Chapter Thirteen, Activity #1: Weather App Widget

Updated May, 2015 to reflect new World Weather Online API version 2.

In this activity, we will expand our Weather Application from the last chapter to include an app widget that can be placed on the home screen. The widget will show an image and information about the current weather.

Project Details

Your modified program will allow the user to select the Weather App as a home screen widget. The widget display will show a brief summary of the current weather conditions.

In order to make your application available as a widget, you need to complete these general tasks:

- Create a layout XML file for your widget display.
- Create a widget definition XML file
- Update your "AndroidManifest.xml"
- Create an **AppWidgetProvider** class to implement your widget logic

How to Complete this Project – Widget Layout XML

To add a widget to your application, you will need to create a widget layout XML file as described in the student textbook. Create an XML layout file called "widget_layout.xml". Your design should contain an **ImageView** control and two **TextView** controls and should end up looking like the example to the right.

To give your widget a blue background (and ensure that the text is readable on any desktop wallpaper), add the following attribute line to your opening <**RelativeLayout**> tag in your layout file:

```
    android:background="#94B6E7"
```

How to Complete this Project – Widget Definition XML

You next need to create a widget definition XML file. Add the attributes described in the student textbook within your <appwidget-provider> element. Specify a 30-minute update interval, a width and height to consume 1 row and two columns in the home screen grid, and select your layout XML file created above.

How to Complete this Project – Update "AndroidManifest.xml"

In your "AndroidManifest.xml" file, add a <receiver> element that specifies the name of your widget class, an <intent-filter> with the APPWIDGET_UPDATE <action>, and a <meta-data> element with the name and resource for your widget XML definition file. We are going to call your widget class **WeatherWidget**.

How to Complete this Project – Create an AppWidgetProvider Class

You next need to create a class called **WeatherWidget** to implement your widget. This class should extend the **AppWidgetProvider** class and should implement several major features.

First, create a **getWeather**() method in the **WeatherWidget** with this function definition:

```
private void getWeather(Context context, RemoteViews rv, int appWidgetId)
```

This method will be responsible for downloading the current weather information and updating the widget controls with the image, current temperature, and description. To do this, add the following logic:

- Get the current location for this widget by calling **Main.getLocation**(), passing in the **`context`** parameter only. This will ensure you share the currently configured location with the **Main** activity.
- Get an **ArrayList<String>** of weather information by calling **Main.downloadWeather**(), passing in the **`context`** and **`location`** string.
- Make sure there is at least one result in the **ArrayList** to process. The first line (element 0) will contain the current weather information, which is what we want to display! If there is at least one line in the **ArrayList**:
    - Create a new **WeatherInfo** object from the first line by calling **WeatherInfo.getCurrentWeatherInfo**(), passing in the first **ArrayList** result variable.
    - Get a Bitmap image by calling **Main.downloadWeatherImage**(), passing in the **`context`** parameter and **`imageURL`** contained in the **WeatherInfo** object.
    - If the returned **Bitmap** image is not **null**, store it in the **ImageView** you defined in your layout XML by calling **setImageViewBitmap**() through the **RemoteViews** parameter.
    - Update the first **TextView** in your layout XML through the **RemoteViews** to contain the **`temperature`** from the **WeatherInfo** plus the character "F" to show it in Fahrenheit.

- Update the second **TextView** in your layout XML through the **RemoteViews** to contain the **`description`** from the **WeatherInfo** object.

Now that you have a handy **getWeather**() method that will update your widget's display with current weather information, where do you call it from? The **onUpdate**() method of course!

