。

4、若步骤3失败，则将步骤1中注册的用户信息删除，将步骤2中上传的头像删除。

*/\*\*  
 \* 注册的线程类  
 \*/* **class** RegisterTask **extends** AsyncTask<Void,Void,EaseMobException>{  
 String **userName**,**nick**,**password**;  
 ProgressDialog **dialog**;  
  
 */\*\*  
 \* 向应用服务器注册，注册成功后，再向环信服务器注册  
 \** ***@param userName*** *\** ***@param nick*** *\** ***@param password*** *\*/* **public** RegisterTask(String userName,String nick,String password){  
 **super**();  
 **this**.**userName** = userName;  
 **this**.**nick** = nick;  
 **this**.**password** = password;  
 }  
  
 @Override  
 **protected void** onPreExecute() {  
 **dialog** = **new** ProgressDialog(**mContext**);  
 **dialog**.setMessage(getResources().getString(R.string.***Is\_the\_registered***));  
 **dialog**.show();  
 }  
  
 @Override  
 **protected** EaseMobException doInBackground(Void... params) {  
 EaseMobException errorEaseMobException = **null**;  
 UserBean user = **new** UserBean(**userName**,**nick**,**password**);  
 **try** {  
 **boolean** isSuccess = NetUtil.*register*(user);  
 **if** (isSuccess) {  
 isSuccess = NetUtil.*uploadAvatar*(**mContext**,**"user\_avatar"**,**userName**);  
 **if**(isSuccess) {  
 *// 调用sdk注册方法* EMChatManager.*getInstance*().createAccountOnServer(**userName**, **password**);  
 *// 保存用户名* SuperWeChatApplication.*getInstance*().setUserName(**userName**);  
 SuperWeChatApplication.*getInstance*().setUserBean(user);  
 }  
 errorEaseMobException = **new** EaseMobException(getResources().getString(R.string.***Registered\_successfully***));  
 } **else** {  
 errorEaseMobException = **new** EaseMobException(getResources().getString(R.string.***Registration\_failed***));  
 }  
 } **catch** (EaseMobException e){  
 errorEaseMobException = e;  
 e.printStackTrace();  
 NetUtil.*unRegister*(**userName**);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **return** errorEaseMobException;  
 }  
  
 @Override  
 **protected void** onPostExecute(EaseMobException result) {  
 **if** (!RegisterActivity.**this**.isFinishing())  
 **dialog**.dismiss();  
 **if**(result == **null**){  
 Utils.*showToast*(**mContext**, getResources().getString(R.string.***Registration\_failed***), 0);  
 }**else** {  
 **if** (result.getMessage().equals(getResources().getString(R.string.***Registered\_successfully***))) {  
 *//保存用户* SuperWeChatApplication.*getInstance*().setUserName(**userName**);  
 Utils.*showToast*(**mContext**, getResources().getString(R.string.***Registered\_successfully***), 0);  
 finish();  
 } **else** {  
 **int** errorCode = result.getErrorCode();  
 **if** (errorCode == EMError.***NONETWORK\_ERROR***) {  
 Utils.*showToast*(getApplicationContext(), getResources().getString(R.string.***network\_anomalies***), Toast.***LENGTH\_SHORT***);  
 } **else if** (errorCode == EMError.***USER\_ALREADY\_EXISTS***) {  
 Utils.*showToast*(getApplicationContext(), getResources().getString(R.string.***User\_already\_exists***), Toast.***LENGTH\_SHORT***);  
 } **else if** (errorCode == EMError.***UNAUTHORIZED***) {  
 Utils.*showToast*(getApplicationContext(), getResources().getString(R.string.***registration\_failed\_without\_permission***), Toast.***LENGTH\_SHORT***);  
 } **else if** (errorCode == EMError.***ILLEGAL\_USER\_NAME***) {  
 Utils.*showToast*(getApplicationContext(), getResources().getString(R.string.***illegal\_user\_name***), Toast.***LENGTH\_SHORT***);  
 } **else** {  
 Utils.*showToast*(getApplicationContext(), getResources().getString(R.string.***Registration\_failed***) + result.getMessage(), Toast.***LENGTH\_SHORT***);  
 }  
 }  
 }  
 }  
 }

1. 海马玩有时不能访问本地tomcat服务器10.0.2.2，修改为本地地址即可，需要添加一个对项目服务器地址修改的方法，放到登录界面，添加修改服务器的方法setServerUrlClickListener();

*/\*\*  
 \* 设置服务端地址的按钮事件监听  
 \*/***private void** setServerUrlClickListener() {  
 findViewById(R.id.***btnUrl***).setOnClickListener(**new** OnClickListener() {  
 String **serverUrl**;  
 @Override  
 **public void** onClick(View v) {  
 **final** SharedPreferences sp=getSharedPreferences(**"server\_url"**, ***MODE\_PRIVATE***);  
 **serverUrl**=sp.getString(**"url"**, **""**);  
 View layout=View.*inflate*(LoginActivity.**this**, R.layout.***diaolog\_serverurl***,**null**);  
 **final** EditText etServerUrl=(EditText) layout.findViewById(R.id.***etServerUrl***);  
 **if**(!**serverUrl**.isEmpty()){  
 etServerUrl.setText(**serverUrl**);  
 }  
  
