http重链机制,可以采用try
{}catch{}的方案实现重连机制

实现长连接，定时任务我们需要定时向Internet发送数据，因为只是为了不然NAT映射表失效，所以只需发送长度为0的数据即可。
额乐儿
采用AlarmManager：AlarmManager类是属于android系统封装好来管理RTC模块的管理类。这里就涉及到RTC模块，要更好地了解两者的区别，就要明白两者真正的区别。在android客户端使用Push推送时，应该使用AlarmManager来实现心跳功能，使其真正实现长连接。

图片请求框架：
glide 哥来的: Google推荐的图片加载库 Glide默认的Bitmap格式是RGB_565
okhttp的 OkHttpClient:
xutil: x.image().bind(holder.mMyImage,URL);
picasso皮卡搜
Fresco 弗雷斯科 是faceBook中使用的强大框架
视频缓存处理
AndroidVideoCache（卡尺）是一个视频/音频缓存库，利用本地代理实现了边下边播，使用起来非常简单。
HttpProxyCacheServer（埔绕可C）是主要类，是一个代理服务器，可以配置缓存文件的数量、缓存文件的大小、缓存文件的目录和缓存文件命名算法，文件缓存均基于LRU算法，利用Builder来配置：

关于UI方面

SHAPE自己画背景图，配合选择器，Tint染色器（疼次）16，xml 6
矢量图无论放大、缩小或旋转等不会失真；最大的缺点是难以表现色彩层次丰富的逼真图像效果。
设置海拔 凸显布局的层次，建议使用阴影效果
默认为矩形，可以设置为矩形（rectangle）、椭圆形(oval)、线性形状(line)、环形(ring)
更华丽可以配合状态选择器，动画转场动画（界面切换动画）
webp,png,图标有四套图，autolayout 一张279k的png图片可以转换成67.5k的webp图片，而且不失真
第一步添加webp支持，添加so包和lib包,添加WebpUtils文件，里面有通过so包来处理webp文件成为byte数组的方法

应用性能优化处理

android UI性能优化

减少布局的层次
布局中公共的部分抽取复用

1、在布局中尽可能不要多层嵌套使用，如果多层嵌套会造成解析布局耗时延长
2、将布局中公共的部分抽取复用
3、代码采用RxJava链式编程
4、采用简单工厂模式创建对象并将已创建的对象保存在集合中
5、在方法中尽可以使用局部变量
6、减少一个方法的逻辑业务量，尽可以抽取成多个方法来调用实现功能
7、采用线程池来管理多线程网络请求
8、视图ListView的复用
9、ViewPaer采用懒加载模式
1）在片段可见时请求数据。此方案仍预加载了前后的页面，但是没有请求数据，只有进入到当前Framgent时才请求数据。
2）直接修改ViewPager源码。通过查看ViewPager源码可知，控制其预加载的是一个常量
DEFAULT_OFFSCREEN_PAGES，其默认值为1，表示当前页面前后各预加载一个页面，在这里我们直接将其设置为0即可，即去掉预加载。但是，这样有一个问题，那就是在使用其他控件时需要传入ViewPager时，这个就不能用了。
3）直接继承ViewPager，结合PagerAdapter实现懒加载。相对复杂
10、网络图片请求创建缓存，压缩
11、基类的抽取方法封装，复用减少减少代码冗余
12、减少一些不必要的循环的使用。
13、Apk的瘦身 混淆

热修复原理： 运行时动态加载jar, so, dex, apk
微信tinker 阿里的开源项目AndFix和Dexposed
创建一个新项目添加脚本依赖添加权限添加补丁的代码：实现在应用启动时（Application的onCreate方法）加载补丁

超文本传输协议（HTTP，HyperText Transfer Protocol)是互联网上应用最为广泛的一种网络协议。服务器端请求和应答的标准（TCP）
启动流程
自定义控件
View的绘制流程
事件分发

传感器

整个sensor传感器框架主要分为4层，其中包括

1、传感器Java部分，frameworks/base/core/java/Android/hardware/ SensorManager.java

2、传感器jni部分，frameworks/base/services/sensorservice/SensorService.cpp

3、传感器硬件抽象层，device/sprd/common/libs/libsensors/sensors.cpp

4、内核驱动层，kernel/drivers/input/misc/ltr_558als.c

信息采集 ： 温度传感器，光线传感器、距离传感器 ，磁场传感器，重力传感器

如何采集数据 ：对于业务处理，简化

取得传感器的管理器 ：SensorManager context.getSystemService()

- 获取某个传感器（） SensorManager.getDefaultSensor(传感器类型);
- 监听传感器，注册监听传感器
- 注销监听传感器
- 取得采样的数据

