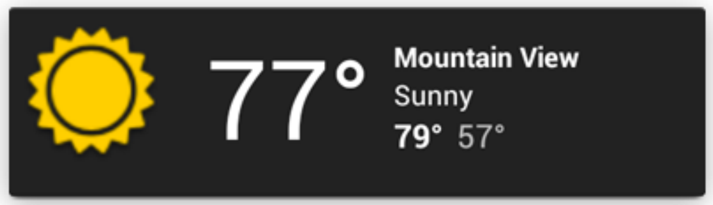
Application Widgets

Xamarin.Android Level 2 Chapter 8

Xamarin Inc.

Overview

Application Widgets (*App Widgets*) can be thought of as mini-applications that are hosted by other applications – the most common host is the Home Screen application in Android. They provide short summaries of information to the user, and are updated at regular intervals by background processes. One example of an App Widget is shown below:



The previous screenshot is an example of the simplest and most common type of App Widget, the *Information Widget*. Information widgets are not the only category of App Widget, the other categories of App Widgets are:

- ➔ Collection Widgets – These are widgets that display lists of information to the user, for example a collection of pictures from the Gallery.
- ➔ Control Widgets – This type of widget will display common functions that the user may toggle with out opening an application first. An example of this would be a music app widget that allows the user to pause and play music.
- ➔ Hybrid Widgets – These are widgets that combine aspects of all of the other widget types.

At an architectural level, an App Widget consists of these three parts:

- ➔ View Layout – this is the UI of the App Widget that will be embedded in the host application.
- ➔ Broadcast Receiver – This is also known as a *widget provider*, and responds to timer updates in order to update the view layout.
- ➔ Timer – This specifies the frequency of the updates. To preserve the battery, updates should not happen to frequently

There is an optional fourth component – an App Widget Configuration Activity. This is a regular Activity that will allow the user to configure an instance of an App Widget.

One key difference between an App Widget and other UI components such as an Activity or a Fragment is that, in an App Widget, the view layout is disconnected from the processes that update it. So, in the screen shot pictured above, the view for the music player is hosted by the Home Screen app, but the Home Screen app has minimal knowledge about what is going on in the App Widget.

This chapter will explain App Widgets with a focus on informational widgets. It will describe the various components that make up a widget, and then walk through a hands on example

Basics

By it's very nature, and App Widget is not one single entity but a loosely coupled collection of components. These components were loosely introduced in the previous section and will be expanded on here. To create an App Widget, it is necessary to implement the of the following components:

- ➔ AppWidgetProvider – This is a broadcast receiver that will update the App Widget with the information display.
- ➔ AppWidgetProviderInfo – This is an object that provides meta-data about the App Widget such as layout information, frequency of updates, and which `AppWidgetProvider` to use.
- ➔ View layout – this is an XML file that defines the initial layout for the App Widget. This is just like the regular XML layout file with the exception that only certain types of views can be used.
- ➔ An App Widget Configuration Activity – This is an optional component for an App Widget. It is an Activity that can be used to collect setup information from the user when the App Widget is created.

AppWidgetProviderInfo

The first time Android interacts with an App Widget , happens when a user wants to add it to an Activity, such as the Home Screen. In order for the system to add an instance of an App Widget to an application, it requires some meta-data about it. This information is provided to the system via an instance of the `AppWidgetProviderInfo` class. App Widgets don't create their own subclass of `AppWidgetProvider` – instead App Widgets store this information in a configuration file called the *App Widget provider information file*. This file is stored in the in `/resources/xml` folder. The following snippet shows an example of this file:

```
<appwidget-provider
    xmlns:android=http://schemas.android.com/apk/res/android
    android:minWidth="150dp"
    android:minHeight="120dp"
    android:updatePeriodMillis="3600000"
    android:initialLayout="@layout/my_widget "
    android:configure="xamarin.sample.ConfigureSampleWidgetActivity"
    android:previewImage="@drawable/some_preview_image_icon"
    android:resizeMode="horizontal|vertical"
    android:previewImage="@drawable/some_preview_image_icon"
    >
</appwidget-provider>
```