 AlertDialog.Builder builder=**new** AlertDialog.Builder(LoginActivity.**this**);  
 builder.setTitle(**"设置服务端的ip地址"**);  
 builder.setView(layout);  
 builder.setPositiveButton(**"设置"**, **new** DialogInterface.OnClickListener() {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {  
 **serverUrl**=etServerUrl.getText().toString();  
 **if**(**serverUrl**.isEmpty()){  
 **return** ;  
 }  
 sp.edit().putString(**"url"**, **serverUrl**).commit();  
 SuperWeChatApplication.*SERVER\_ROOT*=**serverUrl**+**":8080/SuperQQ4Server/Server"**;  
 Toast.*makeText*(LoginActivity.**this**, **"服务端ip地址设置完成"**, Toast.***LENGTH\_LONG***).show();  
 Log.*e*(**"main"**,**"LoginACtivity,SERVER\_ROOT="**+SuperWeChatApplication.*SERVER\_ROOT*);  
 }  
 });  
 builder.setNegativeButton(**"取消"**, **null**);  
 builder.create().show();  
 }  
 });  
}

在SuperWeChatApplication的onCreate方法中添加初始化服务器地址的方法

**public void** initServerUrl() {  
 **final** SharedPreferences sp=getSharedPreferences(**"server\_url"**, ***MODE\_PRIVATE***);  
 String serverUrl=sp.getString(**"url"**, **""**);  
 **if**(serverUrl==**null** || serverUrl.isEmpty()){  
 SuperWeChatApplication.*SERVER\_ROOT*=**"http://10.0.2.2:8080/SuperQQ4Server/Server"**;  
 }**else** {  
 SuperWeChatApplication.*SERVER\_ROOT*=serverUrl+**":8080/SuperQQ4Server/Server"**;  
 }  
 Log.*e*(***TAG***,**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*initServerUrl.serverurl="**+*SERVER\_ROOT*);  
}

# DAY02

1. 在SplashActivity.java中的onStart方法中添加获取已经登录用户的代码

*//将数据库中的当前登录用户保存在内存中*String userName = SuperWeChatApplication.*getInstance*().getUserName();  
UserDao dao = **new** UserDao(SplashActivity.**this**);  
UserBean user = dao.findUserByUserName(userName);  
SuperWeChatApplication.*getInstance*().setUserBean(user);

1. 添加下载联系人的任务类DownloadContactsTask

**public class** DownloadContactsTask **extends** AsyncTask<Void, Void, Boolean> {  
   
 Context **context**;  
 String **userName**;  
 **int pageId**;  
 **int pageSize**;  
  
 **public** DownloadContactsTask(Context context, String userName, **int** pageId,  
 **int** pageSize) {  
 **super**();  
 **this**.**context** = context;  
 **this**.**userName** = userName;  
 **this**.**pageId** = pageId;  
 **this**.**pageSize** = pageSize;  
 }  
  
  
  
 @Override  
 **protected** Boolean doInBackground(Void... params) {  
 **boolean** isSuccess = NetUtil.*downloadContacts*(SuperWeChatApplication.*getInstance*(), **userName**, **pageId**, **pageSize**);   
 **if**(isSuccess){  
 isSuccess = NetUtil.*downloadContactList*(**userName**, **pageId**, **pageSize**);   
 }  
*// else{  
// Toast.makeText(context,"服务器下载失败",Toast.LENGTH\_SHORT).show();  
// }* **if**(isSuccess){  
 ArrayList<GroupBean> groupList = SuperWeChatApplication.*getInstance*().getGroupList();  
 ArrayList<GroupBean>groups = NetUtil.*downloadAllGroup*(**userName**);  
 **if**(groups!=**null**){  
 groupList.addAll(groups);  
 Intent intent = **new** Intent(**"update\_group"**);  
 **context**.sendBroadcast(intent);  
 }  
 }  
 **return** isSuccess;  
 }  
   
 @Override  
 **protected void** onPostExecute(Boolean result) {  
 **if**(result){  
 *//向ContactFragment发送更新联系人的广播* Intent intent = **new** Intent(**"update\_contacts"**);  
 **context**.sendBroadcast(intent);  
 }  
 }  
  
}

1. 在SplashActivity.java中的onStart方法中添加下载联系人的代码

*//下载联系人数据*ArrayList<UserBean> contactList = SuperWeChatApplication.*getInstance*().getContactList();  
**if**(contactList.size() == 0){  
 **new** DownloadContactsTask(SplashActivity.**this**, userName, 0, 20).execute();  
}

1. 在LoginActivity的登录方法中添加下载联系人和头像的代码

*//下载头像*String avatar = SuperWeChatApplication.*getInstance*().getUserBean().getAvatar();  
File file = OnSetAvatarListener.*getAvatarFile*(**mContext**, avatar);  
NetUtil.*downloadAvatar*(file, **"user\_avatar"**, avatar);  
*//下载联系人*ArrayList<UserBean> contactList = SuperWeChatApplication.*getInstance*().getContactList();  
Log.*e*(***TAG***,**"LoginActivity.contactList.size="**+contactList.size());  
**if**(contactList.size()==0){  
 **new** DownloadContactsTask(**mContext**, **currentUsername**, 0, 20).execute();  
}

1. 在SettingsFragment的logout方法中添加情况个人信息的代码

SuperWeChatApplication instance = SuperWeChatApplication.*getInstance*();  
instance.getContactList().clear();  
instance.getContacts().clear();  
instance.getGroupList().clear();  
instance.getGroupMembers().clear();

1. 在MessageAdapter中定义以下变量

*/\*\* 登陆的用户\*/*UserBean **mMyUser**;  
*/\*\* 单聊的聊天对象\*/*UserBean **mToUser**;  
  
*/\*\* 群聊的组成员集合\*/*ArrayList<UserBean> **mGroupMembers**;  
  
