Fragments

Evolve Advanced Track, Chapter 4

Xamarin Inc.

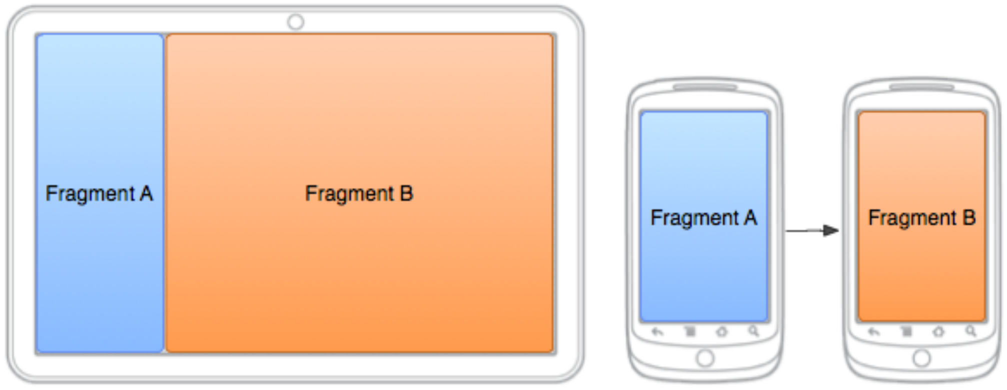
Overview

The larger screen sizes found on most tablets added an extra layer of complexity to Android development—a layout designed for the small screen does not necessarily work as well for larger screens, and vice-versa. In order to reduce the number of complications that this introduced, Android 3.0 added two new features, *Fragments* and the *Android Support Packages*.

Fragments can be thought of as user interface modules. They let the developer divide up the user interface into isolated, reusable parts that can be run in separate Activities. At run time, the Activities themselves will decide which Fragments to use.

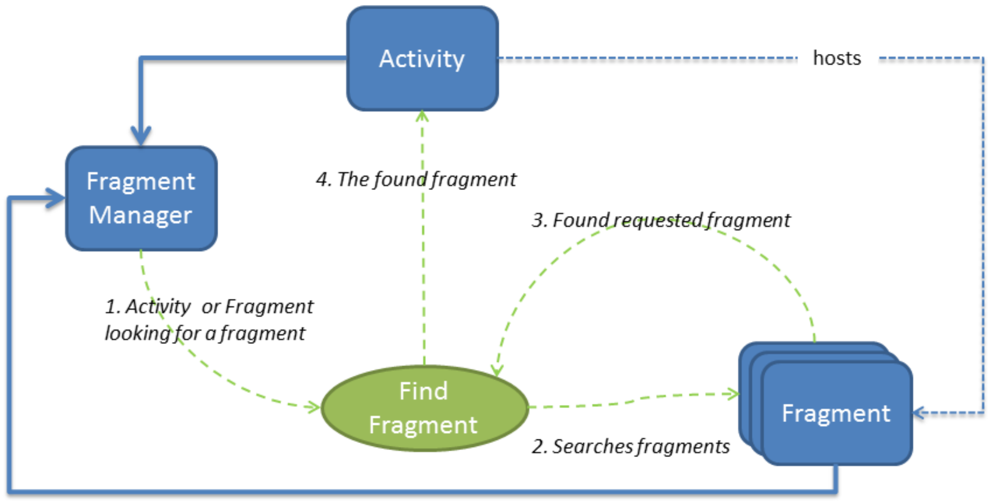
The Android Support Packages were originally called the *Android Compatibility Libraries* and back ported some of the new APIs, such as Fragments to older versions of Android (API Level 10 and below).

To help understand Fragments, consider the image below:



Fragment A contains a list, while *Fragment B* contains details for an item selected in that list. When the application is run on a tablet, it can display both Fragments on the same Activity. When the same application is run on a handset (with its smaller screen size), the Fragments are hosted in two separate Activities. Fragment A and Fragment B are the same on both form factors, but the Activities that host them are different.

To help an Activity coordinate and manage all these Fragments, Android introduced a new class called the *FragmentManager*. Each Activity has its own instance of a `FragmentManager` for adding, deleting, and finding hosted Fragments. The following diagram illustrates the relationship between Fragments and Activities:



In some regards, Fragments can be thought of as composite controls or as mini-Activities. They bundle up pieces of UI into reusable modules that can then be used independently by developers in Activities. A Fragment does have a view hierarchy—just like an Activity—but, unlike an Activity, it can be shared across screens. Views differ from Fragments in that Fragments have their own lifecycle; views do not.

While the Activity is a host to one or more Fragments, it is not directly aware of the Fragments themselves. Likewise, Fragments are not directly aware of other Fragments in the hosting Activity. However, Fragments and Activities are aware of the `FragmentManager` in their Activity. By using the `FragmentManager`, it is possible for an Activity or a Fragment to obtain a reference to a specific instance of a Fragment, and then call methods on that instance. In this way, the Activity or Fragments can communicate and interact with other Fragments.

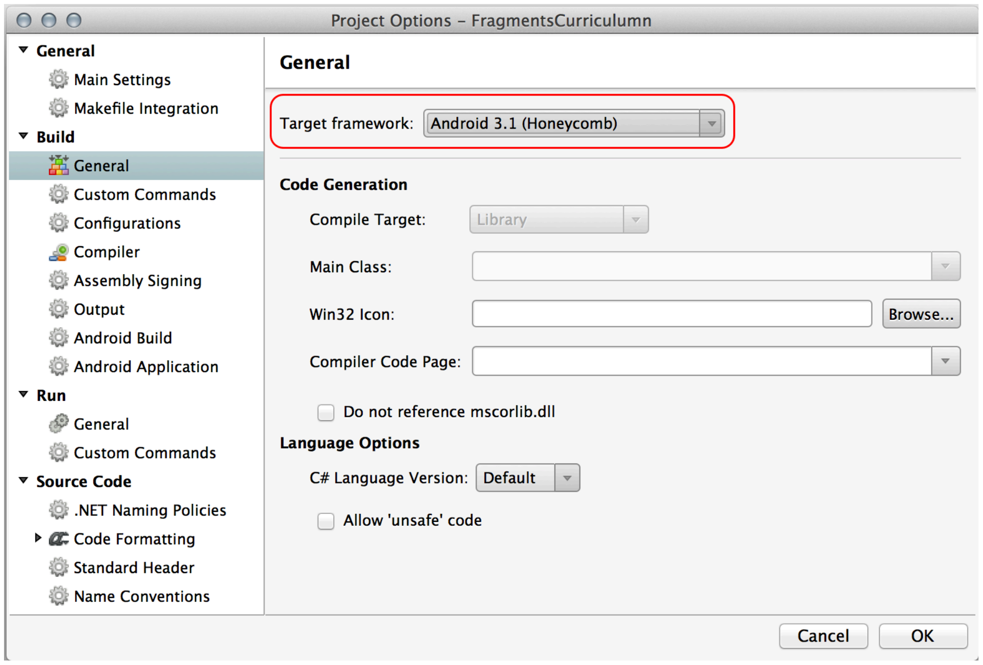
This chapter is an introduction to using Fragments, including:

- Creating Fragments – How to create a basic Fragment and key methods that must be implemented.
- Fragment Management and Transactions – How to manipulate Fragments at run time.
- Android Support Package – How to use the libraries that allow Fragments to be used on older versions of Android.

For a more in depth look at fragments, refer to Xamarin's documentation on Fragments and the Fragments Walkthrough.

Requirements

An Android application must target at least API level 11 (Android 3.0) or higher in order to use Fragments. The Target Framework may be set in the Project Options as shown below:



Target framework: Android 3.1 (Honeycomb)

Code Generation

Compile Target: Library

Main Class:

Win32 Icon: Browse...

Compiler Code Page:

Do not reference mscorlib.dll

Language Options

C# Language Version: Default

Allow 'unsafe' code

It is possible to use Fragments in older versions of Android by using the Android Support Package. How to do this will be covered in more detail further on in this document.

