Android Widget Homepage

Alexander Roan – alr16

This report details the story of how I went about implementing the widget, problems encountered during the implementation and what I learned whilst developing it. I also include some screen shots within the report, and a screen capture video of the app working in the root directory of the assignment folder.

Implementation

I began the implementation of this widget by creating a prototype widget which acted as a testing app for the kind of functionality I needed to include. All of the functionality in this widget was handled by the main class which extended AppWidgetProvider, in the overridden functions OnUpdate and OnRecieve.

I used a tutorial on XML handling [1] to implement the retrieval of data from the two conference URLs. It uses DefaultHttpClient and HttpPost to open a connection to the server and retrieve the information. Initially this was not working at all, as a connection was not being made and was crashing the application. At first, I thought including the internet permission into the manifest would solve the problem. Unfortunately, it did not. After much searching, I found that including two lines of code implementing a StrictMode thread policy which permits network access as well as adding the internet permission to the manifest solved the issue.

Once the data is retrieved the XML handler then uses Dom classes to separate each of the nodes and extract the data from within the XML.

Another tutorial [2] explained how implementing a configuration activity to configure the widget before the widget was displayed on the screen was possible. I set this configuration class to display two buttons, each representing a conference. When one of them is clicked, the corresponding conference URL is requested, its XML data extracted and the relevant data displayed on the widget. This however, proved to be quite buggy. As well as this configuration activity being shown before the widget is placed on the screen, I needed a button on the widget to reopen it when the user wishes to change the conference that they wish to be displayed. When I set the configuration activity to show before the widget is displayed, the button to re configure the widget did not open the configuration activity. In contrast, when the configuration activity was not set to open upon creating the widget, the button worked. Therefore, I decided not to set the configuration activity to open before the widget is placed, meaning that the user could change the configuration of the widget by clicking the button to change the conference.

Once I had attempted implementing some of the basic features on this prototype widget, I began developing the actual project. Much like the prototype, there are 2 layout files: main and configure. The main layout is the widget as displayed on the home screen. It uses a gridlayout, meaning that the API level has to be a minimum of 14, due to this type of layout not being supported in previous versions of android widgets. Inside this gridlayout is a button for changing and refreshing conferences, another gridlayout with two columns which holds four textviews (two for the current session and two for the next session), and a textview to display the current message retrieved from the XML. The configure layout is a simple gridlayout with two columns where a button is present in each one, the first for Conference 1 and the second for Conference 2.

In the code, there are four classes: Main, Configure, ConferenceHandler and XMLHandler.

Main.java is the initial receiver for the widget and extends AppWidgetProvider. The only overridden method in this class is OnUpdate, which simply adds an on click pending intent to the button on the main widget layout, and updates the widget. This pending intent starts the Configure activity.

Configure.java is an activity which sets the desired conference which is to be retrieved over the internet. It sets on click listeners with pending intents to the two buttons in the layout, each of its own conference. These intents activate the ConferenceHandler class when clicked. The action on each of the button's pending intents is the URL of the requested conference number.

ConferenceHandler is an activity which retrieves the conference URL from the action that was set in the pending intents attached to the two buttons on the Configure activity. It then creates an XMLHandler object with the actioned URL to load and extract the desired data and update the widget by displaying that data.

The XMLHandler class is a standard java class with methods which handles the retrieval of data from a desired URL, and assuming the data is in XML and in the format of the conference data, will extract the data into an arraylist of sessions (each a HashMap<String String>), a current time in Date format and the current message. It also finds the current and next sessions in the array list depending on the current time.

Problems

I was not able to implement saving or loading user preferences, mainly because I struggled to understand the application of the AppWIdgetProvider class. The ability to start activities from other activities makes the development of normal applications quite intuitive, but because AppWidgetProvider does not bare any relation to Activity, it does not allow you to start activities. I found this premise quite difficult to deal with when trying to figure out how to save and load up preferences when the widget is started.

I had some problems when it came to debugging once I had implemented the XML retrieval from the URL. The http connection seemed to cause the debugger to time out, even after the retrieval of data had completed and that part of the XML handling had been completed. This meant that I couldn't realistically debug the widget once that part of the implementation had been called. However, once I knew that this part of the code was working, I could comment out calls to it when debugging other parts of the widget so that it wouldn't time out.

Sometimes, after changing something in the code or resources, the widget wouldn't respond to anything at all, giving very little feedback about what was going wrong. The service would sometimes not even show up in the DDMS even though the widget was present on the screen. This proved very frustrating and difficult to deal with when it came to figuring out where the code was falling down, especially when nothing seemed to have changed between builds other than the background colour for example.

Functionality

In terms of the functionality, most of the requirements have been met. In the user interface the current and next session is displayed using the current time as the starting point. The user is able to select a different conference using the Change/Refresh Conference button, and the message is displayed at the bottom of the widget.

The widget retrieves the data from the desired URL and extracts the XML data to be displayed of the screen.

As explained earlier I was unable to implement the saving and loading of the preference as to which conference is being displayed.

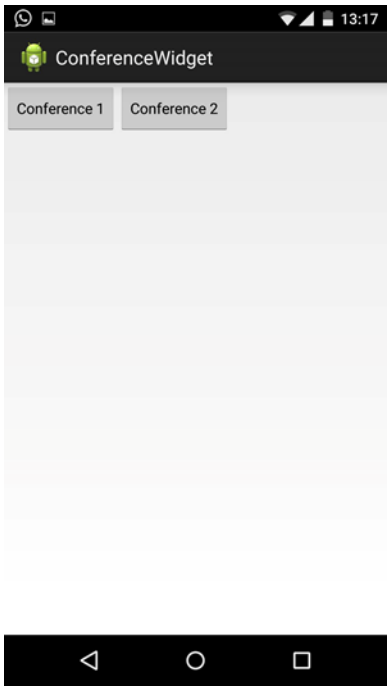
Testing

The following table details testing according to the functional requirements. There is also a video of the app functioning in the root directory of the assignment folder called screenrecording.mp4. As well as this, there are screenshots available in the screenshots folder and

also displayed in this report. The APK for the widget is available in the root directory named ConferenceWidget.apk and the code is available in the ConferenceWidget folder.

Function	Test	Pass/Fail	Notes
FR1. Creating the UI	Placing widget on screen	Pass	Widget is placed on screen
FR2. Retrieve latest data from remote URL	Clicking change/refresh conference button	Pass	Widget downloads latest data from requested url and displays on the screen
FR3. Storing User Preferences		Fail	

Screenshots



What I learned and Conclusion

Before this course I hadn't had much experience with developing mobile applications whatsoever. For this assignment I had to learn how to implement simple android applications before moving on to widgets. I used tutorials and walkthroughs as well as the google documentation on Android and Widgets especially to teach myself how to develop these applications.

I believe that although I have not managed to implement saving and loading of preferences from the widget, the rest of the functionality has been implemented successfully. Due to the steep learning curve from the beginning of this project I believe a mark near 60-65% would be a reasonable amount.

References

[1] – "Android XML Parsing Tutorial", Ravi Tamada. http://www.androidhive.info/2011/11/android-xml-parsing-tutorial/

[2] – "Android app Development : Creating a widget configuration activity in Android app Development", VideosLearning. https://www.youtube.com/watch?v=t4kDM-SudLo