*/\*\* 加载头像\*/*ImageLoader **mImageLoader**;

1. 在构造函数中添加以下代码

*//获取登录的用户***mMyUser** = SuperWeChatApplication.*getInstance*().getUserBean();  
*//实例化下载头像的任务对象***mImageLoader** = RequestManager.*getImageLoader*();  
**if**(chatType == ChatActivity.***CHATTYPE\_SINGLE***){  
 ArrayList<UserBean> contactList = SuperWeChatApplication.*getInstance*().getContactList();  
 UserBean user = **new** UserBean();  
 user.setUserName(username);  
 **int** id = contactList.indexOf(user);  
 **if**(id>0&&contactList!=**null**&&contactList.size()>0&&id<contactList.size()){  
 **mToUser** = contactList.get(id);  
 }  
}**else**{  
 HashMap<String, ArrayList<UserBean>> groupMembers = SuperWeChatApplication.*getInstance*().getGroupMembers();  
 **mGroupMembers** = groupMembers.get(username);  
}

1. 在Utils.java文件中添加获取对方用户信息的方法

*/\*\*  
 \* 返回发送消息者，发送消息者可能是群聊中成员或单聊中的好友  
 \** ***@param chatType****：群聊/单聊  
 \** ***@param groupId****：群聊的groupId或单聊中的登陆者userName  
 \** ***@param userName****：发送消息者的userName  
 \** ***@return*** *\*/***public static** UserBean getMessageFromUser(EMMessage.ChatType chatType, String groupId, String userName){  
 ArrayList<UserBean> userList = **null**;  
 **switch** (chatType) {  
 **case *GroupChat***:*//群聊* HashMap<String,ArrayList<UserBean>> groupMembers = SuperWeChatApplication.*getInstance*().getGroupMembers();  
 *//获取指定groupId的群聊成员集合* userList = groupMembers.get(groupId);  
 **break**;  
 **case *ChatRoom***:  
 **break**;  
 **default**:*//单聊* userList = SuperWeChatApplication.*getInstance*().getContactList();  
 **break**;  
 }  
 *//获取发送消息者* UserBean user = **new** UserBean();  
 user.setUserName(userName);  
 **if**(userList==**null** ||userList.isEmpty()){  
 **return null**;  
 }  
 **int** id = userList.indexOf(user);  
 **if**(id>=0){  
 **return** userList.get(id);  
 }  
 **return null**;  
}

1. 在MessageAdapter的getView方法添加以下代码，用来显示聊天界面双发的头像,并添加点击头像进入个人主页

*// //设置用户头像，注释掉以下  
// setUserAvatar(message, holder.iv\_avatar);*

String path = **null**;

**if** (message.**direct** == EMMessage.Direct.***SEND***) {  
 path = I.***DOWNLOAD\_AVATAR\_URL*** + **mMyUser**.getAvatar();

… …

} **else** {  
 String from = message.getFrom();  
 UserBean fromUser = Utils.*getMessageFromUser*(message.getChatType(), **username**, from);  
 **if**(fromUser!=**null** && fromUser.getAvatar()!=**null**){  
 path = I.***DOWNLOAD\_AVATAR\_URL*** + fromUser.getAvatar();  
 }

… …

**if**(path != **null**){  
 Log.*e*(***TAG***,**"getView,path="**+path);  
 holder.**iv\_avatar**.setTag(path);  
 holder.**iv\_avatar**.setDefaultImageResId(R.drawable.***default\_avatar***);  
 holder.**iv\_avatar**.setErrorImageResId(R.drawable.***default\_avatar***);  
 holder.**iv\_avatar**.setImageUrl(path,**mImageLoader**);  
  
}  
 holder.**iv\_avatar**.setOnClickListener(**new** View.OnClickListener() {  
  
 @Override  
 **public void** onClick(View v) {  
 Intent intent = **new** Intent();  
 intent.setClass(**context**, UserProfileActivity.**class**);  
 intent.putExtra(**"username"**, message.getFrom());  
 **context**.startActivity(intent);  
 }  
 });

将所有的头像布局文件中的ImageView修改为NetworkImageView

1. 在getView代码中，修改Demo中用username代替昵称的代码

*// // 群聊时，显示接收的消息的发送人的名称  
// if ((chatType == ChatType.GroupChat || chatType == ChatType.ChatRoom) && message.direct == EMMessage.Direct.RECEIVE){  
// //demo里使用username代码nick  
// UserUtils.setUserNick(message.getFrom(), holder.tv\_usernick);  
// }  
// if(message.direct == EMMessage.Direct.SEND){  
// UserUtils.setCurrentUserNick(holder.tv\_usernick);  
// }  
 // 显示接收消息中，消息发送者的昵称* **if**(message.**direct** == EMMessage.Direct.***RECEIVE***){  
 UserBean user = Utils.*getMessageFromUser*(chatType, **username**, message.getFrom());  
 **if**(user!=**null**){  
 holder.**tv\_usernick**.setText(user.getNick());  
 }  
 }

1. 在UserUtils类中编写以下方法，该方法用于为指定的user设置header属性:

**public static void** setUserHearder(UserBean user){  
 String userName = user.getUserName();  
 String headerName = **null**;  
 **if**(!TextUtils.*isEmpty*(user.getNick())){  
 headerName = user.getNick();  
 }**else**{  
 headerName = userName;  
 }  
 **if**(userName.equals(Constant.***NEW\_FRIENDS\_USERNAME***)  
 || userName.equals(Constant.***GROUP\_USERNAME***)){  
 user.setHeader(**""**);  
 }**else if**(Character.*isDigit*(headerName.charAt(0))){  
 user.setHeader(**"#"**);  
 }**else**{  
 user.setHeader(HanziToPinyin.*getInstance*()  
 .get(headerName.substring(0, 1)).get(0).**target**.substring(0,  
 1).toUpperCase());  
 **char** header = user.getHeader().toLowerCase().charAt(0);  
 **if**(header<**'a'** || header>**'z'**){  
 user.setHeader(**"#"**);  
 }  
 }  
}

