特别鸣谢本文作者：水的右边为 android 初学者和新手带来如此详细和精彩的文章
另：不要看本文长，耐心看下来会有很大收获。
还：有些人可能说，版主转帖不能证明实力，我只能说精力和能力有限。不过黄色网站的版主也不会去自己拍 A 篇吧　　----- 呵呵纯粹搞笑 废话不多说 看帖吧！

android 开发我的新浪微博客户端-开篇

开始接触学习 android 已经有 3 个礼拜了，一直都是对着 android 的 sdk 文档写 Tutorials 从 Hello World 到 Notepad Tutorial 算是初步入门了吧，刚好最近对微博感兴趣就打算开发个 android 版本的新浪微博客户端作为练手项目,并且以随笔的方式详细的记录开发的全过程。本人对 java 语言以及 eclipse Ide 都是初次应用基本上属于边学边用，做移动设备上的东西也是第一次，总的来说属于无基础、无经验、无天赋的纯三无人员，还请广大同学们多多给予指点。

开发第一件事情，那就是开发工具以及环境，我的配置是 Eclipse Helios (3.6.1) + Adroid2.2，具体的环境搭建我就不罗嗦了，google 一下一大堆，光博客园里都能搜到很多篇了。

开发第二件事情，既然是开发新浪的微博客户端，那就先去新浪申请微博账号然后登陆后到新浪的开放平台，新浪的开放平台提供的新浪微博对外的 api 接口，在我的应用中创建一个新的应用获取 App Key 和 App Secret，这 2 个值后面会有用到先记录下来。在新浪的开放平台中提供了开发文档、SDK、接口测试工具等，本人决定直接通过新浪的 Rest Api 进行开发并不打算使用新浪提供的 SDK,据说新浪提供的 java 版的 SDK 并不能直接用来进行 android 的开发需要进行一定的修改才能使用，只是听说我没有试过不一定准确。

最后在说一下，我准备分为 UI 和功能两部分分别进行说明讲解，据我自己的情况大部分的时间都花在的 UI 的设计和实现上了，编码倒反而工作量小多了，所以特别把 UI 部分分出来讲。

最后还要在说一下，很抱歉上面内容基本上属于废话没有什么实质内容了但是既然是第一篇还是得象征性的交代一下，从下篇开始讲具体的内容。

android 开发我的新浪微博客户端-载入页面 UI 篇(1.1)

本软件设定用户第一个接触到的功能就是页面载入等待功能，这个功能对使用者来说就是一个持续 1、2 秒钟的等待页面，在用户等待的同时程序做一些必要的检查以及数据准备工作，载入页面分为 UI 篇和功能篇，从表及里首先是 UI 的实现，一个软件除功能之外还得有一个光鲜的外表也是非常重要的，尽管本人设计水平一般但是还是亲自操刀用 ps 先做了一下设计效果图如下：

一、接下来的任务就是在 android 中实现这样的效果显示，从这个效果的设计分别把图片分成背景、版本号部分、软件名称和图标、作者名称和 blog 四个部分，按照这样的思路把分别生成 4 张 png 的图片，背景部分考虑实现横屏和竖屏切换额外添加一张横屏背景图，然后新建 android 工程，我这里的名称为 MySinaWeibo,android 版本勾选 2.2,并且创建名为 MainActivity 的 Activity 作为整个软件的起始页面,然后把上面的这些图片保存到项目的 res/drawable-mdpi 文件夹下，关于 res 目录下的 drawable-mdpi、drawable-ldpi,、drawable-hdpi 三个文件夹的区别,mdpi 里面主要放中等分辨率的图片,如 HVGA (320x480)。ldpi 里面主要放低分辨率的图片,如 QVGA (240x320)。hdpi 里面主要放高分辨率的图片,如 WVGA (480x800),FWVGA (480x854)。android 系统会根据机器的分辨率来分别到这几个文件夹里面去找对应的图片，在开发程序时为了兼容不同平台不同屏幕，建议各自文件夹根据需求均存放不同版本图片，我这里就不进行这么多的考虑了。

二、完成图片资源的准备后接下就是 layout 文件的编写， 在 res/layout 文件夹下新建 main.xml 文件，这个 layout 采用 LinearLayout 控件作为顶层控件，然后用 ImageView 控件分别实现版本号图片顶部靠左对齐显示、软件名称和图标图片居中对齐、作者名称和 blog 图片底部靠右对齐。注意在版本号图片显示 ImageView 控件下面添加一个 RelativeLayout 控件作为软件名称和图标图片 ImageVIew 和作者名称和 blog 图片 ImageView 的父控件用来控制居中对齐已经底部对齐的实现，具体代码如下：代码

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
3. android:id="@+id/layout"
4. android:orientation="vertical"
5. android:layout_width="fill_parent"
6. android:layout_height="fill_parent">
7. <ImageView
8. android:layout_width="wrap_content"
9. android:layout_height="wrap_content"
10. android:src="@drawable/ver"
11. android:layout_marginTop="15dip"
12. android:layout_marginLeft="15dip">
13. </ImageView>
14. <RelativeLayout
15. android:layout_width="fill_parent"
16. android:layout_height="fill_parent">
17. <ImageView
18. android:layout_width="wrap_content"
19. android:layout_height="wrap_content"
20. android:src="@drawable/logo"
21. android:layout_centerInParent="true">
22. </ImageView>
23.
24. <ImageView
25. android:layout_width="wrap_content"
26. android:layout_height="wrap_content"
27. android:src="@drawable/dev"
28. android:layout_alignParentBottom="true"
29. android:layout_alignParentRight="true"
30. android:layout_marginRight="5dip"
31. android:layout_marginBottom="35dip">
32. </ImageView>
33. </RelativeLayout>
34. </LinearLayout>
```

复制代码

三、在 ec 打开名为 MainActivity 的 Activity 源代码文件进行编辑，onCreate 部分代码如下：

```
1. public void onCreate(Bundle savedInstanceState) {
2. super.onCreate(savedInstanceState);
3. setContentView(R.layout.main);
4. }
```

复制代码

然后运行项目可以在模拟器中显示，上面的几个图片都按照设计的位置和效果进行显示只是整个页面的背景还是黑色的，接下来就是背景部分的显示实现，由于为了实现横竖屏切换显示，背景图的显示采用代码进行控制显示，首先用如下方法获取当前手机是横屏还是竖屏：

```
1. //获取屏幕方向
2. public static int ScreenOrient(Activity activity)
3. {
4. int orient = activity.getRequestedOrientation();
5. if(orient != ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE && orient !=
   ActivityInfo.SCREEN_ORIENTATION_PORTRAIT) {
6. //宽>高为横屏,反正为竖屏
7. WindowManager windowManager = activity.getWindowManager();
8. Display display = windowManager.getDefaultDisplay();
9. int screenWidth = display.getWidth();
10. int screenHeight = display.getHeight();
11. orient = screenWidth < screenHeight ?
   ActivityInfo.SCREEN_ORIENTATION_PORTRAIT :
   ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE;
12. }
13. return orient;
14. }
```

复制代码

然后编写一个名为 AutoBackground 的公共方法用来实现屏幕背景的自动切换，后面的几乎每一个功能页面都需要用到这个方法

```
1. public static void AutoBackground(Activity activity,View view,int
   Background_v, int Background_h)
2. {
3. int orient=ScreenOrient(activity);
4. if (orient == ActivityInfo.SCREEN_ORIENTATION_PORTRAIT) { //纵向
5. view.setBackgroundResource(Background_v);
6. }else{ //横向
```

```
7.  view.setBackgroundResource(Background_h);
8.  }
9.  }
```

复制代码

完成上述两方法后在 MainActivity 的 onCreate 方法中调用 AutoBackground 方法进行屏幕自动切换：

```
1.  LinearLayout layout=(LinearLayout)findViewById(R.id.layout);
2.  //背景自动适应
3.  AndroidHelper.AutoBackground(this, layout, R.drawable.bg_v,
    R.drawable.bg_h);
```

复制代码

到此完成了载入页面的 UI 部分的实现，测试运行模拟器中查看效果，基本上跟最上面的设计效果图相符，测试效果图如下：


5554:my_avd6
Ver1.0
我的新浪微博
载入中请稍等 . . .
作者：水的右边
http://www.cnblogs.com/hll2008/
eoeAndroid.com

android 开发我的新浪微博客户端-载入页面 sqlite 篇(1.2)

通过上一篇文章（android 开发我的新浪微博客户端-载入页面 UI 篇(1.1)）已经完成了载入页面的 UI 部分的实现，效果如上图,接下来在上面的基础上完成载入页面的功能代码。

首先说明一下新浪微博提供了 OAuth 和 Base OAuth 两种认证方式（如果不知道什么是 OAuth 和 Base OAuth 请自己 google 一下恶补，同时接下来的 2 篇随笔也会对这方面进行详细的说明以及具体实现），本项目是采用 OAuth 认证方式，采用这种方式就需要有用户的新浪 UserID、Access Token、Access Secret 这 3 样东西才能自由便利的调用新浪的开放接口，本项目是这样做的当用户第一次使用软件时进行授权认证获取这 3 样东西的时候存储到 sqlite 库中以便用户下次使用时不需要重新进行繁琐的授权认证操作直接从 sqlite 库中读取出来即可，由于这样的需求载入页面的功能设定是这样：当用户打开软件显示载入页面时开始检查 sqlite 库中是否已经保存有用户的新浪微博的 UserID 号、Access Token、Access Secret 的记录，如果一条记录都没有那就说明用户是第一次使用本软件那么跳到认证授权页面进行授权认证操作（认证授权功能在接下来的两篇中进行实现讲解）获取这 3 个值保存到 sqlite 库中，如果已经包括了记录，那么读取这些记录的 UserID 号、Access Token、Access Secret 值然后根据这 3 个值调用新浪的 api 接口获取这些记录对应的用户昵称和用户头像图标等信息。

上面功能设定中涉及到 sqlite 数据库的创建、数据表的创建、数据记录的添加、数据记录的读取等操作，这里新建名为 SqliteHelper.java 类文件提供 sqlite 数据表的创建、更新等，代码如下：

```
1. public class SqliteHelper extends SQLiteOpenHelper{
2. //用来保存
3. UserID、Access Token、Access Secret
4. 的表名
5. public static final String TB_NAME="users";
6. public SqliteHelper(Context context, String name, CursorFactory
   factory, int version) {
7. super(context, name, factory, version);
8. }
9. //创建表
10. @Override
11. public void onCreate(SQLiteDatabase db) {
12. db.execSQL("CREATE TABLE IF NOT EXISTS "+
13. TB_NAME+"("+
14. UserInfo.ID+" integer primary key,"+
15. UserInfo.USERID+" varchar,"+
16. UserInfo.TOKEN+" varchar,"+
17. UserInfo.TOKENSECRET+" varchar,"+
18. UserInfo.USERNAME+" varchar,"+
19. UserInfo.USERICON+" blob"+
20. ")"
21. );
22. Log.e("Database","onCreate");
23. }
24. //更新表
25. @Override
26. public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
   {
27. db.execSQL("DROP TABLE IF EXISTS " + TB_NAME);
28. onCreate(db);
29. Log.e("Database","onUpgrade");
30. }
31. //更新列
32. public void updateColumn(SQLiteDatabase db, String oldColumn, String
   newColumn, String typeColumn){
33. try{
34. db.execSQL("ALTER TABLE " +
35. TB_NAME + " CHANGE " +
36. oldColumn + " "+ newColumn +
37. " " + typeColumn
38. );
39. }catch(Exception e){
40. e.printStackTrace();
41. }
```

```
42. }
43. }
```

复制代码

接下来新建名为DataHelper.java类文件实现用户记录的创建、更新、删除等，代码如下：

```
1.  public class DataHelper {
2.  //数据库名称
3.  private static String DB_NAME = "mysinaweibo.db";
4.  //数据库版本
5.  private static int DB_VERSION = 2;
6.  private SQLiteDatabase db;
7.  private SqliteHelper dbHelper;
8.
9.  public DataHelper(Context context){
10. dbHelper=new SqliteHelper(context,DB_NAME, null, DB_VERSION);
11. db= dbHelper.getWritableDatabase();
12. }
13.
14. public void Close()
15. {
16. db.close();
17. dbHelper.close();
18. }
19. //获取users表中的UserID、Access Token、Access Secret的记录
20. public List<UserInfo> GetUserList(Boolean isSimple)
21. {
22. List<UserInfo> userList = new ArrayList<UserInfo>();
23. Cursor cursor=db.query(SqliteHelper.TB_NAME, null, null, null, null,
    null, UserInfo.ID+" DESC");
24. cursor.moveToFirst();
25. while(!cursor.isAfterLast()&& (cursor.getString(1)!=null)){
26. UserInfo user=new UserInfo();
27. user.setId(cursor.getString(0));
28. user.setUserId(cursor.getString(1));
29. user.setToken(cursor.getString(2));
30. user.setTokenSecret(cursor.getString(3));
31. if(!isSimple){
32. user.setUserName(cursor.getString(4));
33. ByteArrayInputStream stream = new
    ByteArrayInputStream(cursor.getBlob(5));
34. Drawable icon= Drawable.createFromStream(stream, "image");
35. user.setUserIcon(icon);
```

```
}
userList.add(user);
cursor.moveToNext();
}
cursor.close();
return userList;
}

//判断 users 表中的是否包含某个 UserID 的记录
public Boolean HaveUserInfo(String UserId)
{
Boolean b=false;
Cursor cursor=db.query(SqliteHelper.TB_NAME, null, UserInfo.USERID + "=" + UserId, null, null, null,null);
b=cursor.moveToFirst();
Log.e("HaveUserInfo",b.toString());
cursor.close();
return b;
}

