PDF-XChange Click to buy NOW! www.docu-track.com

AppWidget 基础小结

小小尝试了下 AppWidget，参考网上资料和 demo 小测，得出如下基础小结。

AppWidget 是基于 BroadcastReceiver 组件机制再开发而来的，为此他首先需要遵循 BroadcastReceiver 的开发流程进行开发，其次是根据他自身提供的 AppWidgetProvider、AppWidgetProvderInfo、AppWidgetManger 来进行开发。为此要开发一个 AppWidget 大致流程如下：

一、注册部件

在工程 manifest 中注册用于开发 AppWidget 的 receiver

```
<receiver android:name="AppWidget">
    <intent-filter>
        <action
android:name="android.appwidget.action.APPWIDGET_UPDATE"></action>
    </intent-filter>
    <meta-data android:name="android.appwidget.provider"
        android:resource="@xml/appwidget01" />
    <intent-filter>
        <action android:name="com.qlf.appWidgetUpdate"></action>
    </intent-filter>
</receiver>
```

其中红色部分的代码是固定要写的，绿色部分代码是根据用户自己开发情况来设置的。

通常一个应用程序往往只需要添加一个桌面部件就够了，为此如上述 receiver 这一项没有给出桌面部件的名称和图标，系统会默认用 application 标签中的名称和标签为 widgets 列表中的名称和图标。假如我们需要更加个性化，或者添加多种类型的桌面部件，那么可以根据需求完善桌面部件在桌面部件列表中的图标和名字，参考如下

```
<receiver android:name="AppWidget"
    android:label="test App widget"
    android:icon="@drawable/about_title">
```

一个桌面部件，需要一个 receiver 注册，多个就注册多个 receiver。具体可以参考网上小结 http://txlong-onz.iteye.com/blog/1143532。

AppWidget 是派生自 AppWidgetProvider 的自定义类，主要负责接收通知及其处理。该类派生自 BroadcastReceiver，通过源码可以得知，其主要对 BroadcastReceiver 的 onReceive 函数做了细分，衍生出 onEnabled（创建该类第一个桌面部件时调用）、onUpdate（单个该类桌面部件添加时或所设时间间隔过期时调用）、onDeleted（单个该类桌面部件删除时调用）和 onDisabled（手机该类最后一个部件被删除后调用），后面再详述。

meta-data 标签中的 android:resource 元素指定的 xml 资源，对应就是描述桌面部件的大小、更新频率和配置器（activity）等的 AppWidgetProvderInfo 对象，其资源文件被要求放置于 res/xml 文件夹中，我们拿一个实例来说明下

PDF-XChange Click to buy NOW! www.docu-track.com

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="150dp"
    android:minHeight="120dp"
    android:updatePeriodMillis="0"
    android:initialLayout="@layout/ appwidgetlayout"
    android:configure="com.androidbook.BDayWidget.ConfigureBDayWidgetActivity"
    >
</appwidget-provider>
```

以上包含了 SDK 3.0 之前的 appwidget-provider 标签所有的元素。其中的 minWidth 和 minHeight 为桌面部件需要的宽度和高度，据书上所说，桌面被分割高宽都为为 74dp 的单元格，所以部件的高宽最好设置为(number of cells * 74) - 2，因为最后部件放置时，桌面会将部件大小调整为适合整数个单元格大小。

updatePeriodMillis 为桌面部件更新的时间间隔，单位是毫秒，表示多久时间调用一次 onUpdate 函数。google 建议这个值设置为最小 1 小时，否则设备会被频繁唤醒，而且还建议如果要设置短时的更新时间，推荐使用 AlarmManager 类中的工具来调用 onUpdate。这里该值设置为 0，则表示不进行对 onUpdate 的更新调用，只在添加到桌面时调用。

initialLayout 为桌面部件加载到桌面后的布局，与其它布局类似，只是因为桌面部件布局需要使用跨进程的 RemoteViews 才能访问，而 RemoteViews 所支持的布局和控件是有限的，具体可以查看 android 在线帮助 Dev-Guide 下的 App Widgets 条目。在这里，initialLayout 指定为 appwidgetlayout.xml。其具体内容如下所示

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView android:id="@+id/txtapp"
        android:text="test"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#ffffff"></TextView>
    <Button
        android:id="@+id/btnSend"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Send"></Button>
</LinearLayout>
```

即一个文本框和一个按钮组成。

Configure 为桌面部件的配置器，主要负责配置部件实例，是一个 Activity。

在 3.0 以后还提供诸如 previewImage、autoAdvanceViewId、resizeMode，在这里就不做展开了。

二、编写派生自 AppWidgetProvider 的自定义类

上面注册了桌面部件，命名了一个名为“AppWidget”的 AppWidgetProvider 派生类，这个类的功能就是针对每一个桌面部件实例的消息处理类。正如上面所述，其继承自