蓝牙开发步骤： Android所有关于蓝牙开发的类都在android.bluetooth包下有8个

1、添加相应的权限

<uses-permissionandroid:name="android.permission.BLUETOOTH_ADMIN" /> 额的命

<uses-permissionandroid:name="android.permission.BLUETOOTH" /> 不露处死

2、一般会用到的几个类：

BluetoothAdapter

蓝牙适配器 里面定义了 关闭打开蓝牙，取消发现，获取蓝牙地址，获取蓝牙名称等方法

BluetoothDevice 地外丝

类描述了一个蓝牙设备状态

BluetoothServerSocket 蓝牙的链接服务

三个方法两个重载的accept(埃克塞不扯),accept(inttimeout)两者的区别在于后面的方法指定了过时时间

内存管理机制：

android系统采用垃圾回收机制来管理内存，垃圾什么时候被回收取决于内存调度的阀值，只有低于这个值系统才会按一个列表来关闭用户不需要的东西，而这个列表是按照应用所处在什么进程下（前台，可见，服务，后台，空）以及内存是否充足来确定的，如果内存充足接正常的的进程来处理内存的回收，如果内存不足时可能会强制回收来维持系统的正常运行。

对于一个应用的加载而言是更加根据应用代码所处的不同空间来处理回收释放内存。

应用程序管理机制：

android应用程序管理主要由PackageManager这个类来管理，实现PackageManager这个抽象类的是ContextImpl.java。在ContextImpl.java中，有一个内部静态类叫ApplicationPackageManager，实现了所有PackageManager的接口。

系统窗口管理：是由WindowManagerService负责实现的.WindowManagerService

tcp协议：请求，响应，传输数据

三次握手

客户端

创建Socket连接服务端(指定ip地址,端口号)通过ip地址找对应的服务器

调用Socket的getInputStream()和getOutputStream()方法获取和服务端相连的IO流

```
Socket socket = new Socket("127.0.0.1", 12345);
        InputStream is = socket.getInputStream();                    //获取客户端输入流
        OutputStream os = socket.getOutputStream();                  //获取客户端的输出流
```

服务端
创建ServerSocket(需要指定端口号)
调用ServerSocket的accept()方法接收一个客户端请求，得到一个Socket
调用Socket的getInputStream()和getOutputStream()方法获取和客户端相连的IO流

```
ServerSocket server = new ServerSocket(12345);
        Socket socket = server.accept();                    //接受客户端的请求
        InputStream is = socket.getInputStream();           //获取客户端输入流
        OutputStream os = socket.getOutputStream();         //获取客户端的输出流
```

Socket通信：只是对TCP/IP的封装，是一个调用接口
基于TCP:声明对象指定断口号，调用ACCEPT方法接收客户段数据 客户段 服务器IP 断口号 INPUTSTREAM接收
通过Socket长连接，由服务器端主动把消息发送给客户端
网络通信其实就是Socket通信是对TCP/IP的封装，是将IP，端口进行封装

四大组件及生命周期
Activity:

1.启动Activity：系统会先调用onCreate方法，然后调用onStart方法，最后调用onResume（瑞Z无），Activity进入运行状态。

2.当前Activity被其他Activity覆盖其上或被锁屏：系统会调用onPause方法，暂停当前Activity的执行。

3.当前Activity由被覆盖状态回到前台或解锁屏：系统会调用onResume方法，再次进入运行状态。

4.当前Activity转到新的Activity界面或按Home键回到主屏，自身退居后台：系统会先调用onPause方法，然后调用onStop方法，进入停滞状态。

5.用户后退回到此Activity：系统会先调用onRestart方法，然后调用onStart方法，最后调用onResume方法，再次进入运行状态。

6.当前Activity处于被覆盖状态或者后台不可见状态，即第2步和第4步，系统内存不足，杀死当前Activity，而后用户退回当前Activity：再次调用onCreate方法、onStart方法、onResume方法，进入运行状态。

7.用户退出当前Activity：系统先调用onPause方法，然后调用onStop方法，最后调用onDestory方法，结束当前Activity。

启动模式:

standard默认模式，可以不用写配置。在这个模式下，都会默认在栈顶创建一个新的实例

singleTop 是否处于栈顶，是就复用否则创建一个新的

singleTask保存整个应用只有一个实例，是否存在，存在就将其他的全部出栈，负责创建一个新的

singleInstance启动一个新的任务栈来管理并移到前台 来电界面就是这个模式

service:服务执行一些很耗CPU工作或阻塞的操作，在服务中创建新线程来执行 避免ANR异常
启动：startService,bindservice， stop nubind
fragmean:3.0
兼容3.0以下。采用V4包 fragmean及fragmean管理器改用 通过getSupportfragmeanManager来获取管理器
Activity改用fragmeanActivity

Handler,AsyncTask,EventBus(发布订阅设计模式)，okhttp,Xutils
JNI
SlidingMenu菜单控件
通过代码直接new出来，设置位置，触摸模式，菜单的宽度，滑动效果，设置一个布局（），**Activity继承SlidingActivity**
Photoview图片的放大缩小,
ViewpagerIndicator ViewPager的指示器
ViewPager的适配器必须是继承的FragmentPagerAdapter，并重写getPageIconResId（int position）或者getPageTitle（int position）方法
pagerSlidingTabStrip 就是viewPager上的一条线
SwipeRefreshLayout Google自己的下拉刷新组件 v4包中的
添加下拉刷新监听器 显示或者隐藏刷新进度条 检查是否处于刷新状态 设置进度条的颜色主题

