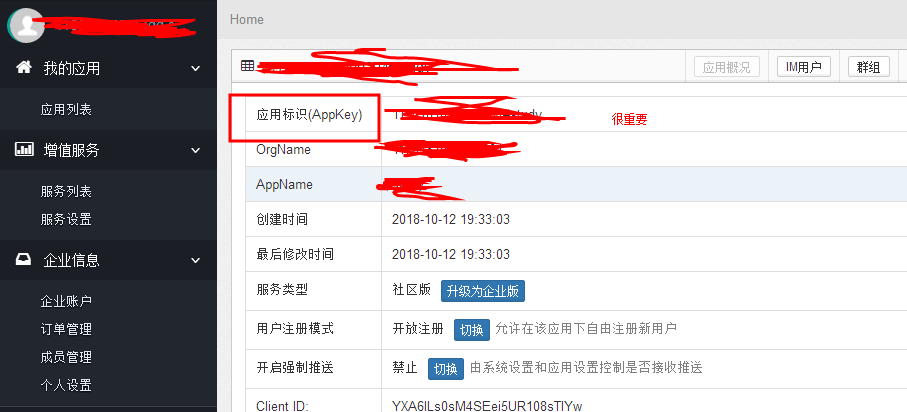
## 环信

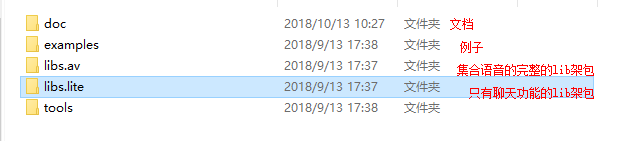
1下载架包：

<https://download-sdk.oss-cn-beijing.aliyuncs.com/downloads%2Feasemob-sdk-3.5.1.zip>