1. 将activity包中的fragment类提取出来放到新建的fragment包中
2. 在ContactlistFragment中定义以下变量

*/\*\* 被操作的当前联系人\*/***private** UserBean **toBeProcessUser**;  
ContactChangeReceiver **mContactChangeReceiver**;  
*/\*\* 联系人集合\*/*ArrayList<UserBean> **mContactList**=**new** ArrayList<UserBean>();

1. 添加ContactChangeReceiver联系人改变的广播消息接收器

**class** ContactChangeReceiver **extends** BroadcastReceiver {  
  
 @Override  
 **public void** onReceive(Context context, Intent intent) {  
 ArrayList<UserBean> contactList = SuperWeChatApplication.*getInstance*().getContactList();  
 **mContactList**.clear();  
 **mContactList**.addAll(contactList);  
 **for**(UserBean user: contactList){  
 UserUtils.*setUserHearder*(user);  
 }  
 Collections.*sort*(**mContactList**,**new** Comparator<UserBean>() {  
  
 @Override  
 **public int** compare(UserBean user1, UserBean user2) {  
 **return** user1.getHeader().compareTo(user2.getHeader());  
 }  
 });  
 refresh();  
 }  
}

1. 在onActivityCreated中添加registerContactChangeReceiver();注册广播接受者：

**private void** registerContactChangeReceiver() {  
 **mContactChangeReceiver**=**new** ContactChangeReceiver();  
IntentFilter filter=**new** IntentFilter(**"update\_contacts"**);  
getActivity().registerReceiver(**mContactChangeReceiver**, filter);  
 }

在onDestroyView中注销广播接受者

@Override  
**public void** onDestroyView() {  
 **super**.onDestroyView();  
 **if**(**mContactChangeReceiver**!=**null**){  
 getActivity().unregisterReceiver(**mContactChangeReceiver**);  
 }  
}

1. 添加initContactList方法来替换getContactList方法，添加固定的三个头部：申请与通知，群聊，聊天室，并将申请与通知放在顶部

**public void** initContactList(){  
 ArrayList<UserBean> contactList = SuperWeChatApplication.*getInstance*().getContactList();  
 Resources res = getActivity().getResources();  
 UserBean chatRoomUser = **new** UserBean();  
 chatRoomUser.setUserName(Constant.***CHAT\_ROBOT***);  
 chatRoomUser.setNick(res.getString(R.string.***chat\_room***));  
 **if**(**mContactList**.indexOf(chatRoomUser)==-1){  
 **mContactList**.add(0,chatRoomUser);  
 }  
 UserBean groupUser = **new** UserBean();  
 String strGroup = getActivity().getResources().getString(  
 R.string.***group\_chat***);  
 groupUser.setUserName(Constant.***GROUP\_USERNAME***);  
 groupUser.setNick(strGroup);  
 groupUser.setHeader(**""**);  
 **if**(**mContactList**.indexOf(groupUser)==-1){  
 **mContactList**.add(0, groupUser);  
 }  
 UserBean newFriends = **new** UserBean();  
 newFriends.setUserName(Constant.***NEW\_FRIENDS\_USERNAME***);  
 String strChat = getActivity().getResources().getString(  
 R.string.***Application\_and\_notify***);  
 newFriends.setNick(strChat);  
 **if**(**mContactList**.indexOf(newFriends)==-1){  
 **mContactList**.add(0, newFriends);  
 }  
}

1. 修改onCreateContextMenu方法中的头部固定信息的数量为2
2. 修改ContactAdapter父类为BaseAdapter并添加以下变量定义

**private** Context **mContext**;  
*//联系人集合***private** ArrayList<UserBean> **mContactList**;  
*//加载头像的任务类***private** ImageLoader **mImageLoader**;

1. 修改构造函数如下：

**public** ContactAdapter(Context context,**int** resource,ArrayList<UserBean>contactList){  
 **mContext** = context;  
 **mContactList** = contactList;  
 **mImageLoader** = RequestManager.*getImageLoader*();  
 **this**.**res** = resource;  
 **layoutInflater** = LayoutInflater.*from*(context);  
}

1. 修改getItem和getCount方法

@Override  
**public** UserBean getItem(**int** position) {  
 **return mContactList**.get(position);  
}  
  
@Override  
**public int** getCount() {  
 **return mContactList**==**null**?0:**mContactList**.size();  
}

1. 修改getView方法，添加头像的设置

*//设置用户头像  
//UserUtils.setUserAvatar(getContext(), username, holder.avatar);***if**(holder.**unreadMsgView** != **null**){  
 holder.**unreadMsgView**.setVisibility(View.***INVISIBLE***);  
}  
String path = I.***DOWNLOAD\_AVATAR\_URL***+user.getAvatar();  
 holder.**avatar**.setDefaultImageResId(R.drawable.***default\_avatar***);  
 holder.**avatar**.setErrorImageResId(R.drawable.***default\_avatar***);  
 holder.**avatar**.setImageUrl(path,**mImageLoader**);