//更新 users 表的记录，根据 UserId 更新用户昵称和用户图标
public int UpdateUserInfo(String userName,Bitmap userIcon,String UserId)
{
ContentValues values = new ContentValues();
values.put(UserInfo.USERNAME, userName);
// BLOB 类型
final ByteArrayOutputStream os = new ByteArrayOutputStream();
// 将 Bitmap 压缩成 PNG 编码，质量为 100%存储
userIcon.compress(Bitmap.CompressFormat.PNG, 100, os);
// 构造 SQLite 的 Content 对象，这里也可以使用 raw
values.put(UserInfo.USERICON, os.toByteArray());
int id= db.update(SqliteHelper.TB_NAME, values, UserInfo.USERID + "=" + UserId, null);
Log.e("UpdateUserInfo2",id+"");
return id;
}

//更新 users 表的记录
public int UpdateUserInfo(UserInfo user)
{
ContentValues values = new ContentValues();
values.put(UserInfo.USERID, user.getUserId());
values.put(UserInfo.TOKEN, user.getToken());
```

```
77. values.put(UserInfo.TOKENSECRET, user.getTokenSecret());
78. int id= db.update(SqliteHelper.TB_NAME, values, UserInfo.USERID + "="
    + user.getUserId(), null);
79. Log.e("UpdateUserInfo",id+"");
80. return id;
81. }
82.
83. //添加 users 表的记录
84. public Long SaveUserInfo(UserInfo user)
85. {
86. ContentValues values = new ContentValues();
87. values.put(UserInfo.USERID, user.getUserId());
88. values.put(UserInfo.TOKEN, user.getToken());
89. values.put(UserInfo.TOKENSECRET, user.getTokenSecret());
90. Long uid = db.insert(SqliteHelper.TB_NAME, UserInfo.ID, values);
91. Log.e("SaveUserInfo",uid+"");
92. return uid;
93. }
94.
95. //删除 users 表的记录
96. public int DelUserInfo(String UserId){
97. int id= db.delete(SqliteHelper.TB_NAME, UserInfo.USERID +"="+UserId,
    null);
98. Log.e("DelUserInfo",id+"");
99. return id;
100.     }
101.     }
```

复制代码

完成上面的代码后，我们需要在载入页面中调用上面的方法实现 sqlite 库中是否已经保存有用户的新浪微博的 UserID 号、Access Token、Access Secret 的记录的功能在 MainActivity 的 onCreate 方法添加代码：

```
1.  public void onCreate(Bundle savedInstanceState) {
2.  super.onCreate(savedInstanceState);
3.  setContentView(R.layout.main);
4.
5.  ......
6.
7.  //获取账号列表
8.  dbHelper=new DataHelper(this);
9.  List<UserInfo> userList= dbHelper.GetUserList(true);
10. if(userList.isEmpty())//如果为空说明第一次使用跳到 AuthorizeActivity
    页面进行 OAuth 认证
```

```
11. {
12. Intent intent = new Intent();
13. intent.setClass(MainActivity.this, AuthorizeActivity.class);
14. startActivity(intent);
15. }
16. else//如果不为空读取这些记录的 UserID 号、Access Token、Access Secret
    值
17. //然后根据这 3 个值调用新浪的 api 接口获取这些记录对应的用户昵称和用户
    头像图标等信息。
18. {
19. for(UserInfo user:userList){
20. ......
21. }
22. }
23. }
```

复制代码

关于载入页面的 sqlite 就说到这里了，下一篇说说 OAuth 认证实现。

android 开发我的新浪微博客户端-OAuth 篇(2.1)

本篇说说关于 OAuth 授权认证的事情，新浪开放 api 都必须在这个基础上才能调用，所以有必要专门来讲讲，前面的文章中已经提到过关于新浪微博提供了 OAuth 和 Base OAuth 两种认证方式，并且本项目采用 OAuth 认证方式，至于为什么采用这个 OAuth 认证而不采用 Base OAuth 认证原因很简单，自从 Twitter 只支持 OAuth 认证方式以来，各大应用都纷纷转向 OAuth 认证方式，而新浪微博的开放平台也将在近日停止 Base OAuth 的认证方式。

OAuth 的基本概念，OAUTH 协议为用户资源的授权提供了一个安全的、开放而又简易的标准。与以往的授权方式不同之处是 OAUTH 的授权不会使第三方触及到用户的帐号信息(如用户名与密码)，即第三方无需使用用户的用户名与密码就可以申请获得该用户资源的授权，因此 OAUTH 是安全的。同样新浪微博提供 OAuth 认证也是为了保证用户账号和密码的安全，在这里通过 OAuth 建立普通新浪微博用户、客户端程序(我们正在开发的这个 android 客户端程序)、新浪微博三者之间的相互信任关系，让客户端程序（我们正在开发的这个 android 客户端程序）不需要知道用户的账号和密码也能浏览、发布微博，这样有效的保护了用户账号的安全性不需要把账号密码透露给客户端程序又达到了通过客户端程序写微博看微博目的。这个是 OAuth 的作用。

结合新浪微博的 OAuth 认证来说说具体的功能实现，首先罗列一下关键字组，下面四组关键字跟我们接下来 OAuth 认证有非常大的关系。

第一组：（App Key 和 App Secret），这组参数就是本系列文本第一篇提到的建一个新的应用获取 App Key 和 App Secret。

第二组：（Request Token 和 Request Secret）

第三组：（oauth_verifier）

第四组：（user_id、Access Token 和 Access Secret）

新浪微博的 OAuth 认证过程，当用户第一次使用本客户端软件时，客户端程序用第一组作为参数向新浪微博发起请求，然后新浪微博经过验证后返回第二组参数给客户端软件同时表示新浪微博信任本客户端软件，当客户端软件获取第二组参数时作为参数引导用户浏览器跳至新浪微博的授权页面，然后用户在新浪的这个授权页面里输入自己的微博账号和密码进行授权，完成授权后根据客户端设定的回调地址把第三组参数返回给客户端软件并表示用户也信任本客户端软件，接下客户端软件把第二组参数和第三组参数作为参数再次向新浪微博发起请求，然后新浪微博返回第四组参数给客户端软件，第四组参数需要好好的保存起来这个就是用来代替用户的新浪账号和密码用的，在后面调用 api 时都需要。从这个过程来看用户只是在新浪微博的认证网页输入过账户和密码并没有在客户端软件里输入过账户和密码，客户端软件只保存了第四组数据并没有保存用户的账户和密码，这样有效的避免了账户和密码透露给新浪微博之外的第三方应用程序，保证 了安全性。

本项目用为了方便开发采用了 oauth-signpost 开源项目进行 OAuth 认证开发，新建 OAuth.java 类文件对 OA 进行简单的封装，OAuth 类主要有 RequestAccessToken、GetAccessToken、SignRequest 三个方法，第一个方法 RequestAccessToken 就是上面过程中用来获取第三组参数用的，GetAccessToken 方法是用来获取第四组参数用，SignRequest 方法是用来调用 api 用。由于采用了 oauth-signpost 开源项目简单了很多。具体代码如下：

```
1.  public class OAuth {
2.  private CommonsHttpOAuthConsumer httpOauthConsumer;
3.  private OAuthProvider httpOauthprovider;
4.  public String consumerKey;
5.  public String consumerSecret;
6.
7.  public OAuth()
8.  {
9.  // 第一组：（App Key 和 App Secret）
10. // 这组参数就是本系列文本第一篇提到的建一个新的应用获取 App Key 和 App
    Secret。
11. this("3315495489","e2731e7grf592c0fd7fea32406f86e1b");
12. }
13. public OAuth(String consumerKey,String consumerSecret)
14. {
15. this.consumerKey=consumerKey;
16. this.consumerSecret=consumerSecret;
17. }
18.
```

```
19. public Boolean RequestAccessToken(Activity activity,String
    callBackUrl){
20. Boolean ret=false;
21. try{
22. httpOauthConsumer = new
    CommonsHttpOAuthConsumer(consumerKey,consumerSecret);
23. httpOauthprovider = new
    DefaultOAuthProvider("http://api.t.sina.com.cn/oauth/request_token",
    "http://api.t.sina.com.cn/oauth/access_token","http://api.t.sina.co
    m.cn/oauth/authorize");
24. String authUrl =
    httpOauthprovider.retrieveRequestToken(httpOauthConsumer,
    callBackUrl);
25. activity.startActivity(new Intent(Intent.ACTION_VIEW,
    Uri.parse(authUrl)));
26. ret=true;
27. }catch(Exception e){
28. }
29. return ret;
30. }
31.
32. public UserInfo GetAccessToken(Intent intent){
33. UserInfo user=null;
34. Uri uri = intent.getData();
35. String verifier =
    uri.getQueryParameter(oauth.signpost.OAuth.OAUTH_VERIFIER);
36. try {
37. httpOauthprovider.setOAuth10a(true);
38. httpOauthprovider.retrieveAccessToken(httpOauthConsumer,verifier);
39. } catch (OAuthMessageSignerException ex) {
40. ex.printStackTrace();
41. } catch (OAuthNotAuthorizedException ex) {
42. ex.printStackTrace();
43. } catch (OAuthExpectationFailedException ex) {
44. ex.printStackTrace();
45. } catch (OAuthCommunicationException ex) {
46. ex.printStackTrace();
47. }
48. SortedSet<String> user_id=
    httpOauthprovider.getResponseParameters().get("user_id");
49. String userId=user_id.first();
50. String userKey = httpOauthConsumer.getToken();
51. String userSecret = httpOauthConsumer.getTokenSecret();
52. user=new UserInfo();
```

```
53. user.setUserId(userId);
54. user.setToken(userKey);
55. user.setTokenSecret(userSecret);
56. return user;
57. }
58.
59. public HttpResponse SignRequest(String token,String
    tokenSecret,String url,List params)
60. {
61. HttpPost post = new HttpPost(url);
62. //HttpClient httpClient = null;
63. try{
64. post.setEntity(new UrlEncodedFormEntity(params,HTTP.UTF_8));
65. } catch (UnsupportedEncodingException e) {
66. e.printStackTrace();
67. }
68. //关闭 Expect:100-Continue 握手
69. //100-Continue 握手需谨慎使用，因为遇到不支持 HTTP/1.1 协议的服务器或
    者代理时会引起问题
70. post.getParams().setBooleanParameter(CoreProtocolPNames.USE_EXPECT_
    CONTINUE, false);
71. return SignRequest(token,tokenSecret,post);
72. }
73.
74. public HttpResponse SignRequest(String token,String
    tokenSecret,HttpPost post){
75. httpOauthConsumer = new
    CommonsHttpOAuthConsumer(consumerKey,consumerSecret);
76. httpOauthConsumer.setTokenWithSecret(token,tokenSecret);
77. HttpResponse response = null;
78. try {
79. httpOauthConsumer.sign(post);
80. } catch (OAuthMessageSignerException e) {
81. e.printStackTrace();
82. } catch (OAuthExpectationFailedException e) {
83. e.printStackTrace();
84. } catch (OAuthCommunicationException e) {
85. e.printStackTrace();
86. }
87. //取得 HTTP response
88. try {
89. response = new DefaultHttpClient().execute(post);
90. } catch (ClientProtocolException e) {
91. e.printStackTrace();
```

```
92. } catch (IOException e) {
93. e.printStackTrace();
94. }
95. return response;
96. }
97. }
```

复制代码

这样就完成了 OAuth 功能类的开发，后面都会用到这个类相关的方法。本篇到这里就算是完结请继续关注后面的文章。

android 开发我的新浪微博客户端-用户授权页面 UI 篇(3.1)

上一篇讲了讲 OAuth 授权认证的事情,大概的介绍了 OAuth 的原理，并且完成了一个 OAuth.java 的类库，提供了几个 OAuth 认证必要的方法，本篇开始具体讲本项目的用户授权功能,用户授权页面是当用户第一次使用本软件的时候自动从载入页面跳转过来的显示的页面，涉及 OAuth 认证相关都是在上一篇的 OAuth.java 的类基础上开发。用户授权页面分为 UI 篇和功能篇两篇，本篇先来讲讲 UI 的实现，这次就不贴 PS 的效果图了直接贴实现后的功能截图如下：

看上面的图,其实这个页面的 UI 实现不复杂,首先是背景部分的实现这个参考 android 开发我的新浪微博客户端-载入页面 UI 篇(1.1)，重点来讲讲这个半透明的弹出对话框窗口是如何实现的，首先新建名为 AuthorizeActivity.java 的 Activity，并且在 AndroidManifest.xml 文件中添加这个 Activity，这样这个 Activity 才能被使用，接下来为这个 Activity 新建名为 authorize.xml 的 Layout,这个 Layout 很简单只负责 logo 小图标显示，背景部分和透明窗口都是有代码来实现，所以非常简单参考 android 开发我的新浪微博客户端-载入页面 UI 篇(1.1)。

完成 Layout 建立后在 AuthorizeActivity 的 onCreate 方法添加如下代码，设置 authorize.xml 为 AuthorizeActivity 的页面 Layout:

```
1.  @Override
2.  public void onCreate(Bundle savedInstanceState) {
3.  super.onCreate(savedInstanceState);
4.  setContentView(R.layout.authorize);
5.  .......
6.  }
```

复制代码

接下来是本文的重点部分，半透明弹窗用 Dialog 控件进行实现，首先为这个半透明弹窗新建一个名为 dialog.xml 的 Layout,这个 Layout 主要是对 4 个元素进行布局，如图所示分别为 i 小图标、信息提示、中间文字、开始按钮，首先用 LinearLayout 对 i 小图标和信息提示进行水平布局，中间文字以一个 TextView 跟在下面，对于开始按钮是用 RelativeLayout 进行底部对齐显示。具体代码如下：

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout
3. xmlns:android="http://schemas.android.com/apk/res/android"
4. android:layout_width="wrap_content"
5. android:layout_height="wrap_content"
6. android:orientation="vertical"
7. android:padding="10dip">
8. <LinearLayout
9. android:layout_width="wrap_content"
10. android:layout_height="wrap_content"
11. android:orientation="horizontal">
12. <ImageView
13. android:layout_width="wrap_content"
14. android:layout_height="wrap_content"
15. android:src="@drawable/info_icon">
16. </ImageView>
17. <TextView
18. android:layout_width="wrap_content"
19. android:layout_height="wrap_content"
20. android:text="信息提示"
21. android:textSize="13px"
22. android:textColor="#219ac6"
23. android:layout_marginLeft="5dip">
24. </TextView>
25. </LinearLayout>
26. <TextView
27. android:id="@+id/text_info"
28. android:layout_marginTop="6px"
29. android:layout_width="200px"
30. android:layout_height="wrap_content"
31. android:textColor="#686767"
32. android:textSize="14px"
33. android:text="第一次使用需要输入您的新浪微博账号和密码进行登录授权
    ">
34. </TextView>
35. <RelativeLayout
36. android:layout_width="fill_parent"
```

```
37. android:layout_height="40px">
38. <LinearLayout
39. android:layout_width="wrap_content"
40. android:layout_height="wrap_content"
41. android:orientation="horizontal"
42. android:layout_centerHorizontal="true"
43. android:layout_alignParentBottom="true">
44. <ImageButton
45. android:id="@+id/btn_start"
46. android:layout_width="80px"
47. android:layout_height="31px"
48. android:src="@drawable/btn_start_selector">
49. </ImageButton>
50. <ImageButton
51. android:id="@+id/btn_cancel"
52. android:layout_width="80px"
53. android:layout_height="31px"
54. android:layout_marginLeft="8px"
55. android:src="@drawable/btn_cancel_selector">
56. </ImageButton>
57. </LinearLayout>
58. </RelativeLayout>
59.
60. </LinearLayout>
```

复制代码

完成了半透明弹窗的 Layout 定义接下来我们要做的就是为它写一个自定义样式来实现我们想要的显示效果，首先我们需准备一个圆角的半透明 png 图片名为 dia_bg.png 并且添加到 drawable 中，接下来再 res/values 文件夹新建名为 dialogStyle.xml 的 resources 样式文件，具体代码如下：

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <resources>
3. <style name="dialog" parent="@android:style/Theme.Dialog">
4. <item name="android:windowFrame">@null</item>
5. <item name="android:windowIsFloating">true</item>
6. <item name="android:windowIsTranslucent">false</item>
7. <item name="android:windowNoTitle">true</item>
8. <item name="android:windowBackground">@drawable/dia_bg</item>
9. <item name="android:backgroundDimEnabled">false</item>
10. </style>
11. </resources>
```

复制代码

这个样式文件的说明如下

parent="@android:style/Theme.Dialog" ：在系统 Dialog 样式基础上，相当于继承系统样式

<item name="android:windowFrame">@null</item> ： Dialog 的 windowFrame 框为无

<item name="android:windowIsFloating">true</item>：是否浮现在 activity 之上

<item name="android:windowIsTranslucent">false</item>:是否半透明

<item name="android:windowNoTitle">true</item>:是否显示 title

<item name="android:windowBackground">@drawable/dia_bg</item>:设置 dialog 的背景

<item name="android:backgroundDimEnabled">false</item>: 背景是否模糊显示

接下来写 java 代码把这个半透明弹窗显示出来，在 AuthorizeActivity 的 onCreate 方法添加如下代码：

```
1. ......
2. View diaView=View.inflate(this, R.layout.dialog, null);
3. dialog=new Dialog(AuthorizeActivity.this,R.style.dialog);
4. dialog.setContentView(diaView);
5. dialog.show();
6. ......
```

复制代码

最后运行查看效果，到这里我们的任务已经完成了。请关注下一篇功能篇。

android 开发我的新浪微博客户端-用户授权页面功能篇(3.2)


5554:my_avd6
12:10 PM
新浪微博
t.sina.com.cn
授权微博开放平台接口访问您的微博帐号，随时享受精彩!
在您授权微博开放平台接口前，请再三确认您信任微博开放平台接口
帐号：
密码：
授权
取消
eoeAndroid.com

在上一篇实现了用户授权页面的 UI，如上图，接下来要做的就是在这个基础上完成功能部分真正实现用户的授权认证，这一篇是 android 开发我的新浪微博客户端-OAuth 篇(2.1)的具体应用篇原理就不多解释了不懂的看 OAuth 篇即可。认证过程从点击开始按钮然后跳转到新浪的授权页面，接着用户在新浪的页面里输入自己的账户和密码确定后返回用户授权页面。首先给开始按钮添加点击事件代码，代码中主要是调用我们前面 android 开发我的新浪微博客户端-OAuth 篇(2.1)完成的 OAuth 类的 RequestAccessToken 方法用来获取 oauth_verifier，具体代码如下：

```
1.  ImageButton stratBtn=(ImageButton)diaView.findViewById(R.id.btn_start);
2.  stratBtn.setOnClickListener(new OnClickListener(){
3.
4.  @Override
5.  public void onClick(View arg0) {
6.  auth=new OAuth();
7.  auth.RequestAccessToken(AuthorizeActivity.this, CallBackUrl);
8.  }
```

```
9.
10. });
```

复制代码

上面的代码中重点来说明一下 RequestAccessToken 方法的第二参数 CallBackUrl，这个参数是用户在新浪的页面中输入账户密码后完成认证后返回的地址，我这里是这样设置的 CallBackUrl = "myapp://AuthorizeActivity" ， 在 AndroidManifest.xml 中配置给 AuthorizeActivity 添加如下配置把 myapp://AuthorizeActivity 指向到 AuthorizeActivity，这样当页面返回到 AuthorizeActivity 中就可以获取到传过来的 oauth_verifier 参数。

```
1. <intent-filter>
2. <action android:name="android.intent.action.VIEW" />
3. <category android:name="android.intent.category.DEFAULT" />
4. <category android:name="android.intent.category.BROWSABLE" />
5. <data android:scheme="myapp" android:host="AuthorizeActivity" />
6. </intent-filter>
```

复制代码

再 AuthorizeActivity 如果来接收返回的 oauth_verifier 参数呢？接下来在 AuthorizeActivity 添加如下方法:

```
1. @Override
2. protected void onNewIntent(Intent intent) {
3. super.onNewIntent(intent);
4. //在这里处理获取返回的 oauth_verifier 参数
5. }
```

复制代码

关于 onNewIntent 的说明是这样的，onCreate 是用来创建一个 Activity 也就是创建一个窗体，但一个 Activty 处于任务栈的顶端，若再次调用 startActivity 去创建它，则不会再次创建。若你想利用已有的 Acivity 去处理别的 Intent 时，你就可以利用 onNewIntent 来处理。在 onNewIntent 里面就会获得新的 Intent，在这里 AuthorizeActivity 是属于已有的 Acivity，所以需要 onNewIntent 来处理接收返回的参数，获取 oauth_verifier 参数后 OAuth 还没有结束从 android 开发我的新浪微博客户端-OAuth 篇(2.1)描述来看还需要进行根据这个参数继续向新浪微博请求获取 User_id、Access Token 和 Access Secret，在这里我把这些操作全部写在了 GetAccessToken 方法中。在 onNewIntent 添加如下代码：

```
1. UserInfo user= auth.GetAccessToken(intent);
2. if(user!=null){
3. DataHelper helper=new DataHelper(this);
4. String uid=user.getUserId();
5. if(helper.HaveUserInfo(uid))
6. {
7. helper.UpdateUserInfo(user);
8. Log.e("UserInfo", "update");
```

```
9.    }else
10.  {
11.  helper.SaveUserInfo(user);
12.  Log.e("UserInfo", "add");
13.  }
14.  }
```

复制代码

通过上面的代码完成了 User_id、Access Token 和 Access Secret 获取并且保存到了 sqlite 库中，这样就完成了用户的 OAuth 认证，当需要调用新浪的 api 时只需要去 sqlite 库中找该用户的 User_id、Access Token 和 Access Secret 即可。到这里本篇就结束了，请关注下一篇。

android 开发我的新浪微博客户端-登录页面 UI 篇(4.1)

首先回顾一下功能流程当用户开启软件显示载入页面时程序首先去 sqlite 库查询是否已经保存有用户的新浪微博的 UserID 号、Access Token、Access Secret 的记录如果没有一条记录那么跳转到用户授权功能页面，这个已经由上面两篇文章实现了，如果有记录那么页面跳转到用户登录页面，也就是本篇以及下篇要实现的功能，本篇讲 UI 的实现，本项目支持多微博账号了，也就是用户可以设置多个微博账号，登录的时候选择其中的一个登录，具体效果如上图,新建名 LoginActivity.java 的 Activity 并且在 AndroidManifest.xml 中进行相应配置，这个页面就是我们要实现的用户登录页面。

看上面的效果，首先页面分 3 部分实现，背景部分、底部菜单部分、用户选择以及头像显示部分，首先在 res/layout 的目录下新建名为 login.xml 的 layout，然后根据页面显示要求编写如下的布局控制：

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout
3. xmlns:android="http://schemas.android.com/apk/res/android"
```

```
4.  android:id="@+id/layout"
5.  android:orientation="vertical"
6.  android:layout_width="fill_parent"
7.  android:layout_height="fill_parent">
8.  <ImageView
9.  android:layout_width="wrap_content"
10. android:layout_height="wrap_content"
11. android:src="@drawable/logo_s"
12. android:layout_marginTop="5dip"
13. android:layout_marginLeft="5dip">
14. </ImageView>
15. <RelativeLayout
16. android:layout_width="fill_parent"
17. android:layout_height="fill_parent">
18. <RelativeLayout
19. android:id="@+id/iconBtn"
20. android:layout_width="90px"
21. android:layout_height="80px"
22. android:background="@drawable/icon_selector"
23. android:layout_above="@+id/selectLayout"
24. android:layout_centerHorizontal="true"
25. android:layout_marginBottom="20dip">
26. <ImageView
27. android:id="@+id/icon"
28. android:layout_width="wrap_content"
29. android:layout_height="wrap_content"
30. android:layout_centerInParent="true">
31. </ImageView>
32. </RelativeLayout>
33.
34. <RelativeLayout
35. android:id="@+id/selectLayout"
36. android:layout_width="wrap_content"
37. android:layout_height="wrap_content"
38. android:layout_centerInParent="true">
39. <EditText
40. android:id="@+id/iconSelect"
41. android:layout_width="200px"
42. android:layout_height="wrap_content"
43. android:maxLength="10"
44. android:paddingLeft="20px"
45. android:editable="false"
46. android:enabled="false"
47. android:textSize="13px"
```

```
48. android:background="@drawable/input_over" >
49. </EditText>
50. <ImageButton
51. android:id="@+id/iconSelectBtn"
52. android:layout_width="wrap_content"
53. android:layout_height="wrap_content"
54. android:layout_marginRight="1.0dip"
55. android:layout_alignTop="@+id/iconSelect"
56. android:layout_alignRight="@+id/iconSelect"
57. android:layout_alignBottom="@+id/iconSelect"
58. android:background="@drawable/more_selector" >
59. </ImageButton>
60. <ImageButton
61. android:id="@+id/login"
62. android:layout_width="40px"
63. android:layout_height="40px"
64. android:layout_marginLeft="5dip"
65. android:layout_alignTop="@+id/iconSelectBtn"
66. android:layout_toRightOf="@+id/iconSelectBtn"
67. android:layout_alignBottom="@+id/iconSelectBtn"
68. android:background="@drawable/btn_in_selector" >
69. </ImageButton>
70. </RelativeLayout>
71.
72. <RelativeLayout
73. android:layout_width="fill_parent"
74. android:layout_height="44dip"
75. android:layout_alignParentBottom="true"
76. android:background="#BB768e95">
77. <LinearLayout
78. android:id="@+id/addLayout"
79. android:layout_width="wrap_content"
80. android:layout_height="wrap_content"
81. android:orientation="vertical"
82. android:layout_alignParentLeft="true"
83. android:gravity="center"
84. android:layout_marginTop="3px">
85. <ImageButton
86. android:id="@+id/addIcon"
87. android:layout_width="wrap_content"
88. android:layout_height="wrap_content"
89. android:background="@drawable/add_selector">
90. </ImageButton>
91. <TextView
```

```
92. android:layout_width="wrap_content"
93. android:layout_height="wrap_content"
94. android:textColor="#ffffff"
95. android:textSize="12px"
96. android:text="添加账号">
97. </TextView>
98. </LinearLayout>
99. <LinearLayout
100.     android:id="@+id/exitLayout"
101.     android:layout_width="wrap_content"
102.     android:layout_height="wrap_content"
103.     android:orientation="vertical"
104.     android:layout_centerInParent="true"
105.     android:gravity="center"
106.     android:layout_marginTop="3px">
107.     <ImageButton
108.     android:id="@+id/exitIcon"
109.     android:layout_width="wrap_content"
110.     android:layout_height="wrap_content"
111.     android:background="@drawable/exit_selector">
112.     </ImageButton>
113.     <TextView
114.     android:layout_width="wrap_content"
115.     android:layout_height="wrap_content"
116.     android:textColor="#ffffff"
117.     android:textSize="12px"
118.     android:text="退出软件">
119.     </TextView>
120.     </LinearLayout>
121.     <LinearLayout
122.     android:id="@+id/delLayout"
123.     android:layout_width="wrap_content"
124.     android:layout_height="wrap_content"
125.     android:orientation="vertical"
126.     android:layout_alignParentRight="true"
127.     android:gravity="center"
128.     android:layout_marginTop="3px">
129.     <ImageButton
130.     android:id="@+id/delIcon"
131.     android:layout_width="wrap_content"
132.     android:layout_height="wrap_content"
133.     android:background="@drawable/del_selector">
134.     </ImageButton>
135.     <TextView
```

```
136.     android:layout_width="wrap_content"
137.     android:layout_height="wrap_content"
138.     android:textColor="#ffffff"
139.     android:textSize="12px"
140.     android:text="删除账号">
141.     </TextView>
142.     </LinearLayout>
143.     </RelativeLayout>
144.     </RelativeLayout>
145.     </LinearLayout>
```