1. 申请环信的账号：

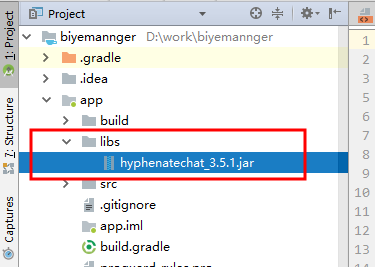


1. 打开下载的zip压缩包



这里我们只要用libs.lite包就可以了

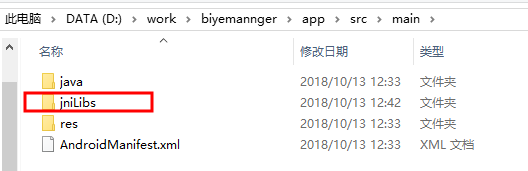
1. 导入libs.lite包里的架包



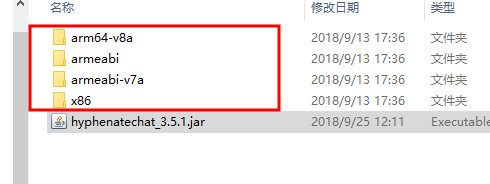
记得还要add as library

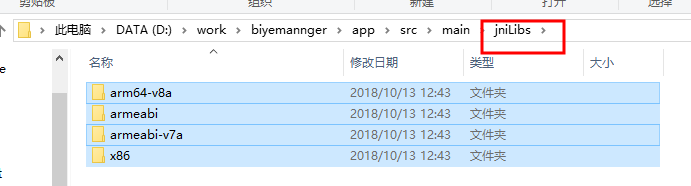


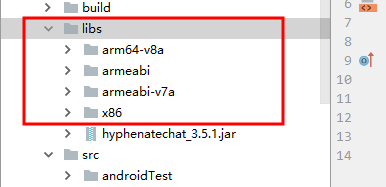
1. 在src main 里面新建jniLibs文件



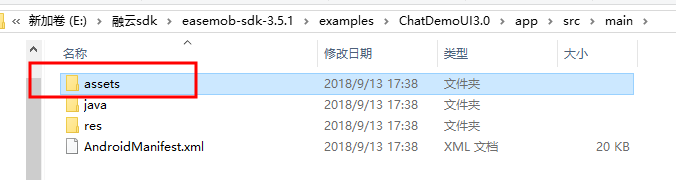
1. 将libs.lite里的其他文件拷贝到jniLibs里面和libs下面

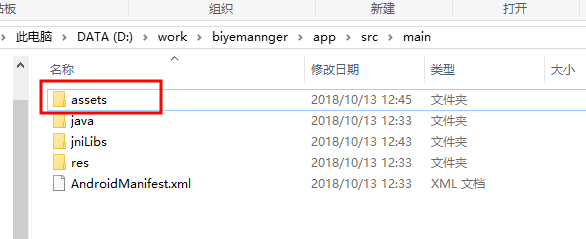






1. 将examples里的assets拷贝到工程的main下面





1. 在applicationContext.xml添加环信的appkey

|  |
| --- |
| *<!-- 设置环信应用的AppKey -->* <**meta-data  android:name="EASEMOB\_APPKEY"  android:value="1170181012099204#study"** />  *<!-- 声明SDK所需的service SDK核心功能-->* <**service android:name="com.hyphenate.chat.EMChatService" android:exported="true"**/> <**service android:name="com.hyphenate.chat.EMJobService"  android:permission="android.permission.BIND\_JOB\_SERVICE"  android:exported="true"** /> *<!-- 声明SDK所需的receiver -->* <**receiver android:name="com.hyphenate.chat.EMMonitorReceiver"**>  <**intent-filter**>  <**action android:name="android.intent.action.PACKAGE\_REMOVED"**/>  <**data android:scheme="package"**/>  </**intent-filter**>  *<!-- 可选filter -->* <**intent-filter**>  <**action android:name="android.intent.action.BOOT\_COMPLETED"**/>  <**action android:name="android.intent.action.USER\_PRESENT"** />  </**intent-filter**> </**receiver**> |

1. build一下，看下有问题没
2. 在运行工程的时候，一定要初始化sdk

|  |
| --- |
| 新建类ECApplication.java复制下面类容  **public class** ECApplication **extends** Application {   *// 上下文菜单* **private** Context **mContext**;   *// 记录是否已经初始化* **private boolean isInit** = **false**;   @Override **public void** onCreate() {  **super**.onCreate();  **mContext** = **this**;   *// 初始化环信SDK* initEasemob();  }   */\*\*  \*  \*/* **private void** initEasemob() {  *// 获取当前进程 id 并取得进程名* **int** pid = android.os.Process.*myPid*();  String processAppName = getAppName(pid);  */\*\*  \* 如果app启用了远程的service，此application:onCreate会被调用2次  \* 为了防止环信SDK被初始化2次，加此判断会保证SDK被初始化1次  \* 默认的app会在以包名为默认的process name下运行，如果查到的process name不是app的process name就立即返回  \*/* Log.*i*(**"asd"**,**"我执行了"**);  **if** (processAppName == **null** || !processAppName.equalsIgnoreCase(**mContext**.getPackageName())) {  *// 则此application的onCreate 是被service 调用的，直接返回* **return**;  }  **if** (**isInit**) {  **return**;  }   *// 调用初始化方法初始化sdk* EMClient.*getInstance*().init(**mContext**, initOptions());  *// 设置开启debug模式* EMClient.