动画
数据库的增删改查
图片压缩
手机适配

项目：
1、触电新闻
是：新闻加直播
用到的开发框架及模式：
FragmentTabHost+Fragment +TabLayout
用到的技术：线程池，自定义可拖动控件，视频 ，mvc
项目的亮点：视频
主要职责：项目的上线发布，需求的讨论，线程池的使用，混淆打包的处理
2、穷游最世界
是：旅游攻略，提供一些门票机票酒店等
用到的开发框架及模式：
FragmentTabHost+Fragment +viewPager mvc
用到的技术：线程池，视图界面的封装抽取
主要职责：初期项目的搭建，通用工具网络请求，第三方登录

设计模式
1. 工厂设计模式 :FragmentFactory
2. 模板设计模式 ：ContentPage、BaseRefreshListFragment
3. MVC模式 MVP
4. 单例模式 GMApplication、 ThreadPollManager
5. 代理模式 ： ThreadPollManager
6. 观察者模式 ， Adapter ， ContentProvider

1. Application
2. CommonUtil
3. 数据缓存、有效性
4. 通过代码实现轮播图
5. xutils框架、ButterKnife、okHttp、CommonPullToRefresh、PhotoView
6. 动态创建Shape、选择
7. 传感器
8. 线程池管理
9. adapter适配多个条目
10. 模块化的封装思想

数据结构：数据之间的关系的一种集合
- [ ] 逻辑结构：

集合结构：数据元素平等，仅仅是同属于一个集合。
线性结构：数据元素之间有着一对一的关系。
树形结构：一对多的层次关系
图形结构：多对多的关系
逻辑结构是针对具体问题的，是为了解决某个问题的。
- [ ] 物理结构：

顺序存储结构，把数据元素存放在地址连续的存储单元里。
链式存储结构，将数据存储在任意的存储单元里。用指针存放数据元素的地址，通过地址找到数据。
- [ ] 排序：冒泡排序：相邻的两对比较找到最大的向前排，选择排序，插入排序：扑克牌的排序方法，希尔排序：，堆排序，归并排序，快速排序。

解决65535个方法的限制：通过multidex包进行多dex编译的方式

Android内自带的集合类：

ArrayMap 等用于HashMap 原理是二分检索法，也因此key的类型都是整型
SparseArray 等用于HashMap 原理是二分检索法，也因此key的类型都是整型
SparseIntArray 等用于HashMap 原理是二分检索法，只能存Integer
SparseLongArray 等用于HashMap 原理是二分检索法，只能存Long
SparseBooleanArray 等用于HashMap 原理是二分检索法，只能存Boolean

Android自带常见注解：

过时：@Deprecated

抑制警告：@SuppressWarnings

覆盖：@Override

隐藏：@hide

参数可以为空，不可为空： **@Nullable** 和 **@NonNull**

指定API版本：@TargetApi(11) 可以使用在TYPE,METHOD,CUSTRUCTOR上面 @SuppressLint('NewApi') 忽略Lint静态检测

@CallSuper 要求方法必须调用父类方法

元注解：

@Target 表示注解作用的目标。

@Retention 表示注解的作用时段

@Documented javadoc所生成的文档就会带上注解信息

@Inherited 该注解表示子类可以集成加载父类上的注解

资源类注解

@AnimatorRes,@AnimRes,@AnyRes,@ArrayRes,@AttrRes,@ BoolRes,@ ColorRes,@ DimensRes,@ DrawableRes,@ FractionRes,@ IdRes,@ IntegerRes,@ InterpolatorRes,@ LayoutRes,@ MenuRes,@ PluralsRes,@ RawRes,@ StringRes,@ StyleableRes,@StyleRes,@ TransitionRes,@ XmlRes

这部分注解是非常有实用性的，有22个

getDrawable(@DrawableRes int id) //getDrawable方法限定了传入的参数必须是Drawable资源文件

另外还有个@ColorInt，限制传入Hex颜色值。

在开发过程中需要传入某种资源文件时，用此类注解，是很有限制作用的做法。

GSON处理不同命名采用的注解：

Gson提供了fromJson() 和toJson() 两个直接用于解析和生成的方法，

前者实现反序列化，后者实现了序列化。同时每个方法都提供了重载方法，我常用的总共有5个

json的字段和值是的名称和类型是一一对应的，但也有一定容错机制

但有时候也会出现一些不和谐的情况

使用PHP作为后台开发语言时很常见

SerializedName注解提供了两个属性，上面用到了其中一个，

别外还有一个属性alternate，接收一个String数组

RxJava:

概念：异步，扩展的观察者模式 是一个实现异步操作的库

异步操作很关键的一点是程序的简洁性，因为在调度过程比较复杂的情况下，异步代码经常会既难写也难被读懂。 Android 创造的 AsyncTask 和Handler ，其实都是为了让异步代码更加简洁。RxJava 的优势也是简洁，但它的简洁的与众不同之处在于，随着程序逻辑变得越来越复杂，它依然能够保持简洁。