复制代码

正对上面的 login.xml 的 layout 进行一下说明，背景部分前面已经讲过了这里也就不重复。

底部菜单实现，原本我是采用 GridView 实现的非常的方便但是后来由于显示位置不好控制改成了用 RelativeLayout 和 LinearLayout 嵌套的方式，实现的比较土但是达到了显示需求，首先是一个最外面的 RelativeLayout 目的是用来实现底部对齐显示，并且把这个 RelativeLayout 的背景设置为浅蓝色半透明的效果，关键这 2 行：android:layout_alignParentBottom="true"和 android:background="#BB768e95"。然后是在 RelativeLayout 内部添加 3 个 LinearLayout 分别是用来显示添加账号、退出软件、删除账号 3 个功能按钮菜单，并且分别设置为左对齐、居中对齐、右对齐，3 个 LinearLayout 都设置为垂直布局 androidrientation="vertical"，然后每 LinearLayout 添加相应的图片和文字。

用户选择以及头像显示部分，这块分成 3 小块，用来显示用户头像的 ImageView、用来显示用户名字并且点击可以出现选择列表的 EditText、用来点击进入当前选择用户首页的功能按钮 ImageButton，这 3 小块的布局实现也是采用 elativeLayout 和 LinearLayout 相互嵌套配合的方式实现的具体参考 login.xml。这里重点说说这个账号选择列表弹出窗口的实现，当点击下拉箭头按钮的时候弹出并显示，这个是用 Dialog 控件实现，首先准备好圆角的半透明背景图 mask_bg.png 然后添加到 res/drawable-mdpi 文件夹下，接着自定义一个 Dialog 样式文件，在 res/values 目录下新建名为 dialogStyles2.xml 的 resources 文件，在用户授权验证页面的时候我们也自定义过类似的 Dialog 的样式，具体解释可以参考前面的户授权验证页面功能，内容如下：

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <resources>
3.  <style name="dialog2" parent="@android:style/Theme.Dialog">
4.  <item name="android:windowFrame">@null</item>
5.  <item name="android:windowIsFloating">true</item>
6.  <item name="android:windowIsTranslucent">false</item>
7.  <item name="android:windowNoTitle">true</item>
8.  <item name="android:windowBackground">@drawable/mask_bg</item>
9.  <item name="android:backgroundDimEnabled">true</item>
10. </style>
11. </resources>
```

复制代码

接下来还需要定义选择列表的 layout，新建名为 dialog2.xml 的 layout 文件，内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:orientation="vertical"
android:padding="4dip">
<ListView
android:id="@+id/list"
android:layout_width="240px"
android:layout_height="220px"
android:divider="#f1f2f2"
android:dividerHeight="1px"
android:layout_margin="5px"
android:background="#ffffff"
android:cacheColorHint="#00000000">
</ListView>
</LinearLayout>
```

复制代码

完成了 layout 和样式文件的编写，接下来就是把 dialogStyles2.xml 样式文件和 dialog2.xml 的列表 layout 用起来，当点击 id 为 iconSelectBtn 的 ImageButton 时显示用户选择窗口，在 LoginActivity 的 onCreate 方法中添加如下代码：

```
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.login);

LinearLayout layout=(LinearLayout)findViewById(R.id.layout);
//背景自动适应
AndroidHelper.AutoBackground(this, layout, R.drawable.bg_v,
R.drawable.bg_h);

ImageButton
iconSelectBtn=(ImageButton)findViewById(R.id.iconSelectBtn);
iconSelectBtn.setOnClickListener(new OnClickListener(){
@Override
public void onClick(View v) {
View diaView=View.inflate(LoginActivity.this, R.layout.dialog2,
null);
dialog=new Dialog(LoginActivity.this,R.style.dialog2);
dialog.setContentView(diaView);
```

```
16. dialog.show();
17.
18. ......
19. }
20.
21. });
```

复制代码

到这里登录的 UI 部分就实现的差不多了，剩下的都是一些功能部分代码用来实现从 sqlite 中账号列表的获取，以及点击选择等交互操作等，这些在下一篇中来继续的讲。

android 开发我的新浪微博客户端-登录页面功能篇(4.2)

上一篇中完成了如上图的 UI 部分的实现，现在继续来讲功能的实现，用户登录操作主要就是账号列表显示和选择账号登录两个功能其他的都是些简单的辅助功能，首先是点击 id 为 iconSelectBtn 的 ImageButton 时显示用户选择窗口，这个时候去数据库中获取账号记录然后在选择窗口中以列表方式显示出来,通过上一篇已经知道 Id 为 list 的 ListView 控件来显示账号列表，首先是从数据库中获取所有的账户记录然后设置默认选中的用户账号代码如下:

```
1.  private void initUser(){
2.  //获取账号列表
3.  dbHelper=new DataHelper(this);
4.  userList = dbHelper.GetUserList(false);
5.  if(userList.isEmpty())
6.  {
7.  Intent intent = new Intent();
8.  intent.setClass(LoginActivity.this, AuthorizeActivity.class);
9.  startActivity(intent);
10. }
11. else
12. {
13. SharedPreferences preferences = getSharedPreferences(Select_Name,
    Activity.MODE_PRIVATE);
14. String str= preferences.getString("name", "");
15. UserInfo user=null;
16. if(str!="")
17. {
18. user=GetUserByName(str);
19. }
20. if(user==null)
21. {
22. user=userList.get(0);
23. }
24. icon.setImageDrawable(user.getUserIcon());
```

```
25. iconSelect.setText(user.getUserName());
26. }
27. }
```

复制代码

这个 initUser() 初始账号的方法在 LoginActivity 的 onCreate 中调用，主要完成两件事情，第一件获取通过 userList = dbHelper.GetUserList(false); 获取所有的账户记录，关于 DataHelper 前面已经有说过了，如果获取的用户记录为空那么就跳转到用户授权功能页面让用户添加账号，如果不为空那么通过 SharedPreferences 去读取用户上一次选择的账号名称，如果没有或者数据库里账号记录不包括这个账户名称那么默认显示记录的第一个账号和头像，如果有那么显示这个账户的名称和头像。关于 SharedPreferences，是 android 提供给开发者用来存储一些简单的数据用的，非常方便类似于网站的 Cookie，在这里我就是用这个来保存上一次用户选择的是哪个账号，非常实用。

接下类首先为 Id 为 list 的 ListView 控件准备数据 Adapter，这个 Adapter 非常简单就是普通的 adapter 继承 BaseAdapter 即可，代码如下：

```
1.  public class UserAdapater extends BaseAdapter{
2.
3.  @Override
4.  public int getCount() {
5.  return userList.size();
6.  }
7.
8.  @Override
9.  public Object getItem(int position) {
10. return userList.get(position);
11. }
12.
13. @Override
14. public long getItemId(int position) {
15. return position;
16. }
17.
18. @Override
19. public View getView(int position, View convertView, ViewGroup parent)
    {
20. convertView =
    LayoutInflater.from(getApplicationContext()).inflate(R.layout.item_
    user, null);
21.
22. ImageView iv = (ImageView) convertView.findViewById(R.id.iconImg);
23. TextView tv = (TextView) convertView.findViewById(R.id.showName);
24. UserInfo user = userList.get(position);
25. try {
```

```
26. //设置图片显示
27. iv.setImageDrawable(user.getUserIcon());
28. //设置信息
29. tv.setText(user.getUserName());
30.
31.
32. } catch (Exception e) {
33. e.printStackTrace();
34. }
35. return convertView;
36. }
```

复制代码

接下就是为这个 ListView 设定数据源 Adapter，在账号选择窗口显示的时候进行设置，添加到 id 为 iconSelectBtn 的 ImageButton 的 OnClickListener 中代码如下：

```
1.  ImageButton
    iconSelectBtn=(ImageButton)findViewById(R.id.iconSelectBtn);
2.  iconSelectBtn.setOnClickListener(new OnClickListener(){
3.  @Override
4.  public void onClick(View v) {
5.  ......
6.  dialog.show();
7.
8.  UserAdapater adapater = new UserAdapater();
9.  ListView listview=(ListView)diaView.findViewById(R.id.list);
10. listview.setVerticalScrollBarEnabled(false);// ListView 去掉下拉条
11. listview.setAdapter(adapater);
12. listview.setOnItemClickListener(new OnItemClickListener(){
13. @Override
14. public void onItemClick(AdapterView<?> arg0, View view,int arg2, long
    arg3) {
15. TextView tv=(TextView)view.findViewById(R.id.showName);
16. iconSelect.setText(tv.getText());
17. ImageView iv=(ImageView)view.findViewById(R.id.iconImg);
18. icon.setImageDrawable(iv.getDrawable());
19. dialog.dismiss();
20. }
21.
22. });
23. }
24.
25. });
```

复制代码

通过上面代码完成了账号选择的功能，接下来给 id 为 login 的 ImageButton 添加 OnClickListener，使得点击后以当前选择账号进入微博首页，代码如下：

```
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.login);
......
ImageButton login=(ImageButton)findViewById(R.id.login);
login.setOnClickListener(new OnClickListener(){
@Override
public void onClick(View v) {
GoHome();
}

});
}

//进入用户首页
private void GoHome(){
if(userList!=null)
{
String name=iconSelect.getText().toString();
UserInfo u=GetUserByName(name);
if(u!=null)
{
ConfigHelper.nowUser=u;//获取当前选择的用户并且保存
}
}
if(ConfigHelper.nowUser!=null)
{
//进入用户首页
Intent intent = new Intent();
intent.setClass(LoginActivity.this, HomeActivity.class);
startActivity(intent);
}
}
```

复制代码

在上面的 GoHome 方法中 ConfigHelper.nowUser 是类型为 UserInfo 的 static 类型用来保存当前登录账号的信息，替代 web 中 session 使用。

最后添加如下方法，用来当这个登录 LoginActivity 结束的时候保存当前选择的账户名称到 SharedPreferences 中，以便帮用户记住登录账号的功能，就是前面的 initUser() 初始账号的方法中会获取保存在 SharedPreferences 中的账户名称，代码如下：

```
@Override
protected void onStop() {
//获得 SharedPreferences 对象
SharedPreferences MyPreferences = getSharedPreferences(Select_Name,
Activity.MODE_PRIVATE);
//获得 SharedPreferences.Editor 对象
SharedPreferences.Editor editor = MyPreferences.edit();
//保存组件中的值
editor.putString("name", iconSelect.getText().toString());
editor.commit();
super.onStop();
}
```

复制代码

至此登录页面功能篇结束，请继续关注下一篇。

android 开发我的新浪微博客户端-用户首页面 UI 篇(5.1)

在前篇完成了用户登录功能后开始用户首页的开发，用户的首页主要的内容是当前登录用户关注的微博列表，本篇先来讲讲 UI 的实现，效果如上图，整个页面分为上、中、下三部分，上面部分是工具条，显示当前登录用户的昵称以及写微博、刷新两个功能按钮；中间部分是当前用户关注的最新微博列表，下面部分是功能切换栏，用来进行各个功能之间的切换。

首先新建名为 HomeActivity.java 的 Activity 作为用户首页，然后在 res/layout 目录下新建名为 home.xml 的 Layout，具体代码如下：

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <LinearLayout
3.  xmlns:android="http://schemas.android.com/apk/res/android"
4.  android:id="@+id/layout"
5.  android:orientation="vertical"
6.  android:layout_width="fill_parent"
7.  android:layout_height="fill_parent">
8.
9.  <RelativeLayout
```

```
10. android:layout_width="fill_parent"
11. android:layout_height="wrap_content"
12. android:layout_margin="3px">
13. <ImageView
14. android:layout_width="wrap_content"
15. android:layout_height="wrap_content"
16. android:src="@drawable/logo_ss">
17. </ImageView>
18. <TextView
19. android:id="@+id/showName"
20. android:layout_width="wrap_content"
21. android:layout_height="wrap_content"
22. android:layout_centerInParent="true"
23. android:textColor="#343434"
24. android:textSize="15px">
25. </TextView>
26. <ImageButton
27. android:id="@+id/writeBtn"
28. android:layout_width="wrap_content"
29. android:layout_height="wrap_content"
30. android:layout_toLeftOf="@+id/refreshBtn"
31. android:background="@drawable/btn_write_selector">
32. </ImageButton>
33. <ImageButton
34. android:id="@+id/refreshBtn"
35. android:layout_width="wrap_content"
36. android:layout_height="wrap_content"
37. android:layout_alignParentRight="true"
38. android:layout_marginLeft="12px"
39. android:background="@drawable/btn_refresh_selector">
40. </ImageButton>
41. </RelativeLayout>
42.
43. <LinearLayout
44. android:layout_width="fill_parent"
45. android:layout_height="wrap_content"
46. android:background="@drawable/hr">
47. </LinearLayout>
48.
49. <RelativeLayout
50. android:layout_width="fill_parent"
51. android:layout_height="fill_parent">
52.
53. <ListView
```

```
android:id="@+id/Msglist"
android:layout_width="fill_parent"
android:layout_height="match_parent"
android:divider="@drawable/divider"
android:dividerHeight="2px"
android:layout_margin="0px"
android:background="#BBFFFFFF"
android:cacheColorHint="#00000000"
android:layout_above="@+id/toolbarLayout"
android:fastScrollEnabled="true"
android:focusable="true">
</ListView>

