What is Context in Android?

The Context in Android is actually the context of what we are talking about and where we are currently present. This will become more clear as we go along with this.

If we go by the name, it's the context of the current state of the application(object). It lets newly-created objects understand what has been going on. You can use the context to get information regarding activity and application.

And also, `Context` is like a handle to the system, it provides access to the resources, databases, and preferences, etc. An Android app has activities. It's like a handle to the environment your application is currently running in. The activity object extends the Context object. It allows access to application-specific resources and gives information about the application environment.

`Context` is almost everywhere in Android Development and it is the most important thing in Android Development, so we must understand to use it correctly.

A wrong use of `Context` can easily lead to memory leaks in an android application.

As there are different types of context in Android, we as an Android Developer often get confused about which context to use at which place. So let's understand what are those, how to use those and when to use which one.

Mainly two types of context:

- **Application Context:** It is the application and we are present in Application. For example - MyApplication(which extends Application class). It is an instance of MyApplication only.
- **Activity Context:** It is the activity and we are present in Activity. For example - MainActivity. It is an instance of MainActivity only.

Application Context

It is an instance that is the singleton and can be accessed in activity via `getApplicationContext()`. This context is tied to the lifecycle of an application. The application context can be used where you need a context whose lifecycle is separate from the current context or when you are passing a context beyond the scope of activity.

Example Use: If you have to create a singleton object for your application and that object needs a context, always pass the application context.

If you pass the activity context here, it will lead to the memory leak as it will keep the reference to the activity and activity will not be garbage collected.

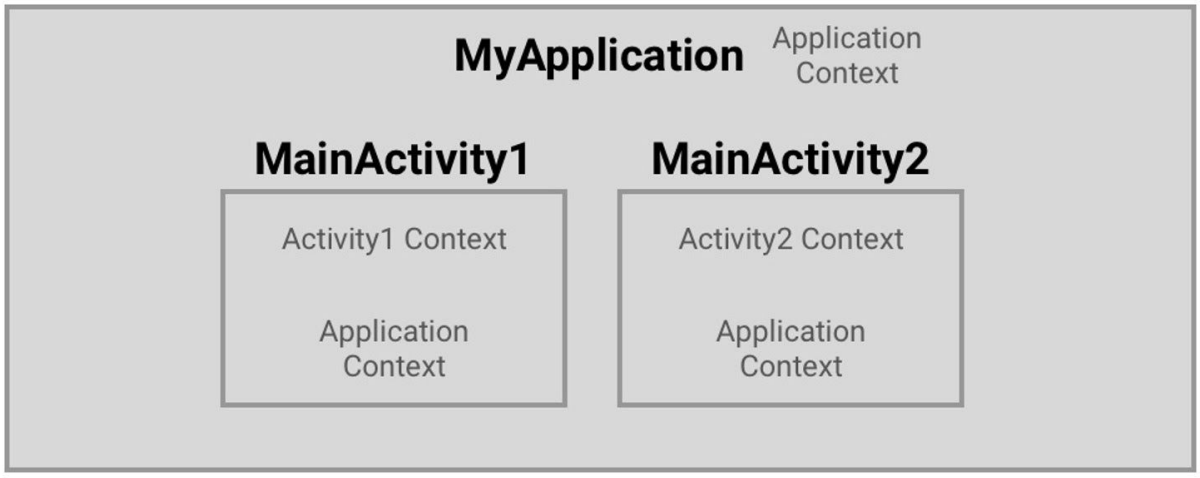
In case, when you have to initialize a library in an activity, always pass the application context, **not the activity context**.

You only use `getApplicationContext()` when you know you need a `Context` for something that may live longer than any other likely `Context` you have at your disposal.

Activity Context
This context is available in an activity. This context is tied to the lifecycle of an activity. The activity context should be used when you are passing the context in the scope of an activity or you need the context whose lifecycle is attached to the current context.

Example Use: If you have to create an object whose lifecycle is attached to an activity, you can use the activity context.

The app hierarchy looks like the following:



- MyApplication [Here we have only Application context present] [Nearest Context is Application Context]
- MainActivity1 [Here we have both Activity(MainActivity1) context and Application context present] [Nearest Context is Activity Context]
- MainActivity2 [Here we have both Activity(MainActivity2) context and Application context present] [Nearest Context is Activity Context]

getContext() in ContentProvider

This context is the application context and can be used similar to the application context. This can be accessed via the `getContext()` method.

When to use which Context?

Let's learn which context to use at which place with an example:

Suppose we have our class MyApplication(which extends the Application class). And, another class MyDB which is Singleton. And MyDB(which is Singleton) needs context. Which context will we be passing?

The answer is Application Context because if we pass the Activity Context (for example MainActivity1), even if MainActivity1 is not in use, the MyDB will be keeping the reference unnecessary which will lead to the memory leaks.

So always remember, in case of Singleton(lifecycle is attached to the application lifecycle), always use the Application Context.

So, now when to use the Activity Context. Whenever you are in Activity, for any UI operations like showing toast, dialogs, and etc, use the Activity Context.

Always try to use the nearest context which is available to you. When you are in Activity, the nearest context is Activity context. When you are in Application, the nearest context is the Application context. If Singleton, use the Application Context.

When not to use `getApplicationContext()` ?

- It's not a complete `Context`, supporting everything that `Activity` does. Various things you will try to do with this `Context` will fail, mostly related to the GUI.
- It can create memory leaks if the `Context` from `getApplicationContext()` holds onto something created by your calls on it that you don't clean up. With an `Activity`, if it holds onto something, once the `Activity` gets

garbage collected, everything else flushes out too. The `Application` object remains for the lifetime of your process.

The Rule of Thumb

- In most cases, use the `Context` directly available to you from the enclosing component you're working within. You can safely hold a reference to it as long as that reference does not extend beyond the lifecycle of that component. As soon as you need to save a reference to a `Context` from an object that lives beyond your Activity or Service, even temporarily, switch that reference you save over to the application context.