Using Fragments

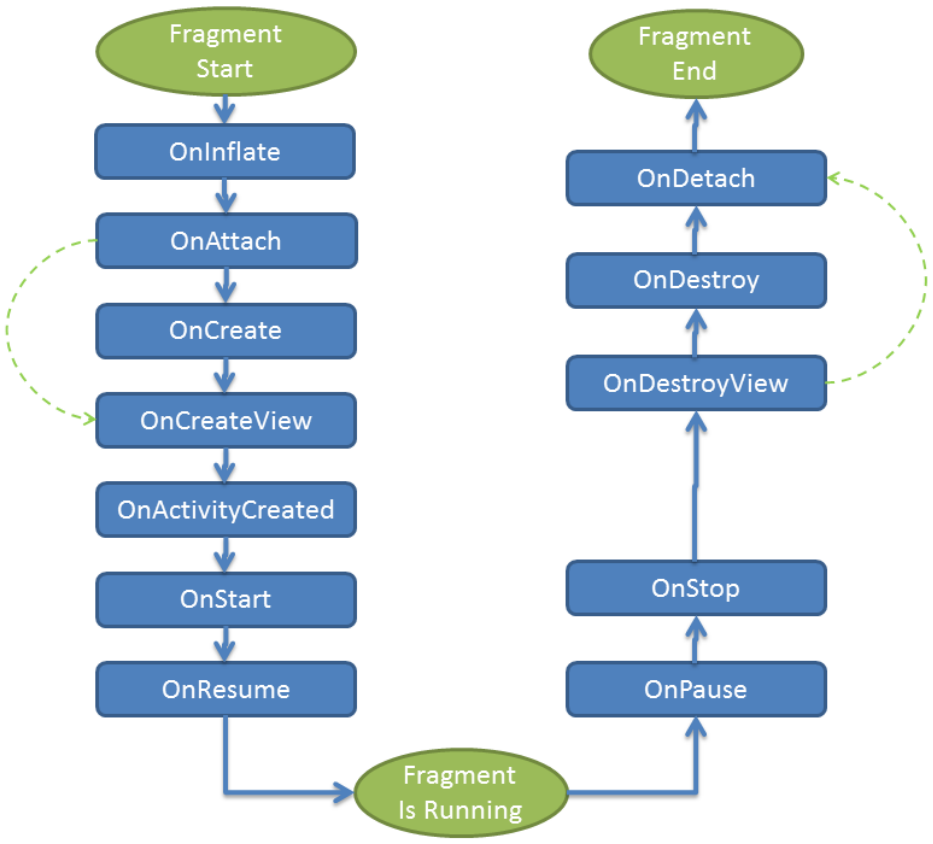
To create a Fragment, a class must inherit from `Android.App.Fragment` and then override the `OnCreateView` method. `OnCreateView` will be called by the hosting Activity when it is time to put the Fragment on the screen, and will return a `View`. The following code is one example of creating the view for a fragment:

```
public override View OnCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState)
{
    return inflater.Inflate(Resource.Layout.Example_Fragment, container,
false);
}
```

In the example above, a layout file is inflated and then added as a child to the ViewGroup specified by the parameter `container`.

Fragment Lifecycle

Fragments have their own lifecycle that is somewhat independent of, but still affected by, the lifecycle of the hosting Activity. For example, when an Activity pauses, all of its associated Fragments are paused. The following diagram outlines the lifecycle of the Fragment.



The table below shows the flow of the various callbacks in the lifecycle of a Fragment as it is being created:

Lifecycle Method	Activity State
`OnInflate()`	Called when the Fragment is being created as part of a view layout. This may be called immediately after the Fragment is created declaratively from an XML layout file. The Fragment is not associated with its Activity yet, but the `Activity`, `Bundle`, and `AttributeSet` from the view hierarchy are passed in as parameters. This method is best used for parsing the `AttributeSet` and for saving the attributes that might be used later by the Fragment.
`OnAttach()`	Called after the Fragment is associated with the Activity. This is the first method to be run when the Fragment is ready to be used. In general, Fragments should not implement a constructor or override the default constructor. Any components that are required for the Fragment should be initialized in this method.

`OnCreate()`	Called by the Activity to create the Fragment. When this method is called, the view hierarchy of the hosting Activity may not be completely instantiated, so the Fragment should not rely on any parts of the Activity's view hierarchy until later on in the Fragment's lifecycle. For example, do not use this method to perform any tweaks or adjustments to the UI of the application. This is the earliest time at which the Fragment may begin gathering the data that it needs. The Fragment is running in the UI thread at this point, so avoid any lengthy processing, or perform that processing on a background thread. This method may be skipped if `SetRetainInstance(true)` is called. This alternative will be described in more detail below.
`OnCreateView()`	Creates the view for the Fragment. This method is called once the Activity's `OnCreate()` method is complete. At this point, it is safe to interact with the view hierarchy of the Activity. This method should return the view that will be used by the Fragment.
`OnActivityCreated()`	Called after `Activity.OnCreate` has been completed by the hosting Activity. Final tweaks to the user interface should be performed at this time.
`OnStart()`	Called after the containing Activity has been resumed. This makes the Fragment visible to the user. In many cases, the Fragment will contain code that would otherwise be in the `OnStart()` method of an Activity.
`OnResume()`	This is the last method called before the user can interact with the Fragment. An example of the kind of code that should be performed in this method would be enabling features of a device that the user may interact with, such as the camera that the location services. Services such as these can cause excessive battery drain, though, and an application should minimize their use in order to preserve battery life.

The next table shows the lifecycle methods that are called as a Fragment is being destroyed:

Lifecycle Method	Activity State
`OnPause()`	The user is no longer able to interact with the Fragment. This situation exists because some other Fragment operation is modifying this Fragment, or the hosting Activity is paused. It is possible that the Activity hosting this Fragment might still be visible, that is, the Activity in focus is partially transparent or does not occupy the full screen. When this method is called, it's the first indication that the user is leaving the Fragment. The Fragment should save any changes.
`OnStop()`	The Fragment is no longer visible. The host Activity may be stopped, or a Fragment operation is modifying it in the Activity. This callback serves the same purpose as `Activity.OnStop`.
`OnDestroyView()`	This method is called to clean up resources associated with the view. This is called when the view associated with the Fragment has been destroyed.
`OnDestroy()`	This method is called when the Fragment is no longer in use. It is still associated with the Activity, but the Fragment is no longer functional. This method should release any resources that are in use by the Fragment, such as a SurfaceView that might be used for a camera. This method may be skipped if `SetRetainInstance(true)` is called. This alternative will be described in more detail below.
`OnDetach()`	This method is called just before the Fragment is no longer associated with the Activity. The view hierarchy of the Fragment no longer exists, and all resources that are used by the Fragment should be released at this point.

Adding a Fragment to an Activity

There are two ways that a Fragment may be hosted inside an Activity:

- **Declaratively** – Fragments can be used declaratively within an `.axml` layout files by using the `<Fragment>` tag.

- **Programmatically** – Fragments can also be instantiated dynamically and then added to a fragment by using the `FragmentManager` class's API.

Programmatic usage via the `FragmentManager` class will be discussed later in this chapter.

Using a Fragment Declaratively

Adding a Fragment inside the layout requires using the `<fragment>` tag and then identifying the Fragment by providing either the `class` attribute or the `android:name` attribute. The following snippet shows how to use the `class` attribute to declare a `fragment`:

```
<?xml version="1.0" encoding="utf-8"?>
<fragment class="com.xamarin.sample.fragments.TitlesFragment"
            android:id="@+id/titles_fragment"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent" />
```

This next snippet shows how to declare a `fragment` by using the `android:name` attribute to identify the Fragment class :

```
<?xml version="1.0" encoding="utf-8"?>
<fragment android:name="com.xamarin.sample.fragments.TitlesFragment"
            android:id="@+id/titles_fragment"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent" />
```

When the Activity is being created, Android will instantiate each Fragment specified in the layout file. The Activity will then call `OnCreateView` on each Fragment and display the View that the Fragment has created.

Fragments that are declaratively added to an Activity are static and will remain on the Activity until it is destroyed; it is not possible to dynamically replace or remove such a Fragment during the lifetime of the Activity to which it is attached.

Each Fragment must be assigned a unique identifier:

- **android:id** – As with other UI elements in a layout file, this is a unique ID.
- **android:tag** – This attribute is a unique string.

If neither of the previous two methods is used, then the Fragment will assume the ID of the container view. In the following example where neither `android:id` nor `android:tag` is provided, Android will assign the ID `fragment_container` to the Fragment:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
               android:id="+@id/fragment_container"
               android:orientation="horizontal"
               android:layout_width="match_parent"
               android:layout_height="match_parent">

        <fragment class="com.example.android.apis.app.TitlesFragment"
               android:layout_width="match_parent"
               android:layout_height="match_parent" />
</LinearLayout>
```

Note: Android does not allow for uppercase characters in package names; it will throw an exception when trying to inflate the view if a package name contains an uppercase character. However, Xamarin.Android is more forgiving, and will tolerate uppercase characters in the namespace.

For example, both of the following snippets will work with Xamarin.Android. However, the second snippet will cause an `android.view.InflateException` to be thrown by a pure Java-based Android application.

```
<fragment                        class="com.example.DetailsFragment"
android:id="@+id/fragment_content"
android:layout_width="match_parent"
android:layout_height="match_parent" />
```

OR

```
<fragment                        class="Com.Example.DetailsFragment"
android:id="@+id/fragment_content"
android:layout_width="match_parent"
android:layout_height="match_parent" />
```

Managing Fragments

To help with managing Fragments and to programmatically add Fragments, Android provides the `FragmentManager` class. Each Activity has an instance of `Android.App.FragmentManager` that will find or dynamically change its Fragments. Each set of these changes is known as a *transaction*, and is performed by using one of the APIs contained in the class `Android.App.FragmentTransation`, which is managed by the `FragmentManager`. An Activity may start a transaction like this:

```
FragmentTransaction fragmentTx = this.FragmentManager.BeginTransaction();
```

These changes to the Fragments are performed in the `FragmentTransaction` instance by using methods such as `Add()`, `Remove()`, and `Replace()`. The changes are then applied by using `Commit()`. The changes in a transaction are not performed immediately. Instead, they are scheduled to run on the Activity's UI thread as soon as possible.

The following example shows how to add a Fragment to an existing container:

```
 // Create a new fragment and a transaction.
FragmentTransaction fragmentTx = this.FragmentManager.BeginTransaction();
DetailsFragment aDifferentDetailsFrag = new DetailsFragment();

// The fragment will have the ID of Resource.Id.fragment_container.
fragmentTx.Add(Resource.Id.fragment_container, aDifferentDetailsFrag);

