1.1 android widget（桌面小部件）实现

发表时间: 2017-06-30 关键字: android, widget

本文介绍如何自己实现一个widget以及各种注意事项。

首先我们先来看一下widget支持的布局，这是第一个“坑”

widget支持的layout和view类型十分有限，只支持下面几种

FrameLayout
LinearLayout
RelativeLayout
GridLayout
And the following widget classes:

AnalogClock
Button
Chronometer
ImageButton
ImageView
ProgressBar
TextView
ViewFlipper
ListView
GridView
StackView
AdapterViewFlipper

ViewStub

也就是说，如果想绘制一条分割线，只能在上面挑选一个view改变宽高设置背景。

如果想做一个toggle button，也只能用imageview发送自定义广播，通过接收广播来改变状态（至于为什么不直接改变image，往后看）

接下来，我们来看widget的逻辑处理组件，AppWidgetProvider

AppWidgetProvider用来接收widget事件，然后根据不同事件对widget进行不同处理。

AppWidgetProvider本质是一个广播接收器,在onReceive方法中处理了一些系统关于widget的广播,app只需重写相关方法即可

所以，在这里你除了可以处理widget相关广播，你也可以处理自定义广播和系统其他广播。

首先我们先在manifest中配置receiver

```
<receiver android:name=".TestWidgetProvider" >
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
        <action android:name="android.appwidget.action.APPWIDGET_ENABLED" />
        <action android:name="android.appwidget.action.APPWIDGET_DISABLED" />
        <action android:name="com.test.widgetdemo.UPDATE_START" />
        <action android:name="com.test.widgetdemo.UPDATE_END" />
    </intent-filter>
    <meta-data android:name="android.appwidget.provider"
               android:resource="@xml/app_widget"/>
</receiver>
```

meta-data中的resource指定了widget更多细节。

```
<appwidget-provider
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:configure="com.test.widgetdemo.WidgetCheckActivity"
    android:initialLayout="@layout/widget"
    android:minHeight="89dip"
    android:minWidth="282dip"
    android:resizeMode="horizontal|vertical"
    android:updatePeriodMillis="1800000"/>
```

关键配置:

1. minHeight/minWidth cell数量与最小宽高的关系: 70 × n − 30 (单位dip)

2. 更新频率 updatePeriodMillis, 最高为30分钟/次, 超过30分钟/次,系统自动重新赋值

3. configure 当添加widget时, 打开此activity用于对widget进行检查或配置。比如在这里可以检查登录状态，账户设置等等。

```
public class WidgetCheckActivity extends Activity {

    protected int mAppWidgetId;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if (savedInstanceState == null) {
            Intent intent = getIntent();
            mAppWidgetId = intent.getIntExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, AppWidgetMar
        } else {
            mAppWidgetId = savedInstanceState.getInt("widgetId");
        }

        Toast.makeText(this, "pass", Toast.LENGTH_LONG).show();

        Intent resultValue = new Intent();
        resultValue.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, mAppWidgetId);
        setResult(RESULT_OK, resultValue);
        finish();
    }

    @Override
    protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putInt("widgetId", mAppWidgetId);
    }
}
```

上面的代码演示了如何获取appWidgetId，如果有需要的话，可以将它在onSaveInstanceState中保存。

当此activity工作完成后，需要返回appWidgetId并关闭activity，否则widget添加不上。

view改变属性、状态和事件都无法使用view相关的方法，所有view只能通过RemoteViews对象来操作。

RemoteViews是你的布局映射的一个对象，使用它，根据你的view id来改变view的属性、状态和事件。比如：

```
views.setTextViewText(textview_id, "文本");
views.setTextColor(textview_id, 文本颜色);
views.setImageViewResource(imageview_id, 图片);
views.setViewVisibility(view_id, 显示状态);
views.setOnClickPendingIntent(view_id, 单击事件);
```

所以，之前说做一个toggle button的时候没有直接通过事件改变背景或src，是因为这里设置的并不是一个OnClickListener，而是一个PendingIntent。我们没有办法直接改变背景，只能通过这个PendingIntent回来，然后再改变背景或者src。

还有一些属性，RemoteViews没有提供直接修改的方法，但提供包装了反射的方法，比如改变背景：

```
views.setInt(view_id, "setBackgroundResource", resId);
```

相关的还有setLong/setShort/setBoolean/setChar/setByte/setString/setUri/setBitmap/setBundle/setIntent等等，不再一一列举

AppWidgetProvider有几个回调方法，这里只列举几个

添加widget

1.屏幕上第一个widget

系统会发送广播ACTION_APPWIDGET_ENABLED

2.之后添加的widget不会有enable或者add之类的广播. 但是每次添加widget依然会打开配置activity. 可以通过配置activity发送自己的初始化广播.注意: 收到此广播和系统的update广播顺序不定

