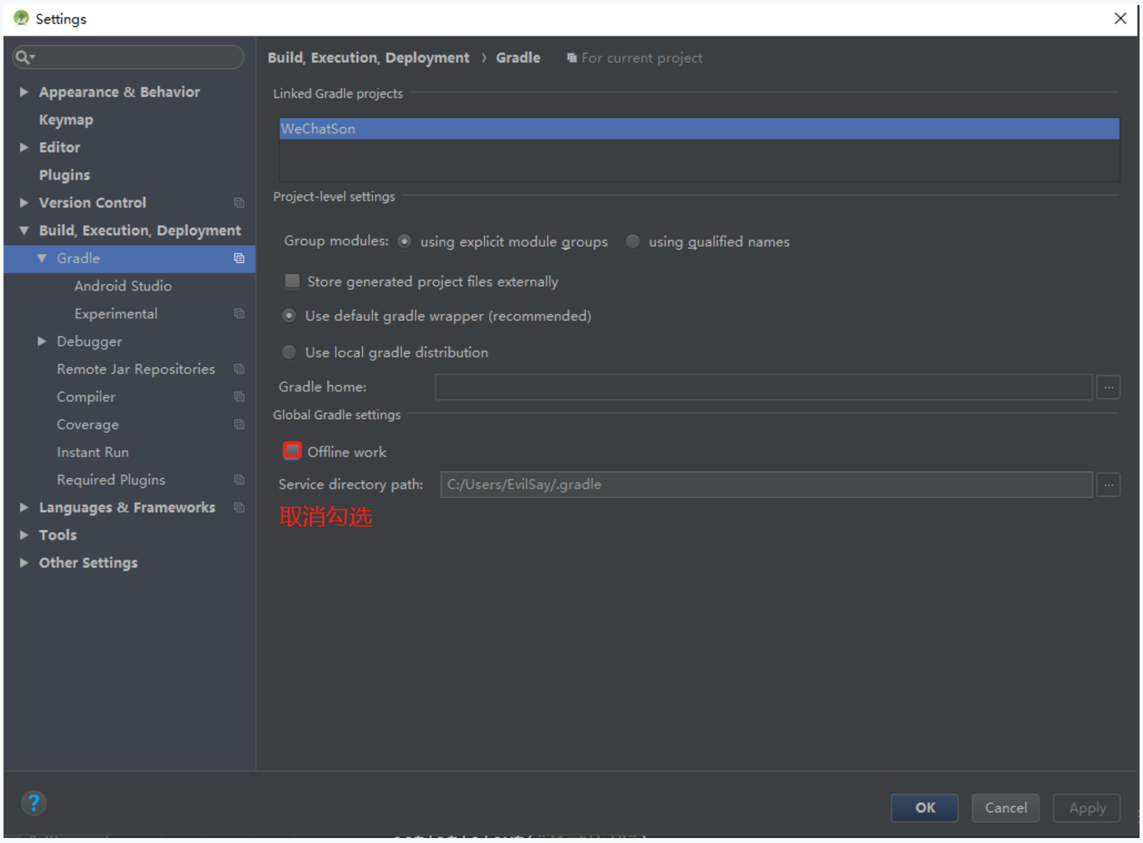
WeChatSon 技术详解

一、AndroidStudio3.0适配到AndroidStudio2.0

0、出现以下错误

> no cached version of com.android.tools.build:gradle:3.0.0 available for off
> **解决方案**

打开File-Setting-Build,Excution,Employment-gradle.


Settings

Build, Execution, Deployment › Gradle   For current project

Linked Gradle projects: WeChatSon

Project-level settings

Group modules: ● using explicit module groups ○ using qualified names

☐ Store generated project files externally

● Use default gradle wrapper (recommended)

○ Use local gradle distribution

Gradle home:

Global Gradle settings

☐ Offline work

Service directory path: C:/Users/EvilSay/.gradle

取消勾选

二、OKhttp<网络请求第方三控件>

稍微讲解一下什么是同步线程,什么是异步线程。同步线程就是当你命令程序干多件事情的时候它会一件一件的干而异步线程线程中是同时来干，在Android开发中，UI线程为优先级最高的同步线程，系统不允许你在UI线程中干其他同步线程中的事情。

0、导入OKhttp

> Android Studio3.0导入方法
> implementation ‘com.squareup.okhttp3:okhttp:3.7.0’
> implementation ‘com.squareup.okio:okio:1.12.0’
> 2.0导入方法
> compile ‘com.squareup.okhttp3:okhttp:3.7.0’
> compile ‘com.squareup.okio:okio:1.12.0’