观察者模式

RxJava 的观察者模式

1) 创建 Observer

2) 创建 Observable

3) Subscribe (订阅)

http网络请求 返回常见状态码：

200（成功） 404（未找到）服务器找不到请求的网页

408（请求超时）服务器等候请求时发生超时

500（服务器内部错误） 服务器遇到错误，无法完成请求。

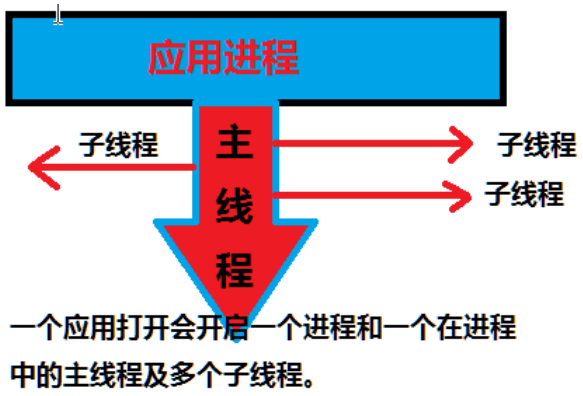
504（网关超时） 服务器作为网关或代理，未及时从上游服务器接收请求。

Android进程与线程基本

当一个程序第一次启动的时候，Android会启动一个LINUX进程和一个主线程。默认的情况下，所有该程序的组件都将在该进程和线程中运行。 同时，Android会为每个应用程序分配一个单独的LINUX用户。

线程是进程的一个实体,是CPU调度和分派的基本单位,它是比进程更小的能独立运行的基本单位。线程比进程更小，基本上不拥有系统资源，故对它的调度所用资源小，能更高效的提高系统内多个程序间并发执行的。

进程是一个具有独立功能的程序关于某个数据集合的一次运行活动。进程是系统进行资源分配和调度的一个独立单位。可以申请和拥有系统资源，是一个动态的概念，是一个活动的实体，是一个“执行中的程序”。不只是程序的代码，还包括当前的活动，数据空间。



进程按照重要性从高到低一共有五个级别

前台－－－〉可见－－－〉服务－－－〉后台－－－〉空

xutils3图片加载源码分析：

x.image().bind(holder.mMyImage,URL);

通过x.image()获得image管理器，通过管理器对象调用了bind(holder.mMyImage,URL);方法

要传入一个img对象和一个URL， 首先就创建了一个任务管理器并开启了一个线程对控件和URL判断，然后处理，图片的GET请求。

- xUtils支持超大文件(超过2G)上传，更全面的http请求协议支持(11种谓词)，拥有更加灵活的ORM，更多的事件注解支持且不受混淆影响；
- xUtils 最低兼容Android 4.0

okhttp源码分析：

即时通信

1、基于第三方IM SDK（环信，融云等）实现,作二次开发；

基于xmpp作优化 smack

优点：开放、标准、可扩展等；

缺点：

由于基于XML设计，报文太大，臃肿，浪费流量（70%的流量是耗费在XMPP本身的标签上）

java方面：

单例设计模式：懒汉式和饿汉式　　区别：一个是不管用不用先创建对象，另个一个是需要的时候调用方法再创建对象

优缺点：饿汉式　浪费内存资源　　懒汉式　　在多线程并发操作时有可能会创建多个对象

JAVA集合：底层数组或链表实现

数组：查寻快，增删慢　　链表：增删快，查询慢

SqLite使用方法:

1、第一需要助手extends SQLiteOpenHelper

2、在构造中创建数据库

3、在onCreate(SQLiteDatabase db)方法 通过execSQL（）创建表

4、在onUpgrade方法中设置数据库版本

5、在要使用的类中获取助手

6、通过助手对象调用相应的增删改查方法对定义的表进行操作

创建内容提供者－－－－〉一个应用要访问另一个应用的私有数据库

有私有数据库的应用创建的内容提供者，其它应用通过

1、创建内容提供者－－－－〉自定义类继承extends ContentProvider 重写相关的方法

2、清单文件中声明

```
<!-- 注册内容提供者
    name表示内容提供者全路径名
    authorities表示访问内容提供者中URI口令的第二个部份
    exported表示是否自愿将内容暴露给第三方，true自愿
  -->
    <provider
    android:name="cn.itheima.provider.UserProvider"
    android:authorities="cn.itheima.provider.itcast"
    android:exported="true">
    </provider>
```

3、在内容提供者类中

4、创建UriMatcher对象

```
//参数一： 口令如果出错了，返回的值
private static UriMatcher uriMatcher = new UriMatcher(0);
static{
        //参数一：口令的第二部份，和清单文件中配置的相同
        //参数二：口令的第三部份
        //参数三：口令如果正确了，返回的值，int类型
        uriMatcher.addURI("cn.itheima.provider.itcast","users",1);
}
```

5、private UserOpenHelper userOpenHelper;　　//声明数据库助手类

6、在onCreate()方法中创建创建数据库助手类对象

```
@Override
public boolean onCreate() {
        userOpenHelper = new UserOpenHelper(this.getContext());
        return false;
}
```