// Commit the transaction.
fragmentTx.Commit();
```

It's possible to save the Fragment transactions to the Activity's back stack by making a call to `FragmentTransaction.AddToBackStack()`. This allows the user to navigate backwards through Fragment changes when the **Back** button is pressed. Without a call to this method, Fragments that are removed will be destroyed and will be unavailable if the user navigates back through the Activity.

The following example shows how to use the `AddToBackStack` method of a `FragmentTransaction` to replace one Fragment, while preserving the state of the first Fragment on the back stack:

```
// Create a new fragment and a transaction.
FragmentTransaction fragmentTx = this.FragmentManager.BeginTransaction();
DetailsFragment aDifferentDetailsFrag = new DetailsFragment();

// Replace the fragment that is in the View fragment_container (if
applicable).
fragmentTx.Replace(Resource.Id.fragment_container, aDifferentDetailsFrag);

// Add the transaction to the back stack.
fragmentTx.AddToBackStack(null);

// Commit the transaction.
fragmentTx.Commit();
```

Communicating with Fragments

The *FragmentManager* provides two methods to help find specific instances of Fragments:

- FindFragmentById – This method will find a Fragment by using the ID that was specified in the layout file or the container ID when the Fragment was added as part of a transaction.
- FindFragmentByTag – This method is used to find a Fragment that has a tag that was provided in the layout file or that was added in a transaction.

Both Fragments and Activities reference the `FragmentManager`, so the same techniques are used to communicate back and forth between them. An application may find a reference Fragment by using one of these two methods, cast that reference to the appropriate type, and then directly call methods on the Fragment. The following snippet provides an example:

It is also possible for the Activity to use the `FragmentManager` to find Fragments:

```
var emailList =
FragmentManager.FindFragmentById<EmailListFragment>(Resource.Id.email_list
_fragment);
emailList.SomeCustomMethod(parameter1, parameter2);
```

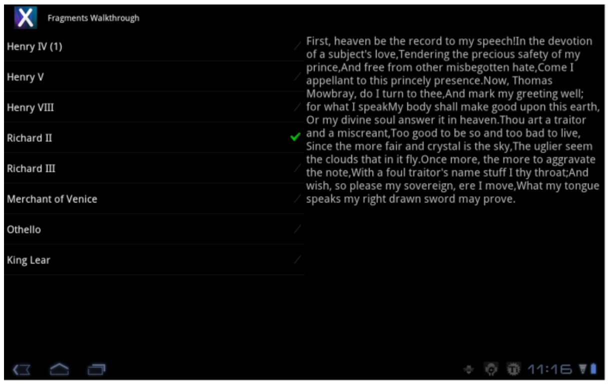
Communicating with the Activity

It is possible for a Fragment to use the `Fragment.Activity` property to reference its host. By casting the Activity to a more specific type, it is possible for an Activity to call methods and properties on its host, as shown in the following example:

```
var myActivity = (MyActivity) this.Activity;
myActivity.SomeCustomMethod();
```

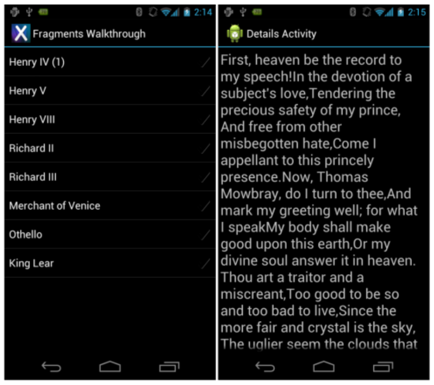
Displaying Lists in a Fragment

The `ListFragment` is a specialized subclass of the `Fragment` class. It is very similar in concept and functionality to the `ListActivity`; it is a wrapper that hosts a ListView in a Fragment. The image below shows a `ListFragment` running on a tablet and a phone:



First, heaven be the record to my speech!In the devotion of a subject's love,Tendering the precious safety of my prince,And free from other misbegotten hate,Come I appellant to this princely presence.Now, Thomas Mowbray, do I turn to thee,And mark my greeting well; for what I speakMy body shall make good upon this earth,Or my divine soul answer it in heaven.Thou art a traitor and a miscreant,Too good to be so and too bad to live,Since the more fair and crystal is the sky,The uglier seem the clouds that in it fly.Once more, the more to aggravate the note,With a foul traitor's name stuff I thy throat;And wish, so please my sovereign, ere I move,What my tongue speaks my right drawn sword may prove.

Tablet



First, heaven be the record to my speech!In the devotion of a subject's love,Tendering the precious safety of my prince,And free from other misbegotten hate,Come I appellant to this princely presence.Now, Thomas Mowbray, do I turn to thee,And mark my greeting well; for what I speakMy body shall make good upon this earth,Or my divine soul answer it in heaven.Thou art a traitor and a miscreant,Too good to be so and too bad to live,Since the more fair and crystal is the sky,The uglier seem the clouds that

Phone

Binding Data With The ListAdapter

The `ListFragment` class already provides a default layout, so it is not necessary to override `OnCreateView` to display the contents of the `ListFragment`. The `ListView` is bound to data by using a `ListAdapter` implementation. The following example shows how using a simple array of strings could do this:

```
public override void OnActivityCreated(Bundle savedInstanceState)
{
    base.OnActivityCreated(savedInstanceState);
    string[] values = new[] { "Android", "iPhone", "WindowsMobile",
                "Blackberry", "WebOS", "Ubuntu", "Windows7", "Max OS X",
                "Linux", "OS/2" };
    this.ListAdapter = new ArrayAdapter<string>(Activity,
Android.Resource.Layout.SimpleExpandableListItem1, values);
}
```

When setting the `ListAdapter`, it is important to use the `ListFragment.ListAdapter` property, and not the `ListView.ListAdapter` property. Using `ListView.ListAdapter` will cause important initialization code to be skipped.

Responding to User Selection

To respond to user selections, an application must override the `OnListItemClick` method. The following example shows one such possibility:

```
 public override void OnListItemClick(ListView l, View v, int index, long
id)
{
    // We can display everything in place with fragments.
    // Have the list highlight this item and show the data.
    ListView.SetItemChecked(index, true);

    // Check what fragment is shown, replace if needed.
    var details =
FragmentManager.FindFragmentById<DetailsFragment>(Resource.Id.details);
    if (details == null || details.ShownIndex != index)
    {
        // Make new fragment to show this selection.
        details = DetailsFragment.NewInstance(index);

