1. 常用分包
   1. 存放各种基类——base
   2. 存放各种实体类——bean
   3. 存放各种工具类——utils
   4. 存放各种适配器——adapter
   5. 存放各种界面的——view
2. 软件实施步骤
   1. 调通网络部分

编写通信工具类

* 1. 调好业务逻辑部分

对网络返回的数据进行json解析，并用一个实体类进行接收，然后把这些数据本来要做什么就用来进行处理。

* 1. 最后才写显示界面

结合实体类编写ListView等控件

1. 网络部分
   1. 在Manifest加入联网权限！
   2. 加入联网工具类HttpClient
   3. 在MainActivity里面开子线程，调用HttpClient工具类访问网络(后面改用Async来进行异步加载)
2. GSON解析
   1. 创建实体类（健康知识列表），用于测试解析。主要是因为这个JSON比较短
   2. 为了方便测试，先把JSON存放在Assets文件夹的一个xml文件里面，然后使用getAssets来读取字符串。
   3. **因为是多层JSON，所以要用JSONObject来取得真正有用的JSON的字符串，然后再用GSON通过反射来获得实体类**

JSONObject obj;

String healthyKnowStr = null;

List<HealKnowCategoryBean> list = null;

try {

//用JSONObject获取指定段的JSON内容

obj = new JSONObject(jsonStr);

healthyKnowListStr = (String) obj.getJSONArray("yi18").toString();

//用GSON来反序列化，生成相应的实体类

list = new Gson().fromJson(healthyKnowListStr,

new TypeToken<List<HealthyKnowListBean>>() {}.getType());

} catch (JSONException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

1. 确定主要功能，实现基本的业务逻辑
   1. 资讯
      1. 社会热点
      2. 食品新闻
      3. 药品新闻
      4. 生活贴上
      5. 医药新闻
      6. 企业要闻
      7. 热点热词
   2. 饮食
      1. 养阴补虚
      2. 补血益气
      3. 壮阳壮腰
      4. 丰胸
      5. 通乳
      6. 产后调理
      7. 美容养颜
      8. 减肥
      9. 亮发
      10. 解酒
      11. 增肥
      12. 明目
      13. 消暑解渴
      14. 润肠通便
      15. 利尿
      16. 活血化瘀
      17. 防癌抗癌
      18. 清热去火
      19. 畏寒怕冷
      20. 益智补脑
      21. 健脾开胃
      22. 延缓衰老
      23. 止血凉血
      24. 祛痰
      25. 其他功能
   3. 药箱（加个搜索栏啦）
      1. 性病用药
      2. 抗结核及麻风类
      3. 血液疾病类
      4. 水电解质及酸碱
      5. 抗寄生虫类
      6. 风湿免疫科
      7. 肿瘤科
      8. 神经/精神
      9. 内分泌失常
      10. 肾病
      11. 肝胆胰用药
      12. 心脑血管
      13. 维生素及营养类
      14. 儿科用药
      15. 妇科用药
      16. 男科用药
      17. 家庭常备
      18. 呼吸系统类
      19. 五官用药
      20. 肠胃用药
      21. 皮肤用药
      22. 感冒发热
      23. 手术用药
      24. 其他
   4. 百科(这里就整个gridview算啦)
      1. 减肥瘦身
      2. 私密生活
      3. 女性保养
      4. 男性健康
      5. 孕婴手册
      6. 育儿宝典
      7. 健康饮食
      8. 医疗护理
      9. 老人健康
      10. 孩子健康
      11. 四季养生
      12. 心理健康
2. 通过异步加载，从网络上面获取用到的信息
   1. 使用异步加载开源框架：AsyncHttpClient。使用它来访问网络数据，不用特定开一个线程出来
   2. 用法：只要自己写一个Handler来接收数据即可
   3. 代码：

public static void GetXml(String url, RequestParams params,final Handler handler){

AsyncHttpClient client = new AsyncHttpClient();

RequestHandle handle = client.get(url, params, new AsyncHttpResponseHandler(){

@Override

public void onSuccess(int statusCode, Header[] headers, byte[] responseBody) {

if(handler != null){

Message msg = new Message();

msg.obj = new String(responseBody);

handler.sendMessage(msg);

}

}

@Override

public void onFailure(int arg0, Header[] arg1, byte[] arg2,Throwable arg3) {

if(handler != null){

Message msg = new Message();

msg.obj = null;

handler.sendMessage(msg);