<LinearLayout
android:id="@+id/loadingLayout"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:orientation="vertical"
android:visibility="invisible"
android:layout_centerInParent="true">
<ProgressBar
android:id="@+id/loading"
android:layout_width="31px"
android:layout_height="31px"
android:layout_gravity="center"
style="@style/progressStyle">
</ProgressBar>
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="正在载入"
android:textSize="12px"
android:textColor="#9c9c9c"
android:layout_gravity="center"
android:layout_below="@+id/loading">
</TextView>
</LinearLayout>


<LinearLayout
android:id="@+id/toolbarLayout"
android:layout_width="fill_parent"
android:layout_height="44dip"
android:layout_alignParentBottom="true">
```

```
98. </LinearLayout>
99. </RelativeLayout>
100. </LinearLayout>
```

复制代码

这个布局首先是一个竖直的根 LinearLayout，在这个根 LinearLayout 里面分别是两个 RelativeLayout， 第一个 RelativeLayout 用来显示页面的工具条，第二个 RelativeLayout 用来显示列表以及底部的功能栏，特别主要在这第二个 RelativeLayout 中有一个 id 为 loadingLayout 的 LinearLayout 是用来显示数据载入中的动画，它的 android:visibility 属性为 invisible(也可以设置成 gone,区别：invisible 这个 View 在 ViewGroupt 中仍保留它的位置，不重新 layout
gone>不可见，但这个 View 在 ViewGroupt 中不保留位置，重新 layout,那后面的 view 就会取代他的位置。)，也就是一开始不显示的意思，接下来看看

```
<ProgressBar

android:id="@+id/loading"
        android:layout_width="31px"
        android:layout_height="31px"

android:layout_gravity="center"

style="@style/progressStyle">
</ProgressBar>
```

这个 ProgressBar 控件就是用来显示动画用的，关键就是 style="@style/progressStyle"，在 res/values 目录下新建名为 loadingstyles.xml，内容如下:

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <resources>
3. <style name="progressStyle" width="38" height="38" parent="@android:style/Widget.ProgressBar.Small">
4. <item name="android:indeterminateDrawable">@anim/loading</item>
5. </style>
6. </resources>
```

复制代码

接着准备好 r1.png - r8.png，

八张不同的小图片分别代表每旋转 45 度图片，八张刚好是 360 度。把这些图片添加到 res/drawable-mdpi 目录中。然后在 res/anim 目录下新建名为 loading.xml 动画文件，内容如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<animation-list android:oneshot="false"
xmlns:android="http://schemas.android.com/apk/res/android">
<item android:duration="200" android:drawable="@drawable/r1" />
<item android:duration="200" android:drawable="@drawable/r2" />
<item android:duration="200" android:drawable="@drawable/r3" />
<item android:duration="200" android:drawable="@drawable/r4" />
<item android:duration="200" android:drawable="@drawable/r5" />
<item android:duration="200" android:drawable="@drawable/r6" />
<item android:duration="200" android:drawable="@drawable/r7" />
<item android:duration="200" android:drawable="@drawable/r8" />
</animation-list>
```

复制代码

关于 Android 播放动画实现我是参考 http://www.eoeandroid.com/forum.php?mod=viewthread&tid=67311&extra=

本篇到这里就结束了，下一篇继续讲用户首页的功能实现，请关注。

android 开发我的新浪微博客户端-用户首页面功能篇(5.2)

上一篇完成用户首页的 UI 实现，本篇接下来讲功能部分的实现，本页面主要的功能就用户关注的最新微博列表，从上一篇中知道本列表是用 ID 为 Msglist 的 ListView 控件来实现，本篇的主要就讲解如果获取微博列表数据给这个 ListView 提供显示数据。ListView 每一条子数据分别由用户头像、用户昵称、发布时间、是否包含照片、微博内容这五部分组成，根据这五部分定义一个名为 WeiBoInfo.java 实体类，代码如下：

```
1.  public class WeiBoInfo {
2.  //文章 id
3.  private String id;
4.  public String getId(){
5.  return id;
6.  }
7.  public void setId(String id){
8.  this.id=id;
9.  }
10. //发布人 id
```

```
11. private String userId;
12. public String getUserId(){
13. return userId;
14. }
15. public void setUserId(String userId){
16. this.userId=userId;
17. }
18.
19. //发布人名字
20. private String userName;
21. public String getUserName(){
22. return userName;
23. }
24. public void setUserName(String userName){
25. this.userName=userName;
26. }
27.
28. //发布人头像
29. private String userIcon;
30. public String getUserIcon(){
31. return userIcon;
32. }
33. public void setUserIcon(String userIcon){
34. this.userIcon=userIcon;
35. }
36.
37. //发布时间
38. private String time;
39. public String getTime(){
40. return time;
41. }
42. public void setTime(String time)
43. {
44. this.time=time;
45. }
46.
47. //是否有图片
48. private Boolean haveImage=false;
49. public Boolean getHaveImage(){
50. return haveImage;
51. }
52. public void setHaveImage(Boolean haveImage){
53. this.haveImage=haveImage;
54. }
```

```
55.
56. //文章内容
57. private String text;
58. public String getText(){
59. return text;
60. }
61. public void setText(String text){
62. this.text=text;
63. }
64.
65. }
```

复制代码

然后在res/layout目录下新建名为weibo.xml的Layout用来控制ListView子项的显示部件，代码很简单不多解释了，直接看下面代码：

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout
3. xmlns:android="http://schemas.android.com/apk/res/android"
4. android:layout_width="wrap_content"
5. android:layout_height="wrap_content"
6. android:orientation="horizontal">
7. <ImageView
8. android:id="@+id/wbicon"
9. android:layout_width="wrap_content"
10. android:layout_height="wrap_content"
11. android:src="@drawable/usericon"
12. android:layout_margin="8px">
13. </ImageView>
14. <LinearLayout
15. android:layout_width="fill_parent"
16. android:layout_height="wrap_content"
17. android:orientation="vertical"
18. android:paddingLeft="0px"
19. android:paddingRight="5px"
20. android:layout_marginTop="5px"
21. android:layout_marginBottom="5px">
22. <RelativeLayout
23. android:layout_width="fill_parent"
24. android:layout_height="wrap_content">
25. <TextView
26. android:id="@+id/wbuser"
27. android:layout_width="wrap_content"
28. android:layout_height="wrap_content"
29. android:textSize="15px"
```

```
30. android:textColor="#424952"
31. android:layout_alignParentLeft="true">
32. </TextView>
33. <ImageView
34. android:id="@+id/wbimage"
35. android:layout_width="wrap_content"
36. android:layout_height="wrap_content"
37. android:layout_marginTop="3px"
38. android:layout_marginRight="5px"
39. android:layout_toLeftOf="@+id/wbtime">
40. </ImageView>
41. <TextView
42. android:id="@+id/wbtime"
43. android:layout_width="wrap_content"
44. android:layout_height="wrap_content"
45. android:layout_alignParentRight="true"
46. android:textColor="#f7a200"
47. android:textSize="12px">
48. </TextView>
49. </RelativeLayout>
50. <TextView
51. android:id="@+id/wbtext"
52. android:layout_width="wrap_content"
53. android:layout_height="wrap_content"
54. android:textColor="#424952"
55. android:textSize="13px"
56. android:layout_marginTop="4px">
57. </TextView>
58. </LinearLayout>
59. </LinearLayout>
```

复制代码

接下来为列表控件定义一个数据 Adapter，代码如下：

```
1.  private List<WeiBoInfo> wbList;
2.
3.  //微博列表 Adapater
4.  public class WeiBoAdapater extends BaseAdapter{
5.
6.  private AsyncImageLoader asyncImageLoader;
7.
8.  @Override
9.  public int getCount() {
10. return wbList.size();
```

```
11. }
12.
13. @Override
14. public Object getItem(int position) {
15. return wbList.get(position);
16. }
17.
18. @Override
19. public long getItemId(int position) {
20. return position;
21. }
22.
23. @Override
24. public View getView(int position, View convertView, ViewGroup parent)
    {
25. asyncImageLoader = new AsyncImageLoader();
26. convertView =
    LayoutInflater.from(getApplicationContext()).inflate(R.layout.weibo,
    null);
27. WeiBoHolder wh = new WeiBoHolder();
28. wh.wbicon = (ImageView) convertView.findViewById(R.id.wbicon);
29. wh.wbtext = (TextView) convertView.findViewById(R.id.wbtext);
30. wh.wbtime = (TextView) convertView.findViewById(R.id.wbtime);
31. wh.wbuser = (TextView) convertView.findViewById(R.id.wbuser);
32. wh.wbimage=(ImageView) convertView.findViewById(R.id.wbimage);
33. WeiBoInfo wb = wbList.get(position);
34. if(wb!=null){
35. convertView.setTag(wb.getId());
36. wh.wbuser.setText(wb.getUserName());
37. wh.wbtime.setText(wb.getTime());
38. wh.wbtext.setText(wb.getText(), TextView.BufferType.SPANNABLE);
39. textHighlight(wh.wbtext,new char[]{'#'},new char[]{'#'});
40. textHighlight(wh.wbtext,new char[]{'@'},new char[]{':',' '});
41. textHighlight2(wh.wbtext,"http://"," ");
42.
43. if(wb.getHaveImage()){
44. wh.wbimage.setImageResource(R.drawable.images);
45. }
46. Drawable cachedImage =
    asyncImageLoader.loadDrawable(wb.getUserIcon(),wh.wbicon, new
    ImageCallback(){
47.
48. @Override
```

```
49. public void imageLoaded(Drawable imageDrawable,ImageView imageView,
    String imageUrl) {
50. imageView.setImageDrawable(imageDrawable);
51. }
52.
53. });
54. if (cachedImage == null) {
55. wh.wbicon.setImageResource(R.drawable.usericon);
56. }else{
57. wh.wbicon.setImageDrawable(cachedImage);
58. }
59. }
60.
61. return convertView;
62. }
```

复制代码

上面的这个 Adapter 实现没有什么特别的很普通，不过这个中使用了 AsyncImageLoader 的方法，这个是用来实现用户头像图标的异步载入显示，这样能提高列表显示的速度，提高用户体验，AsyncImageLoader 的代码如下：

```
1.  public class AsyncImageLoader {
2.  //SoftReference 是软引用，是为了更好的为了系统回收变量
3.  private HashMap<String, SoftReference<Drawable>> imageCache;
4.  public AsyncImageLoader() {
5.  imageCache = new HashMap<String, SoftReference<Drawable>>();
6.  }
7.
8.  public Drawable loadDrawable(final String imageUrl,final ImageView
    imageView, final ImageCallback imageCallback){
9.  if (imageCache.containsKey(imageUrl)) {
10. //从缓存中获取
11. SoftReference<Drawable> softReference = imageCache.get(imageUrl);
12. Drawable drawable = softReference.get();
13. if (drawable != null) {
14. return drawable;
15. }
16. }
17. final Handler handler = new Handler() {
18. public void handleMessage(Message message) {
19. imageCallback.imageLoaded((Drawable) message.obj,
    imageView,imageUrl);
20. }
21. };
```

```
22. //建立新一个新的线程下载图片
23. new Thread() {
24. @Override
25. public void run() {
26. Drawable drawable = loadImageFromUrl(imageUrl);
27. imageCache.put(imageUrl, new SoftReference<Drawable>(drawable));
28. Message message = handler.obtainMessage(0, drawable);
29. handler.sendMessage(message);
30. }
31. }.start();
32. return null;
33. }
34.
35. public static Drawable loadImageFromUrl(String url){
36. URL m;
37. InputStream i = null;
38. try {
39. m = new URL(url);
40. i = (InputStream) m.getContent();
41. } catch (MalformedURLException e1) {
42. e1.printStackTrace();
43. } catch (IOException e) {
44. e.printStackTrace();
45. }
46. Drawable d = Drawable.createFromStream(i, "src");
47. return d;
48. }
49.
50. //回调接口
51. public interface ImageCallback {
52. public void imageLoaded(Drawable imageDrawable,ImageView imageView,
    String imageUrl);
53. }
54. }
```

