**Building a Dynamic UI with Fragments**

To create a dynamic and multi-pane user interface on Android, you need to encapsulate UI components and activity behaviors into modules that you can swap into and out of your activities. You can create these modules with the [Fragment](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\app\Fragment.html) class, which behaves somewhat like a nested activity that can define its own layout and manage its own lifecycle.

When a fragment specifies its own layout, it can be configured in different combinations with other fragments inside an activity to modify your layout configuration for different screen sizes (a small screen might show one fragment at a time, but a large screen can show two or more).

This class shows you how to create a dynamic user experience with fragments and optimize your app's user experience for devices with different screen sizes, all while continuing to support devices running versions as old as Android 1.6.

你可以把fragment当作activity的一部分，它拥有自己的生命周期，可接收自己的输入事件，同时还可以动态的添加或删除（在activity运行时）

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](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\Fragment.html) class using the [Support Library](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\tools\support-library\index.html) so your app remains compatible with devices running system versions as low as Android 1.6.

如果你决定你的程序支持的最小版本高于11，你可以不用support library

**Note:** If you decide 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](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\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](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\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](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\app\Activity.html) class.

一点与activity不同，你必须实现onCreateView这以回调方法来定义Fragment的Layout，实际上，这也是唯一一个需要你必须实现的方法。

One difference when creating a [Fragment](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\Fragment.html) is that you must use the [onCreateView()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\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);  
    }  
}

如activity一样，fragment也拥有其他生命周期方法供你调用以管理它的状态（从activity添加或删除），例如当activity的onPause方法执行时，其所托管的所有fragment也会调用自己的onPause生命周期方法。

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()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\app\Activity.html#onPause()) method is called, any fragments in the activity also receive a call to [onPause()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\Fragment.html#onPause()).

More information about the fragment lifecycle and callback methods is available in the [Fragments](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\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](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\Fragment.html) class must be associated with a parent [FragmentActivity](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentActivity.html). You can achieve this association by defining each fragment within your activity layout XML file.

**Note:** [FragmentActivity](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\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](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\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">  
 每一个<fragment>必须有一个唯一的id，因为activity要靠id来管理fragment  
    <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 about creating layouts for different screen sizes, read [Supporting Different Screen Sizes](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\training\multiscreen\screensizes.html).

Then apply the layout to your activity:

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);  
    }  
}

If you're using the [v7 appcompat library](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\tools\support-library\features.html#v7-appcompat), your activity should instead extend [ActionBarActivity](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v7\app\ActionBarActivity.html), which is a subclass of [FragmentActivity](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentActivity.html) (for more information, read [Adding the Action Bar](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\training\basics\actionbar\index.html)).

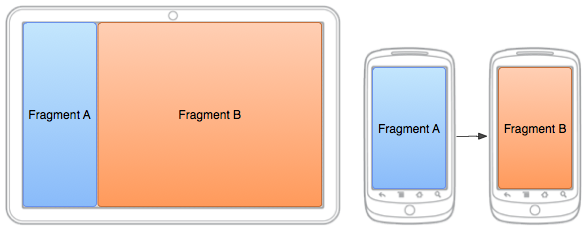
**Note:** When you add a fragment to an activity layout by defining the fragment in the layout XML file, you cannot remove the fragment at runtime. If you plan to swap your fragments in and out during user interaction, you must add the fragment to the activity when the activity first starts, as shown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

# Building a Flexible UI

When designing your application to support a wide range of screen sizes, you can reuse your fragments in different layout configurations to optimize the user experience based on the available screen space.

大屏幕时可能一屏又两个fragment，小屏幕时可能只有一个fragment，可根据不同的情况做不同的配置

For example, on a handset device it might be appropriate to display just one fragment at a time for a single-pane user interface. Conversely, you may want to set fragments side-by-side on a tablet which has a wider screen size to display more information to the user.



## Add a Fragment to an Activity at Runtime

Rather than defining the fragments for an activity in the layout file—as shown in the [previous lesson](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\training\basics\fragments\creating.html) with the <fragment> element—you can add a fragment to the activity during the activity runtime. This is necessary if you plan to change fragments during the life of the activity.

To perform a transaction such as add or remove a fragment, you must use the [FragmentManager](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentManager.html) to create a [FragmentTransaction](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentTransaction.html), which provides APIs to add, remove, replace, and perform other fragment transactions.