}

}

});

}

1. 业务逻辑类编写

用获取健康信息列表为例

* 1. 对于访问API用到的一些参数，用一个List来保存参数名和参数值
  2. 逻辑类代码

/\*

\* 获取健康信息知识列表

\* 请求参数（以下都不是必须的）：

\* page int 请求页数，默认是1

\* limit int 每页返回的条数，默认是20

\* type string 排序方式 ：id：最新时间；count：访问最多，默认是id，按最新时间。

\* id long id：这里是指知识分类的ID 默认为 null ，也就是全部。

\*/

public void GetHealthyKnowList(**List<KeyValuesBean> list**,Handler handler){

String urlStr = GlobalDate.API\_HEAYKNOW\_LIST;

//如果参数列表非null并且非空

if((list != null) && (!list.isEmpty()) ){

urlStr += "?";

//遍历链表，获得并设置参数名和参数值

for(int i=0;i<list.size();i++){

urlStr += list.get(i).getKey() + "=" + list.get(i).getValues();

if(i != (list.size()-1)){

urlStr += "&";

}

}

}

System.out.println("urlStr:"+urlStr);

AsyncHttpClientUtil.GetXml(urlStr, null, handler);

}

* 1. 测试类代码

写一个Handler来异步获取数据并进行JSON解析，获取真正有用的实体类列表

示例代码：

public static void TestGetHealthyKnowList(){

**Handler handler = new Handler(){**

@Override

public void handleMessage(Message msg) {

String jsonStr =(String) msg.obj;

JSONObject obj;

String healthyKnowListStr = null;

List<HealthyKnowListBean> list = null;

try {

//用JSONObject获取指定段的JSON内容

obj = new JSONObject(jsonStr);

healthyKnowListStr = (String) obj.getJSONArray("yi18").toString();

//用GSON来反序列化，生成相应的实体类

list = new Gson().fromJson(healthyKnowListStr,

new TypeToken<List<HealthyKnowListBean>>() {}.getType());

} catch (JSONException e) {

e.printStackTrace();

}

//遍历输出该List

for(int i=0;i<list.size();i++){

String str = null;

str = "title"+list.get(i).getTitle();

str += "author" + list.get(i).getAuthor();

System.out.println(str);

}

//平时用该List来更新UI界面

}

};

/\*

\* 获取健康信息知识列表

\* 请求参数（以下都不是必须的）：

\* page int 请求页数，默认是1

\* limit int 每页返回的条数，默认是20

\* type string 排序方式 ：id：最新时间；count：访问最多，默认是id，按最新时间。

\* id long id：这里是指知识分类的ID 默认为 null ，也就是全部。

\*/

//设置请求参数

List<KeyValuesBean> list = new ArrayList<KeyValuesBean>();

list.add(new KeyValuesBean("page", "20"));

list.add(new KeyValuesBean("limit", "5"));

list.add(new KeyValuesBean("type", "id"));

HealthyKnowEngine hke = new HealthyKnowEngine();

**hke.GetHealthyKnowList(list,handler);**

}

1. 基本的ListView的做法
   1. 写一个Item的布局文件
   2. 写一个adapter
      1. 复写getCount() ，设置条目数目
      2. 复写getView().加载刚刚的布局文件。

**后面优化ListView，主要也是优化该函数**

public View getView(int position, View convertView, ViewGroup parent) {

// TODO Auto-generated method stub

View view = View.inflate(HomeActivity.this, R.layout.list\_item\_home, null);

ImageView iv\_item = (ImageView) view.findViewById(R.id.iv\_item);

TextView tv\_item = (TextView) view.findViewById(R.id.tv\_item);

tv\_item.setText(names[position]);

iv\_item.setImageResource(ids[position]);

return view;

}

* 1. 自定义一个类，并继承于Listview
  2. 把自定义的ListView加入到MainActivity中

1. 网络图片查看——使用开源代码SmartImageView
2. 下拉刷新，滑倒底部显示加载更多的ListView的做法。（以后可以直接用LaztList开源代码）
   1. 下拉刷新
   2. 加载更多
      1. 写一个“加载更多”用到的布局文件。一般为一个Button和一个ProgressBar。在初始化ListView的时候加载该布局。并设置为不可见。
      2. 设置那个Button的按键监听。按键监听的处理函数是从外面的类给进来的。
      3. 定义私有变量lastItem，用来在后面记录当前页面的ListView中最后的Item的序列号
      4. 让Listview设置setOnScrollListener的监听
      5. 重写onScrool()，在里面更新lastItem的值
      6. OnScrollListener 内部有3个常量来记录ListView的滚动状态

SCROLL\_STATE\_FLING

The user had previously been scrolling using touch and had performed a fling.