修改布局文件中的ImageView为NetworkImageView

1. 重写父类的getItemId方法，删除Filter，使用mContext替换getContext
2. 删除中onActivityCreated对getFilter的调用，注释掉notifyDataSetChanged中对copyUserList的操作
3. 修改ContactlistFragment中的adapter

*// 设置adapter***adapter** = **new** ContactAdapter(getActivity(), R.layout.***row\_contact***, **mContactList**);

1. 修改deleteContact的参数为UserBean，并未adapter添加remove方法

**public void** remove(UserBean user){  
 **if**(**mContactList**!=**null**&&!**mContactList**.isEmpty()){  
 **mContactList**.remove(user);  
 notifyDataSetChanged();  
 }  
}

1. 在onActivityCreated中添加setListener()方法，整理
2. 修改PickContactNoCheckboxActivity中的adapter，修改getContactList

*// 设置adapter***contactAdapter** = **new** ContactAdapter(**this**, R.layout.***row\_contact***, **contactList**);

**private void** getContactList() {  
 **contactList**.clear();  
 ArrayList<UserBean> userList = SuperWeChatApplication.*getInstance*().getContactList();  
 userList.addAll(**contactList**);  
 *// 排序* Collections.*sort*(**contactList**, **new** Comparator<UserBean>() {  
  
 @Override  
 **public int** compare(UserBean lhs, UserBean rhs) {  
 **return** lhs.getUserName().compareTo(rhs.getUserName());  
 }  
 });  
}

1. 修改GroupPickContactsActivity中集成ContactAdapter的adapter的构造函数参数为ArrayList<UserBean>
2. 修改GroupPickContactsActivity中获取好友列表的代码

*// 获取好友列表* **final** ArrayList<UserBean> alluserList = **new** ArrayList<UserBean>();  
*// for (User user : ((DemoHXSDKHelper)HXSDKHelper.getInstance()).getContactList().values()) {  
// if (!user.getUsername().equals(Constant.NEW\_FRIENDS\_USERNAME) & !user.getUsername().equals(Constant.GROUP\_USERNAME) & !user.getUsername().equals(Constant.CHAT\_ROOM) & !user.getUsername().equals(Constant.CHAT\_ROBOT))  
// alluserList.add(user);  
// }* ArrayList<UserBean> contactList = SuperWeChatApplication.*getInstance*().getContactList();

1. 修改ForwardMessageActivity中的User为UserBean
2. 在ChatAllHistoryAdapter中添加ContactChangeReceiver来接受联系人改变的广播，添加注册和注销的方法，并在构造函数中调用注册的方法

ContactChangeReceiver **mReceiver**;

**private void** registerContactChangeReceiver(Context context) {  
 **mReceiver** = **new** ContactChangeReceiver();  
 IntentFilter filter = **new** IntentFilter(**"update\_contacts"**);  
 **mContext**.registerReceiver(**mReceiver**, filter);  
}

**class** ContactChangeReceiver **extends** BroadcastReceiver {  
 @Override  
 **public void** onReceive(Context context, Intent intent) {  
 notifyDataSetChanged();  
 context.unregisterReceiver(**this**);

}  
}

1. 在UserUtils中添加用户数组转换为map的方法

**public static** HashMap<String, UserBean> array2Map(ArrayList<UserBean> arraylist){  
 HashMap<String, UserBean> map = **new** HashMap<String, UserBean>();  
 **if**(arraylist==**null**){  
 **return null**;  
 }  
 **for**(UserBean u:arraylist){  
 map.put(u.getUserName(), u);  
 }  
 **return** map;  
}

1. 添加getContactList()方法，获取联系人列表，并转换为map格式

**private void** getContactList(){  
 **mList** = SuperWeChatApplication.*getInstance*().getContactList();  
 **if**(**mList**!=**null**&&**mList**.size()>0){  
 **mConversationUserMap** = UserUtils.*array2Map*(**mList**);  
 }  
}

1. 修改adapter布局文件中头像控件类型为NetworkImageView，在getView中设置头像

*// 获取用户username或者群组groupid*String username = conversation.getUserName();  
getContactList();  
String path;

… …

*// UserUtils.setUserAvatar(getContext(), username, holder.avatar);*holder.**avatar**.setDefaultImageResId(R.drawable.***default\_avatar***);  
holder.**avatar**.setErrorImageResId(R.drawable.***default\_avatar***);  
**if**(**mConversationUserMap**!=**null** && **mConversationUserMap**.size()>0){  
 **if**(**mConversationUserMap**.containsKey(username)){  
 path = **mConversationUserMap**.get(username).getAvatar();  
 **if**(path!=**null**){  
 holder.**avatar**.setImageUrl(I.***DOWNLOAD\_AVATAR\_URL***+path,**mImageLoader**);  
 }  
 }  
}

1. 修改添加好友的布局文件activity\_add\_contact.xml，将ImageView修改为NetworkImageView，将搜索和添加好友的按钮点击事件删除，改为使用代码，添加没有搜索到结果的TextView

<**TextView  
 android:id="@+id/tv\_search\_noting"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:gravity="center\_horizontal"  
 android:paddingTop="50dp"  
 android:text="无结果"  
 android:textSize="20sp"  
 android:visibility="gone"**/>

1. 在AddContactActivity中定义以下变量，并在onCreate中初始化

**private** Button **mbtnAddUser**;  
ImageLoader **mImageLoader**;  
TextView **mtvSearchNoting**;  
Context **mContext**;

**avatar** = (NetworkImageView) findViewById(R.id.***avatar***);   
**mbtnAddUser** = (Button) findViewById(R.id.***indicator***);  
**mtvSearchNoting** = (TextView) findViewById(R.id.***tv\_search\_noting***);  
**mImageLoader** = RequestManager.*getImageLoader*();