AppWidgetProvider

The `AppWidgetProvider` is a specialized broadcast receiver that will handle the event broadcasts that are relevant to an App Widget. It is a convenience class that will take

The following snippet is an example of an `AppWidgetProvider` subclass:

```
[BroadcastReceiver (Label = "@string/widget_name")]
[IntentFilter (new string []
{ "android.appwidget.action.APPWIDGET_UPDATE" })]
[MetaData ("android.appwidget.provider", Resource = "@xml/widget_word")]
public class WordWidget : AppWidgetProvider
{
    public override void OnDeleted(Context context, int[] appWidgetIds) {}
    public override void OnDisabled(Context context) {}
    public override void OnEnabled(Context context) {}
    public override void OnUpdate(Context context, AppWidgetManager
appWidgetManager, int[] appWidgetIds) {}
}
```

`AppWidgetProvider` subclasses must be adorned with the `[IntentFilter]` attribute that tells Android this broadcast receiver will respond to the `APPWIDGET_UPDATE` broadcast. There are other App Widget broadcasts that an App Widget can list for:

- ACTION_APPWIDGET_UPDATED – This
- ACTION_APPWIDGET_DELETED – This
- ACTION_APPWIDGET_ENABLED – This
- ACTION_APPWIDGET_DISABLED – This
- ACTION_APPWIDGET_OPTIONS_CHANGED – This

The `[MetaData]` attribute identifies an XML resource file that contains the meta data information for the AppWidgetProviderInfo class, described below. In this particular example the file is `/Resources/xml/widget_word.xml`.

App Widget View Layout

Layouts for an App Widget defined in XML just like the layouts for other Android views, with the XML files being stored in the folder `/Resources/Layout`. The views that make up the UI for an App Widget differ from other UI's in Android: the views are hosted by one process, such as the Home Screen application, but are controlled by some other background process. These background processes do not have direct access to the views that are in the App Widget; because of the App Widget views are known as *Remote Views*. The background processes must use a proxy class in order to interact with the views in their user interface.

Android provides a special class, `Android.Widget.RemoteView`, which can act as the proxy class between the hosting application and the background process. By using an instance of `RemoteView`, the background process may update UI of the App Widget.

Only certain controls are valid Remote Views. Only the following classes may act as Remote Views. Note that subclasses of the following are not eligible for Remote Views:

- ➔ FrameLayout
- ➔ LinearLayout
- ➔ RelativeLayout
- ➔ AnalogClock
- ➔ Button
- ➔ Chronometer
- ➔ ImageButton
- ➔ ImageView
- ➔ ProgressBar
- ➔ TextView
- ➔ ViewFlipper
- ➔ ListView
- ➔ GridView
- ➔ StackView
- ➔ AdapterViewFlipper

Because the views are disconnected from the process that controls them, a proxy must be used. This class is called, android.widget.RemoteView.

App Widget Definition

The life of an App Widget begins when the user wishes to add an App Widget. Android will query all the installed App Widgets for some meta-data about the App Widget and display that meta-data in the App Widget pick list. The following screen shot shows an example of the of the pick list in Android 4.0:

Walkthrough

1. Create new Xamarin.Android project named `SimpleWidget`.
2. Delete the file `Activity1.cs`. It is not required for this project.
3. Add the following drawables to the project: [CREATE ZIP FILE].
4. Add the following style resources:

```
<resources>

    <style name="WidgetBackground">
        <item name="android:background">@drawable/widget_bg</item>
    </style>
    <style name="BulletPoint">
        <item name="android:textSize">13sp</item>
        <item name="android:textColor">@android:color/black</item>
    </style>

    <style name="Text" />

    <style name="Text.Loading">
        <item name="android:textSize">14sp</item>
        <item name="android:textColor">@android:color/black</item>
    </style>

    <style name="Text.WordTitle">
        <item name="android:textSize">16sp</item>
        <item name="android:textStyle">bold</item>
        <item name="android:textColor">@android:color/black</item>
    </style>

    <style name="Text.WordType">
        <item name="android:textSize">14sp</item>
        <item name="android:textStyle">italic</item>
        <item name="android:textColor">@android:color/black</item>
    </style>

    <style name="Text.Definition">
        <item name="android:textSize">13sp</item>
        <item name="android:textColor">@android:color/black</item>
    </style>

</resources>
```