PDF-XChange Click to buy NOW! www.docu-track.com

BroadcastReceiver，除了 onReceive 函数是通用性处理外，其它几个函数都是对应有固定的 ACTION 通知的。这部分还是直接通过代码说明的方便。

```
public class AppWidget extends AppWidgetProvider
{

    private final String broadCastString = "com.qlf.appWidgetUpdate";
    private final String extr = "com_extr";

    /**
     * 删除一个 AppWidget 时调用
     * */
    @Override
    public void onDeleted(Context context, int[] appWidgetIds){
        super.onDeleted(context, appWidgetIds);
    }

    /**
     * 最后一个 appWidget 被删除时调用
     * */
    @Override
    public void onDisabled(Context context){
        super.onDisabled(context);
    }

    /**
     * AppWidget 的第一个实例被创建时调用
     * */
    @Override
    public void onEnabled(Context context){
        super.onEnabled(context);
    }

    /**
     * 到达指定的更新时间或者当用户向桌面添加 AppWidget 时被调用
     * */
    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager,
            int[] appWidgetIds){
            //创建一个 Intent 对象
            Intent intent = new Intent();
            intent.setAction(broadCastString);
            intent.putExtra(extr, 0);
            intent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS,
appWidgetIds);
```

PDF-XChange Click to buy NOW! www.docu-track.com

```
                //这里是 getBroadcast，当然也可以是 Activity、Receiver 等
                PendingIntent pendingIntent = PendingIntent.getBroadcast(context, appWidgetIds[0], intent, PendingIntent.FLAG_UPDATE_CURRENT);

                //通过 RemoteViews 添加单击事件
                RemoteViews remoteViews = new RemoteViews(context.getPackageName(),R.layout.appwidgetlayout);
                remoteViews.setOnClickPendingIntent(R.id.btnSend, pendingIntent);

                //更新 Appwidget
                appWidgetManager.updateAppWidget(appWidgetIds, remoteViews);
        }

        /**
         * 接受广播事件
         * */
        @Override
        public void onReceive(Context context, Intent intent){
            if (intent.getAction().equals(broadCastString)){
                RemoteViews remoteViews = new RemoteViews(context.getPackageName(),R.layout.appwidgetlayout);
                //获得 appwidget 管理实例，用于管理 appwidget 以便进行更新操作
                AppWidgetManager appWidgetManager = AppWidgetManager.getInstance(context);
                int mTemp = intent.getIntExtra(extr, 0);

                if(mTemp == 0){
                    int[] appWidgetIds = intent.getIntArrayExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS);
                    if (appWidgetIds != null && appWidgetIds.length > 0) {

                        //创建一个 Intent 对象
                        Intent vintent = new Intent();
                        vintent.setAction(broadCastString);
                        vintent.putExtra(extr, 1);
                        vintent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS, appWidgetIds);

                        PendingIntent pendingIntent = PendingIntent.getBroadcast(context, appWidgetIds[0], vintent, PendingIntent.FLAG_UPDATE_CURRENT);
                        remoteViews.setOnClickPendingIntent(R.id.btnSend, pendingIntent);
                        remoteViews.setTextViewText(R.id.btnSend, "hihi");
```

PDF-XChange Click to buy NOW! www.docu-track.com

```
                    //更新 appwidget
                    appWidgetManager.updateAppWidget(appWidgetIds,
remoteViews);
                }
            }else{
                int[]                    appWidgetIds                    =
intent.getIntArrayExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS);
                if (appWidgetIds != null && appWidgetIds.length > 0) {
                    //创建一个 Intent 对象
                    Intent vintent = new Intent();
                    vintent.setAction(broadCastString);
                    vintent.putExtra(extr, 0);
                    vintent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_IDS,
appWidgetIds);

                    PendingIntent pendingIntent = PendingIntent.getBroadcast(context,
appWidgetIds[0], vintent, PendingIntent.FLAG_UPDATE_CURRENT);
                    remoteViews.setOnClickPendingIntent(R.id.btnSend,
pendingIntent);
                    remoteViews.setTextViewText(R.id.btnSend, "send");

                    //更新 appwidget
                    appWidgetManager.updateAppWidget(appWidgetIds,
remoteViews);
                }
            }
        }
        super.onReceive(context, intent);
    }
}
```

上述代码是对网上的一个整理的不错的例子进行每个 widget 桌面实例单独控制的修改而来，以避免在桌面上添加了多个同类桌面部件后，点击其中一个全部都被修改的问题。该例子位于 http://www.cnblogs.com/qianlifeng/archive/2011/03/26/1996407.html

