# Android

## Android Layout字母自动大写，avd上也是：

android:textAllCaps="false"

## Android 让app状态栏透明

代码中：

if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*LOLLIPOP*) {  
 getWindow().addFlags(WindowManager.LayoutParams.*FLAG\_DRAWS\_SYSTEM\_BAR\_BACKGROUNDS*);  
 getWindow().clearFlags(WindowManager.LayoutParams.*FLAG\_TRANSLUCENT\_STATUS*);  
 getWindow().setStatusBarColor(Color.*TRANSPARENT*);  
}

Style中：

要根据api不同进行区别配置

<**item** name="android:windowTranslucentStatus">true</**item**>  
<**item** name="android:windowTranslucentNavigation">true</**item**>  
<**item** name="android:statusBarColor">#00000000</**item**>

Xml中

android:fitsSystemWindows="true"

## android Transaction（事务）

事务（Transaction）是并发控制的单位，是用户定义的一个操作序列。这些操作要么都做，要么都不做，是一个不可分割的工作单位。通过事务，SQL Server能将逻辑相关的一组操作绑定在一起，以便服务器保持数据的完整性

当对多个表进行更新的时候，某条执行失败。为了保持数据的完整性，需要使用事务回滚。

Android –AlertDialog复用自定义添加的view内控件无法获取

((EditText)(dialog.getWindow().findViewById(R.id.*dialpg\_editText*)))

## Android 给虚拟Android设备上传文件

D:\Genymotion\Genymotion\tools>adb push C:\Users\ai\Desktop\WeX5One.apk /sdcard/WeX5One.apk

## Android ARM转X86

An error occured while deploying the file.   
This probably means that the app contains ARM native code and your Genymotion device cannot run ARM instructions. You should either build your native code to x86 or install an ARM translation tool in your device.

部署文件时出错。   
这可能意味着应用程序包含本地ARM代码和你的genymotion设备无法运行ARM指令。你可以建立你的原生代码的x86或在您的设备上安装一个臂的翻译工具。

解决方法

1.用Android Studio 创建一个ARM的虚拟机；（当然这个不是你想要的）

2.下载Genymotion-ARM-Translation-Librarities工具转换包；下载路径：https://pan.baidu.com/s/1kUAftyR

将下载号的工具包直接拖拽到Genymotion中，然后提示重启模拟器

已下载于：D:\point\_to\_desktop\android\other

## Android 波纹效果（Ripple）

在Android L5.0中加入了触摸反馈动画。

其中最明显，最具代表性的就是波纹动画，比如当点击按钮时会从点击的位置产生类似于波纹的扩散效果。

**波纹效果（Ripple）：**

当你使用了Material主题后，波纹动画会自动应用在所有的控件上，我们当然可以来设置其属性来调整到我们需要的效果。

可以通过如下代码设置波纹的背景：

**android:background="?android:attr/selectableItemBackground"**波纹有边界

**android:background="?android:attr/selectableItemBackgroundBorderless"**波纹超出边界

**设置颜色**

我们也可以通过设置xml属性来调节动画颜色，从而可以适应不同的主题：

**android:colorControlHighlight**：设置波纹颜色

**android:colorAccent**：设置checkbox等控件的选中颜色

为view添加波纹效果：

   在布局管理器的属性中添加：

android:clickable="true"

android:background="?android:attr/selectableItemBackground"

为Textview添加波纹效果：

在属性中添加：

android:clickable="true"

android:background="?android:attr/selectableItemBackgroundBorderless"

## Android 在子线程中为什么不能用Toast；

因为Toast在创建的时候会依赖于一个Handler，并且一个Handler是需要有一个Looper才能够创建，而普通的线程是不会自动去创建一个Looper对象，比如说在某个Activity中能new一个Handler是因为Android系统在启动一个Activity的时候会默认的创建一个Looper对象，因此不能够在子线程中显示Toast，你可以在开启的子线程中执行Looper.prepare()来构建一个Looper，然后在显示Toast，但是不要忘记执行Looper.loop()来加载这个Looper，当然，也可以使用主线程的Looper，获取主线程的Looper的方法是Looper.getMainLooper()；同时需要注意的是，同样的不能在子线程中去更新UI界面，因为Toast是相对独立于UI界面的，就好比应用虽然crash掉了，并且已经返回到home界面，但是Toast依然会在hone界面显示出来。