7、写相应的增删改查方法中的操作：

```
//获取访问者传入的口令
int code = uriMatcher.match(uri);    参数一：URI，表示第三方传入的口令
//判断口令是否正确
if(code==1){
//可以访问数据库
SQLiteDatabase db = userOpenHelper.getReadableDatabase();
//查询，返回结果有4条记录
Cursor c = db.query("users", projection, selection, selectionArgs, null, null, sortOrder);
//返回Cursor
return c;
否则抛出异常或给出提示
```

```
/** 查找子控件，不需要强制类型转换 */
    public <T> T findView(int id) {
        T view = (T) findViewById(id);
        return view;
    }
```

访问第三方私有数据库：

```
//创建工具类
ContentResolver cr = this.getContentResolver();
//准备访问内容提供者的口令
//口令的第二部份----清单文件
//口令的第三部份----UserProvider.java
//大小写敏感
Uri uri = Uri.parse("content://cn.itheima.provider.itcast/users");
通过工具类对象开始对数据库进行相应的操作
```

新特性：

Fragment　出于3.0时代用于平板后来才在手机开发中兴起

5.0时　退出很多材料设计的控件

如：**TabLayout**控件

状态选择器，转场动画等

DrawerLayout控件容器

6.0时代权限：如 运行时权限，电话 信息的监听拦截权限

7.0时代：**VR**技术的支持，分屏多任务功能　省流量模式

集成支付宝支付流程

集成前的准备：

1：签约， 签约拿到商户PID 、商户收款账号等应用信息

2：生成商家RSA密钥对（公钥和私钥）并上传公钥到支付宝开发平台：

支付流程：

1、确认支付：要请求服务器（商户信息，金额，时间，用户信息等），生成支付订单（使用RSA私钥对订单信息作加密处理后并做为一个参数返回给客户端）

2、客户端调用支付宝PayTask的pay方法实现支付，需要传递支付信息和服务器返回的signedOrderInfo。

3、处理支付结果（成功、失败、取消）

集成：

1. 导入支付宝jar包

2. 配置清单文件:

　　1）添加权限

　　2）声明Activity

3.集成相关java代码

　最重要的 请求服务器“生成订单接口”

集成地图的定位搜索等功能

1、创建应用，指定发布版本的安全码SHA1　应用包名

2、添加相应的权限，配置KEY

3、添加jar包so包

4、获取MapView设置相应的生命周期　显示地图

5、定位AMapLocationClient　需要添加服务

Android分四层，

最底层　　**linux内核（Linux Kernel），**

　　函数库层（Libraries），

　　应用框架层（Application Framework）

最上层　　应用层（**Applications**）。

mvc与mvp

MVC全称是Model - View - Controller，是模型(model)－视图(view)－控制器(controller)的缩写, MVC是一种框架模式而非设计模式

- **Model** ： 数据模型，包括实体bean和业务bean。 业务bean封装对数据的读写操作，包括：数据库删改查，网络请求，文件操作等。
- **View**： 用户界面，包括Activity、Fragment，显示布局和控件。
- **Controller**： 控制器, 在传统**Android**开发中，**Activity**和**Fragment**既充当了**View**的角色，同时也充当了**Controller**的角色。 Activity或Fragment需要调用网格请求类请求数据，同时可能需要调用DAO操作数据库保存缓存数据。

MVC的特点及存在问题（**Android**应用）

- 视图层是可以直接访问数据层的， 从而视图层可以持有数据层的成员变量。
- 因为View可以调用Model，从而不可避免的还要包括一些业务逻辑代码，从而造成View要更改或利用
- 以上也可能会造成View的代码过多甚至臃肿，难以维护。

MVP三个角色 **/Presenter**负责逻辑的处理，**Model**提供数据，**View**负责显示

- **Model** ： 数据模型，包括实体bean和业务bean。 业务bean封装对数据的读写操作，包括：数据库删改查，网络请求，文件操作等。

- **View**： 用户界面，包括Activity、Fragment，显示布局和控件。
- **Presenter**： 控制器（提出者，主持人），单独抽取出来的一层代码。
- 优点：
    - 实现了View与Model的解耦，Activity与Fragment只充当View的角色，不再有过多的业务代码，解决了MVC中Activity和Fragment代码臃肿的问题。
    - 实现了分层，每层的职责十分清晰，容易扩展和管理。

  缺点：
    - 会增加很多类，Presenter类和View层的接口，对于很多情况其实没必要

Android平台进行数据存储的方式,分别如下:

1 使用SharedPreferences存储数据 瑟尔的泼妇论色是

适用范围：保存少量的数据，且这些数据的格式非常简单：字符串型、基本类型的值。比如应用程序的各种配置信息（如是否打开音效、是否使用震动效果、小游戏的玩家积分等），解锁口 令密码等