SCROLL\_STATE\_IDLE

The view is not scrolling.

SCROLL\_STATE\_TOUCH\_SCROLL

The user is scrolling using touch, and their finger is still on the screen

* + 1. 重写onScrollStateChanged()，通过判断:
       1. lastItem的值是否等于adapter.getCount()
       2. scrollState是否等于OnScrollListener.**SCROLL\_STATE\_IDLE**

假如两个条件都符合，则算是已经拉到最低，显示”加载更多” 的布局界面。

* + 1. 在外部的按键处理函数中，要对ListView用到的List进行Add数据。并且数据加入完后，要通过**adapter.notifyDataSetChanged()**来刷新数据

1. 使用Fragment + ViewPager搭建软件基本框架
   1. 参考以前的代码即可以搭建
   2. 设置ViewPager不可以滑动
   3. 设置底部导航条的ImageButton背景为透明，然后再他们的Layout统一设置其背景色
2. 嵌套ViewPager解决触摸冲突的方法

自己自定义一个ViewPager作为子Viewpager，并且复写触摸拦截和触摸事件

看代码：

public class ChildViewPager extends ViewPager {

private float mDX, mDY, mLX, mLY;

int mLastAct = -1;

boolean mIntercept = false;

public ChildViewPager(Context context, AttributeSet attrs) {

super(context, attrs);

}

public ChildViewPager(Context context) {

super(context);

}

@Override

public boolean **onInterceptTouchEvent**(MotionEvent ev) {

switch (ev.getAction()) {

case MotionEvent.ACTION\_DOWN:

mDX = mDY = 0f;

mLX = ev.getX();

mLY = ev.getY();

break;

case MotionEvent.ACTION\_MOVE:

final float curX = ev.getX();

final float curY = ev.getY();

mDX += Math.abs(curX - mLX);

mDY += Math.abs(curY - mLY);

mLX = curX;

mLY = curY;

if (mIntercept && mLastAct == MotionEvent.ACTION\_MOVE) {

return false;

}

if (mDX > mDY) {

mIntercept = true;

mLastAct = MotionEvent.ACTION\_MOVE;

return false;

}

}

mLastAct = ev.getAction();

mIntercept = false;

return super.onInterceptTouchEvent(ev);

}

@Override

public boolean **onTouchEvent**(MotionEvent arg0) {

if (arg0.getAction() == MotionEvent.ACTION\_DOWN) {

// 此句代码是为了通知他的父ViewPager现在进行的是本控件的操作，不要对我的操作进行干扰

getParent().requestDisallowInterceptTouchEvent(true);

}

if (arg0.getAction() == MotionEvent.ACTION\_MOVE) {

// 此句代码是为了通知他的父ViewPager现在进行的是本控件的操作，不要对我的操作进行干扰

getParent().requestDisallowInterceptTouchEvent(true);

}

return super.onTouchEvent(arg0);

}

}

1. 定义统一显示列表内容的ListView的Fragment基类
   1. 先优化已经写好的含有ListView的Fragment的代码
   2. 参考别人的代码规范，只抽象出InitView和initDate两个方法即可
2. **ViewPager的三个难题（重点）**
   1. ViewPager里面再嵌套一个ViewPager

重写onTouchEvent事件，当处于Down或者Move事件时，调用方法：**getParent().requestDisallowInterceptTouchEvent(true);**

如：

@Override

public boolean onTouchEvent(MotionEvent arg0) {

if (arg0.getAction() == MotionEvent.ACTION\_DOWN) {

// 此句代码是为了通知他的父ViewPager现在进行的是本控件的操作，不要对我的操作进行干扰

**getParent().requestDisallowInterceptTouchEvent(true);**

}

if (arg0.getAction() == MotionEvent.ACTION\_MOVE) {

// 此句代码是为了通知他的父ViewPager现在进行的是本控件的操作，不要对我的操作进行干扰

**getParent().requestDisallowInterceptTouchEvent(true);**

}

return super.onTouchEvent(arg0);