*getInstance*().setDebugMode(**true**);   *// 设置初始化已经完成* **isInit** = **true**;  }   */\*\*  \* SDK初始化的一些配置  \* 关于 EMOptions 可以参考官方的 API 文档  \* http://www.easemob.com/apidoc/android/chat3.0/classcom\_1\_1hyphenate\_1\_1chat\_1\_1\_e\_m\_options.html  \*/* **private** EMOptions initOptions() {   EMOptions options = **new** EMOptions();  *// 设置Appkey，如果配置文件已经配置，这里可以不用设置  // options.setAppKey("lzan13#hxsdkdemo");  // 设置自动登录* options.setAutoLogin(**true**);  *// 设置是否需要发送已读回执* options.setRequireAck(**true**);  *// 设置是否需要发送回执，* options.setRequireDeliveryAck(**true**);  *// 设置是否根据服务器时间排序，默认是true* options.setSortMessageByServerTime(**false**);  *// 收到好友申请是否自动同意，如果是自动同意就不会收到好友请求的回调，因为sdk会自动处理，默认为true* options.setAcceptInvitationAlways(**false**);  *// 设置是否自动接收加群邀请，如果设置了当收到群邀请会自动同意加入* options.setAutoAcceptGroupInvitation(**false**);  *// 设置（主动或被动）退出群组时，是否删除群聊聊天记录* options.setDeleteMessagesAsExitGroup(**false**);  *// 设置是否允许聊天室的Owner 离开并删除聊天室的会话* options.allowChatroomOwnerLeave(**true**);  *// 设置google GCM推送id，国内可以不用设置  // options.setGCMNumber(MLConstants.ML\_GCM\_NUMBER);  // 设置集成小米推送的appid和appkey  // options.setMipushConfig(MLConstants.ML\_MI\_APP\_ID, MLConstants.ML\_MI\_APP\_KEY);* **return** options;  }   */\*\*  \* 根据Pid获取当前进程的名字，一般就是当前app的包名  \*  \** ***@param pid*** *进程的id  \** ***@return*** *返回进程的名字  \*/* **private** String getAppName(**int** pid) {  String processName = **null**;  ActivityManager activityManager =  (ActivityManager) **this**.getSystemService(Context.***ACTIVITY\_SERVICE***);  List list = activityManager.getRunningAppProcesses();  Iterator i = list.iterator();  **while** (i.hasNext()) {  ActivityManager.RunningAppProcessInfo info =  (ActivityManager.RunningAppProcessInfo) (i.next());  **try** {  **if** (info.**pid** == pid) {  *// 根据进程的信息获取当前进程的名字* processName = info.**processName**;  *// 返回当前进程名* **return** processName;  }  } **catch** (Exception e) {  e.printStackTrace();  }  }  *// 没有匹配的项，返回为null* **return** processName;  } }  修改AndroidManifest.xml name写成刚才建的类    添加权限  *<!-- Required -->* <**uses-permission android:name="android.permission.VIBRATE"** /> <**uses-permission android:name="android.permission.INTERNET"** /> <**uses-permission android:name="android.permission.RECORD\_AUDIO"** /> <**uses-permission android:name="android.permission.CAMERA"** /> <**uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"** /> <**uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"** /> <**uses-permission android:name="android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS"  tools:ignore="ProtectedPermissions"** /> <**uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"** /> <**uses-permission android:name="android.permission.GET\_TASKS"** /> <**uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE"** /> <**uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE"** /> <**uses-permission android:name="android.permission.WAKE\_LOCK"** /> <**uses-permission android:name="android.permission.MODIFY\_AUDIO\_SETTINGS"** /> <**uses-permission android:name="android.permission.READ\_PHONE\_STATE"** /> <**uses-permission android:name="android.permission.RECEIVE\_BOOT\_COMPLETED"** /> |