1. 整理搜索和添加方法到setListener()中，并在onCreate中调用

**private void** setListener(){  
 searchContactListener();  
 addContactListener();  
}

1. 仿照微信的搜索流程，修改搜索方法

**public void** searchContactListener() {  
 **searchBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **final** String name = **editText**.getText().toString();  
 String saveText = **searchBtn**.getText().toString();  
 **if**(SuperWeChatApplication.*getInstance*().getUserName().equals(name)){  
 String str = getString(R.string.***not\_add\_myself***);  
 startActivity(**new** Intent(**mContext**, AlertDialog.**class**).putExtra(**"msg"**, str));  
 **return**;  
 }  
  
 **toAddUsername** = name;  
 **if**(TextUtils.*isEmpty*(**toAddUsername**)) {  
 String st = getResources().getString(R.string.***Please\_enter\_a\_username***);  
 startActivity(**new** Intent(**mContext**, AlertDialog.**class**).putExtra(**"msg"**, st));  
 **return**;  
 }  
 *//* ***TODO 从服务器获取此contact,如果不存在提示不存在此用户* try** {  
 String path = **new** ApiParams()  
 .with(I.***KEY\_REQUEST***, I.***REQUEST\_FIND\_USER***)  
 .with(I.User.***USER\_NAME***, **toAddUsername**)  
 .getUrl(SuperWeChatApplication.*SERVER\_ROOT*);  
 executeRequest(**new** GsonRequest<UserBean>(path, UserBean.**class**,  
 responseListener(), errorListener()));  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 });  
}

1. 添加回调函数，处理界面的显示

**private** Response.Listener<UserBean> responseListener() {  
 **return new** Response.Listener<UserBean>() {  
 @Override  
 **public void** onResponse(UserBean user) {  
 Log.*e*(***TAG***,**"user="**+user);  
 **if**(user!=**null**){  
 **mtvSearchNoting**.setVisibility(View.***GONE***);  
 *//如果已是好友，则跳转到好友个人主页* ArrayList<UserBean> contactList = SuperWeChatApplication.*getInstance*().getContactList();  
 **if**(contactList.contains(user)){  
 Intent intent = **new** Intent();  
 intent.setClass(**mContext**, UserProfileActivity.**class**);  
 intent.putExtra(**"username"**, user.getUserName());  
 intent.putExtra(**"user"**, user);  
 **mContext**.startActivity(intent);  
 }**else**{  
 *//服务器存在此用户，显示此用户和添加按钮* **searchedUserLayout**.setVisibility(View.***VISIBLE***);  
 **nameText**.setText(user.getNick()==**null**?user.getUserName():user.getNick());  
 **avatar**.setDefaultImageResId(R.drawable.***default\_avatar***);  
 **avatar**.setErrorImageResId(R.drawable.***default\_avatar***);  
 **avatar**.setImageUrl(I.***DOWNLOAD\_AVATAR\_URL***+user.getAvatar(),**mImageLoader**);  
 }  
 }**else**{  
 **searchedUserLayout**.setVisibility(View.***GONE***);  
 **mtvSearchNoting**.setVisibility(View.***VISIBLE***);  
 }  
 }  
 };  
}

1. 修改添加好友的方法

**public void** addContactListener(){  
 **mbtnAddUser**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **if**(((DemoHXSDKHelper) HXSDKHelper.*getInstance*()).getContactList().containsKey(**nameText**.getText().toString())){  
 *//提示已在好友列表中，无需添加* **if**(EMContactManager.*getInstance*().getBlackListUsernames().contains(**nameText**.getText().toString())){  
 startActivity(**new** Intent(**mContext**, AlertDialog.**class**).putExtra(**"msg"**, **"此用户已是你好友(被拉黑状态)，从黑名单列表中移出即可"**));  
 **return**;  
 }  
 **return**;  
 }  
 **progressDialog** = **new** ProgressDialog(**mContext**);  
 String stri = getResources().getString(R.string.***Is\_sending\_a\_request***);  
 **progressDialog**.setMessage(stri);  
 **progressDialog**.setCanceledOnTouchOutside(**false**);  
 **progressDialog**.show();  
 **new** Thread(**new** Runnable() {  
 **public void** run() {  
 **try** {  
 *//demo写死了个reason，实际应该让用户手动填入* String s = getResources().getString(R.string.***Add\_a\_friend***);  
 EMContactManager.*getInstance*().addContact(**toAddUsername**, s);  
 runOnUiThread(**new** Runnable() {  
 **public void** run() {  
 **progressDialog**.dismiss();  
 String s1 = getResources().getString(R.string.***send\_successful***);  
 Toast.*makeText*(getApplicationContext(), s1, Toast.***LENGTH\_SHORT***).show();  
 }  
 });  
 } **catch** (**final** Exception e) {  
 runOnUiThread(**new** Runnable() {  
 **public void** run() {  
 **progressDialog**.dismiss();  
 String s2 = getResources().getString(R.string.***Request\_add\_buddy\_failure***);  
 Toast.*makeText*(getApplicationContext(), s2 + e.getMessage(), Toast.***LENGTH\_SHORT***).show();  
 }  
 });  
 }  
 }  
 }).start();  
 }  
 });  
}

1. 修改activity\_user\_profile.xml中的ImageView为NetworkImageView
2. 在UserProfileActivity中添加以下变量的定义

ImageLoader **mImageLoader**;  
UserBean **mUser**;