1、GET请求<异步线程>

```
    /**
     * 构造一个OkHttp GET方法的请求
     * @param callback 异步回调
     */
     // GET方式请求 <异步线程>
    public static void httpUtils(Callback callback){
        //声明一个OkHttpClient并调用它的方法
        OkHttpClient client = new OkHttpClient();
        //访问请求
        Request request     = new Request.Builder()
                //URL地址
                .url(MAIL_LIST)
                .build();
//          调用HttpClient中的newCall中的网络异步回调
        client.newCall(request).enqueue(callback);
    }
// 调用方法
        //httpUtils静态方法,请注意这里会开一条异步线程但是要在里面设置Activ
ity的控件需要在onFailure或者onResponse中另开一条UI线程
        HttpUtils.httpUtils(new Callback() {
            /**
             * 网络请求失败
             * @param call 异步回调方法。网络连接失败可以在此重新请求
             * @param e    异常捕获
             */
            @Override
            public void onFailure(Call call, IOException e) {}
            /**
             * 网络请求成功
             * @param call
             * @param response 响应体
             * @throws IOException
             */
            @Override
            public void onResponse(Call call, Response response) th
rows IOException {
                response.body().string(); //拿到返回的JSON或者其他响应
体
                //另外开启一条UI线程
                runOnUiThread(new Runnable() {
            @Override
            public void run() {
                Toast.makeText(getApplicationContext(),response.bod
y().string(), LENGTH_SHORT).show();
            }
        });
            }
        });
    }
```

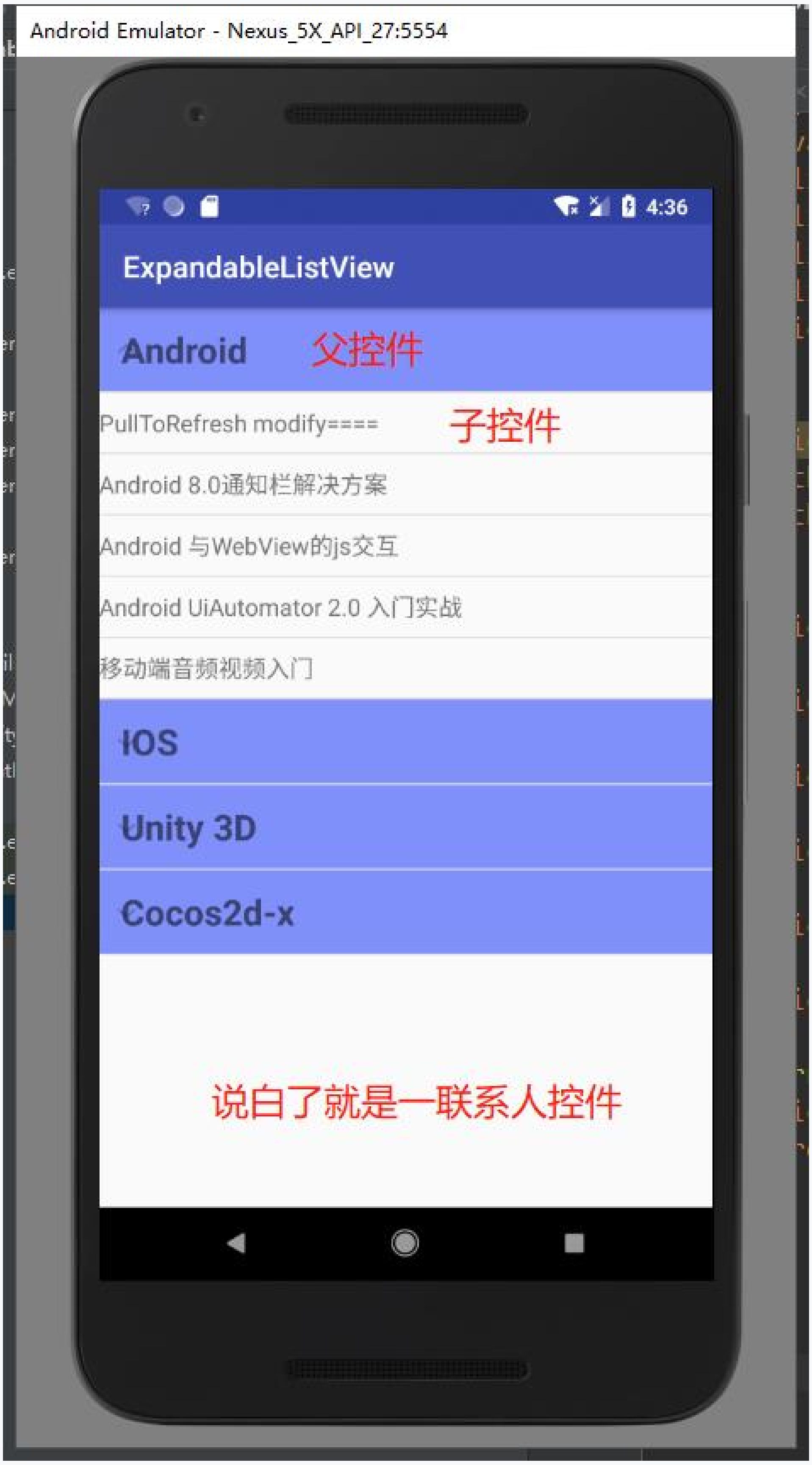
2、POST请求<异步线程>

```
    /**
     * 构造一个OkHttp POST方法的请求
     */
    public static void httpPostUtils_Reg(String name,String user,St
ring password,Callback callback){
        OkHttpClient client = new OkHttpClient();
        //PSOT 表单创建
        RequestBody body = new FormBody.Builder()
                .add("name",name)
                .add("user",user)
                .add("password",password)
                .build();
        //访问请求
        Request request     = new Request.Builder()
                .url(LOG_ADDRESS)
                //提交表单
                .post(body)
                .build();
//          网络异步回调
        client.newCall(request).enqueue(callback);
    }
    //调用方法
            httpPostUtils_Reg("123", "123", new Callback() {
            @Override
            public void onFailure(Call call, IOException e) {}
            @Override
            public void onResponse(Call call, Response response) th
rows IOException { }
        });
```