1. 开始测试
2. 新建空的活动 ECChatActivity ECLoginActivity ECMainActivity

|  |
| --- |
| login界面  *<?***xml version="1.0" encoding="utf-8"***?>* <**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:paddingBottom="16dp"  android:paddingLeft="16dp"  android:paddingRight="16dp"  android:paddingTop="16dp"  tools:context=".ECLoginActivity"**>  <**LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical"** >   <**EditText  android:id="@+id/ec\_edit\_username"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:hint="username"  android:inputType="textImeMultiLine"** />   <**EditText  android:id="@+id/ec\_edit\_password"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:hint="password"  android:inputType="textPassword"** />   <**Button  android:id="@+id/ec\_btn\_sign\_up"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="注册"** />   <**Button  android:id="@+id/ec\_btn\_sign\_in"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="登录"** />  </**LinearLayout**> </**RelativeLayout**>  chat界面*<?***xml version="1.0" encoding="utf-8"***?>* <**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:paddingBottom="16dp"  android:paddingLeft="16dp"  android:paddingRight="16dp"  android:paddingTop="16dp"  tools:context=".ECChatActivity"**>  *<!--输入框-->* <**RelativeLayout  android:id="@+id/ec\_layout\_input"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_alignParentBottom="true"**>   <**Button  android:id="@+id/ec\_btn\_send"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_alignParentRight="true"  android:text="Send"**/>   <**EditText  android:id="@+id/ec\_edit\_message\_input"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_alignParentLeft="true"  android:layout\_toLeftOf="@id/ec\_btn\_send"**/>  </**RelativeLayout**>   *<!--展示消息内容-->* <**TextView  android:id="@+id/ec\_text\_content"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:layout\_above="@id/ec\_layout\_input"  android:maxLines="15"  android:scrollbars="vertical"**/> </**RelativeLayout**>  main界面  *<?***xml version="1.0" encoding="utf-8"***?>* <**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:paddingBottom="16dp"  android:paddingLeft="16dp"  android:paddingRight="16dp"  android:paddingTop="16dp"  tools:context=".ECMainActivity"**>  <**LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical"**>   <**EditText  android:id="@+id/ec\_edit\_chat\_id"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:hint="对方的username"**/>   <**Button  android:id="@+id/ec\_btn\_start\_chatee"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="发起聊天"**/>   <**Button  android:id="@+id/ec\_btn\_sign\_out"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="退出登录"**/>  </**LinearLayout**>  </**RelativeLayout**> |