1. 在onCreate中初始化mImageLoader
2. 在initView中初始化NetworkImageView

**headAvatar** = (NetworkImageView) findViewById(R.id.***user\_head\_avatar***);  
**headAvatar**.setDefaultImageResId(R.drawable.***default\_avatar***);  
**headAvatar**.setErrorImageResId(R.drawable.***default\_avatar***);

1. 修改initListener方法

**private void** initListener() {  
 Intent intent = getIntent();  
 String username = intent.getStringExtra(**"username"**);  
 **mUser** = (UserBean) intent.getSerializableExtra(**"user"**);  
 **boolean** enableUpdate = intent.getBooleanExtra(**"setting"**, **false**);  
**... ...**  
 **if** (**mUser** == **null**) {  
 **mUser** = SuperWeChatApplication.*getInstance*().getUserBean();  
 }  
 **tvUsername**.setText(**mUser**.getUserName());  
 **tvNickName**.setText(**mUser**.getNick());  
 **headAvatar**.setImageUrl(I.***DOWNLOAD\_AVATAR\_URL***+**mUser**.getAvatar(),**mImageLoader**);  
}

1. 将MessageAdapter中getView方法里面点击头像跳转到个人首页的方法修改如下：

holder.**iv\_avatar**.setOnClickListener(**new** View.OnClickListener() {  
  
 @Override  
 **public void** onClick(View v) {  
 Intent intent = **new** Intent();  
 intent.setClass(**context**, UserProfileActivity.**class**);  
 intent.putExtra(**"username"**, message.getFrom());  
 intent.putExtra(**"user"**, Utils.*getMessageFromUser*(message.getChatType(), **username**, message.getFrom()));  
 **context**.startActivity(intent);  
 }  
});

1. 在MainActivity.MyContactList.onContactAdded中添加以下代码

**boolean** exists = **false**;  
**for** (String username : usernameList) {  
 User user = setUserHead(username);  
 *// 添加好友时可能会回调added方法两次* **if** (!localUsers.containsKey(username)) {  
 **userDao**.saveContact(user);  
 exists = **true**;  
 }  
 toAddUsers.put(username, user);  
}  
localUsers.putAll(toAddUsers);  
*// 刷新ui***if** (**currentTabIndex** == 1)  
 **contactListFragment**.refresh();  
 SuperWeChatApplication instance = SuperWeChatApplication.*getInstance*();  
 String currentUserName = instance.getUserName();  
ArrayList<UserBean> contactList = instance.getContactList();  
ArrayList<String> addContactList = **new** ArrayList<String>();  
 **for**(String username : usernameList){  
 UserBean u = **new** UserBean(username);  
 **if**(!contactList.contains(u)){  
 addContactList.add(username);  
 }  
 }  
**if**(exists && addContactList.size()>0){  
 **try** {  
 **for** (String name : addContactList) {  
 String path = **new** ApiParams()  
 .with(I.***KEY\_REQUEST***, I.***REQUEST\_ADD\_CONTACT***)  
 .with(I.User.***USER\_NAME***, currentUserName)  
 .with(I.Contact.***NAME***, name)  
 .getUrl(SuperWeChatApplication.*SERVER\_ROOT*);  
 executeRequest(**new** GsonRequest<ContactBean>(path, ContactBean.**class**,  
 responseListener(name), errorListener()));  
 }  
 }**catch** (Exception e){  
 e.printStackTrace();  
 }  
}

1. 添加回调函数responseListener

**private** Response.Listener<ContactBean> responseListener(**final** String contactName) {  
 **return new** Response.Listener<ContactBean>() {  
 @Override  
 **public void** onResponse(ContactBean contact) {  
 **if**(contact!=**null** && **"ok"**.equals(contact.getResult())) {  
 *//获取好友集合* HashMap<Integer, ContactBean> contacts = SuperWeChatApplication.*getInstance*().getContacts();  
 *//若添加成功，则将新好友保存在内存集合* contacts.put(contact.getMyuid(), contact);  
 *//从服务端下载新好友* **try** {  
 String path = **new** ApiParams()  
 .with(I.***KEY\_REQUEST***, I.***REQUEST\_FIND\_USER***)  
 .with(I.User.***USER\_NAME***, contactName)  
 .getUrl(SuperWeChatApplication.*SERVER\_ROOT*);  
 executeRequest(**new** GsonRequest<UserBean>(path, UserBean.**class**,  
 responseUserBeanListener(), errorListener()));  
 }**catch** (Exception e){  
 e.printStackTrace();  
 }  
 }  
 }  
 };  
}

1. 添加回调函数responseUserBeanListener

**private** Response.Listener<UserBean> responseUserBeanListener(){  
 **return new** Response.Listener<UserBean>(){  
 @Override  
 **public void** onResponse(UserBean user) {  
 **if**(user!=**null**) {  
 *//获取登陆用户的所有好友* ArrayList<UserBean> contactList = SuperWeChatApplication.*getInstance*().getContactList();  
 contactList.add(user);  
 *//若有新好友，向ContactListFragment发送刷新牧场好友的广播* Intent intent = **new** Intent(**"update\_contacts"**);  
 **mContext**.sendStickyBroadcast(intent);  
 Utils.*showToast*(**mContext**, R.string.***Add\_buddy\_success***, Toast.***LENGTH\_SHORT***);  
 }  
 }  
 };  
}