**If your activity allows the fragments to be removed and replaced, you should add the initial fragment(s) to the activity during the activity's**[**onCreate()**](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\app\Activity.html#onCreate(android.os.Bundle))**method.**

当你动态添加fragment时，有一个重要的规则---fragment所在的Lauout中必须有一个容器View

An important rule when dealing with fragments—especially those that you add at runtime—is that the fragment must have a container [View](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\view\View.html) in the layout in which the fragment's layout will reside.

The following layout is an alternative to the layout shown in the [previous lesson](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\training\basics\fragments\creating.html) that shows only one fragment at a time. In order to replace one fragment with another, the activity's layout includes an empty [FrameLayout](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\widget\FrameLayout.html) that acts as the fragment container.

名称一样的layout文件，但没有了large的修饰，说明此文件时用于那些小屏幕的设备

Notice that the filename is the same as the layout file in the previous lesson, but the layout directory does not have the large qualifier, so this layout is used when the device screen is smaller than large because the screen does not fit both fragments at the same time.

res/layout/news\_articles.xml:

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@+id/fragment\_container"  
    android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent" />

Inside your activity, call [getSupportFragmentManager()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentActivity.html#getSupportFragmentManager()) to get a [FragmentManager](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentManager.html) using the Support Library APIs. Then call [beginTransaction()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentManager.html#beginTransaction()) to create a [FragmentTransaction](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentTransaction.html) and call [add()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentTransaction.html#add(android.support.v4.app.Fragment, java.lang.String)) to add a fragment.

You can perform multiple fragment transaction for the activity using the same [FragmentTransaction](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentTransaction.html). When you're ready to make the changes, you must call [commit()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentTransaction.html#commit()).

For example, here's how to add a fragment to the previous 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);  
        // Check that the activity is using the layout version with  
        // the fragment\_container FrameLayout  
        if (findViewById(R.id.fragment\_container) != null) {  
            // However, if we're being restored from a previous state,  
            // then we don't need to do anything and should return or else  
            // we could end up with overlapping fragments.  
            if (savedInstanceState != null) {  
                return;  
            }  
            // Create a new Fragment to be placed in the activity layout  
            HeadlinesFragment firstFragment = new HeadlinesFragment();  
              
            // In case this activity was started with special instructions from an  
            // Intent, pass the Intent's extras to the fragment as arguments  
            firstFragment.setArguments(getIntent().getExtras());  
              
            // Add the fragment to the 'fragment\_container' FrameLayout  
            getSupportFragmentManager().beginTransaction()  
                    .add(R.id.fragment\_container, firstFragment).commit();  
        }  
    }  
}

Because the fragment has been added to the [FrameLayout](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\widget\FrameLayout.html) container at runtime—instead of defining it in the activity's layout with a <fragment> element—the activity can remove the fragment and replace it with a different one.

## Replace One Fragment with Another

The procedure to replace a fragment is similar to adding one, but requires the [replace()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentTransaction.html#replace(int, android.support.v4.app.Fragment)) method instead of [add()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentTransaction.html#add(android.support.v4.app.Fragment, java.lang.String)).

Keep in mind that when you perform fragment transactions, such as replace or remove one, it's often appropriate to allow the user to navigate backward and "undo" the change. To allow the user to navigate backward through the fragment transactions, you must call [addToBackStack()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentTransaction.html#addToBackStack(java.lang.String)) before you commit the [FragmentTransaction](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentTransaction.html).

当你remove或replace一个fragment并把它放入回退栈时，被remove的fragment为stop状态，并没有被destory，当用户返回时，fragment就执行restart方法，如果你未将fragment加入回退栈，当你remove或replace时fragment就会被destroy。

**Note:**  you remove or replace a fragment and add the transaction to the back stack, the fragment that is removed is stopped (not destroyed). If the user navigates back to restore the fragment, it restarts. If you do not add the transaction to the back stack, then the fragment is destroyed when removed or replaced.

Example of replacing one fragment with another:

// Create fragment and give it an argument specifying the article it should show  
ArticleFragment newFragment = new ArticleFragment();  
Bundle args = new Bundle();  
args.putInt(ArticleFragment.ARG\_POSITION, position);  
newFragment.setArguments(args);  
  
FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();  
  
// Replace whatever is in the fragment\_container view with this fragment,  
// and add the transaction to the back stack so the user can navigate back  
transaction.replace(R.id.fragment\_container, newFragment);  
transaction.addToBackStack(null);  
  
// Commit the transaction  
transaction.commit();

The [addToBackStack()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentTransaction.html#addToBackStack(java.lang.String)) method takes an optional string parameter that specifies a unique name for the transaction. The name isn't needed unless you plan to perform advanced fragment operations using the[FragmentManager.BackStackEntry](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentManager.BackStackEntry.html) APIs.