        // Execute a transaction, replacing any existing
        // fragment with this one inside the frame.
        var ft = FragmentManager.BeginTransaction();
        ft.Replace(Resource.Id.details, details);
        ft.SetTransition(FragmentTransit.FragmentFade);
        ft.Commit();
    }
}
```

In the code above, when the user selects an item in the `ListFragment`, a new Fragment is displayed in the hosting Activity, showing more details about the item that was selected.

Providing Backwards Compatibility with the Android Support Package

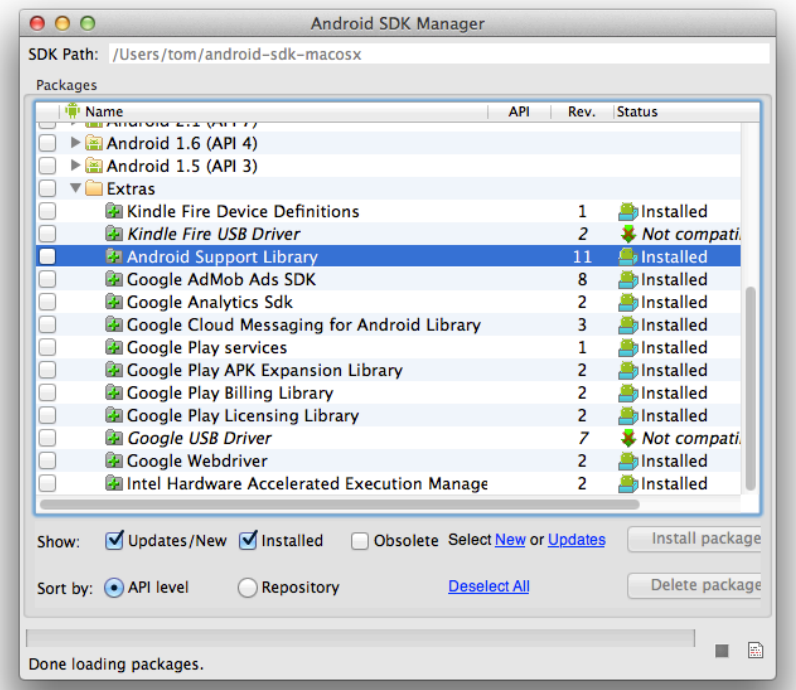
The usefulness of Fragments would be limited without backwards compatibility with pre-Android 3.0 (API Level 11) devices. To provide this capability, Google introduced the Android Support Package (originally called the *Android Compatibility Library* when it was released) which back ports some of the APIs from newer versions of Android to older versions of Android. It is the Android Support Package that enables devices running Android 1.6 (API level 4) to Android 2.3.3. (API level 10).

Note: Not all fragment subclasses are available through the Android Support Package. For more details refer to Xamarin's documentation on Fragments and the Fragments Walkthrough.

Adding the Support Package

The Android Support Package is not automatically added to a Xamarin.Android application. In order to add this support package, the following steps must be performed:

1. Download the Android Support pack from the Android SDK Manager, under the Extras menu, as shown in the following screenshot:



Android SDK Manager

SDK Path: /Users/tom/android-sdk-macosx

Packages

Name	API	Rev.	Status
Android 1.6 (API 4)			
Android 1.5 (API 3)			
Extras			
Kindle Fire Device Definitions		1	Installed
Kindle Fire USB Driver		*2*	*Not compati*
Android Support Library		11	Installed
Google AdMob Ads SDK		8	Installed
Google Analytics Sdk		2	Installed
Google Cloud Messaging for Android Library		3	Installed
Google Play services		1	Installed
Google Play APK Expansion Library		2	Installed
Google Play Billing Library		2	Installed
Google Play Licensing Library		2	Installed
Google USB Driver		*7*	*Not compati*
Google Webdriver		2	Installed
Intel Hardware Accelerated Execution Manage		2	Installed

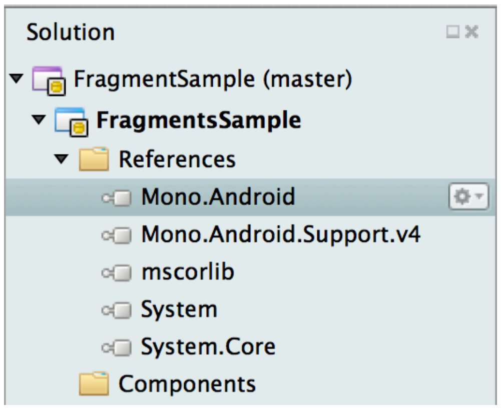
Show: [✓ Updates/New] [✓ Installed] Obsolete Select New or Updates

Sort by: [● API level] Repository Deselect All

Done loading packages.

The necessary libraries may be found as in the Android SDK directory `<sdk>/extras/android/support`.

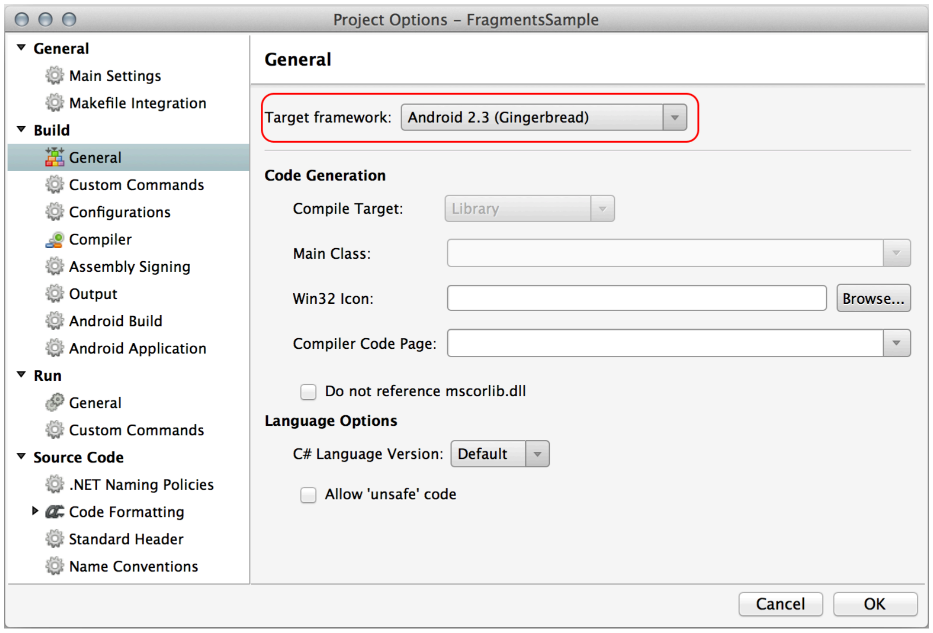
2. The next step is to add a reference to the `Mono.Android.Support.v4` assembly, as shown in the following screenshot:



```
Solution
FragmentSample (master)
  FragmentsSample
    References
      Mono.Android
      Mono.Android.Support.v4
      mscorlib
      System
      System.Core
    Components
```

After these steps have been performed, it becomes possible to use Fragments in earlier versions of Android. The Fragment APIs will work the same now in these earlier versions, with the following exceptions:

- Change the Target Android Version – The application no longer needs to target Android 3.0 or higher. Instead, applications may target an older version of Android, as shown below:


Project Options – FragmentsSample

General

Target framework: Android 2.3 (Gingerbread)

Code Generation

Compile Target: Library

Main Class:

Win32 Icon:

Compiler Code Page:

☐ Do not reference mscorlib.dll

Language Options

C# Language Version: Default

☐ Allow 'unsafe' code

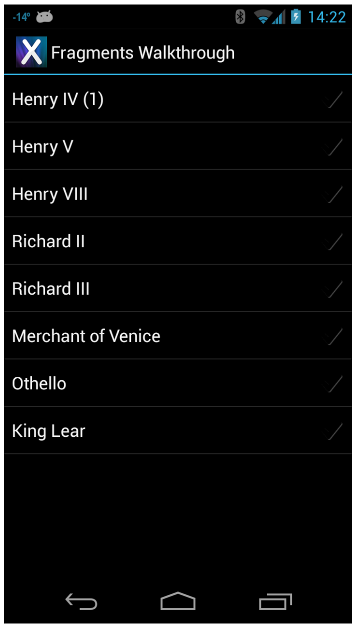
- Extend FragmentActivity – The Activities that are hosting Fragments must now inherit from `Android.App.FragmentActivity`, and not from `Android.App.Activity`.
- Update Namespaces – Classes that inherit from `Android.App.Fragment` must now inherit from `Android.Support.v4.App.Fragment`. Remove the using statement "`using Android.App;`" at the top of the source code file and replace it with "`using Android.Support.v4.App`".
- Use SupportFragmentManager – `Android.App.FragmentActivity` exposes a `SupportingFragmentManager` property that must be used to get a reference to the `FragmentManager`. For example:

```
FragmentTransaction fragmentTx =
this.SupportingFragmentManager.BeginTransaction();
DetailsFragment detailsFrag = new DetailsFragment();
fragmentTx.Add(Resource.Id.fragment_container, detailsFrag);
fragmentTx.Commit();
```

With these changes in place, it will be possible to run a Fragment-based application on older versions of Android such as Gingerbread.

Walkthrough

Now that you understand the theory behind fragments it is time to do a practical walk through of creating a fragments. The application in this walkthrough will present a list of Shakespearean plays. When the user selects a play, the application will display a passage from the selected play. The application will look different depending on the device it is running on. On a phone, there will be one screen that display a list of Shakespeare's plays:


14:22

Fragments Walkthrough

Henry IV (1)

Henry V

Henry VIII

Richard II

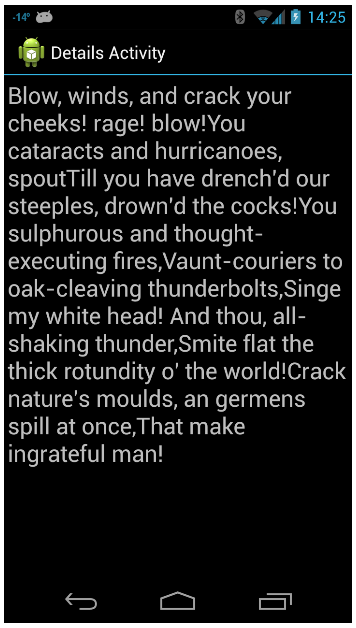
Richard III

Merchant of Venice

Othello

King Lear

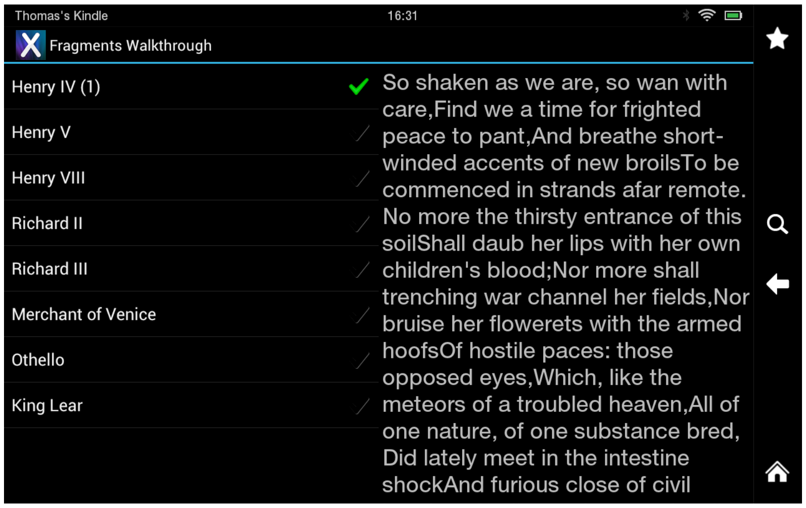
And another list that will display a quote from selected play:


14:25

Details Activity

Blow, winds, and crack your cheeks! rage! blow!You cataracts and hurricanoes, spoutTill you have drench'd our steeples, drown'd the cocks!You sulphurous and thought-executing fires,Vaunt-couriers to oak-cleaving thunderbolts,Singe my white head! And thou, all-shaking thunder,Smite flat the thick rotundity o' the world!Crack nature's moulds, an germens spill at once,That make ingrateful man!

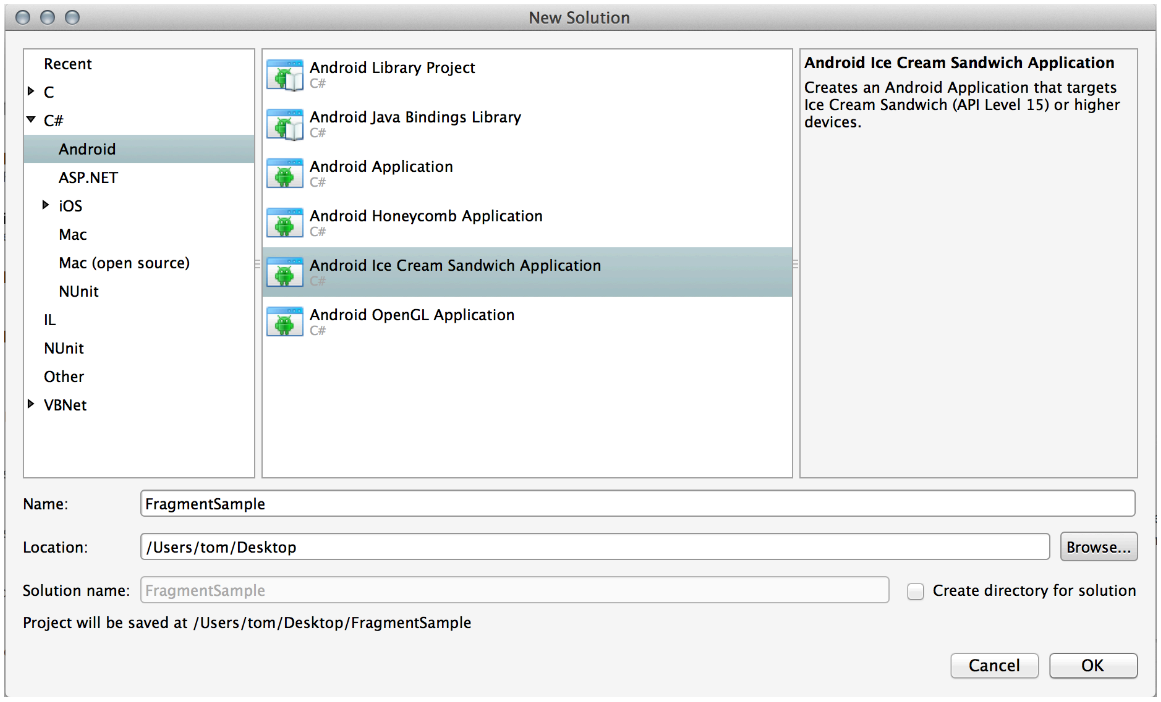
In the case of a tablet we want the application to display a list of plays on the left hand side. When the user selects a play, the quote should be displayed on the right hand side of the screen, as shown in the following screenshot:



So shaken as we are, so wan with care,Find we a time for frighted peace to pant,And breathe short-winded accents of new broilsTo be commenced in strands afar remote. No more the thirsty entrance of this soilShall daub her lips with her own children's blood;Nor more shall trenching war channel her fields,Nor bruise her flowerets with the armed hoofsOf hostile paces: those opposed eyes,Which, like the meteors of a troubled heaven,All of one nature, of one substance bred, Did lately meet in the intestine shockAnd furious close of civil

With this in mind, lets get started on the walkthrough.

1. Start by creating a new **Android Ice Cream Sandwich Application** in Xamarin Studio named `FragmentSample`, as shown in the following screenshot:



Android Ice Cream Sandwich Application
Creates an Android Application that targets Ice Cream Sandwich (API Level 15) or higher devices.

Name: FragmentSample
Location: /Users/tom/Desktop
Solution name: FragmentSample
Create directory for solution
Project will be saved at /Users/tom/Desktop/FragmentSample

2. Next, we need to create the `MainActivity` class. This is the startup Activity for the application. When the Xamarin.Android project is created, the startup Activity is called `Activity1`. Rename the file and class to `MainActivity.cs` and `MainActivity`, respectively.

3. Next, edit `MainActivity` so that it resembles the following code:

```
[Activity(Label = "Fragments Walkthrough", MainLauncher = true)]
public class MainActivity : Activity
{
    protected override void OnCreate(Bundle bundle)
```

```
        {
            base.OnCreate(bundle);
            SetContentView(Resource.Layout.Main);
        }
    }
```

4. Edit the file `Resources/layout/Main.axml` so that it contains the following 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 class="fragmentsample.TitlesFragment"
            android:id="@+id/titles_fragment"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent" />
</LinearLayout>
```

By default, this layout file will be loaded by all devices. This file will add a fragment called `TitlesFragment` to the layout. `TitlesFragment` will display a list of plays from Shakespeare. We will create this fragment shortly – for now let's focus on creating our layout files.

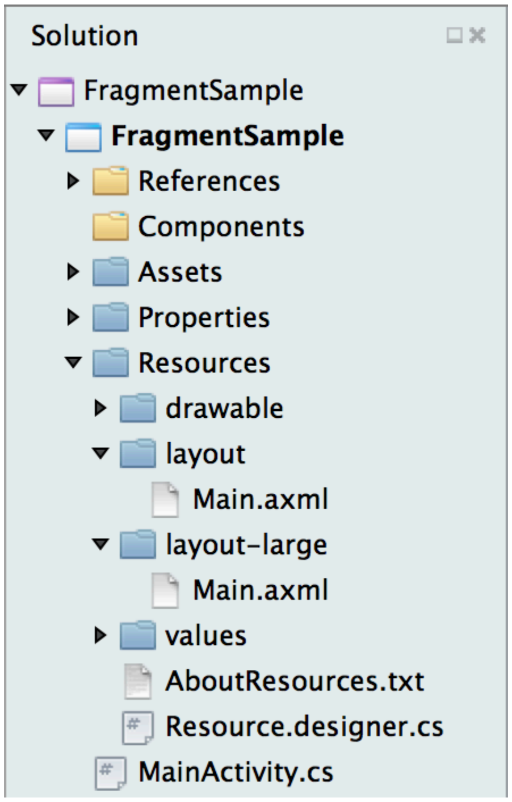
5. Add a new resource folder named `Resources/layout-large`, and create a new layout called `Main.axml` and edit the contents of the file so that it contains the following XML:

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

  <fragment class="fragmentsample.TitlesFragment"
            android:id="@+id/titles_fragment"
            android:layout_weight="1"
            android:layout_width="0px"
            android:layout_height="match_parent" />

  <FrameLayout android:id="@+id/details"
               android:layout_weight="1"
               android:layout_width="0px"
               android:layout_height="match_parent" />

</LinearLayout>
```

When Android detects that it is running on a device with a large screen, it will inflate this layout file instead of the file `Resources/layout/Main.axml`. Android knows that the file `Resources/layout-large/Main.axml` is for large devices because of the special name of the directory. Notice that the resource directory has the suffix `-large`. This suffix is known as a *resource qualifier* and is used identify resources for specific devices.

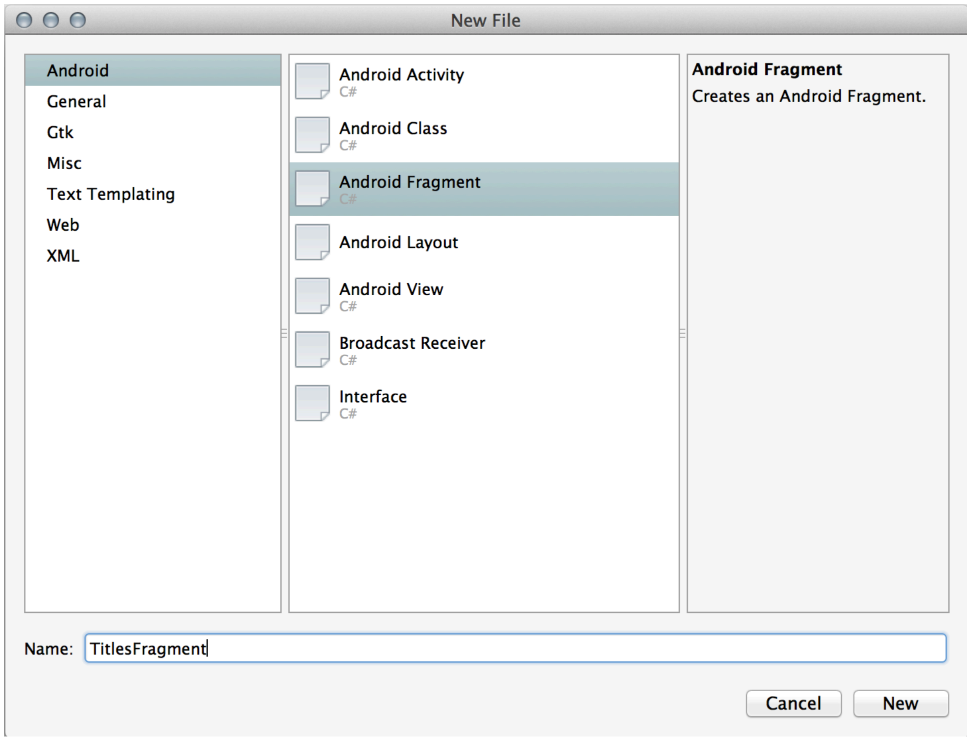
This layout will display two Views – the left side of the screen will display the `TitlesFragment` while the right side of the screen holds a `FrameLayout`. The `FrameLayout` is a placeholder. At runtime, `MainActivity` will replace the `FrameLayout` with a dynamically loaded fragment showing a quote from the Shakespearean play that the user selected in `TitlesFragment`.

When you are done with these two steps, the structure of your project should resemble the following screenshot:



```
Solution
▼ FragmentSample
  ▼ FragmentSample
    ▶ References
      Components
    ▶ Assets
    ▶ Properties
    ▼ Resources
      ▶ drawable
      ▼ layout
          Main.axml
      ▼ layout-large
          Main.axml
      ▶ values
        AboutResources.txt
        Resource.designer.cs
      MainActivity.cs
```

6. Next we need to create `TitlesFragment` that will display the titles of the various plays. Add a new file named `TitlesFragment` to the project as shown in the following screenshot:



Android Fragment
Creates an Android Fragment.

Name: TitlesFragment

7. After adding `TitlesFragment` has been added, change the class so that it inherits from `Android.App.ListFragment.` `ListFragment` is a specialized fragment type that includes list functionality. Edit TitlesFragment so that it resembles the following code snippet:

```
public class TitlesFragment : ListFragment
{
        private bool _isDualPane = false;
        private int _currentPlayId;
}
```

8. This application requires some data to display, namely a list of plays from Shakespeare and some quotes from those plays. To keep things simple, we will hold this kind of information in some arrays. Add a new class to the project called Shakespeare, and paste in the following code for it:

```
public class Shakespeare
{
        public static string[] Dialogue =
        {
                "So shaken as we are, so wan with care," +
                "Find we a time for frighted peace to pant," +
                "And breathe short-winded accents of new broils" +
                "To be commenced in strands afar remote." +
                "No more the thirsty entrance of this soil" +
                "Shall daub her lips with her own children's blood;" +
                "Nor more shall trenching war channel her fields," +
                "Nor bruise her flowerets with the armed hoofs" +
                "Of hostile paces: those opposed eyes," +
                "Which, like the meteors of a troubled heaven," +
                "All of one nature, of one substance bred," +
                "Did lately meet in the intestine shock" +
                "And furious close of civil butchery" +
                "Shall now, in mutual well-beseeming ranks," +
                "March all one way and be no more opposed" +
                "Against acquaintance, kindred and allies:" +
                "The edge of war, like an ill-sheathed knife," +
                "No more shall cut his master. Therefore, friends," +
                "As far as to the sepulchre of Christ," +
                "Whose soldier now, under whose blessed cross" +
                "We are impressed and engaged to fight," +
                "Forthwith a power of English shall we levy;" +
                "Whose arms were moulded in their mothers' womb" +
                "To chase these pagans in those holy fields" +
                "Over whose acres walk'd those blessed feet" +
                "Which fourteen hundred years ago were nail'd" +
                "For our advantage on the bitter cross." +
                "But this our purpose now is twelve month old," +
                "And bootless 'tis to tell you we will go:" +
                "Therefore we meet not now. Then let me hear" +
                "Of you, my gentle cousin Westmoreland," +
                "What yesternight our council did decree" +
                "In forwarding this dear expedience.",
                "Hear him but reason in divinity," +
                "And all-admiring with an inward wish" +
                "You would desire the king were made a prelate:" +
                "Hear him debate of commonwealth affairs," +
                "You would say it hath been all in all his study:" +
                "List his discourse of war, and you shall hear" +
                "A fearful battle render'd you in music:" +
                "Turn him to any cause of policy," +
                "The Gordian knot of it he will unloose," +
                "Familiar as his garter: that, when he speaks," +
                "The air, a charter'd libertine, is still," +
                "And the mute wonder lurketh in men's ears," +
                "To steal his sweet and honey'd sentences;" +
                "So that the art and practic part of life" +
                "Must be the mistress to this theoric:" +
                "Which is a wonder how his grace should glean it," +
                "Since his addiction was to courses vain," +
                "His companies unletter'd, rude and shallow," +
                "His hours fill'd up with riots, banquets, sports," +
                "And never noted in him any study," +
                "Any retirement, any sequestration" +
                "From open haunts and popularity.",
                "I come no more to make you laugh: things now," +
                "That bear a weighty and a serious brow," +
                "Sad, high, and working, full of state and woe," +
                "Such noble scenes as draw the eye to flow," +
                "We now present. Those that can pity, here" +
                "May, if they think it well, let fall a tear;" +
                "The subject will deserve it. Such as give" +
                "Their money out of hope they may believe," +
                "May here find truth too. Those that come to see" +
                "Only a show or two, and so agree" +
                "The play may pass, if they be still and willing," +
                "I'll undertake may see away their shilling" +
                "Richly in two short hours. Only they" +
                "That come to hear a merry bawdy play," +
                "A noise of targets, or to see a fellow" +
                "In a long motley coat guarded with yellow," +
                "Will be deceived; for, gentle hearers, know," +
                "To rank our chosen truth with such a show" +
                "As fool and fight is, beside forfeiting" +
                "Our own brains, and the opinion that we bring," +
                "To make that only true we now intend," +
                "Will leave us never an understanding friend." +
                "Therefore, for goodness' sake, and as you are known" +
                "The first and happiest hearers of the town," +
                "Be sad, as we would make ye: think ye see" +
                "The very persons of our noble story" +
                "As they were living; think you see them great," +
                "And follow'd with the general throng and sweat" +
                "Of thousand friends; then in a moment, see" +
                "How soon this mightiness meets misery:" +
                "And, if you can be merry then, I'll say" +
                "A man may weep upon his wedding-day.",
                "First, heaven be the record to my speech!" +
                "In the devotion of a subject's love," +
                "Tendering the precious safety of my prince," +
                "And free from other misbegotten hate," +
                "Come I appellant to this princely presence." +
                "Now, Thomas Mowbray, do I turn to thee," +
                "And mark my greeting well; for what I speak" +
                "My body shall make good upon this earth," +
                "Or my divine soul answer it in heaven." +
                "Thou art a traitor and a miscreant," +
                "Too good to be so and too bad to live," +
                "Since the more fair and crystal is the sky," +
                "The uglier seem the clouds that in it fly." +
                "Once more, the more to aggravate the note," +
                "With a foul traitor's name stuff I thy throat;" +
                "And wish, so please my sovereign, ere I move," +
                "What my tongue speaks my right drawn sword may prove.",
                "Now is the winter of our discontent" +
                "Made glorious summer by this sun of York;" +
                "And all the clouds that lour'd upon our house" +
                "In the deep bosom of the ocean buried." +
                "Now are our brows bound with victorious wreaths;" +
                "Our bruised arms hung up for monuments;" +
                "Our stern alarums changed to merry meetings," +
                "Our dreadful marches to delightful measures." +
                "Grim-visaged war hath smooth'd his wrinkled front;" +
                "And now, instead of mounting barded steeds" +
                "To fright the souls of fearful adversaries," +
                "He capers nimbly in a lady's chamber" +
                "To the lascivious pleasing of a lute." +
                "But I, that am not shaped for sportive tricks," +
                "Nor made to court an amorous looking-glass;" +
                "I, that am rudely stamp'd, and want love's majesty" +
                "To strut before a wanton ambling nymph;" +
                "I, that am curtail'd of this fair proportion," +
                "Cheated of feature by dissembling nature," +
                "Deformed, unfinish'd, sent before my time" +
                "Into this breathing world, scarce half made up," +
                "And that so lamely and unfashionable" +
                "That dogs bark at me as I halt by them;" +
                "Why, I, in this weak piping time of peace," +
                "Have no delight to pass away the time," +
                "Unless to spy my shadow in the sun" +
                "And descant on mine own deformity:" +
                "And therefore, since I cannot prove a lover," +
                "To entertain these fair well-spoken days," +
                "I am determined to prove a villain" +
                "And hate the idle pleasures of these days." +
                "Plots have I laid, inductions dangerous," +
                "By drunken prophecies, libels and dreams," +
                "To set my brother Clarence and the king" +
                "In deadly hate the one against the other:" +
                "And if King Edward be as true and just" +
                "As I am subtle, false and treacherous," +
                "This day should Clarence closely be mew'd up," +
                "About a prophecy, which says that 'G'" +
                "Of Edward's heirs the murderer shall be." +
                "Dive, thoughts, down to my soul: here" +
                "Clarence comes.",
                "To bait fish withal: if it will feed nothing else," +
                "it will feed my revenge. He hath disgraced me, and" +
                "hindered me half a million; laughed at my losses," +
                "mocked at my gains, scorned my nation, thwarted my" +
                "bargains, cooled my friends, heated mine" +
                "enemies; and what's his reason? I am a Jew. Hath" +
                "not a Jew eyes? hath not a Jew hands, organs," +
                "dimensions, senses, affections, passions? fed with" +
                "the same food, hurt with the same weapons, subject" +
                "to the same diseases, healed by the same means," +
                "warmed and cooled by the same winter and summer, as" +
                "a Christian is? If you prick us, do we not bleed?" +
                "if you tickle us, do we not laugh? if you poison" +
                "us, do we not die? and if you wrong us, shall we not" +
                "revenge? If we are like you in the rest, we will" +
                "resemble you in that. If a Jew wrong a Christian," +
                "what is his humility? Revenge. If a Christian" +
                "wrong a Jew, what should his sufferance be by" +
                "Christian example? Why, revenge. The villany you" +
                "teach me, I will execute, and it shall go hard but I" +
                "will better the instruction.",
                "Virtue! a fig! 'tis in ourselves that we are thus" +
                "or thus. Our bodies are our gardens, to the which" +
                "our wills are gardeners: so that if we will plant" +
                "nettles, or sow lettuce, set hyssop and weed up" +
                "thyme, supply it with one gender of herbs, or" +
                "distract it with many, either to have it sterile" +
                "with idleness, or manured with industry, why, the" +
                "power and corrigible authority of this lies in our" +
                "wills. If the balance of our lives had not one" +
                "scale of reason to poise another of sensuality, the" +
                "blood and baseness of our natures would conduct us" +
                "to most preposterous conclusions: but we have" +
                "reason to cool our raging motions, our carnal" +
                "stings, our unbitted lusts, whereof I take this that" +
                "you call love to be a sect or scion.",
                "Blow, winds, and crack your cheeks! rage! blow!" +
                "You cataracts and hurricanoes, spout" +
                "Till you have drench'd our steeples, drown'd the cocks!" +
                "You sulphurous and thought-executing fires," +
                "Vaunt-couriers to oak-cleaving thunderbolts," +
                "Singe my white head! And thou, all-shaking thunder," +
                "Smite flat the thick rotundity o' the world!" +
                "Crack nature's moulds, an germens spill at once," +
                "That make ingrateful man!"
        };
        public static string[] Titles =
        {
                "Henry IV (1)",
                "Henry V",
                "Henry VIII",
                "Richard II",
                "Richard III",
                "Merchant of Venice",
                "Othello",
                "King Lear"
        };
}
```

9. Now we must update `TitlesFragment` so that it will display a list of plays. The Fragment will implement `OnCreateActivity` (another fragment lifecycle method) and provide an `Adapter` that `ListFragment` will use to populate the list. Edit `TitlesFragment`, and override `OnActivityCreated` as shown in the following snippet:

```
public override void OnActivityCreated(Bundle savedInstanceState)
{
    base.OnActivityCreated(savedInstanceState);

    var adapter = new ArrayAdapter<String>(Activity,
Android.Resource.Layout.SimpleListItemChecked, Shakespeare.Titles);
    ListAdapter = adapter;

    if (savedInstanceState != null)
    {
        _currentPlayId = savedInstanceState.GetInt("current_play_id", 0);
    }
    var detailsFrame = Activity.FindViewById<View>(Resource.Id.details);
    _isDualPane = detailsFrame != null && detailsFrame.Visibility == 
ViewStates.Visible;
    if (_isDualPane)
    {
        ListView.ChoiceMode = (int) ChoiceMode.Single;
        ShowDetails(_currentPlayId);
    }
}
```

This method uses an `ArrayAdapter<string>` and populates it with titles from the `Shakespeare` class that was added in the previous step.

After the adapter has been created, the fragment will try to determine which play is currently selected by looking in the instance state of the application. If there is a key called `current_play_id` in the instance state, then the value of that key will be stored in the instance variable `_currentPlayId`.

The code above tries to get a reference to the `FrameLayout` that is hosted by `MainActivity`. If `FrameLayout` does exist, then the `_isDualPane` flag is set to `true` and we know that the application is running on a device with a large screen. If the application is running on a large screen device, then the method `ShowDetails` will be invoked. The `ShowDetails` method will be covered in the next step.

10. Next lets implement `ShowDetails`. Add the following method to `TitlesFragment`:

```
private void ShowDetails(int playId)
{
        _currentPlayId = playId;
        if (_isDualPane)
        {
                ListView.SetItemChecked(playId, true);

                var details = 
FragmentManager.FindFragmentById(Resource.Id.details) as DetailsFragment;
                if (details == null || details.ShownPlayId != playId)
                {
                        // Make new fragment to show this selection.
                        details = DetailsFragment.NewInstance(playId);

                        // Execute a transaction, replacing any existing
                        // fragment with this one inside the frame.
                        var ft = FragmentManager.BeginTransaction();
                        ft.Replace(Resource.Id.details, details);
                        ft.Commit();
                }
        }
        else
        {
                var intent = new Intent();

                intent.SetClass(Activity, typeof (DetailsActivity));
                intent.PutExtra("current_play_id", playId);
                StartActivity(intent);
        }
    }
```

`ShowDetails` will display the quotation for the selected play. In the case of tablets, the `_isDualPane` flag will be set to `true`, and so the quote will be displayed in a fragment next to the `TitlesFragment`. If the selected play `id` is not already displayed, then a new `DetailsFragment` is created, and then loaded into the `FrameLayout` on the Activity. For other devices that do not have a large display—phones, for example—`_isDualPane` will be set to `false` so a new `DetailsActivity` will be started.

11. `TitlesFragment` is a list, and must respond to user selections in the list. To do this, `TitlesFragment` will override the method `OnListItemClick`. Implement `OnListItemClick` as shown in the following code snippet:

```
public override void OnListItemClick(ListView l, View v, int position,
long id)
{
    ShowDetails(position);
}
```

12. Next lets add a new Activity called `DetailsActivity`. Smaller devices will host `DetailsFragment` in an instance of this Activity:

```
[Activity(Label = "Details Activity")]
public class DetailsActivity : FragmentActivity
{
    protected override void OnCreate(Bundle bundle)
    {
        base.OnCreate(bundle);
        var index = Intent.Extras.GetInt("current_play_id", 0);

        var details = DetailsFragment.NewInstance(index);
        var fragmentTransaction = FragmentManager.BeginTransaction();
        fragmentTransaction.Add(Android.Resource.Id.Content, details);
        fragmentTransaction.Commit();
    }
}
```

Notice that no layout file is loaded for `DetailsActivity`. Instead, `DetailsFragment` is loaded into the root view of the Activity. This root view has a special ID `Android.Resource.Id.Content`.

13. Finally, we need to create `DetailsFragment`. This fragment will display a quote from the selected play that is displayed in `TitlesFragment`. The following code shows the complete `DetailsFragment`:

```
internal class DetailsFragment : Fragment
{
    public static DetailsFragment NewInstance(int playId)
    {
        var detailsFrag = new DetailsFragment {Arguments = new Bundle()};
        detailsFrag.Arguments.PutInt("current_play_id", playId);
        return detailsFrag;
    }
    public int ShownPlayId
    {
        get { return Arguments.GetInt("current_play_id", 0); }
    }

    public override View OnCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState)
    {
        if (container == null)
        {
            // Currently in a layout without a container, so no reason to
create our view.
            return null;
        }

        var scroller = new ScrollView(Activity);

        var text = new TextView(Activity);
        var padding =
Convert.ToInt32(TypedValue.ApplyDimension(ComplexUnitType.Dip, 4,
Activity.Resources.DisplayMetrics));
        text.SetPadding(padding, padding, padding, padding);
        text.TextSize = 24;
        text.Text = Shakespeare.Dialogue[ShownPlayId];

        scroller.AddView(text);