5. Replace the contents of `/values/strings.xml` with the following XML:

```
<resources>
    <string name="app_name">Wiktionary simple example</string>

    <string name="widget_name">Wiktionary Sample</string>

    <string name="widget_loading">Loading word\nof day\u2026</string>
</resources>
```

6. Request internet permissions in `AssemblyInfo.cs`:

```
[assembly: UsesPermission (Android.Manifest.Permission.Internet)]
```

7. Add a class `WordEntry`. This class will hold the data for the Word of the Day and contains the logic for retrieve the Word of the Day from the internet.

```
class WordEntry
{
        public string Title { get; set; }
        public string Description { get; set; }
        public string Link { get; set; }

        public WordEntry ()
        {
                Title = string.Empty;
                Description = string.Empty;
                Link = string.Empty;
        }

        public static WordEntry GetWordOfTheDay ()
        {
                var entry = new WordEntry ();

                try {
                        string url = "http://toolserver.org/~enwikt/wotd/";

                        XDocument doc = XDocument.Load (url);

                        XElement today = doc.Root.Element
("channel").Element ("item");

                        entry.Title = today.Element ("title").Value;
                        entry.Description = today.Element
("description").Value;
                        entry.Link = today.Element ("link").Value;

                        // Remove the date from the title
                        entry.Title = entry.Title.Substring
(entry.Title.IndexOf (':') + 1).Trim ();

                } catch (Exception ex) {
                        entry.Title = "Error";
                        entry.Description = ex.Message;
                }

                return entry;
        }
}
```

8. Create the class `UpdateService` class that will be used to fetch the Word of the Day and display it in the App Widget:

```
[Service]
public class UpdateService : Service
{
        private Context context;

        public override void OnStart (Intent intent, int startId)
        {
                // Build the widget update for today
                RemoteViews updateViews = buildUpdate (this);

                // Push update for this widget to the home screen
                ComponentName thisWidget = new ComponentName (this,
"simplewidget.WordWidget");
                AppWidgetManager manager = AppWidgetManager.GetInstance
(this);
                manager.UpdateAppWidget (thisWidget, updateViews);
        }
        public override IBinder OnBind (Intent intent)
        {
                // We don't need to bind to this service
                return null;
        }

        // Build a widget update to show the current Wiktionary
        // "Word of the day." Will block until the online API returns.
        public RemoteViews buildUpdate (Context context)
        {
                var entry = WordEntry.GetWordOfTheDay ();

                // Build an update that holds the updated widget contents
                var updateViews = new RemoteViews (context.PackageName,
Resource.Layout.widget_word);

                updateViews.SetTextViewText (Resource.Id.word_title,
entry.Title);
                updateViews.SetTextViewText (Resource.Id.definition,
entry.Description);

                // When user clicks on widget, launch to Wiktionary
definition page
                if (!string.IsNullOrEmpty (entry.Link)) {
                        Intent defineIntent = new Intent (Intent.ActionView,
Android.Net.Uri.Parse (entry.Link));

                        PendingIntent pendingIntent =
PendingIntent.GetActivity (context, 0, defineIntent, 0);
                        updateViews.SetOnClickPendingIntent
(Resource.Id.widget, pendingIntent);
                }

                return updateViews;
        }
}
```

9. Add the following XML to a file `/layout/widget_message.xml`. This will be used to display the Word of the Day in the App Widget:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/widget"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:focusable="true"
    style="@style/WidgetBackground">
    <ImageView
        android:id="@+id/icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:src="@drawable/star_logo" />
    <TextView
        android:id="@+id/word_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="14dip"
        android:layout_marginBottom="1dip"
        android:includeFontPadding="false"
        android:singleLine="true"
        android:ellipsize="end"
        style="@style/Text.WordTitle" />
    <TextView
        android:id="@+id/bullet"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/word_title"
        android:paddingRight="4dip"
        android:includeFontPadding="false"
        android:singleLine="true"
        style="@style/BulletPoint" />
    <TextView
        android:id="@+id/definition"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/word_title"
        android:layout_toRightOf="@id/bullet"
        android:paddingRight="5dip"
        android:paddingBottom="4dip"
        android:includeFontPadding="false"
        android:lineSpacingMultiplier="0.9"
        android:maxLines="4"
        android:fadingEdge="vertical"
        style="@style/Text.Definition" />
</RelativeLayout>
```

10. Implement an `AppWidgetProvider`:

```
[BroadcastReceiver (Label = "@string/widget_name")]
[IntentFilter (new string []
{ "android.appwidget.action.APPWIDGET_UPDATE" })]
[MetaData ("android.appwidget.provider", Resource = "@xml/widget_word")]
public class WordWidget : AppWidgetProvider
{
        public override void OnUpdate (Context context, AppWidgetManager
appWidgetManager, int[] appWidgetIds)
        {
                // To prevent any ANR timeouts, we perform the update in a
service
                context.StartService (new Intent (context, typeof
(UpdateService)));
        }
}
```

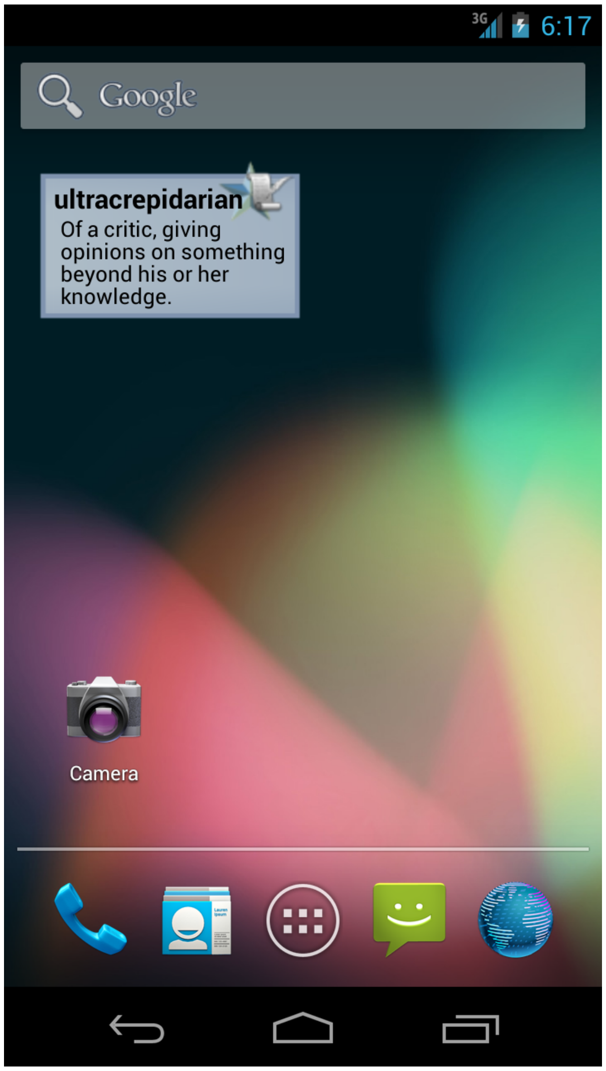
11. Create the meta data for the `AppWidgetProviderInfo` object:

```
<appwidget-provider
xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="146dip"
    android:minHeight="72dip"
    android:updatePeriodMillis="86400000"
    android:initialLayout="@layout/widget_message" />
```

12. Create the initial layout file that the App Widget will display when it is first created (added to the screen?)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/widget"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    style="@style/WidgetBackground">
    <TextView
        android:id="@+id/message"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="12dip"
        android:padding="10dip"
        android:gravity="center"
        android:text="@string/widget_loading"
        style="@style/Text.Loading" />
</LinearLayout>
```

13. Add the App Widget to the Home Screen, it should appear similar to the following:



ultracrepidarian

Of a critic, giving opinions on something beyond his or her knowledge.

Good show! You created your first App Widget

Summary

This chapter covered how to create an App Widget in Xamarin.Android. It introduced architecture and explained how to create an App Widget. It then finished up with a practical hands-on walkthrough that created a Word of the Day App Widget.