注脚1可能没说清楚，昨天写的时候搞清了现象但没找出原因，刚刚找到了原因：——局部内部类的对象可以访问同一个方法中被定义为final的局部变量。定义为final后，编译程序的实现方法：将所有的局部内部类对象要访问的final型局部变量，都拷贝成为该内部类对象中的一个数据成员。这样，即使栈中局部变量（含final）已死亡，但由于拷贝到内部类中的数据成员任是final,其值永不变，因而局部内部类对象在方法内局部变量死亡后，照样可以访问final型局部变量。就是代码里的下面这个位置：

MainActivity.this.runOnUiThread(new Runnable(){.............}) //201行开始

new Runnable(){.............}是一个匿名内部类（可以理解为Runnable的子类），这里作为runOnUiThread方法的实参（局部内部类的对象）。

参考的原文在这里：<http://blog.csdn.net/cpp597455873/article/details/46287853>

## Android 真机或虚拟机上安装失败 Build -ReBuilde

Installation failed with message Failed to finalize session : INSTALL\_FAILED\_INVALID\_APK: /data/app/vmdl1841863905.tmp/11\_app-debug signatures are inconsistent. It is possible that this issue is resolved by uninstalling an existing version of the apk if it is present, and then re-installing.

WARNING: Uninstalling will remove the application data!

Do you want to uninstall the existing application?

<http://stackoverflow.com/questions/42668595/installation-failed-to-finalize-session-signatures-are-inconsistent-android>

## Android 修改状态栏字体颜色

<http://blog.csdn.net/java04/article/details/51542583>

Android 6.0开始，谷歌官方提供了支持，在style属性中配置android:windowLightStatusBar   
即可， 设置为true时，当statusbar的背景颜色为淡色时，statusbar的文字颜色会变成灰色，为false时同理。

<style name="statusBarStyle" parent="@android:style/Theme.DeviceDefault.Light">

<item name="android:statusBarColor">@color/status\_bar\_color</item>

<item name="android:windowLightStatusBar">false</item>

</style>

## Android snackbar 无动画

手机 – 设置 – 辅助功能 – 服务 – fooView（关闭）

## Android Studio 导入jar包后编译成功但代码提示无法使用

导入后放到 src/main/libs

# java

## Java 局部内部类使用局部final变量

(1).内部类是外部类的一个成员，就像外部类的成员方法一样，所以内部类有权限访问外部类的所有成员，包括private的。  
(2).内部类不能访问外部类方法中的局部变量，除非变量是final的(一般发生在方法中定义的内部类)。这是因为局部变量的生命周期原因。

(1).所谓“局部内部类”就是在对象的方法成员内部定义的类。而方法中的类，访问同一个方法中的局部变量，却必须要加上一个final。  
(2).原因是编译程序实现上的困难：内部类对象的生命周期会超过局部变量的生命期。局部变量的生命期：当该方法被调用时，该方法中的局部变量在栈中被创建，当方法调用结束时，退栈，这些局部变量全部死亡。而内部类对象生命期，与其它类一样，当创建一个局部内部类对象后，只有当没有其它人再引用它时，它才能死亡。所以完全可能一个方法已调用结束（局部变量已死亡），但该局部类的对象仍然活着。即：局部类的对象生命期会超过局部变量。  
(3).局部内部类的对象访问同一个方法中的局部变量，那么这就要求只要局部内部类对象还活着，那么栈中的那些它要访问的局部变量就不能“死亡”（否则：它都死了，还访问个什么呢？）。这就是说：局部变量的生命期至少等于或大于局部内部类对象的生命期。  
(4).解决方法：局部内部类的对象可以访问同一个方法中被定义为final的局部变量。定义为final后，编译程序的实现方法：将所有的局部内部类对象要访问的final型局部变量，都拷贝成为该内部类对象中的一个数据成员。这样，即使栈中局部变量（含final）已死亡，但由于它是final,其值永不变，因而局部内部类对象在变量死亡后，照样可以访问final型局部变量。  
(5).归纳总结：局部内部类对象中包含有要访问的final型局部变量的一个拷贝，成为它的数据成员。因此，正是在这个意义上，final型局部变量的生命期，超过其方法的一次调用。严格来说，方法调用结束，所有的局部变量（含final）全死亡了。但：局部内部类对象中有final型局部变量的拷贝。

## java 24 \* 60 \* 60 \* 1000

Date d=new Date();

SimpleDateFormat df=new SimpleDateFormat("yyyy-MM-dd");

System.out.println("today:"+df.format(d));

System.out.println("the day befor yestoday:" + df.format(new Date(d.getTime() - (long)2 \* 24 \* 60 \* 60 \* 1000)));

System.out.println("the day after tomorrow:" + df.format(new Date(d.getTime() + (long)3 \* 24 \* 60 \* 60 \* 1000)));