1. ECloginActivity.java

|  |
| --- |
| *// 弹出框* **private** ProgressDialog **mDialog**;  *// username 输入框* **private** EditText **mUsernameEdit**;  *// 密码输入框* **private** EditText **mPasswordEdit**;  *// 注册按钮* **private** Button **mSignUpBtn**;  *// 登录按钮* **private** Button **mSignInBtn**; Context **context**;  @Override **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);   setContentView(R.layout.***activity\_eclogin***);  initView();  }  **private** EMOptions initOptions() {   EMOptions options = **new** EMOptions();  *// 设置Appkey，如果配置文件已经配置，这里可以不用设置  // options.setAppKey("lzan13#hxsdkdemo");  // 设置自动登录* options.setAutoLogin(**true**);  *// 设置是否需要发送已读回执* options.setRequireAck(**true**);  *// 设置是否需要发送回执，* options.setRequireDeliveryAck(**true**);  *// 设置是否根据服务器时间排序，默认是true* options.setSortMessageByServerTime(**false**);  *// 收到好友申请是否自动同意，如果是自动同意就不会收到好友请求的回调，因为sdk会自动处理，默认为true* options.setAcceptInvitationAlways(**false**);  *// 设置是否自动接收加群邀请，如果设置了当收到群邀请会自动同意加入* options.setAutoAcceptGroupInvitation(**false**);  *// 设置（主动或被动）退出群组时，是否删除群聊聊天记录* options.setDeleteMessagesAsExitGroup(**false**);  *// 设置是否允许聊天室的Owner 离开并删除聊天室的会话* options.allowChatroomOwnerLeave(**true**);  *// 设置google GCM推送id，国内可以不用设置  // options.setGCMNumber(MLConstants.ML\_GCM\_NUMBER);  // 设置集成小米推送的appid和appkey  // options.setMipushConfig(MLConstants.ML\_MI\_APP\_ID, MLConstants.ML\_MI\_APP\_KEY);* **return** options;  }  */\*\*  \* 初始化界面控件  \*/* **private void** initView() {  **mUsernameEdit** = findViewById(R.id.***ec\_edit\_username***);  **mPasswordEdit** = findViewById(R.id.***ec\_edit\_password***);   **mSignUpBtn** = findViewById(R.id.***ec\_btn\_sign\_up***);  **mSignUpBtn**.setOnClickListener(**new** View.OnClickListener() {  @Override **public void** onClick(View v) {  signUp();  }  });   **mSignInBtn** = findViewById(R.id.***ec\_btn\_sign\_in***);  **mSignInBtn**.setOnClickListener(**new** View.OnClickListener() {  @Override **public void** onClick(View v) {  Log.*i*(**"zjc"**,**"我进来了"**);  signIn();  }  });  }   */\*\*  \* 注册方法  \*/* **private void** signUp() {  *// 注册是耗时过程，所以要显示一个dialog来提示下用户* **mDialog** = **new** ProgressDialog(**this**);  **mDialog**.setMessage(**"注册中，请稍后..."**);  **mDialog**.show();   **new** Thread(**new** Runnable() {  @Override **public void** run() {  **try** {  String username = **mUsernameEdit**.getText().toString().trim();  String password = **mPasswordEdit**.getText().toString().trim();  EMClient.*getInstance*().createAccount(username, password);  runOnUiThread(**new** Runnable() {  @Override **public void** run() {  **if** (!ECLoginActivity.**this**.isFinishing()) {  **mDialog**.dismiss();  }  Toast.*makeText*(ECLoginActivity.**this**, **"注册成功"**, Toast.***LENGTH\_LONG***).show();  }  });  } **catch** (**final** HyphenateException e) {  e.printStackTrace();  runOnUiThread(**new** Runnable() {  @Override **public void** run() {  **if** (!ECLoginActivity.**this**.isFinishing()) {  **mDialog**.dismiss();  }  */\*\*  \* 关于错误码可以参考官方api详细说明  \* http://www.easemob.com/apidoc/android/chat3.0/classcom\_1\_1hyphenate\_1\_1\_e\_m\_error.html  \*/* **int** errorCode = e.getErrorCode();  String message = e.getMessage();  Log.*d*(**"lzan13"**,  String.*format*(**"sign up - errorCode:%d, errorMsg:%s"**, errorCode,  e.getMessage()));  **switch** (errorCode) {  *// 网络错误* **case** EMError.***NETWORK\_ERROR***:  Toast.*makeText*(ECLoginActivity.**this**,  **"网络错误 code: "** + errorCode + **", message:"** + message,  Toast.***LENGTH\_LONG***).show();  **break**;  *// 用户已存在* **case** EMError.***USER\_ALREADY\_EXIST***:  Toast.*makeText*(ECLoginActivity.**this**,  **"用户已存在 code: "** + errorCode + **", message:"** + message,  Toast.***LENGTH\_LONG***).show();  **break**;  *// 参数不合法，一般情况是username 使用了uuid导致，不能使用uuid注册* **case** EMError.***USER\_ILLEGAL\_ARGUMENT***:  Toast.*makeText*(ECLoginActivity.**this**,  **"参数不合法，一般情况是username 使用了uuid导致，不能使用uuid注册 code: "** + errorCode  + **", message:"** + message, Toast.***LENGTH\_LONG***).show();  **break**;  *// 服务器未知错误* **case** EMError.***SERVER\_UNKNOWN\_ERROR***:  Toast.*makeText*(ECLoginActivity.**this**,  **"服务器未知错误 code: "** + errorCode + **", message:"** + message,  Toast.