

ANDROID UI – FRAGMENTS

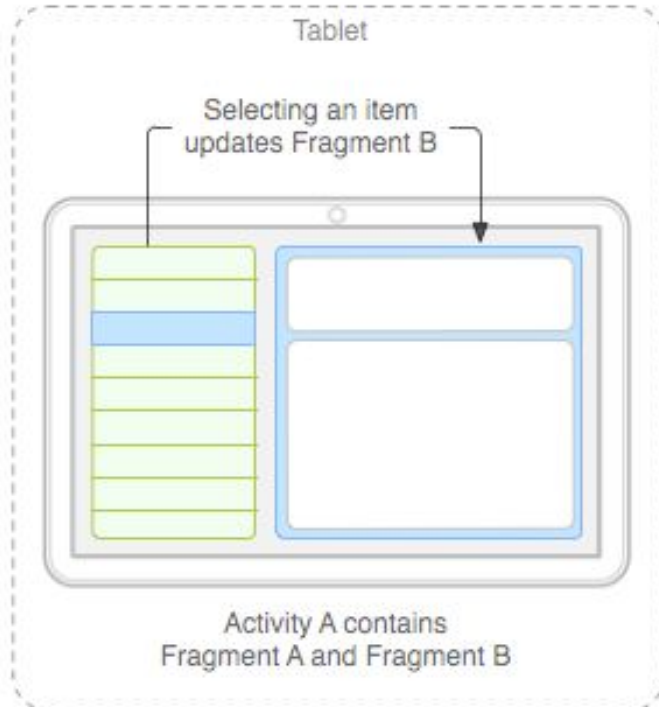


Fragment

- An activity is a container for views
- When you have a larger screen device than a phone –like a tablet it can look too simple to use phone interface here.
- □ Fragments
 - Mini-activities, each with its own set of views
 - One or more fragments can be embedded in an Activity
 - You can do this dynamically as a function of the device type (tablet or not) or orientation

Fragment Idea

- Fragments
 - Mini-activities, each with its own set of views
 - One or more fragments can be embedded in an Activity
 - You can do this dynamically as a function of the device type (tablet or not) or orientation



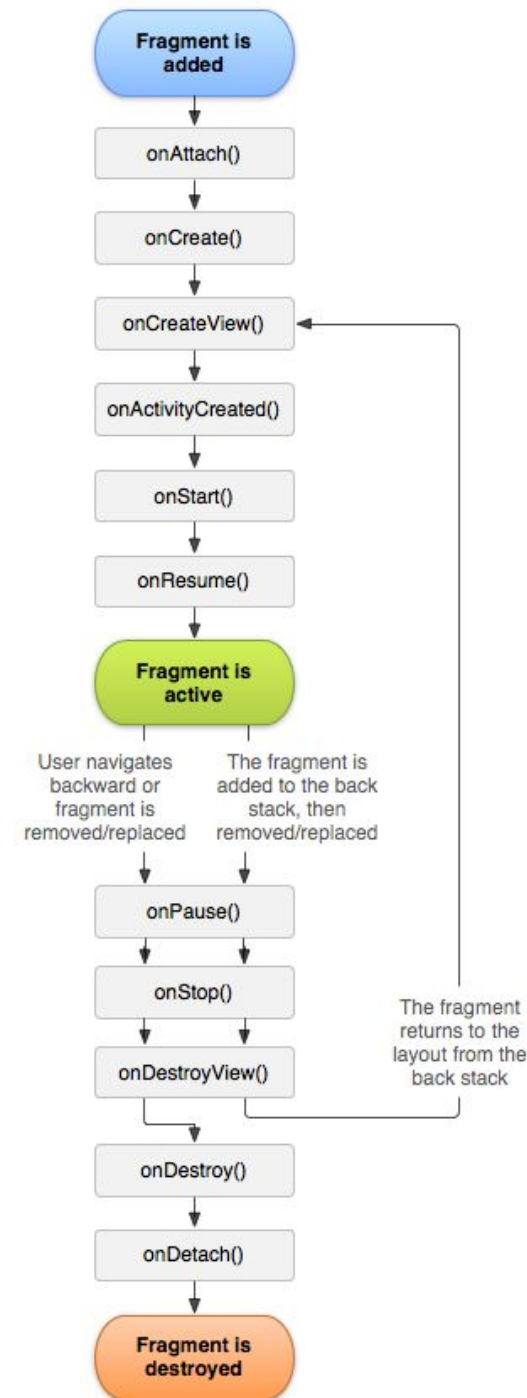
You might decide to run a tablet in portrait mode with the handset model of only one fragment in an Activity