You next need to create an **onUpdate**() method that will be called every 30 minutes to update all widgets:

```
public void onUpdate(Context context, AppWidgetManager appWidgetManager,
                     int[] appWidgetIds)
{
    super.onUpdate(context, appWidgetManager, appWidgetIds);
    // other logic here
}
```

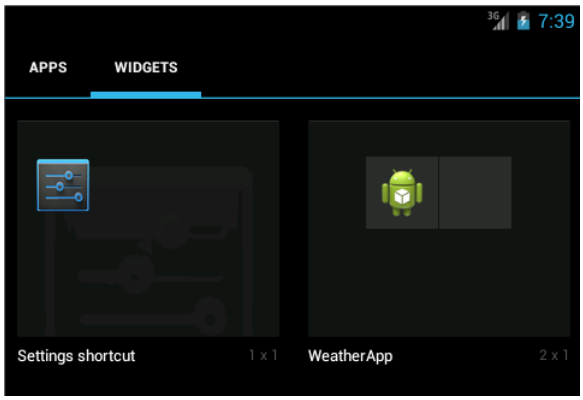
For each widget, you want to install a **PendingIntent** that will launch the **Main** weather activity if the user clicks on the image. You also want to update the widget's display based on its configured location with new weather information.

Underneath your call to **super.onUpdate**(), perform these steps:

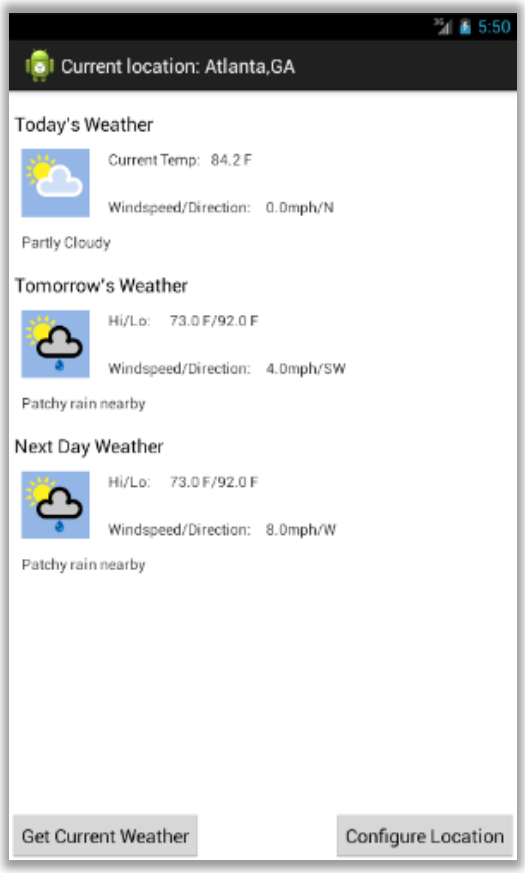
- Loop through all of the **`appWidgetIds`** that were passed into this method. The user could have more than one Weather widget on their home page!
- For each widget:
  - Create an explicit **Intent** with the input context targeting the **Main.class**
  - Create a **PendingIntent** with **PendingIntent.getActivity**() to hold your explicit **Intent**.
  - Create a **RemoteViews** for the widget using the context's package name and the widget's XML layout
  - Set the **PendingIntent** into your **ImageView** object from the XML layout using the **RemoteViews**
  - Call your **getWeather**() method to update the weather information for that widget, passing in the context, **RemoteViews**, and current appWidgetId.
  - Call the **AppWidgetManager**'s **updateAppWidget**() method with your appWidgetId and **RemoteViews** object.

Project Output

When finished, after you run your application in the emulator, you should be able to exit the normal weather app and return to the home screen. From there, choose the Applications button and select the "Widgets" tab at the top of the screen. Scroll across and you should see your "Weather App" widget listed. Add it to your home screen, and verify that it begins showing you accurate weather information for your current location.



If you click on the widget's image, your **Main** activity should launch and show you the more detailed information screen, including tomorrow and the next day's data.



Note that your app widget will use the same configuration information as the **Main** activity for now, and it only updates every 30 minutes. So any configuration changes you make after adding the widget will not take effect immediately.

You should be able to add a new widget to the home screen, however, that will use the **Main** activity's current configuration right away.