三、ExpandableListView

0、介绍:

ExpandableListView 一个垂直滚动的显示两个级别（Child,Group）列表项的视图，列表项来自ExpandableListAda pter， 组可以单独展开。


Android Emulator - Nexus_5X_API_27:5554

4:36

ExpandableListView

Android 父控件

PullToRefresh modify==== 子控件

Android 8.0通知栏解决方案

Android 与WebView的js交互

Android UiAutomator 2.0 入门实战

移动端音频视频入门

iOS

Unity 3D

Cocos2d-x

说白了就是一联系人控件

1、适配器及基础数据源

BaseExpandableListAdapter，将基础数据源链接到一个ExpandableListView中
数据源创建
``` java
```

```java
public class ChapterChat {

    private Integer id;

    private String name;
    /*以下变量名为数据库表名以及数据库键值 不重要*/
    public static final String TABLE_NAME = "tb_chapter";

    public static final String COD_NAME = "name";

    public static final String COD_ID = "_id";
     //子条目数据
    public List<ChapterChatItem> chapterChatItems = new ArrayList<>();

    public ChapterChat(Integer id, String name) {
    this.id = id;
    this.name = name;
}

//     添加数据源对象
public void addChild(ChapterChatItem chapterChatItem) {
    chapterChatItem.setPid(getId());
    chapterChatItems.add(chapterChatItem);
}

//     指定数据值
public void addChild(Integer id, String name) {
    ChapterChatItem chapterChatItem = new ChapterChatItem(id, name);
    chapterChatItem.setPid(getId());
    chapterChatItems.add(chapterChatItem);
}

public ChapterChat() {

}

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
        this.name = name;
    }

    public List<ChapterChatItem> getChapterChatItems() {
        return chapterChatItems;
    }

    public void setChapterChatItems(List<ChapterChatItem> chapterChatItems) {
        this.chapterChatItems = chapterChatItems;
    }
    }
    //子条目
    public class ChapterChatItem {
    private Integer id;
    private String name;
    private Integer pid;
    public static final String TABLE_NAME = "tb_item";
    public static final String COD_NAME = "name";
    public static final String COD_PID = "pid";
    public static final String COD_ID = "_id";

    public ChapterChatItem() {

    }

    public ChapterChatItem(Integer id, String name) {
        this.id = id;
        this.name = name;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Integer getPid() {
        return pid;
    }

    public void setPid(Integer pid) {
        this.pid = pid;
```

```
}

@Override
public String toString() {
    return "ChapterChatItem{" +
            "id=" + id +
            ", name='" + name + '\'' +
            ", pid=" + pid +
            '}';
}
}
```