Fragment

- A Fragment represents a behavior or a portion of user interface in an Activity.
- You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities.
- You can think of a fragment as a modular section of an activity, which has its own lifecycle, receives its own input events, and which you can add or remove while the activity is running (sort of like a "sub activity" that you can reuse in different activities).

Fragment Lifecycle

- Fragment in an Activity---Activity Lifecycle influences
 - Activity paused □ all its fragments paused
 - Activity destroyed □ all its fragments paused
 - Activity running □ manipulate each fragment independently.
- Fragment transaction □ add, remove, etc.
 - adds it to a **back stack** that's managed by the activity—each back stack entry in the activity is a record of the fragment transaction that occurred.
 - The back stack allows the user to reverse a fragment transaction (navigate backwards), by pressing the **Back button**.



Fragment inside Activity

- it lives in a ViewGroup inside the activity's view hierarchy
- fragment has its own view layout.
- via XML: Insert a fragment into your activity layout by declaring the fragment in the activity's layout file, as a <fragment> element,
- via CODE: from your application code by adding it to an existing ViewGroup.
- you may also use a fragment without its own UI as an invisible worker for the activity.

Fragment – extend a Fragment class

- ❑ via CODE: extend `android.app.Fragment` OR one of its subclasses ([DialogFragment](#) via CODE: extend `android.app.Fragment` OR one of its subclasses (`DialogFragment`, [ListFragment](#) via CODE: extend `android.app.Fragment` OR one of its subclasses (`DialogFragment`, `ListFragment`, [PreferenceFragment](#) via CODE: extend `android.app.Fragment` OR one of its subclasses (`DialogFragment`, `ListFragment`, `PreferenceFragment`, [WebViewFragment](#))
- ❑ **IMPORTANT:** must include a public empty constructor. The framework will often re-instantiate a fragment class when needed, in

Fragment methods (callback functions)

- onAttach(Activity) called once the fragment is associated with its activity.
- onCreate(Bundle) called to do initial creation of the fragment.
- onCreateView(LayoutInflater, ViewGroup, Bundle) creates and returns the view hierarchy associated with the fragment.
- onActivityCreated(Bundle) onActivityCreated(Bundle) tells the fragment that its activity has completed its own Activity.onCreate.
- onStart() makes the fragment visible to the user (based on its containing activity being started).
- onResume() makes the fragment interacting with the user (based on its containing activity being resumed).

Fragment methods (callback functions)

As a fragment is no longer being used, it goes through a reverse series of callbacks:

- ❑ [onPause\(\)](#) fragment is no longer interacting with the user either because its activity is being paused or a fragment operation is modifying it in the activity.
- ❑ [onStop\(\)](#) fragment is no longer visible to the user either because its activity is being stopped or a fragment operation is modifying it in the activity.
- ❑ [onDestroyView\(\)](#) allows the fragment to clean up resources associated with its View.
- ❑ [onDestroy\(\)](#) called to do final cleanup of the fragment's state.
- ❑ [onDetach\(\)](#) called immediately prior to the fragment no longer being associated with its activity.

Fragments and their UI

- Most fragments will have a UI
- Will have its own layout
- you must implement the onCreateView() you must implement the `onCreateView()` callback method, which the Android system calls when it's time for the fragment to draw its layout. Your implementation of this method must return a View that is the root of your fragment's layout.

Fragments and their UI – onCreateView() using XML

- Can implement onCreateView using XML

```
public static class ExampleFragment extends Fragment {
```

```
    @Override
```


```
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
                           Bundle savedInstanceState) {
```

*Activity parent's
ViewGroup*



A blue arrow points from the *ViewGroup* parameter in the `onCreateView` method signature to a box containing the text *Activity parent's ViewGroup*.

