**Dynamic User Interface**

1. Fragment Lifecycle

A fragment represents a behaviour or a portion of user interface in an Activity. Fragments can be combines in a single activity to build a multiplane UI and reuse a fragment in multiple activities. A fragment must always be embedded in an activity and the fragments lifecycle is directly affected by the host activity lifecycle.

When a fragment is added as part of an activity layout it lives in a ViewGroup inside the activity’s view hierarchy and the fragment defines its own view layout. A fragment is inserted into an activity layout by declaring the fragment in the activities layout file as a <fragment> element or from your application code by adding it to an existing ViewGroup.

To create a fragment you must crate a subclass of Gragment. The Fragment class has code that contains callback methods similar to an activity, such as onCreate(), onStart(), onPause(), and onStop().

onCreate()

The system calls this when creating the fragment. Within your implementation, you should initialize essential components of the fragment that you want to retain when the fragment is paused or stopped, then resumed.

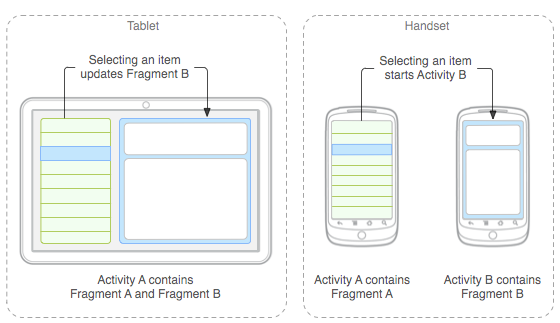
onCreateView()

The system calls this when it's time for the fragment to draw its user interface for the first time. To draw a UI for your fragment, you must return a View from this method that is the root of your fragment's layout. You can return null if the fragment does not provide a UI.

onPause()

The system calls this method as the first indication that the user is leaving the fragment (though it does not always mean the fragment is being destroyed). This is usually where you should commit any changes that should be persisted beyond the current user session (because the user might not come back).

1. Dynamic User Interface



You should design each fragment as a modular and reusable activity component. That is, because each fragment defines its own layout and its own behavior with its own lifecycle callbacks, you can include one fragment in multiple activities, so you should design for reuse and avoid directly manipulating one fragment from another fragment. This is especially important because a modular fragment allows you to change your fragment combinations for different screen sizes. When designing your application to support both tablets and handsets, you can reuse your fragments in different layout configurations to optimize the user experience based on the available screen space. For example, on a handset, it might be necessary to separate fragments to provide a single-pane UI when more than one cannot fit within the same activity.