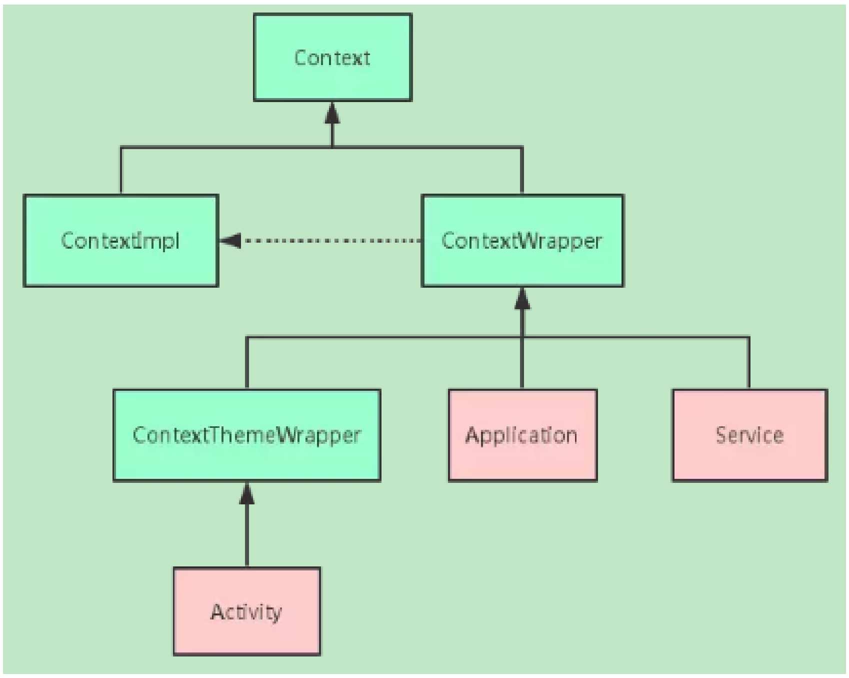
Fully understand Context in Android

Can we do Activity mActivity = new Activity()?

Android application is using java, but not like java that you can create a class and main() function, then able to run. Android is based on components including Activity, Service, Content Provider and Broadcast receiver. You are not able to create a new component simply by using 'new' since all of these components have their own context. Activity extends from Context, Service and Application also extend from Context.

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.



From source code, we can get the previous class diagram. ContextImpl is implementing all context abstract functions, and ContextWrapper wraps all functions from context and using ContextImpl class implementation. The attachBaseContext function is ContextWrapper class is making sure the context is attached only once. ContextThemeWrapper which as its name, apply theme from application or Activity which is defined as 'android:theme' in AndroidManifest.xml file. Since both Application and Service both do not need theme, so they inheritant directly from ContextWrapper. During the the activity, application and service are initiated, a new ContextImpl object will be created each time which implements the functions in Context.

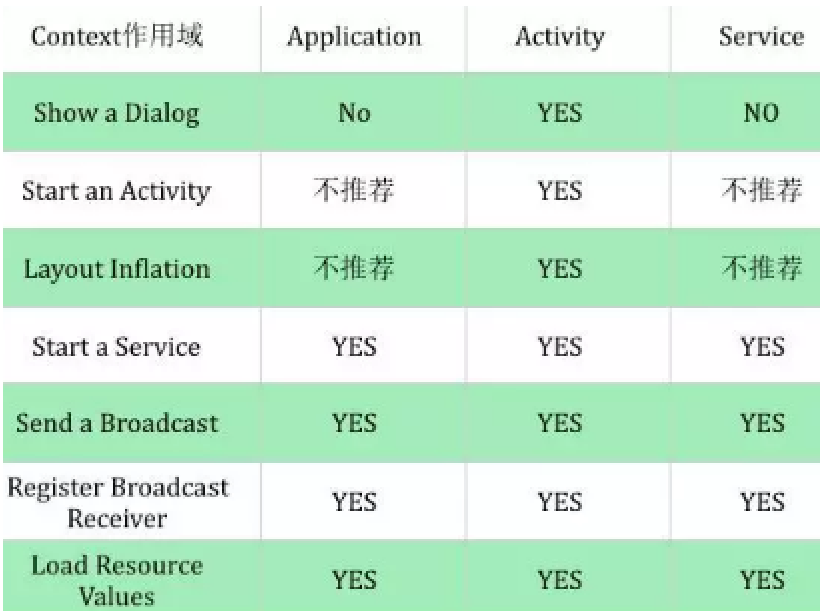
```
public class ContextWrapper extends Context {
    Context mBase;
    public ContextWrapper(Context base) {
        mBase = base;
    }

    /**
     * Set the base context for this ContextWrapper.  All calls will then be
     * delegated to the base context.  Throws
     * IllegalStateException if a base context has already been set.
     *
     * @param base The new base context for this wrapper.
     */
    protected void attachBaseContext(Context base) {
        if (mBase != null) {
            throw new IllegalStateException("Base context already set");
        }
        mBase = base;
    }
    ......
}
```