删除widget

1.删除widget系统会发送ACTION_APPWIDGET_DELETED广播

2.删除最后一个widget系统会发出ACTION_APPWIDGET_DISABLED广播

更新widget时，系统会发送ACTION_APPWIDGET_UPDATE广播

widget大小改变时，系统会发送ACTION_APPWIDGET_OPTIONS_CHANGED广播

所以，如果要做一个刷新按钮，我们的步骤大概应该是这样

首先布局，一个ImageView和一个ProgressBar

```
<FrameLayout
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="center_vertical|right">
    <ImageView
        android:id="@+id/refresh"
        android:layout_width="17dip"
        android:layout_height="17dip"
        android:layout_gravity="center"
        android:scaleType="fitXY"
        android:src="@drawable/app_widget_refresh"/>
    <ProgressBar
        android:id="@+id/progress_bar"
        android:layout_width="17dip"
        android:layout_height="17dip"
        android:layout_gravity="center"
        android:indeterminateDrawable="@drawable/app_widget_progress"
        android:indeterminateDuration="2000"
        android:visibility="gone"/>
</FrameLayout>
```

然后初始化View状态并设置事件监听，然后更新widget

```
AppWidgetManager appWidgetManager = AppWidgetManager.getInstance(context);
int[] ids = getAppWidgetIds(context);
int N = ids.length;
for (int i = 0; i < N; i++) {
    RemoteViews views = new RemoteViews(context.getPackageName(), R.layout.widget);
```

```
    views.setOnClickPendingIntent(R.id.refresh, makeUpdateStartPendingIntent(context));
    appWidgetManager.updateAppWidget(ids[i], views);
}
```

这里make一个启动Service的PendingIntent

```
private static PendingIntent makeUpdateStartPendingIntent(Context context) {
    Intent intent = new Intent(context, WidgetHelperService.class);
    intent.putExtra("action", ACTION_UPDATE_START);
    return PendingIntent.getService(context, REQUEST_REFRESH, intent, PendingIntent.FLAG_UPDATE
}
```

然后通过service再发送广播，至于为什么不直接发送广播，稍后会告诉大家。

这是我们可以通过AppWidgetProvider来接收并处理此广播，用于进行更新数据操作以及刷新view

下面只是模拟一下刷新数据的动作，两秒中后，刷新按钮恢复之前的状态。

```
private static void showProgressBar(final Context context, boolean show) {
    AppWidgetManager appWidgetManager = AppWidgetManager.getInstance(context);
    int[] ids = getAppWidgetIds(context);
    int N = ids.length;
    for (int i = 0; i < N; i++) {
        final RemoteViews views = new RemoteViews(context.getPackageName(), R.layout.widget);
        if (show) {
            views.setViewVisibility(R.id.progress_bar, View.VISIBLE);
            views.setViewVisibility(R.id.refresh, View.GONE);
            new Handler().postDelayed(new Runnable() {
                @Override
                public void run() {
                    context.sendBroadcast(new Intent(ACTION_UPDATE_END));
                }
            }, 2000L);
        } else {
            views.setViewVisibility(R.id.progress_bar, View.GONE);
            views.setViewVisibility(R.id.refresh, View.VISIBLE);
        }
```

```
        appWidgetManager.partiallyUpdateAppWidget(ids[i], views);
    }
}
```

当然，在这个例子里，我们可以直接在run函数中操作view而不需要通过广播

更新widget有两个方法

```
AppWidgetManager.updateAppWidget(int appWidgetId, RemoteViews views)
AppWidgetManager.partiallyUpdateAppWidget(int appWidgetId, RemoteViews views)
```

注意：

1. 如果你发现，有时view事件无法相应，那么你可能是在调用partiallyUpdateAppWidget之前没有调用partiallyUpdateAppWidget， partiallyUpdateAppWidget在widget没有调用updateAppWidget之前，它是无效的（实际部分版本有效，5.0+？）。

2. 如果你发现view事件还是有时无法相应，那么你可能是通过PendingIntent.getBroadcast给view设置了一个发送广播的PendingIntent，要注意的是，在强制停止app之后，widget中的view无法再发送广播了，但是可以打开activity和service，所以我的做法是，通过view打开service，然后通过service发送广播

最后来说一下，为什么之前给view设置事件想发送广播的时候，没有直接使用PendingIntent.getBroadcast而是使用PendingIntent.getService

当app被强制停止后，高版本android中app是无法再接收广播的，无论是以哪种注册形式注册接收器。有的厂商的rom杀死app后也是如此。所以我们只能使用迂回战术，用户点击launcher中widget触发事件打开我们自己的service，这时我们的app已经复活，然后再给自己发广播，我们就可以收到了。