        return scroller;
    }
}
```

In order for `DetailsFragment` to function properly, it must have the index of the play that is selected in the `TitlesFragment`. There are many ways to provide this value to `DetailsFragment`; in this example, the play `Id` is placed into a Bundle and that Bundle is stored to the Arguments property of `DetailsFragment`. The property `ShownPlayId` is provided as a wrapper to abstract away the details of retrieving the `current_play_id` from the Arguments.

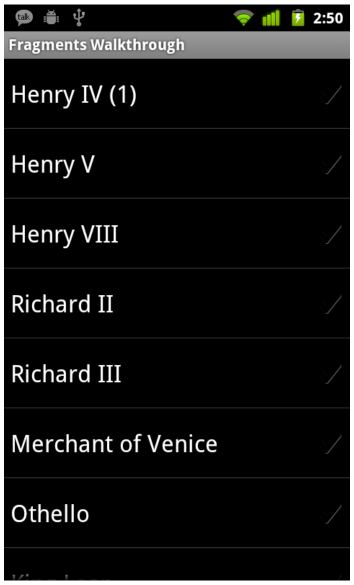
`OnCreateView` is called when the fragment needs to draw its user interface and should return an `Android.Views.View` object. In most cases, this is a `View` inflated from an existing layout file. In the case of the above example, the fragment will programmatically build the view that will be used for display.

14. Finally, run the application on any device with Android 3.0 or higher. The application should change is UI as appropriate depending on the size of the device.

Congratulations! You've now created an application that uses fragments to simplify development across form factors. The Activity behaves like a switchboard, coordinating communication between the two Fragments and updating the UI in response to use selections.

Backwards Compatibility With Older Devices

In this next section, we're going to extend this application so that it will work on pre-Android 3.0 devices. We will add the Android Support Packages to the application, and then make some changes to the existing codebase. Once this is done, our application will run on older versions of Android. The following screenshots show the application running on Android 2.3:



2:50

Fragments Walkthrough

Henry IV (1)

Henry V

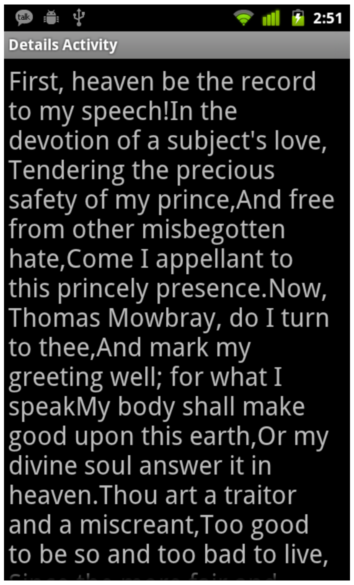
Henry VIII

Richard II

Richard III

Merchant of Venice

Othello



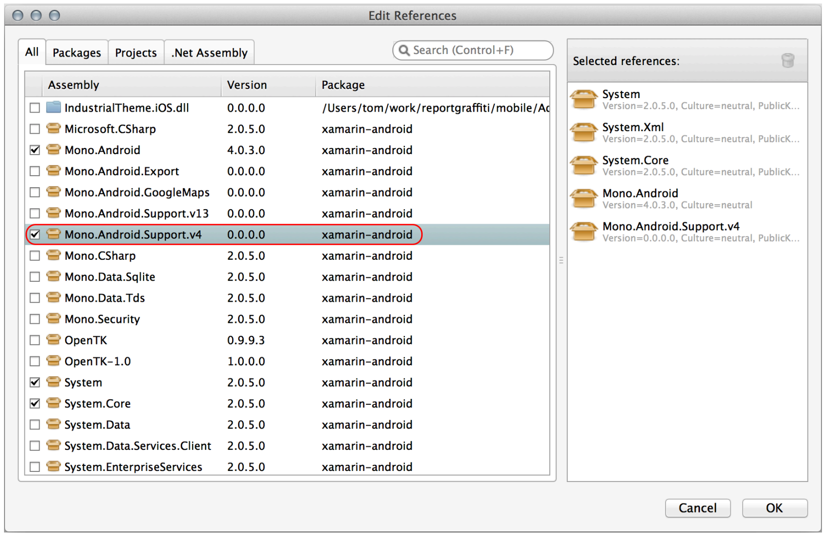
2:51

Details Activity

First, heaven be the record to my speech!In the devotion of a subject's love, Tendering the precious safety of my prince,And free from other misbegotten hate,Come I appellant to this princely presence.Now, Thomas Mowbray, do I turn to thee,And mark my greeting well; for what I speakMy body shall make good upon this earth,Or my divine soul answer it in heaven.Thou art a traitor and a miscreant,Too good to be so and too bad to live,

Now, lets get started making changes to this application.

1. The first thing we need to do is to reference the assembly `Mono.Android.Support.v4`, as shown in the following screenshot:

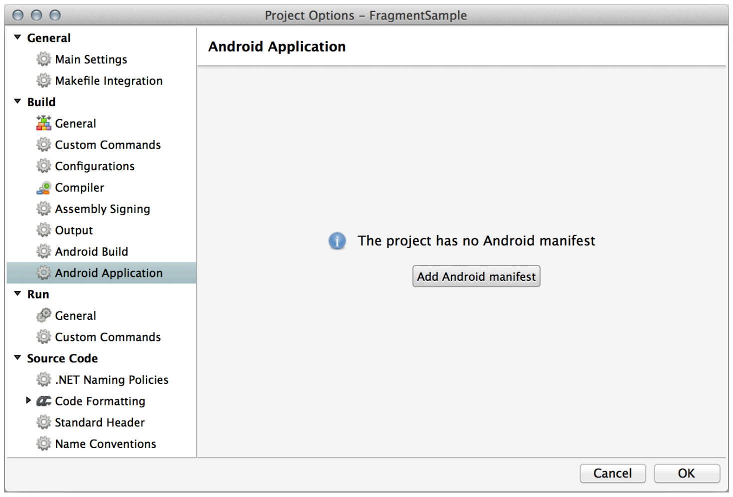


	Assembly	Version	Package
☐	IndustrialTheme.iOS.dll	0.0.0.0	/Users/tom/work/reportgraffiti/mobile/Ac
☐	Microsoft.CSharp	2.0.5.0	xamarin-android
☑	Mono.Android	4.0.3.0	xamarin-android
☐	Mono.Android.Export	0.0.0.0	xamarin-android
☐	Mono.Android.GoogleMaps	0.0.0.0	xamarin-android
☐	Mono.Android.Support.v13	0.0.0.0	xamarin-android
☑	Mono.Android.Support.v4	0.0.0.0	xamarin-android
☐	Mono.CSharp	2.0.5.0	xamarin-android
☐	Mono.Data.Sqlite	2.0.5.0	xamarin-android
☐	Mono.Data.Tds	2.0.5.0	xamarin-android
☐	Mono.Security	2.0.5.0	xamarin-android
☐	OpenTK	0.9.9.3	xamarin-android
☐	OpenTK-1.0	1.0.0.0	xamarin-android
☑	System	2.0.5.0	xamarin-android
☑	System.Core	2.0.5.0	xamarin-android
☐	System.Data	2.0.5.0	xamarin-android
☐	System.Data.Services.Client	2.0.5.0	xamarin-android
☐	System.EnterpriseServices	2.0.5.0	xamarin-android

Selected references:

- System
  Version=2.0.5.0, Culture=neutral, PublicK...
- System.Xml
  Version=2.0.5.0, Culture=neutral, PublicK...
- System.Core
  Version=2.0.5.0, Culture=neutral, PublicK...
- Mono.Android
  Version=4.0.3.0, Culture=neutral
- Mono.Android.Support.v4
  Version=0.0.0.0, Culture=neutral, PublicK...

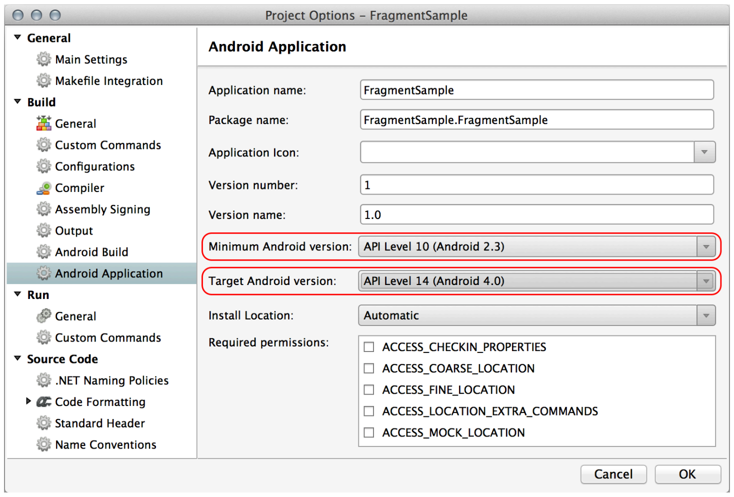
2. Next we must change the properties of our application and set the **Minimum Android version** and the **Target Android version**. Select the **Project Options** of the project, and click on **Android Application**. Notice that this project does not have an `AndroidManifest.xml` file:



Android Application

The project has no Android manifest

Click on the button **Add Android manifest**, and set the Minimum Android version to **API Level 10 (Android 2.3)** and the Target Android version to **API Level 14 (Android 4.0)** as shown in the following screenshot:



Android Application

Application name: FragmentSample
Package name: FragmentSample.FragmentSample
Application Icon:
Version number: 1
Version name: 1.0
Minimum Android version: API Level 10 (Android 2.3)
Target Android version: API Level 14 (Android 4.0)
Install Location: Automatic
Required permissions:
- ☐ ACCESS_CHECKIN_PROPERTIES
- ☐ ACCESS_COARSE_LOCATION
- ☐ ACCESS_FINE_LOCATION
- ☐ ACCESS_LOCATION_EXTRA_COMMANDS
- ☐ ACCESS_MOCK_LOCATION

3. In order for the application run on older versions of Android, any Activity that uses fragments must inherit from `Android.Support.V4.App.FragmentActivity`. This class is provided Android Support Package to support backwards compatibility for fragments. Change to `MainActivity` as shown in the following code snippet:

```
[Activity(Label = "Fragments Walkthrough", MainLauncher = true)]
public class MainActivity : FragmentActivity
{
    protected override void OnCreate(Bundle bundle)
    {
        base.OnCreate(bundle);
        SetContentView(Resource.Layout.Main);
    }
}
```

4. Likewise, `DetailsActivity` must be changed from an `Activity` to a `FragmentActivity` as well. The Android Support Package provides a wrapper class, *SupportFragmentManager*, which provides a backward compatible version

of the FragmentManager Change DetailsActivity so that it will use SupportFragmentManager as shown in the following code snippet:

```
[Activity(Label = "Details Activity")]
public class DetailsActivity : FragmentActivity
{
    protected override void OnCreate(Bundle bundle)
    {
        base.OnCreate(bundle);
        var index = Intent.Extras.GetInt("index", 0);

        var details = DetailsFragment.NewInstance(index);
        var fragmentTransaction = 
SupportFragmentManager.BeginTransaction();
        fragmentTransaction.Add(Android.Resource.Id.Content, details);
        fragmentTransaction.Commit();
    }
}
```

5. Next edit DetailsFragment so that it subclasses Android.Support.V4.App.Fragment and not Android.App.Fragment. This is as simple as changing the statement using Android.App; to using Android.Support.V4.App;.
6. The final class that needs to be changed is TitlesFragment. It too needs to subclass Android.Support.V4.App.Fragment and not Android.App.Fragment.
7. Run the application again. This time it will not matter what version of Android the device has – the application should work flawlessly.

At this point sit back and give yourself a pat on the back. You just created your first fragment-based application that will run on all current versions of Android, and will take advantage of the larger screens that tablets provide.

Summary

Fragments provide a way to decompose an Android application into re-usable components that can be hosted in Activities. This chapter introduced Fragments, and then went on to explain the Fragment lifecycle. Then you were shown how to add a Fragment to an application and use it in Activities. The specialized ListFragment was introduced and discussed. To help with backwards compatibility you were then introduced to the Android Support Package. Finally we finished up with a walkthrough, creating an application that used a ListFragment and a Fragment that would dynamically adapt its display to the device it's running on.