}

* 1. 在ViewPager里面嵌套ListView。

左右滑动则切换ViewPager，上下滑动则滚动ListView

重写Viewpager里面设置事件拦截器(onInterceptTouchEvent)，如果左右滑动超多5dp，则把触摸事件拦截下来，不让其传递给下面的ListView。否则，则最后会被ListView的onTouchEvent消费了该滑动事件

例子：

@Override

public boolean onInterceptTouchEvent(MotionEvent ev) {

switch (ev.getAction()) {

case MotionEvent.ACTION\_DOWN:

mLX = ev.getX();

mLY = ev.getY();

break;

case MotionEvent.ACTION\_MOVE:

final float curX = ev.getX();

final float curY = ev.getY();

mDX += Math.abs(curX - mLX);

mDY += Math.abs(curY - mLY);

mLX = curX;

mLY = curY;

**//如果是左右滑动，那么就在这里进行拦截**

**//目的是让里面嵌套的ListView能够相互不影响**

**if(mDX-mDY > 5){**

**return true;**

**}**

}

}

* 1. 禁止ViewPager滑动

重写ontouch方法，让其直接返回true

@Override

public boolean onTouchEvent(MotionEvent arg0) {

return false;

}

1. Fragment的嵌套问题
   1. **首先ViewPager不能作为顶层容器（即类似于LinearLayout那些），否则会报错**
   2. 主页面是一个FragmentActivity，二级页面是一个Fragment，二级页面里面又包含一个含有Viewpager的Fragment，所以就涉及了Fragment的嵌套问题

**原本在Fragment的VIewpager的setAdapter，是使用getSupportFragmentManager，现在改为getChildFragmentManager**

1. 修改软件主框架，改为使用FragmentTabHost
   1. 原因：因为不用到ViewPager的滑动和预加载，所以没有必要用ViewPager，而且ViewPager会和下一层的VIewPager冲突
2. Fragment的问题
   1. 调用gerActivity()，返回空指针

方法：重写onAttach(Activity activity)方法

@Override

public void onAttach(Activity activity) {

super.onAttach(activity);

context = activity;

}

* 1. Fragment如何获得appliction的全局变量

一般在Fragment里面是调用getActivity.然后再像Activity那样从application那里拿数据

* 1. 两个Fragment之间通信

使用：getSupportFragmentManager().findFragmentByTag 找到Fragment，然后再用Bundle来携带数据

1. 创建初始化页面
   1. 创建一个activity并且在manifest.xml中配置
   2. 创建动画文件，让启动的时候，软件名称能够旋转播放
2. 软件优化，V1.0上线
   1. 把剩余的其他3个主页面也跟第一个一样搭建起来。至此软件框架基本完成
   2. 优化utils里面的async工具类，取消enginer类，直接通过async工具类就能从网上的API拿数据
   3. 优化Context内容页和主页面的配色
   4. 修改最低要求的版本号，从4.0改为2.3，提高测试通过率
3. 下一版本展望
   1. 第三方应用库
      1. 推送：极光推送，个推
      2. 应用统计：友盟
      3. 广告：多盟，有米
      4. 社交分享：shareSDK
   2. 离线下载功能
   3. 收藏功能
   4. 加入githun上面开源的界面特效
   5. 对软件进行加固，防止反编译
4. 引入shareSDK，加入社交分享功能
   1. 对于微信朋友圈或者微信分享，需要在微信的开放平台里面注册账号，获取相应的appkey才行。不然只能分享文字，不能分享网址和图片
   2. 对于第三方包使用问题解决：
      1. 注意第三方包的配置文件(manifest和properties)要求的Target-SDK，如果本机没有，则改为本机有的SDK
      2. 如果第三方包提供了一个library的project，导入进去一直出错的话那就不如直接把该project的文件放进去工程里面
5. 优化ListView

修改ListView的adapter的getview函数:

* 1. 加入内部类 viewholder
  2. 复用converview

1. 加入缓存功能
   1. 有了缓存之后，缓存数据的步骤变为：内存->SD卡->网络
   2. 内存：使用map容器，对于图片则以url作为key。对于Map的处理实际使用的方法：
      1. LruCache
      2. 弱引用
   3. SD卡：使用数据库进行插入删除
   4. 网络：访问API获取数据，并保存到内存和SD卡里
   5. **原来本来用的smartImageView是具有缓存功能。写本地缓存可以参考该开源项目**