BaseExpandableListAdapter 参数详解以及示例代码

```
public class ChapterChatAdapter extends BaseExpandableListAdapter {
//拿到父列表List数据
private List<ChapterChat> chapterChats;
    //传当前上下文的视图，不了解context的话可以去Google一下
    private Context context;
     //对于一个没有被载入或者想要动态载入的界面，都需要使用LayoutInflater.from()来
载入
     private LayoutInflater layoutInflater;

public ChapterChatAdapter(List<ChapterChat> chapterChats, Context contex
t) {
    this.chapterChats = chapterChats;
    this.context = context;
    //          传入当前上下文的布局
    this.layoutInflater = LayoutInflater.from(context);
}

//获得父列表项的数目
@Override
public int getGroupCount() {
    return chapterChats.size();
}

//获得子列表项的数目
@Override
public int getChildrenCount(int groupPosition) {
    return chapterChats.get(groupPosition).getChapterChatItems().size();
}

//拿到集合中第N个数据<个人理解>
@Override
public Object getGroup(int groupPosition) {
    return chapterChats.get(groupPosition);
}
```

```
//获取子列表项对应的Item
@Override
public Object getChild(int groupPosition, int childPosition) {
    return chapterChats.get(childPosition).getChapterChatItems().size();
}

//获得父列表项的Id
@Override
public long getGroupId(int groupPosition) {
    return groupPosition;
}

//获得子列表项的Id
@Override
public long getChildId(int groupPosition, int childPosition) {
    return childPosition;
}

//我不知道是干嘛这个大家可以试一试把他改成true，缘分报bug!
@Override
public boolean hasStableIds() {
    return false;
}

/**
 * 父类列表的视图
 *
 * @param groupPosition 父列表集合下标
 * @param isExpanded    获取父列表点击状态，展开为TURE,为展开为false
 * @param convertView   父列表视图加载
 * @param parent        <当前父列表视图>布局的容器
 * @return
 */
@Override
public View getGroupView(int groupPosition, boolean isExpanded, View conv
ertView, ViewGroup parent) {
    ChapterViewHolder chapterViewHolder = null;
    if (convertView == null) {
        // 加载视图参数解释--第一个传的父控件的视图<你需要自己创建一个>，第二个参 数<
使用自己layout中的控件参数>，第三个参数不使用默认控件参数
        convertView = layoutInflater.inflate(R.layout.parent_food, paren
t, false);
        chapterViewHolder = new ChapterViewHolder();
        chapterViewHolder.textView = convertView.findViewById(R.id.parent
_text);
        chapterViewHolder.imageView = convertView.findViewById(R.id.img);
        //对象重用
        convertView.setTag(chapterViewHolder);
    } else {
        chapterViewHolder = (ChapterViewHolder) convertView.getTag();
    }
    ChapterChat chapterChat = chapterChats.get(groupPosition);
    chapterViewHolder.textView.setText(chapterChat.getName());
    chapterViewHolder.imageView.setImageResource(R.drawable.img_exb);
    chapterViewHolder.imageView.setSelected(isExpanded);
    return convertView;
}

/**
 * 子列表视图
 * 参数详解同上
 *
 * @param groupPosition
 * @param childPosition
 * @param isLastChild
 * @param convertView
 * @param parent
 * @return
 */
@Override
public View getChildView(int groupPosition, int childPosition, boolean is
LastChild, View convertView, ViewGroup parent) {
    ChildViewHolder childViewHolder = null;
    if (convertView == null) {
        // 这个视图传的就是子控件了，也需要自己创建一个
        convertView = layoutInflater.inflate(R.layout.item_child_chapter,
 parent, false);
        childViewHolder = new ChildViewHolder();
        childViewHolder.textView = convertView.findViewById(R.id.id_text_
view);
        convertView.setTag(childViewHolder);
    } else {
        childViewHolder = (ChildViewHolder) convertView.getTag();
    }
    ChapterChatItem chapterChatItem = chapterChats.get(groupPosition).get
ChapterChatItems().get(childPosition);
    childViewHolder.textView.setText(chapterChatItem.getName());
    return convertView;
}

//列表项是否能否触发事件，返回true则为可以响应点击
@Override
public boolean isChildSelectable(int groupPosition, int childPosition) {
    return true;
}

// 内部类
class ChapterViewHolder {
    TextView textView;
    ImageView imageView;

}

class ChildViewHolder {
    TextView textView;
}
}
```

5、总结

0、没有