复制代码

完成上述的工作后，接下来就是显示微薄列表， 在 HomeActivity 的 onCreate 方法中调用 loadList();代码如下：

```
1.  @Override
2.        public void onCreate(Bundle savedInstanceState) {
3.                super.onCreate(savedInstanceState);
4.                setContentView(R.layout.home);
5.
```

```
                。。。。。。
                loadList();
        }

private void loadList(){
                if(ConfigHelper.nowUser==null)
                {

                }
                else
                {
                        user=ConfigHelper.nowUser;
                        //显示当前用户名称
                        TextView
showName=(TextView)findViewById(R.id.showName);
                        showName.setText(user.getUserName());

                        OAuth auth=new OAuth();
                        String url =
"http://api.t.sina.com.cn/statuses/friends_timeline.json";
                        List params=new ArrayList();
                        params.add(new BasicNameValuePair("source",
auth.consumerKey));
                        HttpResponse response
=auth.SignRequest(user.getToken(), user.getTokenSecret(), url,
params);
                        if (200 ==
response.getStatusLine().getStatusCode()){
                                try {
                                        InputStream is =
response.getEntity().getContent();
                                        Reader reader = new
BufferedReader(new InputStreamReader(is), 4000);
                                        StringBuilder buffer = new
StringBuilder((int) response.getEntity().getContentLength());
                                        try {
                                                char[] tmp = new char[1024];
                                                int l;
                                                while ((l =
reader.read(tmp)) != -1) {
                                                        buffer.append(tmp, 0,
l);
                                                }
                                        } finally {
```

```
                                        reader.close();
                                }
                                String string = buffer.toString();
                                //Log.e("json", "rs:" + string);
                                response.getEntity().consumeConte
nt();
                                JSONArray data=new
JSONArray(string);
                                for(int i=0;i<data.length();i++)
                                {
                                        JSONObject
d=data.getJSONObject(i);
                                        //Log.e("json", "rs:" +
d.getString("created_at"));
                                        if(d!=null){
                                                JSONObject
u=d.getJSONObject("user");
                                                if(d.has("retweeted_s
tatus")){
                                                        JSONObject
r=d.getJSONObject("retweeted_status");
                                                }

                                                //微博 id
                                                String
id=d.getString("id");
                                                String
userId=u.getString("id");
                                                String
userName=u.getString("screen_name");
                                                String
userIcon=u.getString("profile_image_url");
                                                Log.e("userIcon",
userIcon);
                                                String
time=d.getString("created_at");
                                                String
text=d.getString("text");
                                                Boolean haveImg=false;
                                                if(d.has("thumbnail_p
ic")){
                                                        haveImg=true;
                                                        //String
thumbnail_pic=d.getString("thumbnail_pic");
```

```
                                                    //Log.e("thum
bnail_pic", thumbnail_pic);
                                                }

                                                Date date=new
Date(time);
                                                time=ConvertTime(date)
;
                                                if(wbList==null){
                                                        wbList=new
ArrayList<WeiBoInfo>();
                                                }
                                                WeiBoInfo w=new
WeiBoInfo();
                                                w.setId(id);
                                                w.setUserId(userId);
                                                w.setUserName(userNam
e);
                                                w.setTime(time);
                                                w.setText(text);

                                                w.setHaveImage(haveIm
g);
                                                w.setUserIcon(userIco
n);
                                                wbList.add(w);
                                        }
                                }

                        }catch (IllegalStateException e) {
                                e.printStackTrace();
                        } catch (IOException e) {
                                e.printStackTrace();
                        } catch (JSONException e) {
                                e.printStackTrace();
                        }
                }

                if(wbList!=null)
                {
                        WeiBoAdapater adapater = new
WeiBoAdapater();
                            ListView
Msglist=(ListView)findViewById(R.id.Msglist);
```

```
                                Msglist.setOnItemClickListener(new
    OnItemClickListener(){
                                    @Override
                                    public void
    onItemClick(AdapterView<?> arg0, View view,int arg2, long arg3) {
                                        Object
    obj=view.getTag();
                                        if(obj!=null){
                                            String
    id=obj.toString();
                                            Intent intent =
    new Intent(HomeActivity.this,ViewActivity.class);
                                            Bundle b=new
    Bundle();
                                            b.putString("key",
    id);
                                            intent.putExtras(
    b);
                                            startActivity(int
    ent);
                                        }
                                    }

                                });
                                Msglist.setAdapter(adapater);
                            }
                        }
                        loadingLayout.setVisibility(View.GONE);
                    }
```

上面的 loadList() 方法通过新浪 Api 接口 http://api.t.sina.com.cn/statuses/friends_timeline.json 获取当前登录用户及其所关注用户的最新微博消息，然后显示到列表中。

这样就完成了用户首页功能的开发。

android 开发我的新浪微博客户端-阅读微博 UI 篇(6.1)

上一篇完成了微博列表的功能，本篇接着做预读微博的功能，本篇主要讲讲 UI 部分的实现，最终实现的效果如上图所示。整个显示页面从上往下分为四部分，第一部分顶部工具条、第二部分作者头像和名称、第三部分微博正文、第四部分功能按钮区。新建名为 ViewActivity.java 作为阅读微博的页面，再 res/layout 目录下新建名为 view.xml 的 Layout，代码如下：

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

```
12.  <ImageView
13.  android:layout_width="wrap_content"
14.  android:layout_height="wrap_content"
15.  android:src="@drawable/logo_ss">
16.  </ImageView>
17.  <TextView
18.  android:id="@+id/showName"
19.  android:layout_width="wrap_content"
20.  android:layout_height="wrap_content"
21.  android:layout_centerInParent="true"
22.  android:textColor="#343434"
23.  android:text="阅读微博"
24.  android:textSize="16px">
25.  </TextView>
26.  <ImageButton
27.  android:id="@+id/returnBtn"
28.  android:layout_width="wrap_content"
29.  android:layout_height="wrap_content"
30.  android:layout_toLeftOf="@+id/homeBtn"
31.  android:background="@drawable/bnt_return_selector">
32.  </ImageButton>
33.  <ImageButton
34.  android:id="@+id/homeBtn"
35.  android:layout_width="wrap_content"
36.  android:layout_height="wrap_content"
37.  android:layout_alignParentRight="true"
38.  android:layout_marginLeft="12px"
39.  android:background="@drawable/btn_home_selector">
40.  </ImageButton>
41.  </RelativeLayout>
42.  <LinearLayout
43.  android:layout_width="fill_parent"
44.  android:layout_height="wrap_content"
45.  android:background="@drawable/hr">
46.  </LinearLayout>
47.
48.  <RelativeLayout
49.  android:id="@+id/user_bg"
50.  android:layout_width="fill_parent"
51.  android:layout_height="78px"
52.  android:paddingTop="8px"
53.  android:paddingLeft="15px"
54.  android:background="@drawable/u_bg_v">
55.      <ImageView
```

```
56.         android:id="@+id/user_icon"
57.         android:layout_width="wrap_content"
58.         android:layout_height="wrap_content"
59.         android:layout_alignParentLeft="true"
60.         android:src="@drawable/usericon">
61.         </ImageView>
62.         <TextView
63.         android:id="@+id/user_name"
64.         android:layout_width="wrap_content"
65.         android:layout_height="wrap_content"
66.         android:layout_toRightOf="@+id/user_icon"
67.         android:layout_marginLeft="10px"
68.         android:layout_marginTop="18px"
69.         android:textColor="#000000">
70.         </TextView>
71.         <ImageView
72.         android:layout_width="wrap_content"
73.         android:layout_height="wrap_content"
74.         android:layout_alignParentRight="true"
75.         android:layout_marginRight="5px"
76.         android:layout_marginTop="10px"
77.         android:src="@drawable/sjjt">
78.         </ImageView>
79.   </RelativeLayout>
80.   <RelativeLayout
81.      android:layout_width="fill_parent"
82.      android:layout_height="fill_parent">
83.         <ScrollView
84.         android:layout_width="fill_parent"
85.         android:layout_height="fill_parent"
86.         android:paddingLeft="17px"
87.         android:paddingRight="17px"
88.         android:paddingBottom="5px"
89.         android:layout_above="@+id/menu_layout">
90.         <LinearLayout
91.         android:layout_width="fill_parent"
92.         android:layout_height="fill_parent"
93.         android:orientation="vertical">
94.              <TextView
95.              android:id="@+id/text"
96.              android:layout_width="wrap_content"
97.              android:layout_height="wrap_content"
98.              android:textColor="#000000"
99.              android:textSize="15px">
```

```
100.            </TextView>
101.            <ImageView
102.            android:id="@+id/pic"
103.            android:layout_width="wrap_content"
104.            android:layout_height="wrap_content">
105.            </ImageView>
106.        </LinearLayout>
107.        </ScrollView>
108.
109.        <LinearLayout
110.        android:id="@+id/loadingLayout"
111.        android:layout_width="wrap_content"
112.        android:layout_height="wrap_content"
113.        android:orientation="vertical"
114.        android:visibility="gone"
115.        android:layout_centerInParent="true">
116.            <ProgressBar
117.            android:id="@+id/loading"
118.            android:layout_width="31px"
119.            android:layout_height="31px"
120.             android:layout_gravity="center"
121.             style="@style/progressStyle">
122.             </ProgressBar>
123.             <TextView
124.             android:layout_width="wrap_content"
125.             android:layout_height="wrap_content"
126.             android:text="正在载入"
127.             android:textSize="12px"
128.             android:textColor="#9c9c9c"
129.             android:layout_gravity="center"
130.             android:layout_below="@+id/loading">
131.             </TextView>
132.        </LinearLayout>
133.
134.        <TableLayout
135.        android:id="@+id/menu_layout"
136.        android:layout_width="fill_parent"
137.        android:layout_height="wrap_content"
138.        android:gravity="center"
139.        android:layout_alignParentBottom="true"
140.        android:layout_marginBottom="5px">
141.            <TableRow
142.            android:layout_width="wrap_content"
143.            android:layout_height="wrap_content"
```

```
144.            android:gravity="center">
145.            <Button
146.            android:id="@+id/btn_gz"
147.            android:layout_width="wrap_content"
148.            android:layout_height="wrap_content"
149.            android:textColor="#3882b8"
150.            android:textSize="15px"
151.            android:text="        关注(1231)"
152.            android:background="@drawable/lt_selector">
153.            </Button>
154.            <Button
155.            android:id="@+id/btn_pl"
156.            android:layout_width="wrap_content"
157.            android:layout_height="wrap_content"
158.            android:textColor="#3882b8"
159.            android:textSize="15px"
160.            android:text="        评论(31)"
161.            android:background="@drawable/rt_selector">
162.            </Button>
163.            </TableRow>
164.            <TableRow
165.            android:layout_width="wrap_content"
166.            android:layout_height="wrap_content"
167.            android:gravity="center">
168.            <Button
169.            android:layout_width="wrap_content"
170.            android:layout_height="wrap_content"
171.            android:textColor="#3882b8"
172.            android:textSize="15px"
173.            android:layout_gravity="left"
174.            android:text="刷新"
175.            android:background="@drawable/lb_selector">
176.            </Button>
177.            <Button
178.            android:layout_width="wrap_content"
179.            android:layout_height="wrap_content"
180.            android:textColor="#3882b8"
181.            android:textSize="15px"
182.            android:text="收藏"
183.            android:background="@drawable/rb_selector">
184.            </Button>
185.            </TableRow>
186.
187.        </TableLayout>
```

```
188.
189.   </RelativeLayout>
190. </LinearLayout>
```

复制代码

上面这个布局实现起来并不复杂， 主要看看功能按钮区的 4 个按钮的点击上去的切换背景的效果，以关注按钮为例子看这行设置，android:background="@drawable/lt_selector"，在 res/drawable-mdpi 目录下新建名为 lt_selector.xml 用来实现点击上去切换图片的效果，具体代码如下：

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <selector xmlns:android="http://schemas.android.com/apk/res/android">
3.  <item android:state_focused="false" android:state_selected="false"
    android:state_pressed="false" android:drawable="@drawable/tbtn_1" />
4.  <item android:state_pressed="true" android:drawable="@drawable/tbtn_h_1" />
5.  </selector>
```

复制代码

本篇虽然看 layout 文件非常的长，其实仔细看看非常的简单了没有什么难和复杂的了，就是按照前面的经验控制好图片以及控件的显示位置和样式即可，本篇中用了一个 ScrollView 控件这个是前面没有用到过的，主要是用来当微博的内容超出显示区域的时候出现滚动条用的这个非常容易使用，所以就简单写一下到此结束了，请继续关注下一篇阅读微博的功能篇。

android 开发我的新浪微博客户端-阅读微博功能篇(6.2)

注：最近由于 OAuth 上传图片碰到了难题，一直在做这方面的研究导致博客很久没有更新。

在上面一篇中已经实现了预读微博的 UI 界面，效果如上图，接下来完成功能部分的代码，当用户在上一个列表界面的列表中点击某一条微博的时候显示这个阅读微博的界面，在这个界面中根据传来的微博 ID，然后根据这个 ID 通过 api 获取微博的具体内容进行显示。

在 ViewActivity.class 的 onCreate 方法中添加如下代码：

```
1.  private UserInfo user;
2.  private String key="";
3.  @Override
4.  public void onCreate(Bundle savedInstanceState) {
5.  super.onCreate(savedInstanceState);
6.  setContentView(R.layout.view);
7.
8.  。。。。。
9.
```

```
10. //获取上一个页面传递过来的 key，key 为某一条微博的 id
11. Intent i=this.getIntent();
12. if(!i.equals(null)){
13. Bundle b=i.getExtras();
14. if(b!=null){
15. if(b.containsKey("key")){
16. key = b.getString("key");
17. view(key);
18. }
19. }
20. }
21. }
```

复制代码

接下来就是 view 方法具体获取微博内容的方法，在这个方法中如果获取的本条微博如果包含图片那么就用前面 AsyncImageLoader 的方法异步载入图片并且进行显示，同时在这个方法中还要获取本条微博被转发的次数以及评论的次数，具体代码如下：

```
1. private void view(String id){
2. user=ConfigHelper.nowUser;
3. OAuth auth=new OAuth();
4. String url = "http://api.t.sina.com.cn/statuses/show/:id.json";
5. List params=new ArrayList();
6. params.add(new BasicNameValuePair("source", auth.consumerKey));
7. params.add(new BasicNameValuePair("id", id));
8. HttpResponse response =auth.SignRequest(user.getToken(), user.getTokenSecret(),
   url, params);
9. if (200 == response.getStatusLine().getStatusCode()){
10. try {
11. InputStream is = response.getEntity().getContent();
12. Reader reader = new BufferedReader(new InputStreamReader(is), 4000);
13. StringBuilder buffer = new StringBuilder((int)
   response.getEntity().getContentLength());
14. try {
15. char[] tmp = new char[1024];
16. int l;
17. while ((l = reader.read(tmp)) != -1) {
18. buffer.append(tmp, 0, l);
19. }
20. } finally {
21. reader.close();
22. }
23. String string = buffer.toString();
24. //Log.e("json", "rs:" + string);
```

```
response.getEntity().consumeContent();
JSONObject data=new JSONObject(string);
if(data!=null){
JSONObject u=data.getJSONObject("user");
String userName=u.getString("screen_name");
String userIcon=u.getString("profile_image_url");
Log.e("userIcon", userIcon);
String time=data.getString("created_at");
String text=data.getString("text");