***LENGTH\_LONG***).show();  **break**;  **case** EMError.***USER\_REG\_FAILED***:  Toast.*makeText*(ECLoginActivity.**this**,  **"账户注册失败 code: "** + errorCode + **", message:"** + message,  Toast.***LENGTH\_LONG***).show();  **break**;  **default**:  Toast.*makeText*(ECLoginActivity.**this**,  **"ml\_sign\_up\_failed code: "** + errorCode + **", message:"** + message,  Toast.***LENGTH\_LONG***).show();  **break**;  }  }  });  } **catch** (Exception e) {  e.printStackTrace();  }  }  }).start();  }   */\*\*  \* 登录方法  \*/* **private void** signIn() {  **mDialog** = **new** ProgressDialog(**this**);  **mDialog**.setMessage(**"正在登陆，请稍后..."**);  **mDialog**.show();  String usernamee = **mUsernameEdit**.getText().toString().trim();  String passworde= **mPasswordEdit**.getText().toString().trim();  **if** (TextUtils.*isEmpty*(usernamee) || TextUtils.*isEmpty*(passworde)) {  Toast.*makeText*(ECLoginActivity.**this**, **"用户名和密码不能为空"**, Toast.***LENGTH\_LONG***).show();  **return**;  }  EMClient.*getInstance*().login(usernamee, passworde, **new** EMCallBack() {  */\*\*  \* 登陆成功的回调  \*/* @Override **public void** onSuccess() {  runOnUiThread(**new** Runnable() {  @Override **public void** run() {  Log.*i*(**"zjc"**,**"wo chenggongl "**);  **mDialog**.dismiss();  *// 加载所有会话到内存* EMClient.*getInstance*().chatManager().loadAllConversations();  *// 加载所有群组到内存，如果使用了群组的话  // EMClient.getInstance().groupManager().loadAllGroups();  // 登录成功跳转界面* Log.*i*(**"zjc"**,**"我进来了llllll"**);  Intent intent = **new** Intent(ECLoginActivity.**this**, ECMainActivity.**class**);  startActivity(intent);  finish();  }  });  }   */\*\*  \* 登陆错误的回调  \** ***@param i*** *\** ***@param s*** *\*/* @Override **public void** onError(**final int** i, **final** String s) {  runOnUiThread(**new** Runnable() {  @Override **public void** run() {  **mDialog**.dismiss();  Log.*d*(**"lzan13"**, **"登录失败 Error code:"** + i + **", message:"** + s);  */\*\*  \* 关于错误码可以参考官方api详细说明  \* http://www.easemob.com/apidoc/android/chat3.0/classcom\_1\_1hyphenate\_1\_1\_e\_m\_error.html  \*/* **switch** (i) {  *// 网络异常 2* **case** EMError.***NETWORK\_ERROR***:  Toast.*makeText*(ECLoginActivity.**this**,  **"网络错误 code: "** + i + **", message:"** + s, Toast.***LENGTH\_LONG***).show();  **break**;  *// 无效的用户名 101* **case** EMError.***INVALID\_USER\_NAME***:  Toast.*makeText*(ECLoginActivity.**this**,  **"无效的用户名 code: "** + i + **", message:"** + s, Toast.***LENGTH\_LONG***).show();  **break**;  *// 无效的密码 102* **case** EMError.***INVALID\_PASSWORD***:  Toast.*makeText*(ECLoginActivity.**this**,  **"无效的密码 code: "** + i + **", message:"** + s, Toast.***LENGTH\_LONG***).show();  **break**;  *// 用户认证失败，用户名或密码错误 202* **case** EMError.***USER\_AUTHENTICATION\_FAILED***:  Toast.*makeText*(ECLoginActivity.**this**,  **"用户认证失败，用户名或密码错误 code: "** + i + **", message:"** + s, Toast.***LENGTH\_LONG***)  .show();  **break**;  *// 用户不存在 204* **case** EMError.***USER\_NOT\_FOUND***:  Toast.*makeText*(ECLoginActivity.**this**,  **"用户不存在 code: "** + i + **", message:"** + s, Toast.***LENGTH\_LONG***).show();  **break**;  *// 无法访问到服务器 300* **case** EMError.***SERVER\_NOT\_REACHABLE***:  Toast.*makeText*(ECLoginActivity.**this**,  **"无法访问到服务器 code: "** + i + **", message:"** + s, Toast.***LENGTH\_LONG***).show();  **break**;  *// 等待服务器响应超时 301* **case** EMError.***SERVER\_TIMEOUT***:  Toast.*makeText*(ECLoginActivity.**this**,  **"等待服务器响应超时 code: "** + i + **", message:"** + s, Toast.***LENGTH\_LONG***)  .show();  **break**;  *// 服务器繁忙 302* **case** EMError.***SERVER\_BUSY***:  Toast.*makeText*(ECLoginActivity.**this**,  **"服务器繁忙 code: "** + i + **", message:"** + s, Toast.***LENGTH\_LONG***).show();  **break**;  *// 未知 Server 异常 303 一般断网会出现这个错误* **case** EMError.***SERVER\_UNKNOWN\_ERROR***:  Toast.*makeText*(ECLoginActivity.**this**,  **"未知的服务器异常 code: "** + i + **", message:"** + s, Toast.***LENGTH\_LONG***).show();  **break**;  **default**:  Toast.*makeText*(ECLoginActivity.**this**,  **"ml\_sign\_in\_failed code: "** + i + **", message:"** + s,  Toast.***LENGTH\_LONG***).show();  **break**;  }  }  });  }   @Override **public void** onProgress(**int** i, String s) {   }  });  } |