How many Context in an Application?

From above, we can conclude that Number(Context)=Activity+Service+1(Application). For other components like Context Provider, Broadcast Receiver which are not subclass of Context. So they are not counted, but we need pass context object to them when create a new one.

What Context can do?

Context is powerful, but it still has some restrictions. Since context is implemented by ContextImpl, so in most situations we can use activity, service and application context in common. However, in some situation like start a Activity or pop up a dialog, it must use Activity context since a new Activity is based on another Activity to form a stack, also a popup dialog need to show on top of Activity except some system alert dialog.



Context作用域	Application	Activity	Service
Show a Dialog	No	YES	NO
Start an Activity	不推荐	YES	不推荐
Layout Inflation	不推荐	YES	不推荐
Start a Service	YES	YES	YES
Send a Broadcast	YES	YES	YES
Register Broadcast Receiver	YES	YES	YES
Load Resource Values	YES	YES	YES

- If we use ApplicationContext to start an Activity with standard lauch mode, it will throw expcetion "ndroid.util.AndroidRuntimeException: Calling startActivity from

outside of an Activity context requires the FLAG_ACTIVITY_NEW_TASK flag. Is this really what you want?" This exception is because the non Activity context do not have the back stack. So from service and application is only to use FLAG_ACTIVITY_NEW_TASK (this activity will become the start of a new task on this history stack.) to start an Activity, which is not recommented.
- It is able to inflate the layout, but will only apply the default theme of the android system, those custom theme will not able to apply since only Activity extends ContextThemeWrapper class.

Ways to get Context:

- view.getContext() it's the activity context which the view is hosted
- Activity.getApplicationContext() get the current application context, when we need context, this global context need to be considerd at first.
- Activity.this when some UI controler need context or start an Activity. Toast can use ApplicationContext.

Difference between getApplication() and getApplicationContext()

```
    @Override
    public Context getApplicationContext() {
        return (mPackageInfo != null) ?
                mPackageInfo.getApplication() : mMainThread.getApplication();
    }
```

Actually both functions return application object since application itself is a context. So why android provide two functions? The reason is because getApplication() is only able to be used in Activity and Service. In other components like BroadcastReceiver is only able to use getApplicationContext() to get application object.

Improper use of context can easily cause memory leak

```
public class Singleton{
    private static Singleton instance;
    private Context mContext;

    private Singleton(){};

    public static Singleton getInstance(Context context){
        if(instance==null){
            synchronized(Singleton.class){
                if(instance==null){
                    instance = new Singleton();
                    mContext = context;
                }
            }
        }
        return instance;
    }
}
```

Since Singleton will live a long time in the application which cause the context which is hold by it will not able to be GC.

```
public class MainActivity extends Activity {
    private static ImageView mImageView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ...
        mImageView = (ImageView)findViewById(R.id.imageId);
}
```

Since mImageView is static, and the current context which is Activity.this is referenced by imageview, so the whole Activity is leaked.

Some tips to use context:

Since the application lives unitl the process terminated, so we can conclude the following suggestions

- When application context is able to handle, do use application context as first priority.
- Do not use/reference Activity which the lifecycle is longer than Activity.
- Do not use non static inner class inside activity since inner class has implicite refrence to outer class. If use static inner classes, do use weak reference of the objects from outer class.