TextView utv=(TextView)findViewById(R.id.user_name);
utv.setText(userName);
TextView ttv=(TextView)findViewById(R.id.text);
ttv.setText(text);

ImageView iv=(ImageView)findViewById(R.id.user_icon);
AsyncImageLoader asyncImageLoader = new AsyncImageLoader();
Drawable cachedImage = asyncImageLoader.loadDrawable(userIcon,iv, new ImageCallback(){
@Override
public void imageLoaded(Drawable imageDrawable,ImageView imageView, String imageUrl) {

imageView.setImageDrawable(imageDrawable);
}
});
if (cachedImage == null)
{
iv.setImageResource(R.drawable.usericon);
}
else
{
iv.setImageDrawable(cachedImage);
}
if(data.has("bmiddle_pic")){
String picurl=data.getString("bmiddle_pic");
String picurl2=data.getString("original_pic");

ImageView pic=(ImageView)findViewById(R.id.pic);
pic.setTag(picurl2);
pic.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View v) {
Object obj=v.getTag();
```

```
67. Intent intent = new Intent(ViewActivity.this,ImageActivity.class);
68. Bundle b=new Bundle();
69. b.putString("url", obj.toString());
70. intent.putExtras(b);
71. startActivity(intent);
72. }
73. });
74. Drawable cachedImage2 = asyncImageLoader.loadDrawable(picurl,pic, new
    ImageCallback(){
75. @Override
76. public void imageLoaded(Drawable imageDrawable,ImageView imageView, String
    imageUrl) {
77. showImg(imageView,imageDrawable);
78. }
79. });
80. if (cachedImage2 == null)
81. {
82. //pic.setImageResource(R.drawable.usericon);
83. }
84. else
85. {
86. showImg(pic,cachedImage2);
87. }
88. }
89. }
90. }catch (IllegalStateException e) {
91. e.printStackTrace();
92. } catch (IOException e) {
93. e.printStackTrace();
94. } catch (JSONException e) {
95. e.printStackTrace();
96. }
97. }
98. url = "http://api.t.sina.com.cn/statuses/counts.json";
99. params=new ArrayList();
100. params.add(new BasicNameValuePair("source", auth.consumerKey));
101. params.add(new BasicNameValuePair("ids", id));
102. response =auth.SignRequest(user.getToken(), user.getTokenSecret(), url, params);
103. if (200 == response.getStatusLine().getStatusCode()){
104. try {
105. InputStream is = response.getEntity().getContent();
106. Reader reader = new BufferedReader(new InputStreamReader(is), 4000);
107. StringBuilder buffer = new StringBuilder((int)
    response.getEntity().getContentLength());
```

```
108. try {
109. char[] tmp = new char[1024];
110.int l;
111.while ((l = reader.read(tmp)) != -1) {
112.buffer.append(tmp, 0, l);
113.}
114.} finally {
115.reader.close();
116.}
117.String string = buffer.toString();
118.response.getEntity().consumeContent();
119.JSONArray data=new JSONArray(string);
120. if(data!=null){
121. if(data.length()>0){
122. JSONObject d=data.getJSONObject(0);
123. String comments=d.getString("comments");
124. String rt=d.getString("rt");
125. Button btn_gz=(Button)findViewById(R.id.btn_gz);
126. btn_gz.setText("  转发("+rt+")");
127. Button btn_pl=(Button)findViewById(R.id.btn_pl);
128. btn_pl.setText("  评论("+comments+")");
129. }
130. }
131. }
132. catch (IllegalStateException e) {
133. e.printStackTrace();
134. } catch (IOException e) {
135. e.printStackTrace();
136. } catch (JSONException e) {
137. e.printStackTrace();
138. }
139. }
140. }
```

复制代码

在上面的方法中对于微博中包含的图片显示尺寸进行了特别的处理，如果直接把获取的图片显示在 ImageView 中，因为当图片宽高超过手机屏幕的时候，系统会自动按照手机的屏幕按比例缩放图片进行显示，但是我发现一个现象图片的高虽然是按照比例缩小了，但是图片占据的高仍旧是原来图片的高度照成真实图片和文字内容之间多了很高的一块空白，这个现象非常的奇怪，所以我写了如下方法进行处理：

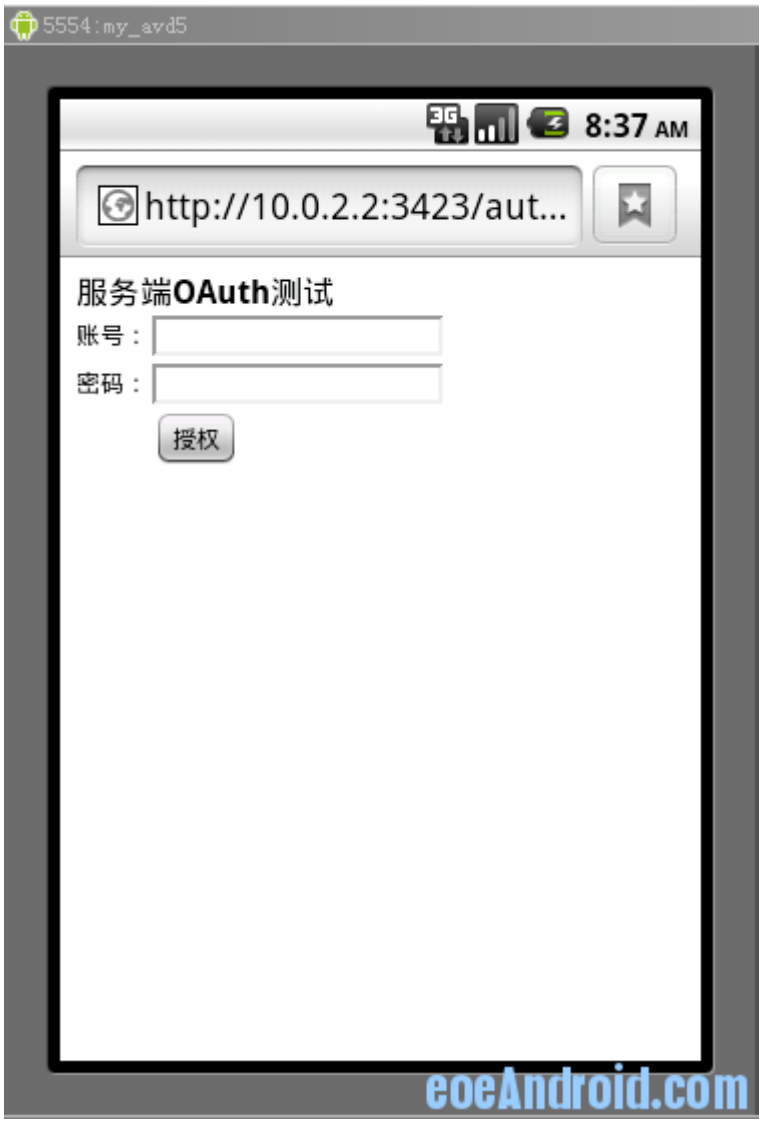
```
1.  private void showImg(ImageView view,Drawable img){
2.  int w=img.getIntrinsicWidth();
3.  int h=img.getIntrinsicHeight();
```

```
4.  Log.e("w", w+"/"+h);
5.  if(w>300)
6.  {
7.  int hh=300*h/w;
8.  Log.e("hh", hh+"");
9.  LayoutParams para=view.getLayoutParams();
10. para.width=300;
11. para.height=hh;
12. view.setLayoutParams(para);
13. }
14. view.setImageDrawable(img);
15. }
```

复制代码

本篇到这里就结束了，请继续关注下一篇。

关于微博服务端 API 的 OAuth 认证实现



新浪微博跟 update 相关的 api 已经挂了很多天了一直没有恢复正常，返回错误：40070 Error limited application access api!，新浪开放平台的论坛里 n 多的人都在等这个恢复，新浪官方也相当的恶心出问题了连个公告都没有，既不说什么原因又不说什么时候能恢复。还是有版主说是 api 正在升级礼拜 1 恢复正常今天都礼拜 2 了还是不行。基于这个原因我的 android 版的新浪微博客户端已经停工好几天了，刚好是跟 update 相关的一些功能。　　客户端开发不成了，就自己做做服务端程序，提供类似新浪微博 rest api 服务， api 其实说简单也很简单了，无法是通过链接对外提供 json 或者 xml 格式的数据和接收外部提供的数据进去相应的存储、删除、更新等操作。过程中碰到的最麻烦的问题就是 OAuth 认证功能了，在做 android 版的新浪微博客户端时候也花了蛮长的时间对 OAuth 认证进行研究，在客户端原先是采用了 oauth-signpost 开源项目，后来由于某些原因就放弃了这个开源类库，自己重新写了 OAuth 认证部分的实现， 现在做服务端的 OAuth 认证，其实有过做客户端的经验做服务端也差不多，简单的说无非是客户端对参数字符串进行签名然后把签名值传输到服务端，服务端也对同样对参数字符串进行签名，把从客户端传过来的签名值进去比较，简单的说就这么个过程，具体实现肯定比这个要复杂多了，不明真相的同学可以

google 一下 OAuth 进行深入的学习研究了。 服务端程序用 asp.net 和 C#编写了而非 java，理由很简单本人对.net 更加熟悉。由于想快速的实现效果采用了 oauth-dot-net 开源项目并没有全部自己写。

一、首先新建名为 Rest Api 的 ASP.NET Web 应用程序，然后添加 oauth-dot-net 开源项目相关的几个 dll(Castle.Core.dll 、Castle.MicroKernel.dll 、Castle.Windsor.dll 、CommonServiceLocator.WindsorAdapter.dll 、Microsoft.Practices.ServiceLocation.dll 、OAuth.Net.Common.dll 、OAuth.Net.Components.dll、OAuth.Net.ServiceProvider.dll)。

二、在 Web.config 文件里添加相应的配置，具体可以参考 OAuth.Net.Examples.EchoServiceProvider 项目，然后在 Global.asax.cs 添加如下代码：

```
1.  public override void Init()
2.          {
3.              IServiceLocator injector =
4.                  new WindsorServiceLocator(
5.                      new WindsorContainer(
6.                          new XmlInterpreter(
7.                              new ConfigResource("oauth.net.components"))));
8.
9.              ServiceLocator.SetLocatorProvider(() => injector);
10.         }
```

复制代码

接下来是比较重要，就是 request_token、authorize、access_token 的实现，OAuth 认证实现的几个过程，不理解可以看 android 开发我的新浪微博客户端-OAuth 篇(2.1) ，具体代码实现很多是参考 OAuth.Net.Examples.EchoServiceProvider 示例项目。

三、 首先新建 ConsumerStore.cs 类，用来存储 Consumer 信息，由于测试项目所以存储在内存中并没有考虑保存到数据库，真实项目的时候请把相应的 Consumer 信息保存到数据库中。Consumer 信息对应新浪微博其实就是应用的 App Key 和 App Secret，当开发者在新浪微博建一个新的应用获取 App Key 和 App Secret，所以完整的应该还需要一个开发一个提供给第三方开发者申请获取 App Key 和 App Secret 的功能页面，这里就不具体实现，直接在代码里写死了一个名为测试应用的 Consumer， App Key：2433927322， App Secret：87f042c9e8183cbde0f005a00db1529f，这个提供给客户端测试用。 具体代码如下：

```
1.  public sealed class ConsumerStore : InMemoryConsumerStore, IConsumerStore
2.      {
```

```
3.                      internal static readonly IConsumer FixedConsumer = new OAuthConsumer("2433927322", "87f042c9e8183cbde0f005a00db1529f", "测试应用", ConsumerStatus.Valid);
4.
5.          public ConsumerStore()
6.          {
7.              this.ConsumerDictionary.Add(
8.                  ConsumerStore.FixedConsumer.Key,
9.                  ConsumerStore.FixedConsumer);
10.         }
11.
12.         public override bool Add(IConsumer consumer)
13.         {
14.             throw new NotSupportedException("Consumers cannot be added to this store--it is fixed.");
15.         }
16.
17.         public override bool Contains(string consumerKey)
18.         {
19.             return ConsumerStore.FixedConsumer.Key.Equals(consumerKey);
20.         }
21.
22.         public override bool Update(IConsumer consumer)
23.         {
24.             throw new NotSupportedException("Consumers cannot be updated in this store--it is fixed.");
25.         }
26.
27.         public override bool Remove(IConsumer consumer)
28.         {
29.             throw new NotSupportedException("Consumers cannot be removed from this store--it is fixed.");
30.         }
31.     }
```

复制代码

四、接下来就是 request_token 功能，新建 RequestTokenHandler.cs，这个是 OAuth.Net.ServiceProvider.RequestTokenHandler 子类，并且是 httpHandlers 所以需要在 Web.config 中添加 httpHandlers 配置，这个用来接收客户端程序的请求，返回给客户端程序 Request Token 和 Request Secret 用，具体代码如下：

```
1. public sealed class RequestTokenHandler : OAuth.Net.ServiceProvider.RequestTokenHandler
2. {
```

