COMP30510 Mobile Application Development

Widgets

Dr. Abraham Campbell

University College Dublin

Abey.campbell@ucd.ie

App Widgets

- Widgets (App Widgets) are miniature (interactive) application views that can be embedded in other applications and receive periodic updates
  - Home screen is an application!
  - Locked screen too
- A widget can be thought of as extremely efficient and very limited Activity

Interaction with Widgets

- Always remember that a user will only interact with a widget with either
  - Touch
  - Vertical swipe
- The WidgetCategory attribute allows your widget to be displayed in multiply locations
  - Home Screen
  - LockScreen in Android 4.2

Security and Performance Matters

- App Widgets should be extremely fast and very efficient as screen updates should happen immediately, otherwise users are frustrated
- Embedded widgets inherit permissions of its activity
  - Thus widgets are extremely limited by design both for performance and security reasons.

Creating an App Widget

- Design UI Layout
- Add Metadata description in XML
- Implement Intent Receiver

Designing Layout

- App Widget UI is a typical layout, with certain limitations
- Default grid layout is 4x4 cells, each 74dp minimum
- Calculating minimum # of pixels for an app widget:

  minsize = (Cell count * 74dp) 2dp

Appwidget Provider Metadata

```
<?xml ... >
<appwidgetprovider
xmlns:android="..."
android:initialLayout = \
"@layout/my_widget_layout"
android:minWidth="146dp"
android:minHeight="146dp"
android:label="My App Widget"
android:updatePeriodMillis="3600000"
/>
```

App Widget Intent Receiver

```
import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.content.Context;
public class MyAppWidget extends AppWidgetProvider
{
        @Override
        public void onUpdate(Context context,
                        AppWidgetManager
                        appWidgetManager,
                        int[] appWidgetIds)
                        {
                                // TODO Update the Widget UI.
                        }
}
```

Other methods

```
@Override
public void onDeleted(Context context,  int[] appWidgetIds)
{… super…}

@Override
public void onDisabled(Context context,)
{… super…}

@Override
public void onEnabled(Context context,)
{… super…}
```

App Widget in Manifest

```
<receiver android:name=".MyAppWidget"
                android:label="My App Widget">
        <intentfilter>
                <action android:name=
"android.appwidget.action.APPWIDGET_UPDATE" />
        </intentfilter>
<metadata android:name = "android.appwidget.provider"
    android:resource =  "@xml/my_app_widget_info" />
</receiver>
```

App Widget Limitations

- Using a App Widget it can only support the follow layout Limitations
  - FrameLayout
  - LinearLayout
  - RelativeLayout
  - GridLayout

App Widget Limitations Cont'd

- The only UI elements allowed are
  - AnalogClock
  - Button
  - Chronometer
  - ImageButton
  - ImageView
  - ProgressBar
  - TextView
  - ViewFlipper
  - ListView
  - GridView
  - StackView
  - AdapterViewFlipper
  - As of android 3.0 -> Collection view widgets

App Widget Limitations Cont'd

- App Widget interaction is limited to:
  - Adding a click listener to one or more views within the layout
  - Changing the UI based on selection changes

App Widgets: Other

- Typically Remote Views can be (now best practice) used to modify app widgets
- App Widgets can support multiple intent filters
- App Widgets can also be nicely integrated with Alarm Manager (for automatic periodic updates, etc)
- App Widgets can also have configuration

Code example from reto meier

```
package com.paad.PA4AD_Ch14_MyWidget;

import android.app.PendingIntent;
import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.content.Context;
import android.content.Intent;
import android.widget.RemoteViews;

public class MyAppWidget extends AppWidgetProvider {

 /**
  * Listing 14-8: Using a Remote View within the App Widget Provider's onUpdate Handler
  */
 @Override
 public void onUpdate(Context context,
                      AppWidgetManager appWidgetManager,
                      int[] appWidgetIds) {
  // Iterate through each widget, creating a RemoteViews object and
  // applying the modified RemoteViews to each widget.
  final int N = appWidgetIds.length;
  for (int i = 0; i < N; i++) {
   int appWidgetId = appWidgetIds[i];

}
```

```
            // Create a Remote View
            /**
             * Listing 14-5: Creating Remote Views
             */
            RemoteViews views = new RemoteViews(context.getPackageName(),
                                        R.layout.my_widget_layout);

            // TODO Update the UI
            /**
             * Listing 14-11: Adding a Click Listener to an App Widget
             */
            Intent intent = new Intent(context, MyActivity.class);
            PendingIntent pendingIntent =
              PendingIntent.getActivity(context, 0, intent, 0);
            views.setOnClickPendingIntent(R.id.widget_text, pendingIntent);

            // Notify the App Widget Manager to update the widget using
            // the modified remote view.
            appWidgetManager.updateAppWidget(appWidgetId, views);
          }
        }

public static String FORCE_WIDGET_UPDATE =
          "com.paad.mywidget.FORCE_WIDGET_UPDATE";

        @Override
        public void onReceive(Context context, Intent intent) {
          super.onReceive(context, intent);

          if (FORCE_WIDGET_UPDATE.equals(intent.getAction())) {
            // TODO Update widget
          }
        }
```