Bundle that provides data about the previous
instance of the fragment, if the fragment is being resumed



A blue arrow points from the `Bundle savedInstanceState` parameter in the `onCreateView` method signature to a box containing the text Bundle that provides data about the previous instance of the fragment, if the fragment is being resumed.

```
    // Inflate the layout for this fragment
```

```
    return inflater.inflate(R.layout.example_fragment, container, false);
```

```
}
```



A blue arrow points from a box containing the text **Have *example_fragment.xml* file that contains the layout
This will be contained in resource layout folder.** to the `R.layout.example_fragment` parameter in the `inflater.inflate` method call.

**Have *example_fragment.xml* file that contains the layout
This will be contained in resource layout folder.**

```
}
```

OPTION1 –adding to an Activity via Activity layout XML.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="com.example.news.ArticleListFragment"
        android:id="@+id/list"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
    <fragment android:name="com.example.news.ArticleReaderFragment"
        android:id="@+id/viewer"
        android:layout_weight="2"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
</LinearLayout>
```

2 fragment classes



Need unique ids for each so system can restore the fragment if the activity is restarted

OPTION2 –creating and adding to an Activity via CODE.

*/*Inside Activity Code where you want to add Fragment (dynamically anywhere or in onCreate() callback)*

**/*

//get FragmentTransaction associated with this Activity

FragmentManager fragmentManager = getFragmentManager();

FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

//Create instance of your Fragment

ExampleFragment fragment = new ExampleFragment();

This points to the Activity ViewGroup in which the fragment should be placed, specified by resource ID

//Add Fragment instance to your Activity

fragmentTransaction.add(R.id.fragment_container, fragment);

fragmentTransaction.commit();

Managing Fragments

FragmentManager methods:

- Get fragments that exist in Activity =
 - findFragmentById() (for fragments that provide a UI in the activity layout)
 - findFragmentByTag() (for fragments that do or don't provide a UI).
- Pop fragments off the back stack,
 - popBackStack() (simulating a *Back* command by the user).
- Register a listener for changes to the back stack,
 - addOnBackStackChangeListener().

Fragment Transactions – adding, removing and replacing dynamically

// Create new fragment and transaction

```
Fragment newFragment = new ExampleFragment();
```

```
FragmentTransaction transaction getFragmentManager().beginTransaction();
```

// Replace whatever is in the fragment_container view with this fragment


// and add the transaction to the back stack

```
transaction.replace(R.id.fragment_container, newFragment);
```


```
transaction.addToBackStack(null);
```

// Commit the transaction

```
transaction.commit();
```



newFragment replaces whatever fragment (if any) is currently in the layout container identified by the **R.id.fragment_container**



If you do not call addToBackStack() when you perform a transaction that removes a fragment, then that fragment is destroyed when the transaction is committed and the user cannot navigate back to it. Whereas, if you do call addToBackStack() then the fragment is not destroyed and the user can navigate back to it.

OPTION 3- Adding Fragment that has NO UI using Code

- use a fragment to provide a background behavior for the activity without presenting additional UI.
- use add(Fragment, String) (supplying a unique string "tag" for the fragment, rather than a view ID).
 - it's not associated with a view in the activity layout, it does not receive a call to onCreateView(). So you don't need to implement that method.
- If you want to get the fragment from the activity later, you need to use findFragmentByTag().

Create your own Fragment class or use known sub-classes

- DialogFragment Displays a floating dialog. Using this class to create a dialog is a good alternative to using the dialog helper methods in the Activity class, because you can incorporate a fragment dialog into the back stack of fragments managed by the activity, allowing the user to return to a dismissed fragment.
- ListFragment Displays a list of items that are managed by an adapter (such as a SimpleCursorAdapter that are managed by an adapter (such as a SimpleCursorAdapter), similar to ListActivity that are managed by an adapter (such as a SimpleCursorAdapter), similar to ListActivity. It provides several methods for managing a list view, such as the onListItemClick() callback to handle click events.
- PreferenceFragment PreferenceFragment Displays a hierarchy of Preference PreferenceFragment Displays a hierarchy of Preference objects as a list similar to PreferenceActivity. This is useful when creating a