1. 修改ContactAdapter中申请与通知的未读消息标识判断

**int** c=((DemoHXSDKHelper) HXSDKHelper.*getInstance*()).getContactList().get(Constant.***NEW\_FRIENDS\_USERNAME***).getUnreadMsgCount();  
**if**(user.getUnreadMsgCount() > 0 || c >0){  
 holder.**unreadMsgView**.setVisibility(View.***VISIBLE***);  
*// holder.unreadMsgView.setText(user.getUnreadMsgCount()+"");*}**else**{  
 holder.**unreadMsgView**.setVisibility(View.***INVISIBLE***);  
}

1. 在MainActivity.MyContactList.onContactDeleted()中添加以下代码

*//删除应用服务器的好友关系*HashMap<Integer, ContactBean> contacts = SuperWeChatApplication.*getInstance*().getContacts();  
ArrayList<UserBean> contactList = SuperWeChatApplication.*getInstance*().getContactList();  
ArrayList<ContactBean> deleteContacts = **new** ArrayList<ContactBean>();  
ArrayList<UserBean> deleteContactList = **new** ArrayList<UserBean>();  
*//删除内存中好友，删除的好友存放在deleteContactList和deleteContacts集合中* **for**(UserBean contactUser : contactList){  
 **if**(usernameList.contains(contactUser.getUserName())){  
 ContactBean contact = contacts.remove(contactUser.getId());  
 deleteContacts.add(contact);  
 deleteContactList.add(contactUser);  
 }  
}  
**if**(deleteContacts.size()>0){  
 contactList.removeAll(deleteContactList);*//删除内存中好友  
 // 删除应用服务器的联系人记录* **try** {  
 **for**(ContactBean contact : deleteContacts) {  
 String path = **new** ApiParams()  
 .with(I.***KEY\_REQUEST***, I.***REQUEST\_DELETE\_CONTACT***)  
 .with(I.Contact.***MYUID***, contact.getMyuid() + **""**)  
 .with(I.Contact.***CUID***, contact.getCuid() + **""**)  
 .getUrl(SuperWeChatApplication.*SERVER\_ROOT*);  
 Log.*e*(***TAG***,**"delete contacts,path="**+path);  
 executeRequest(**new** GsonRequest<Boolean>(path, Boolean.**class**,  
 responseDeleteContactListener(), errorListener()));  
 }  
 }**catch** (Exception e){  
 e.printStackTrace();  
 }  
}

1. 添加回调函数responseDeleteContactListener

**private** Response.Listener<Boolean> responseDeleteContactListener(){  
 **return new** Response.Listener<Boolean>() {  
 @Override  
 **public void** onResponse(Boolean isSuccess) {  
 **if**(isSuccess){  
 Intent intent = **new** Intent(**"update\_contacts"**);  
 **mContext**.sendBroadcast(intent);  
 }  
 }  
 };  
}

1. 在ContactlistFragment中添加变量并在重写父类的onAttach 中初始化

MainActivity **mContext**;

@Override  
**public void** onAttach(Context context) {  
 **super**.onAttach(context);  
 **mContext** = (MainActivity) context;  
}

1. 在deleteContact方法中添加以下代码

*//删除应用服务器的好友关系*HashMap<Integer, ContactBean> contacts = SuperWeChatApplication.*getInstance*().getContacts();  
ArrayList<UserBean> contactList = SuperWeChatApplication.*getInstance*().getContactList();  
ArrayList<ContactBean> deleteContacts = **new** ArrayList<ContactBean>();  
ArrayList<UserBean> deleteContactList = **new** ArrayList<UserBean>();  
*//删除内存中好友，删除的好友存放在deleteContactList和deleteContacts集合中***for**(UserBean contactUser : contactList){  
 **if**(tobeDeleteUser.equals(contactUser)){  
 ContactBean contact = contacts.remove(contactUser.getId());  
 deleteContacts.add(contact);  
 deleteContactList.add(contactUser);  
 }  
}  
**if**(deleteContacts.size()>0){  
 contactList.removeAll(deleteContactList);*//删除内存中好友  
 // 删除应用服务器的联系人记录* **try** {  
 **for**(ContactBean contact : deleteContacts) {  
 String path = **new** ApiParams()  
 .with(I.***KEY\_REQUEST***, I.***REQUEST\_DELETE\_CONTACT***)  
 .with(I.Contact.***MYUID***, contact.getMyuid() + **""**)  
 .with(I.Contact.***CUID***, contact.getCuid() + **""**)  
 .getUrl(SuperWeChatApplication.*SERVER\_ROOT*);  
 Log.*e*(***TAG***,**"fragment delete contacts,path="**+path);  
 **mContext**.executeRequest(**new** GsonRequest<Boolean>(path, Boolean.**class**,  
 responseDeleteContactListener(), **mContext**.errorListener()));  
 }  
 }**catch** (Exception e){  
 e.printStackTrace();  
 }  
}  
EMContactManager.*getInstance*().deleteContact(tobeDeleteUser.getUserName());  
*// 删除db和内存中此用户的数据*EMUserDao dao = **new** EMUserDao(getActivity());  
dao.deleteContact(tobeDeleteUser.getUserName());

1. 添加回调函数responseDeleteContactListener

**private** Response.Listener<Boolean> responseDeleteContactListener(){  
 **return new** Response.Listener<Boolean>() {  
 @Override  
 **public void** onResponse(Boolean isSuccess) {  
 **if**(isSuccess){  
 Intent intent = **new** Intent(**"update\_contacts"**);  
 **mContext**.sendBroadcast(intent);  
 }  
 }  
 };  
}

1. d
2. d
3. d
4. d
5. d
6. d
7. d
8. d
9. d
10. d
11. d
12. d
13. d
14. d
15. d
16. d
17. d
18. d
19. d
20. d
21. d
22. d
23. d
24. dd