24\*60\*60结果强转为long型代表一天

## Java中子类和父类的初始化顺序-静态优先，父类优先

类的初始化顺序是：

1、初始化父类中的静态成员变量和静态代码块。

2、初始化子类中的静态成员变量和静态代码块。

3、初始化父类中的普通成员变量和代码块，在执行父类中的构造方法。

4、初始化子类中的普通成员变量和代码块，在执行子类中的构造方法。

## Java transient关键字

java语言的关键字，变量修饰符，如果用transient声明一个实例变量，当对象存储时，它的值不需要维持。换句话来说就是，用transient关键字标记的成员变量不参与序列化过程。

## Java >> n &<< n

>> n （右移 除2的n次方）

<< n （左移 乘2的n次方）

## Java Collection接口的toArry方法注意

1. public Object[] toArray()方法：方法中会重新构造一个Object[]数组，然后将集合中每个元素赋给数组，由于new的是Object数组，所以无法向上转型回之前的数组。
2. public <T> T[] toArray(T[] a)方法：返回传入的数组a指定的类型，a的length小于集合size时会利用反射重新构建一个指定类型数组，并把集合中所有元素赋给数组。

# 其他

## 大创答辩（互联网+综合养老服务平台）

Git是一个版本控制系统（Version Control System，VCS）。

版本控制是一种记录一个或若干文件内容变化，以便将来查阅特定版本修订情况的系统。

有了版本控制系统，就可以不用担心文件丢失，不小心误修改文件等等“事故”，而且你可以随便回到历史记录的某个时刻。

SVN, CVS这类早期的集中式版本控制系统，都有一个单一的集中管理的服务器，保存所有文件的修订版本，而协同工作的人们都通过客户端连到这台服务器，取出最新的文件或者提交更新

在分布式版本控制系统里，客户端并不只提取最新版本的文件快照，而是把代码仓库完整地镜像下来。

这么一来，任何一处协同工作用的服务器发生故障，事后都可以用任何一个镜像出来的本地仓库恢复。因为每一次的提取操作，实际上都是一次对代码仓库的完整备份。

GitLab是一个开源的版本管理系统，实现一个自托管的Git项目仓库，可通过Web界面进行访问公开的或者私人项目。它拥有与Github类似的功能，能够浏览源代码，管理缺陷和注释。可以管理团队对仓库的访问，它非常易于浏览提交过的版本并提供一个文件历史库。团队成员可以利用内置的简单聊天程序(Wall)进行交流。它还提供一个代码片段收集功能可以轻松实现代码复用，便于日后有需要的时候进行查找。在线DEMO

互联网和浏览器已经普及和扎根，企业应用的前端UI和样式组件，都会沿着w3c的html+css+js路线发展，资源也越来越多，比如bootstrap等。已经没有任何企业，能离开w3c的html+css+js标准，

后端和前端正好相反，技术正走向多元化。node，Python，java并存

WeX5采用的是Apache许可证开源免费

WeX5对跨平台多前端应用开发的支持极好，一次开发，多平台运行。

目前，WeX5能开发的跨平台多前端应用有：​

移动app（苹果ios app、安卓Android app）

微信应用（包括公众号、服务号和企业号应用）

企业web app（PC、平板和手机）

其他轻应用（百度直达号等）

WeX5的UI组件技术完全是开放主流，基于jquery和bootstrap技术，经WeX5高度优化，在移动上的表现很好，接近原生，真的很赞。

WeX5 的UI基于模块化开发，采用增强的RequireJS模块化技术

Requirejs是一个JavaScript文件和模块加载器。requireJS允许你把你的javascript代码独立成文件和模块，同时管理每个模块间的依赖关系。

风格样式库基于bootstrap技术

WeX5采用混合应用（hybrid app）开发模式。能轻松调用手机设备的系统和硬件能力，如相机、地图、LBS定位、指南针、通讯录、文件、语音、电池。。。等。

基于cordova框架 Cordova提供了一组设备相关的API，通过这组API，移动应用能够以JavaScript访问原生的设备功能，如摄像头、麦克风等。

WeX5的IDE基于eclipse，提供了一个完全可视化、组件化、拖拽式开发环境。

打包发布无任何限制，无需任何费用

轻松打包发布成高性能的原生app，包括安卓Android的apk，苹果ios的ipa

一键发布微信应用（微信公众号、服务号、企业号均可）

一键发布成百度轻应用、web应用等

WeX5的后端完全开放，可通过http、Websocket等协议连接各种后端中间件或云服务（java、node、php、.net等）

总而言之，WeX5是多平台、跨前端应用开发的优秀工具