1. ECChatActivity.java

|  |
| --- |
| **package** com.example.a11699.biyemannger;  **import** android.os.Handler; **import** android.os.Message; **import** android.support.v7.app.AppCompatActivity; **import** android.os.Bundle; **import** android.text.TextUtils; **import** android.text.method.ScrollingMovementMethod; **import** android.util.Log; **import** android.view.View; **import** android.widget.Button; **import** android.widget.EditText; **import** android.widget.TextView;  **import** com.hyphenate.EMCallBack; **import** com.hyphenate.EMMessageListener; **import** com.hyphenate.chat.EMClient; **import** com.hyphenate.chat.EMCmdMessageBody; **import** com.hyphenate.chat.EMConversation; **import** com.hyphenate.chat.EMMessage; **import** com.hyphenate.chat.EMTextMessageBody;  **import** java.util.List;  **public class** ECChatActivity **extends** AppCompatActivity **implements** EMMessageListener {   *// 聊天信息输入框* **private** EditText **mInputEdit**;  *// 发送按钮* **private** Button **mSendBtn**;   *// 显示内容的 TextView* **private** TextView **mContentText**;   *// 消息监听器* **private** EMMessageListener **mMessageListener**;  *// 当前聊天的 ID* **private** String **mChatId**;  *// 当前会话对象* **private** EMConversation **mConversation**;   @Override **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***activity\_ecchat***);   *// 获取当前会话的username(如果是群聊就是群id)* **mChatId** = getIntent().getStringExtra(**"ec\_chat\_id"**);  **mMessageListener** = **this**;   initView();  initConversation();  }   */\*\*  \* 初始化界面  \*/* **private void** initView() {  **mInputEdit** = (EditText) findViewById(R.id.***ec\_edit\_message\_input***);  **mSendBtn** = (Button) findViewById(R.id.***ec\_btn\_send***);  **mContentText** = (TextView) findViewById(R.id.***ec\_text\_content***);  *// 设置textview可滚动，需配合xml布局设置* **mContentText**.setMovementMethod(**new** ScrollingMovementMethod());   *// 设置发送按钮的点击事件* **mSendBtn**.setOnClickListener(**new** View.OnClickListener() {  @Override **public void** onClick(View v) {  String content = **mInputEdit**.getText().toString().trim();  **if** (!TextUtils.*isEmpty*(content)) {  **mInputEdit**.setText(**""**);  *// 创建一条新消息，第一个参数为消息内容，第二个为接受者username* EMMessage message = EMMessage.*createTxtSendMessage*(content, **mChatId**);  *// 将新的消息内容和时间加入到下边* **mContentText**.setText(**mContentText**.getText()  + **"\n发送："** + content  + **" - time: "** + message.getMsgTime());  *// 调用发送消息的方法* EMClient.*getInstance*().chatManager().sendMessage(message);  *// 为消息设置回调* message.setMessageStatusCallback(**new** EMCallBack() {  @Override **public void** onSuccess() {  *// 消息发送成功，打印下日志，正常操作应该去刷新ui* Log.*i*(**"lzan13"**, **"send message on success"**);  }   @Override **public void** onError(**int** i, String s) {  *// 消息发送失败，打印下失败的信息，正常操作应该去刷新ui* Log.*i*(**"lzan13"**, **"send message on error "** + i + **" - "** + s);  }   @Override **public void** onProgress(**int** i, String s) {  *// 消息发送进度，一般只有在发送图片和文件等消息才会有回调，txt不回调* }  });  }  }  });  }   */\*\*  \* 初始化会话对象，并且根据需要加载更多消息  \*/* **private void** initConversation() {   */\*\*  \* 初始化会话对象，这里有三个参数么，  \* 第一个表示会话的当前聊天的 useranme 或者 groupid  \* 第二个是绘画类型可以为空  \* 第三个表示如果会话不存在是否创建  \*/* **mConversation** = EMClient.*getInstance*().chatManager().getConversation(**mChatId**, **null**, **true**);  *// 设置当前会话未读数为 0* **mConversation**.markAllMessagesAsRead();  **int** count = **mConversation**.getAllMessages().size();  **if** (count < **mConversation**.getAllMsgCount() && count < 20) {  *// 获取已经在列表中的最上边的一条消息id* String msgId = **mConversation**.getAllMessages().get(0).getMsgId();  *// 分页加载更多消息，需要传递已经加载的消息的最上边一条消息的id，以及需要加载的消息的条数* **mConversation**.loadMoreMsgFromDB(msgId, 20 - count);  }  *// 打开聊天界面获取最后一条消息内容并显示* **if** (**mConversation**.getAllMessages().size() > 0) {  EMMessage messge = **mConversation**.getLastMessage();  EMTextMessageBody body = (EMTextMessageBody) messge.getBody();  *// 将消息内容和时间显示出来* **mContentText**.setText(  **"聊天记录："** + body.getMessage() + **" - time: "** + **mConversation**.getLastMessage()  .getMsgTime());  }  }   */\*\*  \* 自定义实现Handler，主要用于刷新UI操作  \*/* Handler **mHandler** = **new** Handler() {  @Override **public void** handleMessage(Message msg) {  **switch** (msg.**what**) {  **case** 0:  EMMessage message = (EMMessage) msg.**obj**;  *// 这里只是简单的demo，也只是测试文字消息的收发，所以直接将body转为EMTextMessageBody去获取内容* EMTextMessageBody body = (EMTextMessageBody) message.getBody();  *// 将新的消息内容和时间加入到下边* **mContentText**.setText(**mContentText**.getText()  + **"\n接收："** + body.getMessage()  + **" - time: "** + message.getMsgTime());  **break**;  }  }  };   @Override **protected void** onResume() {  **super**.onResume();  *// 添加消息监听* EMClient.*getInstance*().chatManager().addMessageListener(**mMessageListener**);  }   @Override **protected void** onStop() {  **super**.onStop();  *// 移除消息监听* EMClient.*getInstance*().chatManager().removeMessageListener(**mMessageListener**);  }  */\*\*  \* --------------------------------- Message Listener -------------------------------------  \* 环信消息监听主要方法  \*/  /\*\*  \* 收到新消息  \*  \** ***@param list*** *收到的新消息集合  \*/* @Override **public void** onMessageReceived(List<EMMessage> list) {  *// 循环遍历当前收到的消息* **for** (EMMessage message : list) {  Log.*i*(**"lzan13"**, **"收到新消息:"** + message);  **if** (message.getFrom().equals(**mChatId**)) {  *// 设置消息为已读* **mConversation**.markMessageAsRead(message.getMsgId());   *// 因为消息监听回调这里是非ui线程，所以要用handler去更新ui* Message msg = **mHandler**.obtainMessage();  msg.**what** = 0;  msg.**obj** = message;  **mHandler**.sendMessage(msg);  } **else** {  *//* ***TODO 如果消息不是当前会话的消息发送通知栏通知*** }  }  }   */\*\*  \* 收到新的 CMD 消息  \*/* @Override **public void** onCmdMessageReceived(List<EMMessage> list) {  **for** (**int** i = 0; i < list.size(); i++) {  *// 透传消息* EMMessage cmdMessage = list.get(i);  EMCmdMessageBody body = (EMCmdMessageBody) cmdMessage.getBody();  Log.*i*(**"lzan13"**, **"收到 CMD 透传消息"** + body.action());  }  }   */\*\*  \* 收到新的已读回执  \*  \** ***@param list*** *收到消息已读回执  \*/* @Override **public void** onMessageRead(List<EMMessage> list) {}   */\*\*  \* 收到新的发送回执  \** ***TODO 无效 暂时有bug*** *\*  \** ***@param list*** *收到发送回执的消息集合  \*/* @Override **public void** onMessageDelivered(List<EMMessage> list) {}   */\*\*  \* 消息撤回回调  \*  \** ***@param list*** *撤回的消息列表  \*/* @Override **public void** onMessageRecalled(List<EMMessage> list) {}   */\*\*  \* 消息的状态改变  \*  \** ***@param message*** *发生改变的消息  \** ***@param object*** *包含改变的消息  \*/* @Override **public void** onMessageChanged(EMMessage message, Object object) {} } |