保存基于XML文件存储的key-value键值对数据，通常用来存储一些简单的配置信息

2 、文件存储数据

3 、SQLite数据库存储数据 应用场景 短信，手机联系人，手机中的文件等数据

4、将JAVAbean封装为json数据存为文件

上传到网络存储数据 应用场景手机联系人同步技术

fastJson与**Gson**的区别 都是用来解析**JSON**数据的

在解析的数据量小于**200**条时，其实都差不多

在数据量大于**200**时 **fastJson**的解析速度及性能要远远高于**Gson**的

Gson的注解可以解决不同服务所写的**JSON**数据的键的处理

FastJson中@JSONField注释使用作用与Gson的注解作用相同。

内存泄漏与内存溢出OOM

内存泄漏的含义： 对象已经结束了不需要使用了，但是**GC**却不能回收，**GC**不能回收是由于另个一个正在使用的对象持有它的引用导致不能被回收，而停留在堆内存中，造成内存泄漏。长时间积累会导致内存溢出**OOM**

解决的方案：为了防止内存溢出，处理一些比较占用内存并且生命周期长的对象时，可以尽量使用软引用和弱引用。

1、 一个单例构造方向中需要一个上下文，在调用的过程中传入的是Activity的上下文。只要这个单例没有被释放，那么这个Activity也不会被释放一直到进程退出才会释放。
能使用Application的Context就不要使用Activity的Content，Application的生命周期伴随着整个进程的周期
2、 mHandler是Handler的非静态匿名内部类的实例，所以它持有外部类Activity的引用，我们知道消息队列是在一个Looper线程中不断轮询处理消息，那么当这个Activity退出时消息队列中还有未处理的消息或者正在处理消息，而消息队列中的Message持有mHandler实例的引用，mHandler又持有Activity的引用，所以导致该Activity的内存资源无法及时回收，引发内存泄漏。
创建一个静态Handler内部类，然后对Handler持有的对象使用弱引用，这样在回收时也可以回收Handler持有的对象，这样虽然避免了Activity泄漏，不过Looper线程的消息队列中还是可能会有待处理的消息，所以我们在Activity的Destroy时或者Stop时应该移除消息队列中的消息
3、 对于使用了BraodcastReceiver，ContentObserver，File，Cursor，Stream，Bitmap等资源的使用，应该在Activity销毁时及时关闭或者注销，否则这些资源将不会被回收，造成内存泄漏
在Activity销毁时及时关闭或者注销

每一个应用都是运行在一个独立的DVM虚拟机上的，根据不同的手机分配的可用内存可能是32M,或者64M,对于一个应用而言是不可以向手机内存任意索取空间的。
手机应用一般同期都比较短，一个应用用完就会关闭，而手机一般都是很少主动去重启的，如果一但发生内存泄漏，随着时间的积累程序就会

出现OOM。
OOM一但出现手机应用就会出现卡顿，手机发烫等问题。
如何避免应用泄漏：
及时让GC来回收堆内存中的垃圾，由于堆内存是一块不连续的内存区域，如果频繁的new/remove会产生大量碎片，消耗应用的性能，手动调用System.gc()，有可能GC不一定会执行，原因就是GC的回收机制

对象在堆内存中的引用类型	回收时机	生命周期
强引用	GC任何时候都不能回收	只能等进程终止
软引用	GC只在内存不够时才能回收	内存不足会被回收
弱引用	GC只要一运行就会回收	GC终止
虚引用	任何时候都可能被GC回收	随时可能

图片加载框架：

Facebook开源的安卓上的图片加载框架,也可以说是至今为止安卓上最强大的图片加载框架。
相对于其他几个图片加载框架,Fresco主要的优点在于更好的内存管理和更强大的功能,更便捷的使用,缺点则是体积比较大,引入后会导致应用apk增加1.5M到2M的大小

picasso皮卡搜是Square公司开源的一个Android图形缓存库，地址http://square.github.io/picasso/，可以实现图片下载和缓存功能。仅仅只需要一行代码就能完全实现图片的异步加载：

```
1.  Picasso.with(context).load("http://i.imgur.com/DvpvklR.png").into(imageView);
```

图片压缩，二级缓存

Lrucacha，主要是get和set方法，存储的结构采用了LinkedHashMap，这种map内部实现了lru算法（近期最少使用算法）。

POST请求发现问题的方法是用fiddler链接手机，进行抓包测试 或者用postman的工具也可以

Fragment是**Android** 3.0新增的概念，Fragment用来描述一些行为或一部分用户界面在一个Activity中，你可以合并多个fragment在一个单独的activity中建立多个UI面板，同时重用fragment在多个activity中.你可以认为fragment作为一个activity中的一节模块，fragment有自己的生命周期，接收自己的输入事件，你可以添加或移除从运行中的activity.
一个fragment必须总是嵌入在一个activity中，同时fragment的生命周期受activity而影响

设置及获取横竖屏的方法：

设置：

1、 在清单文件的activity节点下进行设置 android:screenOrientation = " 横屏、竖屏"。

```
<activity android:name="com.yogi.ScreenOrientationActivity"
android:screenOrientation="portrait" >
设置横竖屏，横屏改成:landscape即可
        <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
</activity>
```

2、 在代码中进行设置:

设置横屏代码：setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);//横屏

设置竖屏代码：setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);//竖屏

```
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
```

设置全屏的方式:

1、在xml文件中进行设置 在activity或者application使用的theme中设置 WindowFullScreen = true;

getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.LayoutParams.FLAG_FULLSCREEN);

获取：

Configuration mConfiguration = this.getResources().getConfiguration(); //获取设置的配置信息

int ori = mConfiguration.orientation ; //获取屏幕方向

获取手机网络类型状态：

(ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE); 获取系统的网络服务

connManager.getActiveNetworkInfo(); 获取当前网络类型

获取手机唯一标识：

通过 **IMEI，MEID**或者**ESN**码　只针对于手机有效

```
TelephonyManager tm = (TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);

String DEVICE_ID = tm.getDeviceId();
```

通过**ANDROID_ID**　设备第一次启动产生的**ID**值

在设备首次启动时，系统会随机生成一个64位的数字，并把这个数字以16进制字符串的形式保存下来，这个16进制的字符串就是ANDROID_ID，当设备被wipe后该值会被重置。可以通过下面的方法获取：

```
import android.provider.Settings;

String ANDROID_ID = Settings.System.getString(getContentResolver(), Settings.System.ANDROID_ID);
```

ANDROID_ID可以作为设备标识，但需要注意：

- 厂商定制系统的Bug：不同的设备可能会产生相同的ANDROID_ID：9774d56d682e549c。
- 厂商定制系统的Bug：有些设备返回的值为null。
- 设备差异：对于CDMA设备，ANDROID_ID和TelephonyManager.getDeviceId() 返回相同的值。

Installtion ID : UUID　应用第一次安装时产生的**ID**值

以上四种方式都有或多或少存在的一定的局限性或者bug，在这里，有另外一种方式解决，就是使用UUID，该方法无需访问设备的资源，也跟设备类型无关。

这种方式是通过在程序安装后第一次运行后生成一个ID实现的，但该方式跟设备唯一标识不一样，它会因为不同的应用程序而产生不同的ID，而不是设备唯一ID。

因此经常用来标识在某个应用中的唯一ID（即Installtion ID），或者跟踪应用的安装数量。

打包流程（面试，重点）

1. 通过appt.exe生成R文件
2. 通过aidl.exe处理aidl文件
3. 通过javac.exe编译java源文件
4. 解压第三方jar包
5. 通过bx.bat把class转换成dex
6. 通过appt.exe编译资源文件得到resources.ap_
7. 通过apkbuilder生成未签名的apk
8. 通过jarsigner对apk进行签名；
9. 通过zipalign对apk进行优化（可选）；

使用ItemTouchHelper高效地实现 今日头条 、网易新闻 的频道排序、移动

使用RecyclerView包的ItemTouchHelper

Demo使用的方式。只能用RecyclerView实现，但是性能、功能都很强大，实现也非常简单，ItemTouchHelper处理好了关于在RecyclerView上添加拖动排序与滑动删除的所有事情。

通过ItemTouchHelper.Callback的onMove回调方法，对数组集合进行交换位置，并通过notifyItemMove方法刷新界面，RecyclerView默认的item动画为DefaultItemAnimator，它的notifyItemMove方法使范围内item有一个很自然的位移动画。

ANR异常

在UI线程中进行过多的耗时操作就会堵塞主线程从而造成ANR，具体造成ANR的原因有三个：**Activity**耗时操作超过**5s**，**Broadcast Receiver**（广播收音机） 超过**10s**，**Service**超过**15s**。

Android拍照

CAMERA 卡莫
SurfaceView 色非死唯有
Result 瑞早吃 **onActivityResult**
Bitmap bm = (Bitmap) data.getExtras().get("data");
getExtras 埃克斯尊死
调用系统的相机：
1、定义控件 **SurfaceView**
2、在**onActivityResult** 中接收回传的数据
3、通过**getExtras** 埃克斯尊死 转为**Bitmap** 进处压缩处理
4、展示图片

设计模式：
观察者模式、**rxJava**
简单工厂方法、
抽象工厂模式、**baseAtivity**
单例模式、**Eventbus**
代理模式、线程池
命令模式、

xml:可扩展标记语言,传输和存储数据,应用程序之间进行数据传输的最常用的工具
json:

PhotoView
2、进行缩小放大操作 有个手势的，当放手的时候，会有个动画效果

图片的压缩处理分析：

Options（奥普绳子）对象
设置inJustDecodeBounds = true不返回实际的BITMAP,也不给分配内存空间，但是可以获取图片信息，
然后通过工厂去获取BitmapFactory.decodeFile("/mnt/sdcard/dog.jpg",options);
获取图片的宽高
options.inSampleSize设置压缩比例 再重新申请空间，
然后通过工厂获取到**bitmap**对象再设置到控件。**Android**中图片是以**bitmap**形式存在的，那么**bitmap**所占内存，直接影响到了应用所占内存大小
图片长度 **x** 图片宽度 **x** 一个像素点占用的字节数
1、质量压缩

