### 源码分析相关面试题

* [Volley源码分析](http://www.jianshu.com/p/ec3dc92df581)
* [注解框架实现原理](http://www.jianshu.com/p/20da6d6389e1)
* [okhttp3.0源码分析](http://www.jianshu.com/p/9ed2c2f2a52c)
* [onSaveInstanceState源码分析](http://www.jianshu.com/p/cbf9c3557d64)
* [静默安装和源码编译](http://www.jianshu.com/p/2211a5b3c37f)

### Activity相关面试题

* [保存Activity的状态](http://www.jianshu.com/p/cbf9c3557d64)

### 与XMPP相关面试题

* [XMPP协议优缺点](http://www.jianshu.com/p/2c04ac3c526a)
* [极光消息推送原理](http://www.jianshu.com/p/d88dc66908cf)

### 与性能优化相关面试题

* [内存泄漏和内存溢出区别](http://www.jianshu.com/p/5dd645b05c76)
* [UI优化和线程池实现原理](http://www.jianshu.com/p/c22398f8587f)
* [代码优化](http://www.jianshu.com/p/ebd41eab90df)
* [内存性能分析](http://www.jianshu.com/p/2665c31b9c2f)
* [内存泄漏检测](http://www.jianshu.com/p/1514c7804a06)
* [App启动优化](http://www.jianshu.com/p/f0f73fefdd43)
* [与IPC机制相关面试题](http://www.jianshu.com/p/de4793a4c2d0)

### 与登录相关面试题

* [oauth认证协议原理](http://www.jianshu.com/p/2a6ecbf8d49d)
* [token产生的意义](http://www.jianshu.com/p/9b7ce2d6c195)
* [微信扫一扫实现原理](http://www.jianshu.com/p/a9d1f21bd5e0)

### 与开发相关面试题

* [迭代开发的时候如何向前兼容新旧接口](http://www.jianshu.com/p/cbecadec98de)
* [手把手教你如何解决as jar包冲突](http://www.jianshu.com/p/30fdc391289c)
* [context的原理分析](http://www.jianshu.com/p/2706c13a1769)
* [解决ViewPager.setCurrentItem中间很多页面切换方案](http://www.jianshu.com/p/38ab6d856b56)

### 与人事相关面试题

* [人事面试宝典](http://www.jianshu.com/p/d61b553ff8c9)

### 本文配套视频

* [配套视频](https://v.qq.com/x/page/y0396os8vc6.html)

## 谈一下你对Android中的context的理解，在一个应用程序中有多少个context实例？

### 1. 什么是Context?



img

通过金山词霸解释：上下文环境，什么是环境，这个词只可意会不可言传，为了大家更好的理解，举一个栗子，比如：我想点鸡，我在麦当劳跟服务员说我想点鸡，服务员给端上来一只香喷喷的烤鸡，如下图：



img

但是我换一个环境 ，去红灯区点鸡，妈咪就会给带来一只呆萌可爱的失足少女，如下图：



img

这就是环境，一样的东西不同地方，就表示不一样的意思。

### Context，中文直译为“上下文”，SDK中对其说明如下：

Interface to global information about an application environment. This is an abstract class whose implementation is provided by the Android system. It allows access to application-specific resources and classes, as well as up-calls for application-level operations such as launching activities, broadcasting and receiving intents, etc

### 从上可知一下三点,即：

* 它描述的是一个应用程序环境的信息，即上下文。
* 该类是一个抽象(abstract class)类，Android提供了该抽象类的具体实现类(后面我们会讲到是ContextIml类)。
* 通过它我们可以获取应用程序的资源和类，也包括一些应用级别操作，例如：启动一个Activity，发送广播，接受Intent信息 等

首先看它们的继承关系



img

### 2. 什么时候创建Context实例

熟悉了Context的继承关系后，我们接下来分析应用程序在什么情况需要创建Context对象的？应用程序创建Context实例的情况有如下几种情况：

1. 创建Application 对象时， 而且整个App共一个Application对象
2. 创建Service对象时
3. 创建Activity对象时

### 因此应用程序App共有的Context数目公式为：

总Context实例个数 = Service个数 + Activity个数 + 1（Application对应的Context实例）

1、创建Application对象的Context: 首先新建一个MyApplication并让它继承自Application，然后在AndroidManifest.xml文件中对MyApplication进行指定，如下所示：

<application
 android:name=".MyApplication"
 android:allowBackup="true"
 android:icon="@drawable/ic\_launcher"
 android:label="@string/app\_name"
 android:theme="@style/AppTheme" >
 ......
</application>

指定完成后，当我们的程序启动时Android系统就会创建一个MyApplication的实例,通过如下代码获取到它的实例:

public class MainActivity extends Activity {

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity\_main);
 MyApplication myApp = (MyApplication) getApplication();
 Log.d("TAG", "getApplication is " + myApp);
 }

}

可以看到，代码很简单，只需要调用getApplication()方法就能拿到我们自定义的Application的实例了，打印结果如下所示：



img

那么除了getApplication()方法，其实还有一个getApplicationContext()方法，这两个方法看上去好像有点关联，那么它们的区别是什么呢？我们将代码修改一下：

public class MainActivity extends Activity {

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity\_main);
 MyApplication myApp = (MyApplication) getApplication();
 Log.d("TAG", "getApplication is " + myApp);
 Context appContext = getApplicationContext();
 Log.d("TAG", "getApplicationContext is " + appContext);
 }
}

同样，我们把getApplicationContext()的结果打印了出来，现在重新运行代码，结果如下图所示：



img

打印出的结果是一样的呀，连后面的内存地址都是相同的，看来它们是同一个对象。其实这个结果也很好理解，Application本身就是一个Context，所以这里获取getApplicationContext()得到的结果就是MyApplication本身的实例。

那么有的朋友可能就会问了，既然这两个方法得到的结果都是相同的，那么Android为什么要提供两个功能重复的方法呢？实际上这两个方法在作用域上有比较大的区别。getApplication()方法的语义性非常强，一看就知道是用来获取Application实例的，但是这个方法只有在Activity和Service中才能调用的到。那么也许在绝大多数情况下我们都是在Activity或者Service中使用Application的，但是如果在一些其它的场景，比如BroadcastReceiver中也想获得Application的实例，这时就可以借助getApplicationContext()方法了，如下所示：

public class MyReceiver extends BroadcastReceiver {

 @Override
 public void onReceive(Context context, Intent intent) {
 MyApplication myApp = (MyApplication) context.getApplicationContext();
 Log.d("TAG", "myApp is " + myApp);
 }

}

也就是说，getApplicationContext()方法的作用域会更广一些，任何一个Context的实例，只要调用getApplicationContext()方法都可以拿到我们的Application对象。

* 欢迎关注微信公众号,长期推荐技术文章和技术视频

微信公众号名称：Android干货程序员

img

img