Creating a Fragment

THIS LESSON TEACHES YOU TO

1. Create a Fragment Class
2. Add a Fragment to an Activity using XML

YOU SHOULD ALSO READ

- Fragments

TRY IT OUT

Download the sample

FragmentBasics.zip

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). This lesson shows how to extend the `Fragment` (/reference/android/support/v4/app/Fragment.html) class using the Support Library so your app remains compatible with devices running system versions as old as Android 1.6.

> **Note:** If you decide for other reasons that the minimum API level your app requires is 11 or higher, you don't need to use the Support Library and can instead use the framework's built in `Fragment` (/reference/android/app/Fragment.html) class and related APIs. Just be aware that this lesson is focused on using the APIs from the Support Library, which use a specific package signature and sometimes slightly different API names than the versions included in the platform.

Create a Fragment Class

To create a fragment, extend the `Fragment` (/reference/android/support/v4/app/Fragment.html) class, then override key lifecycle methods to insert your app logic, similar to the way you would with an `Activity` (/reference/android/app/Activity.html) class.

One difference when creating a `Fragment` (/reference/android/support/v4/app/Fragment.html) is that you must use the `onCreateView()` (/reference/android/support/v4/app/Fragment.html#onCreateView(android.view.LayoutInflater, android.view.ViewGroup, android.os.Bundle)) callback to define the layout. In fact, this is the only callback you need in order to get a fragment running. For example, here's a simple fragment that specifies its own layout:

```
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.ViewGroup;

public class ArticleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.article_view, container, false);
    }
}
```

Just like an activity, a fragment should implement other lifecycle callbacks that allow you to manage its state as it is added or removed from the activity and as the activity transitions between its lifecycle states. For instance, when the activity's `onPause()` (/reference/android/app/Activity.html#onPause()) method is called, any fragments in the activity also receive a call to `onPause()` (/reference/android/support/v4/app/Fragment.html#onPause()).

More information about the fragment lifecycle and callback methods is available in the Fragments (/guide/components/fragments.html) developer guide.

Add a Fragment to an Activity using XML

While fragments are reusable, modular UI components, each instance of a Fragment (/reference/android/support/v4/app/Fragment.html) class must be associated with a parent FragmentActivity (/reference/android/support/v4/app/FragmentActivity.html). You can achieve this association by defining each fragment within your activity layout XML file.

> **Note:** FragmentActivity (/reference/android/support/v4/app/FragmentActivity.html) is a special activity provided in the Support Library to handle fragments on system versions older than API level 11. If the lowest system version you support is API level 11 or higher, then you can use a regular Activity (/reference/android/app/Activity.html).

Here is an example layout file that adds two fragments to an activity when the device screen is considered "large" (specified by the `large` qualifier in the directory name).

`res/layout-large/news_articles.xml:`

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <fragment android:name="com.example.android.fragments.HeadlinesFragment"
              android:id="@+id/headlines_fragment"
              android:layout_weight="1"
              android:layout_width="0dp"
              android:layout_height="match_parent" />

    <fragment android:name="com.example.android.fragments.ArticleFragment"
              android:id="@+id/article_fragment"
              android:layout_weight="2"
              android:layout_width="0dp"
              android:layout_height="match_parent" />

</LinearLayout>
```

> **Tip:** For more information about creating layouts for different screen sizes, read Supporting Different Screen Sizes (/training/multiscreen/screensizes.html).

Here's how an activity applies this layout:

```
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;

public class MainActivity extends FragmentActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.news_articles);
    }
}
```

> **Note:** When you add a fragment to an activity layout by defining the fragment in the layout XML file, you *cannot* remove the fragment at runtime. If you plan to swap your fragments in and out during user interaction, you must add the fragment to the activity when the activity first starts, as shown in the next lesson.