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
        int quality = Integer.valueOf(editText.getText().toString());  输入压缩值
   bit.compress(CompressFormat.JPEG, quality, baos);     bitmap.compress 康普ruai吃
        byte[] bytes = baos.toByteArray();
        bm = BitmapFactory.decodeByteArray(bytes, 0, bytes.length);
```

质量压缩不会减少图片的像素，它是在保持像素的前提下改变图片的位深及透明度等，来达到压缩图片的目的

png图片是无损的，不能进行压缩。

2、采样率压缩 **BitmapFactory.Options options = new BitmapFactory.Options();**

3、缩放法压缩（**martix**） 没脆可是

```
Matrix matrix = new Matrix();
        matrix.setScale(0.5f, 0.5f);  缩放变换
        bm = Bitmap.createBitmap(bit, 0, 0, bit.getWidth(),
              bit.getHeight(), matrix, true);
```

4、**RGB_565**法

```
 BitmapFactory.Options options2 = new BitmapFactory.Options();
        options2.inPreferredConfig = Bitmap.Config.RGB_565;

        bm = BitmapFactory.decodeFile(Environment
              .getExternalStorageDirectory().getAbsolutePath()
```

5、**createScaledBitmap** 将图片压缩成用户所期望的长度和宽度 有损压缩

适配

1、版本适配

2、屏幕适配

Android中的**Matrix**是一个**3 x 3**的矩阵

Matrix的对图像的处理可分为四类基本变换：

Translate 平移变换

Rotate 旋转变换

Scale 缩放变换

Skew 错切变换

GreenDao框架

一个精简的库

execSQL **A**个**rai**可

create 克瑞**A**吃

getWritableDatabase() **ruai**的博，得特贝司

update 啊不得特

insert 银色吃

初始化：**dragHelper = ViewDragHelper.create(this, mCallback);** 拽个黑而佩 克瑞**A**吃

填充结束，获取子控件 **onFinishInflate()** 匪类吃弗雷斯

测量 **onMeasure()** **onSizeChanged()** 陈志

处理的回调：**ViewDragHelper.Callback**

触摸事件传给**ViewDragHelper**处理：**dragHelper.processTouchEvent(event);**

ItemTouchHelper.Callback

Android hdpi ldpi mdpi xhdpi xxhdpi适配

		密度
ldpi	**240*320**	**120dp**
mdpi	**320*480**	**160**
hdpi	**480*800**	**240**
xhdpi	**720*1280**	**320**
xxhdpi	**1080*1920**	**480**

比例倍数关系：

idpi 是 **mdpi** 的**0.75**倍

hdpi 是 **mdpi** 的 **1.5**

xhdpi **mdpi** **2**

xxhdpi **mdpi** **3**

系统类关系结构

view的直接继承子类有12个

如：ImageView、ProgressBar、SurfaceView、TextView、ViewGroup

ViewGroup子类：

如：**LinearLayout** **FrameLayout**

ImageView子类：**ImageButton**

Textview常见子类：

Button **EditText**

几个兼容类的知识

Activity 基础类

Fragment **3.0**出现的基础类

FragmentActivity v4中,提供了**supportFragmentManager,**可以去管理**v4**包中的**Fragment**

ActionBarActivity v7中,为了使用**v7**中的**actionbar**

AppCompatActivity 高版本的**v7**中,管理**v4**包中的**Fragment,**以及使用**v7**中的**ActionBar**

ImageView.ScaleType / **android**:scaleType值的意义区别：

CENTER /center 按图片的原来size居中显示，当图片长/宽超过View的长/宽，则截取图片的居中部分显示

CENTER_CROP / centerCrop 按比例扩大图片的size居中显示，使得图片长(宽)等于或大于View的长(宽)

CENTER_INSIDE / centerInside 将图片的内容完整居中显示，通过按比例缩小或原来的size使得图片长/宽等于或小于View的长/宽

FIT_CENTER / fitCenter 把图片按比例扩大/缩小到View的宽度，居中显示

FIT_END / fitEnd 把图片按比例扩大/缩小到View的宽度，显示在View的下部分位置

FIT_START / fitStart 把图片按比例扩大/缩小到View的宽度，显示在View的上部分位置

FIT_XY / fitXY 把图片不按比例扩大/缩小到View的大小显示

MATRIX / matrix 用矩阵来绘制，动态缩小放大图片来显示

给控件设置外部字体样式：

```
TextView title = findView(R.id.CA_text_title);
```

typeFace 默认有四种系统的样式：

Typeface.DEFAULT //常规字体类型

Typeface.DEFAULT_BOLD //黑体字体类型

Typeface.MONOSPACE //等宽字体类型

Typeface.SANS_SERIF //sans serif字体类型

常用的字体风格名称还有：

Typeface.BOLD //粗体

Typeface.BOLD_ITALIC //粗斜体

Typeface.ITALIC //斜体

Typeface.NORMAL //常规

```
//创建字体样式对象
Typeface typeFace =Typeface.createFromAsset(getAssets(),"fonts/AK.TTF");
title.setTypeface(typeFace);
```

//在assets目录下创建fonts目录存放TTF格式的字体