1. ECMainActivity.java

|  |
| --- |
| *// 发起聊天 username 输入框* **private** EditText **mChatIdEdit**;  *// 发起聊天* **private** Button **mStartChatBtn**;  *// 退出登录* **private** Button **mSignOutBtn**;   @Override **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);   *// 判断sdk是否登录成功过，并没有退出和被踢，否则跳转到登陆界面* **if** (!EMClient.*getInstance*().isLoggedInBefore()) {  Intent intent = **new** Intent(ECMainActivity.**this**, ECLoginActivity.**class**);  startActivity(intent);  finish();  **return**;  }   setContentView(R.layout.***activity\_ecmain***);   initView();  }   */\*\*  \* 初始化界面  \*/* **private void** initView() {   **mChatIdEdit** = findViewById(R.id.***ec\_edit\_chat\_id***);   **mStartChatBtn** = findViewById(R.id.***ec\_btn\_start\_chatee***);  **mStartChatBtn**.setOnClickListener(**new** View.OnClickListener() {  @Override **public void** onClick(View v) {  *// 获取我们发起聊天的者的username* String chatId = **mChatIdEdit**.getText().toString().trim();  **if** (!TextUtils.*isEmpty*(chatId)) {  *// 获取当前登录用户的 username* String currUsername = EMClient.*getInstance*().getCurrentUser();  **if** (chatId.equals(currUsername)) {  Toast.*makeText*(ECMainActivity.**this**, **"不能和自己聊天"**, Toast.***LENGTH\_SHORT***).show();  **return**;  }  *// 跳转到聊天界面，开始聊天* Intent intent = **new** Intent(ECMainActivity.**this**, ECChatActivity.**class**);  intent.putExtra(**"ec\_chat\_id"**, chatId);  startActivity(intent);  } **else** {  Toast.*makeText*(ECMainActivity.**this**, **"Username 不能为空"**, Toast.***LENGTH\_LONG***).show();  }  }  });   **mSignOutBtn** = findViewById(R.id.***ec\_btn\_sign\_out***);  **mSignOutBtn**.setOnClickListener(**new** View.OnClickListener() {  @Override **public void** onClick(View v) {  signOut();  }  });  }   */\*\*  \* 退出登录  \*/* **private void** signOut() {  *// 调用sdk的退出登录方法，第一个参数表示是否解绑推送的token，没有使用推送或者被踢都要传false* EMClient.*getInstance*().logout(**false**, **new** EMCallBack() {  @Override **public void** onSuccess() {  Log.*i*(**"lzan13"**, **"logout success"**);  *// 调用退出成功，结束app* finish();  }   @Override **public void** onError(**int** i, String s) {  Log.*i*(**"lzan13"**, **"logout error "** + i + **" - "** + s);  }   @Override **public void** onProgress(**int** i, String s) {   }  });  } } |

