As the name suggests, it's the context of current state of the application/object. It can also be defined as *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.*

Context is the bridge between components. You use it to communicate between components, instantiate components, and access components.

**Your own components**

We use context to instantiate our components with Activity, Content Provider, BroadcastReceiver, and so on. We use it to access resources and filesystems as well.

**Your component and a system component**

Context acts as an entry point to the Android system. Some well-used System components are WifiManager, Vibrator, and PackageManager. You can access WifiManager using context.getSystemService(Context.WIFI\_SERVICE).

In this same way, you can use context to access the filesystem dedicated to your app as a user in OS.

Your own component and some other app’s component

Communicating between your own components and other app’s components is almost identical if you use the intent-filter approach. After-all, every components is an equal citizen in Android.

Simpler terms::

* It's like access of android activity to the app's resource.
* It's similar to, when you visit a hotel, you want breakfast, lunch & dinner in the suitable timings, right?
* There are many other things you like during the time of stay. How do you get these things?
* You ask the room-service person to bring these things for you.
* Here the room-service person is the context considering you are the single activity and the hotel to be your app, finally the breakfast, lunch & dinner have to be the resources.

One more Example

Every boss has an assistant to look after, to do all less important and time consuming tasks. If a file or a cup of coffee is needed, assistant is on the run. Some bosses barely know what’s going on in the office, so they ask their assistants regarding this too. They do some work themselves but for most other things they need help of their assistants.

In this scenario,

* Boss –  is the **Android application**
* Assistant – is **context**
* Files/Cup of coffee  – are **resources**

**Things that involve context are:**

1. *Loading a resource.*
2. *Launching a new activity.*
3. *Creating views.*
4. *obtaining system service.*

Context is the base class for *Activity*, *Service*, *Application* .... etc

*Another way to describe this: Consider context as remote of a TV & channel's in the television are resources, services, using intents etc - - - Here remote acts as an access to get access to all the different resources into foreground.*

* *So, Remote has access to channels such as resources, services, using intents etc ....*
* *Likewise ..... Whoever has access to remote naturally has access to all the things such as resources, services, using intents etc*

**Different invoking methods by which you can get context**

* getApplicationContext()
* getContext()
* getBaseContext()
* or this (when in the activity class)
* **Creating new objects**: Creating new views, adapters, listeners:
* TextView tv = new TextView(getContext());

ListAdapter adapter = new SimpleCursorAdapter(getApplicationContext(), ...);

* **Accessing standard common resources**: Services like LAYOUT\_INFLATER\_SERVICE, SharedPreferences:
* context.getSystemService(LAYOUT\_INFLATER\_SERVICE)

getApplicationContext().getSharedPreferences(\*name\*, \*mode\*);

* **Accessing components implicitly**: Regarding content providers, broadcasts, intent

getApplicationContext().getContentResolver().query(uri, ...);