# Communicating with Other Fragments

In order to reuse the Fragment UI components, you should build each as a completely self-contained（独立的）, modular（模块化的） component that defines its own layout and behavior. Once you have defined these reusable Fragments, you can associate them with an Activity and connect them with the application logic to realize the overall composite UI.

Often you will want one Fragment to communicate with another, for example to change the content based on a user event.

所有题fragment之间的通信都是通过所托管的activity完成的，两个fragment之间永远无法直接通信

All Fragment-to-Fragment communication is done through the associated Activity. Two Fragments should never communicate directly.

## Define an Interface

To allow a Fragment to communicate up to its Activity, you can define an interface in the Fragment class and implement it within the Activity. The Fragment captures the interface implementation during its onAttach() lifecycle method and can then call the Interface methods in order to communicate with the Activity.

Here is an example of Fragment to Activity communication:

public class HeadlinesFragment extends ListFragment {  
    OnHeadlineSelectedListener mCallback;  
  
    // Container Activity must implement this interface  
    public interface OnHeadlineSelectedListener {  
        public void onArticleSelected(int position);  
    }  
  
    @Override  
    public void onAttach(Activity activity) {  
        super.onAttach(activity);  
          
        // This makes sure that the container activity has implemented  
        // the callback interface. If not, it throws an exception  
        try {  
            mCallback = (OnHeadlineSelectedListener) activity;  
        } catch (ClassCastException e) {  
            throw new ClassCastException(activity.toString()  
                    + " must implement OnHeadlineSelectedListener");  
        }  
    }  
      
    ...  
}

Now the fragment can deliver messages to the activity by calling the onArticleSelected() method (or other methods in the interface) using the mCallback instance of the OnHeadlineSelectedListenerinterface.

For example, the following method in the fragment is called when the user clicks on a list item. The fragment uses the callback interface to deliver the event to the parent activity.

    @Override  
    public void onListItemClick(ListView l, View v, int position, long id) {  
        // Send the event to the host activity  
        mCallback.onArticleSelected(position);  
    }

## Implement the Interface

In order to receive event callbacks from the fragment, the activity that hosts it must implement the interface defined in the fragment class.

For example, the following activity implements the interface from the above example.

public static class MainActivity extends Activity  
        implements HeadlinesFragment.OnHeadlineSelectedListener{  
    ...  
      
    public void onArticleSelected(int position) {  
        // The user selected the headline of an article from the HeadlinesFragment  
        // Do something here to display that article  
    }  
}

## Deliver a Message to a Fragment

The host activity can deliver messages to a fragment by capturing the [Fragment](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\Fragment.html) instance with[findFragmentById()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\support\v4\app\FragmentManager.html#findFragmentById(int)), then directly call the fragment's public methods.

For instance, imagine that the activity shown above may contain another fragment that's used to display the item specified by the data returned in the above callback method. In this case, the activity can pass the information received in the callback method to the other fragment that will display the item:

public static class MainActivity extends Activity  
        implements HeadlinesFragment.OnHeadlineSelectedListener{  
    ...  
    public void onArticleSelected(int position) {  
        // The user selected the headline of an article from the HeadlinesFragment  
        // Do something here to display that article  
        ArticleFragment articleFrag = (ArticleFragment)  
                getSupportFragmentManager().findFragmentById(R.id.article\_fragment);  
        if (articleFrag != null) {  
            // If article frag is available, we're in two-pane layout...  
            // Call a method in the ArticleFragment to update its content  
            articleFrag.updateArticleView(position);  
        } else {  
            // Otherwise, we're in the one-pane layout and must swap frags...  
            // Create fragment and give it an argument for the selected article  
            ArticleFragment newFragment = new ArticleFragment();  
            Bundle args = new Bundle();  
            args.putInt(ArticleFragment.ARG\_POSITION, position);  
            newFragment.setArguments(args);  
            FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();  
  
            // Replace whatever is in the fragment\_container view with this fragment,  
            // and add the transaction to the back stack so the user can navigate back  
            transaction.replace(R.id.fragment\_container, newFragment);  
            transaction.addToBackStack(null);  
            // Commit the transaction  
            transaction.commit();  
        }  
    }  
}