```
3. protected override void IssueRequestToken(HttpContext httpContext, OAuthRequestContext requestContext)
4. {
5. //产生 RequestToken
6. IRequestToken token = this.GenerateRequestToken(httpContext, requestContext);
7.
8. requestContext.RequestToken = token;
9. Uri callbackUri;
10. if (Uri.TryCreate(requestContext.Parameters.Callback, UriKind.Absolute, out callbackUri))
11. {
12. if (!ServiceProviderContext.CallbackStore.ContainsCallback(token))
13. {
14. //保存 Callback 地址了
15. ServiceProviderContext.CallbackStore.AddCallback(token, callbackUri);
16. }
17. }
18. else
19. OAuthRequestException.ThrowParametersRejected(new string[] { Constants.CallbackParameter }, "Not a valid Uri.");
20.
21.
22. //把 token.Token 和 token.Secret 输出到客户端，
23. requestContext.ResponseParameters[Constants.TokenParameter] = token.Token;
24. requestContext.ResponseParameters[Constants.TokenSecretParameter] = token.Secret;
25. }
26.
27. protected override IRequestToken GenerateRequestToken(HttpContext httpContext, OAuthRequestContext requestContext)
28. {
29.
30. return ServiceProviderContext.TokenGenerator.CreateRequestToken(requestContext.Consumer, requestContext.Parameters);
31. }
32. }
```

复制代码

五、 接着是 authorize 功能，新建名为 authorize.aspx 的页面，用来给用户输入账号和密码进行授权的页面，这个页面很简单具体如下图，在这个页面中获取用户输入的账户和密码跟数据库中存储的用户账号和密码进行验证，如果验证通过返回之前客户端提供的 callback 地址，并且给这个地址添加一个校验码，具体代码如下：

```
1. public partial class authorize : System.Web.UI.Page
```

```
2.  {
3.  protected void Page_Load(object sender, EventArgs e)
4.  {
5.
6.  }
7.
8.  protected void Button1_Click(object sender, EventArgs e)
9.  {
10. if (loginName.Text == "test" && password.Text == "123")
11. {
12. string toke = Request.Params["oauth_token"];
13. IRequestToken tk = ServiceProviderContext.TokenStore.GetRequestToken(toke);
14. Uri callback = ServiceProviderContext.CallbackStore.GetCalback(tk);
15. string oauth_verifier = ServiceProviderContext.VerificationProvider.Generate(tk);
16. Response.Redirect(callback.ToString() + "?oauth_verifier=" + oauth_verifier);
17. }
18.
19. }
20. }
```

复制代码

六、接下来就是 access_token 功能，新建 AccessTokenHandler.cs ， 这个是 OAuth.Net.ServiceProvider.AccessTokenHandler 子类，并且是 httpHandlers 所以需要在 Web.config 中添加 httpHandlers 配置，这个用来接收客户端程序的请求，返回给客户端程序 Access Token 和 Access Secret 用，具体代码如下：

```
1.  public sealed class AccessTokenHandler : OAuth.Net.ServiceProvider.AccessTokenHandler
2.      {
3.              protected override void IssueAccessToken(HttpContext httpContext, OAuthRequestContext requestContext)
4.          {
5.              //产生 access token
6.                  IAccessToken accessToken = this.GenerateAccessToken(httpContext, requestContext);
7.
8.              accessToken.Status = TokenStatus.Authorized;
9.
10.             // 把 accessToken 和 accessSecret 输出到客户端，
11.                 requestContext.ResponseParameters[Constants.TokenParameter] = accessToken.Token;
12.             requestContext.ResponseParameters[Constants.TokenSecretParameter] = accessToken.Secret;
13.         }
14.
```

```
15.                 protected override IAccessToken GenerateAccessToken(HttpContext
    httpContext,  OAuthRequestContext requestContext)
16.         {
17.                                                                         return
    ServiceProviderContext.TokenGenerator.CreateAccessToken(requestContext.Consu
    mer, requestContext.RequestToken);
18.         }
19.     }
20.
21. public class TokenGenerator : ITokenGenerator
22.     {
23.                 internal static readonly IRequestToken FixedRequestToken = new
    OAuthRequestToken("requestkey",
24.             "requestsecret",
25.             ConsumerStore.FixedConsumer,
26.             TokenStatus.Authorized,
27.             null,
28.             ServiceProviderContext.DummyIdentity,
29.             new string[] { });
30.
31.                 internal static readonly IAccessToken FixedAccessToken = new
    OAuthAccessToken(
32.             "accesskey",
33.             "accesssecret",
34.             ConsumerStore.FixedConsumer,
35.             TokenStatus.Authorized,
36.             TokenGenerator.FixedRequestToken);
37.
38.                 public IRequestToken CreateRequestToken(IConsumer consumer,
    OAuthParameters parameters)
39.         {
40.             return TokenGenerator.FixedRequestToken;
41.         }
42.
43.                 public IAccessToken CreateAccessToken(IConsumer consumer,
    IRequestToken requestToken)
44.         {
45.             return TokenGenerator.FixedAccessToken;
46.         }
47.     }
48.
49. public class TokenStore : InMemoryTokenStore, ITokenStore
50.     {
51.         public TokenStore()
```

```
52.         {
53.             this.RequestTokenDictionary.Add(
54.                 TokenGenerator.FixedRequestToken.Token,
55.                 TokenGenerator.FixedRequestToken);
56.
57.             this.AccessTokenDictionary.Add(
58.                 TokenGenerator.FixedAccessToken.Token,
59.                 TokenGenerator.FixedAccessToken);
60.         }
61.
62.         public override bool Add(IRequestToken token)
63.         {
64.             throw new NotSupportedException("Tokens cannot be added to the token store--it is fixed.");
65.         }
66.
67.         public override bool Add(IAccessToken token)
68.         {
69.             throw new NotSupportedException("Tokens cannot be added to the token store--it is fixed.");
70.         }
71.
72.         public override bool Update(IRequestToken token)
73.         {
74.             throw new NotSupportedException("Tokens cannot be updated in the token store--it is fixed.");
75.         }
76.
77.         public override bool Update(IAccessToken token)
78.         {
79.             throw new NotSupportedException("Tokens cannot be updated in the token store--it is fixed.");
80.         }
81.
82.         public override bool Remove(IRequestToken token)
83.         {
84.             throw new NotSupportedException("Tokens cannot be removed from the token store--it is fixed.");
85.         }
86.
87.         public override bool Remove(IAccessToken token)
88.         {
89.             throw new NotSupportedException("Tokens cannot be removed from the token store--it is fixed.");
```

```
90.         }
91.     }
```

复制代码

这样就完成了一个最最简单小型的服务端 OAuth 认证，然后用 android 客户端进行测试 ok 通过。 注意点： 一、android 模拟器访问本地服务地址为 10.0.2.2，比如 http://localhost:3423/authorize.aspx 在模拟器中用 http://10.0.2.2:3423/authorize.aspx 。 二、OAuth.Net 类库的 OAuth.Net.Common 项目中的 interface ICallbackStore 添加了一个 Uri GetCalback(IRequestToken token);并且在具体的实现类 InMemoryCallbackStore 添加了实习代码： public Uri GetCalback(IRequestToken token) {

```
        lock (this.callbackStore)
        {
            if (this.callbackStore.ContainsKey(token))
            {
                return this.callbackStore[token];
            }
            else
            {
                return null;
            }
        }
    }
```

三、为了能用我前面做的给新浪用的 android 客户端，对于类库源代码 AccessTokenHandler 的 ParseParameters 方法做了如下修改，因为新浪请求 api 的时候都会加一个 source 的参数： protected virtual void ParseParameters(HttpContext httpContext, OAuthRequestContext requestContext) {

```
        .......
        parameters.AllowOnly(
                Constants.ConsumerKeyParameter,
                Constants.TokenParameter,
                Constants.SignatureMethodParameter,
                Constants.SignatureParameter,
                Constants.TimestampParameter,
                Constants.NonceParameter,
                Constants.VerifierParameter,
                Constants.VersionParameter, // (optional)
                Constants.RealmParameter, // (optional)
                "source");
```

```
        ......
    }
```

android 开发我的新浪微博客户端-大图浏览以及保存篇(7)

在阅读微博的功能篇中，如果微博包含了图片就会在微博正文下面显示该张图片，但是这个图片只是张缩略图，这样就需要提供一个能放大缩小查看这张图片的功能，当点击正文中的缩略图的时候显示一个简单的图片浏览器功能，提供图片的放大、缩小、拖拽操作方便用户查看图片，同时也提供保存图片到手机的功能。本功能的 UI 比较简单就不单独分篇讲了，具体的实现效果如上图。

新建 ImageActivity.java 作为图片浏览 Activity，在 res/layout 下新建 image.xml 的 Layout 作为图片浏览的布局文件，image.xml 布局代码很简单了就不详细解释了直接贴代码：

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <LinearLayout
3.    xmlns:android="http://schemas.android.com/apk/res/android"
4.    android:layout_width="wrap_content"
```

```
5.  android:layout_height="wrap_content"
6.  android:orientation="vertical">
7.  <LinearLayout
8.  android:layout_width="fill_parent"
9.  android:layout_height="41px"
10. android:background="@drawable/imagebar_bg">
11.
12. <RelativeLayout
13. android:layout_width="fill_parent"
14. android:layout_height="fill_parent"
15. android:layout_weight="2">
16. <Button
17. android:id="@+id/returnBtn"
18. android:layout_width="63px"
19. android:layout_height="29px"
20. android:layout_centerInParent="true"
21. android:text="返回"
22. android:textColor="#ffffff"
23. android:background="@drawable/btn1_bg">
24. </Button>
25. </RelativeLayout>
26.
27. <RelativeLayout
28. android:layout_width="fill_parent"
29. android:layout_height="fill_parent"
30. android:layout_weight="1">
31. <TextView
32. android:layout_width="wrap_content"
33. android:layout_height="wrap_content"
34. android:layout_centerInParent="true"
35. android:text="浏览图片"
36. android:textColor="#ffffff">
37. </TextView>
38. </RelativeLayout>
39.
40. <RelativeLayout
41. android:layout_width="fill_parent"
42. android:layout_height="fill_parent"
43. android:layout_weight="2">
44. <Button
45. android:id="@+id/downBtn"
46. android:layout_width="60px"
47. android:layout_height="29px"
48. android:layout_centerInParent="true"
```

```
49.  android:text="下载"
50.  android:textColor="#ffffff"
51.  android:background="@drawable/btn2_bg">
52.  </Button>
53.  </RelativeLayout>
54.
55.  </LinearLayout>
56.  <RelativeLayout
57.  android:layout_width="fill_parent"
58.  android:layout_height="fill_parent">
59.
60.  <MySinaWeiBo.ui.ImageZoomView
61.      xmlns:android="http://schemas.android.com/apk/res/android"
62.      android:id="@+id/pic"
63.      android:layout_width="fill_parent"
64.      android:layout_height="fill_parent">
65.  </MySinaWeiBo.ui.ImageZoomView>
66.
67.  <ZoomControls
68.  android:id="@+id/zoomCtrl"
69.  android:layout_width="wrap_content"
70.  android:layout_height="wrap_content"
71.  android:layout_alignParentRight="true"
72.  android:layout_alignParentBottom="true">
73.  </ZoomControls>
74.  </RelativeLayout>
75. </LinearLayout>
```

复制代码

上面的代码中用到了一个自定义控件 MySinaWeiBo.ui.ImageZoomView，这个就是整个功能的核心部分，用来实现图片的放大、缩小、拖拽的一个图片显示控件，这个控件非我原创，是参考了 Android one finger zoom tutorial 这篇博客写出来的，所以我在这里也不贴实现代码了，有兴趣的大家可以直接看看这个文章。　　接下要做的就是用这个 ImageZoomView 来显示图片，在阅读微博内容的页面中当点击内容中的缩略图片的时候会把这个缩略图对应的原图的 url 传给当前的这个 ImageActivity，那么在 ImageActivity 的 onCreate 方法中根据这个 url 获取图片并且设置给 ImageZoomView。在 onCreate 方法中代码如下：

```
1.  @Override
2.  public void onCreate(Bundle savedInstanceState) {
3.  。。。。。
4.  Intent i=this.getIntent();
5.  if(i!=null){
6.  Bundle b=i.getExtras();
7.  if(b!=null){
```

```
8.  if(b.containsKey("url")){
9.  String url = b.getString("url");
10. mZoomView=(ImageZoomView)findViewById(R.id.pic);
11. Drawable img= AsyncImageLoader.loadImageFromUrl(url);
12.
13.
14. image=drawableToBitmap(img);
15. mZoomView.setImage(image);
16.
17. mZoomState = new ZoomState();
18. mZoomView.setZoomState(mZoomState);
19. mZoomListener = new SimpleZoomListener();
20. mZoomListener.setZoomState(mZoomState);
21.
22. mZoomView.setOnTouchListener(mZoomListener);
23. resetZoomState();
24. }
25. }
26. }
27. 。。。。。。
28. }
```

复制代码

```
1.  private void resetZoomState() {
2.  mZoomState.setPanX(0.5f);
3.  mZoomState.setPanY(0.5f);
4.
5.  final int mWidth = image.getWidth();
6.  final int vWidth= mZoomView.getWidth();
7.  Log.e("iw:",vWidth+"");
8.  mZoomState.setZoom(1f);
9.  mZoomState.notifyObservers();
10.
11. }
12.
13. public Bitmap drawableToBitmap(Drawable drawable) {
14. Bitmap bitmap = ((BitmapDrawable)drawable).getBitmap();
15. return bitmap;
16. }
```

复制代码

```
1.  ZoomControls zoomCtrl = (ZoomControls) findViewById(R.id.zoomCtrl);
2.  zoomCtrl.setOnZoomInClickListener(new OnClickListener(){
```

```
3.  @Override
4.  public void onClick(View view) {
5.  float z= mZoomState.getZoom()+0.25f;
6.  mZoomState.setZoom(z);
7.  mZoomState.notifyObservers();
8.  }
9.
10. });
11. zoomCtrl.setOnZoomOutClickListener(new OnClickListener(){
12.
13. @Override
14. public void onClick(View v) {
15. float z= mZoomState.getZoom()-0.25f;
16. mZoomState.setZoom(z);
17. mZoomState.notifyObservers();
18. }
19.
20. });
```

复制代码

这样一个简单的图片浏览器功能就完成了，支持放大缩小并且还能拖拽，基本上达到应用需求。

各位路过的大虾，有空也请点点下面的广告，就当帮兄弟缴地租了。