关于 AppWidgetManager 和 RemoteViews

上述代码简单演示了 AppWidgetProvider 的消息处理流程，我们可以看到其内部的调用都是通过 AppWidgetManager 和 RemoteViews 来实现的。不像一般程序开发，一个实例是有一个类的对象化来实现的，在 appWidget 中，每一个具体放置于桌面上的部件，并不是某一个类的对象化，这些桌面部件都是通过 AppWidgetManager 这个客户端的中介与真实的 AppWidgetService 交互（其间采用了基于 AIDL 技术的 C/S 机制），从应用程序开发的角度来上说，我们可以理解为使用 AppWidgetManager 通过 RemoteViews 为信息载体来实现对每

PDF-XChange Click to buy NOW! www.docu-track.com

个桌面部件的控制，最主要的三个函数为

public void updateAppWidget(int appWidgetId, RemoteViews views)

public void updateAppWidget(int[] appWidgetIds, RemoteViews views)

public void updateAppWidget(ComponentName provider, RemoteViews views)

前面两个根据源码可以归为一类，即将 RemoteViews 的更改派发给指定的一个或多个 appWidget 实例，而第三个函数用于更新一类 appWidget 的所有实例。

PendingIntent 注意

在这里需要补充说明的是，PendingIntent 作为对 Intent 的封装，被称为挂起的 Intent，有一个很特殊的现象，当 intent 的内容除了 Extra 内容不同，其它都一致时，会被认为是同一个 Intent，为此我们要通过 Intent 传递信息，必须从获得 PendingIntent 三个函数入手：

public static PendingIntent getActivity(Context context, int requestCode, Intent intent, int flags)

public static PendingIntent getBroadcast(Context context, int requestCode, Intent intent, int flags)

public static PendingIntent getService(Context context, int requestCode, Intent intent, int flags)

requestCode 是一个目前没有使用的特殊值，网上通常采用默认的 0 值，这里可以通过给 int 变量赋不同值来达到区分相同内容 Intent 的目的。在 appWidget 中，这个值用每个桌面实例的 appWidgetId 最好了。不过我有疑问的是，这个 requestCode 不知道在哪里可以通过什么方式获取不？我是没有找到这个值的获取方式，为此虽然这个值设置为 appWidgetId，但是还是需要通过 Intent 的 Extra 信息来传递跟桌面每个部件对应 id 值。

同时上述三函数的调用中会存在 Extra 数据不被传递的问题，这个问题，可以通过修改第四个参数 flags 来实现，这个值可以有如下四种取值，简单描述如下：

int FLAG_ONE_SHOT（0x40000000）:该 PendingIntent 只能用一次，在 send()方法执行后，自动取消；

int FLAG_NO_CREATE（0x20000000）：如果该 PendingIntent 不存在，直接返回 null 而不是创建一个 PendingIntent；

int FLAG_CANCEL_CURRENT（0x10000000）：如果该 PendingIntent 已经存在，则在生成新的之前取消当前的；

int FLAG_UPDATE_CURRENT（0x8000000）：如果该 PendingIntent 已经存在，则用新传入的 Intent 更新当前的数据。

为了用新的 Intent 替换 PendingIntent 中旧的 Intent，保持最新意图得到执行，为此我们在这里建议使用 FLAG_UPDATE_CURRENT。

三、编写部件配置器

简单的桌面部件不需要编写部件配置，不过一旦在 AppWidgetProvderInfo 信息中添加了配置器，那么就需要编写该 Activity，这个类的写法在 sdk 中分步骤描述得很详细，我就不做翻译了，直接贴上示例代码如下

First, get the App Widget ID from the Intent that launched the Activity:

```
Intent intent = getIntent();
```

PDF-XChange Click to buy NOW! www.docu-track.com

```
Bundle extras = intent.getExtras();
if (extras != null) {
    mAppWidgetId = extras.getInt(
            AppWidgetManager.EXTRA_APPWIDGET_ID,
            AppWidgetManager.INVALID_APPWIDGET_ID);
}
```

2.Perform your App Widget configuration.

3.When the configuration is complete, get an instance of the AppWidgetManager by calling getInstance(Context):

```
AppWidgetManager appWidgetManager = AppWidgetManager.getInstance(context);
```

4.Update the App Widget with a RemoteViews layout by calling updateAppWidget(int, RemoteViews):

```
RemoteViews views = new RemoteViews(context.getPackageName(),
R.layout.example_appwidget);
appWidgetManager.updateAppWidget(mAppWidgetId, views);
```

Finally, create the return Intent, set it with the Activity result, and finish the Activity:

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

四、几个小问题

上述是涉及开发方面的问题，个人在亲测时还发现如下几个问题，有心人看看是否为共性问题，网上暂时也没有搜到相关资料

程序安装到 SD 卡后选择窗口小部件列表中没有对应的项

由于考虑到我的手机内存不多，所以将程序安装到 SD 卡，结果在桌面长按，弹出类似如下的“添加到桌面”对话框，选中 widget 对应的“小工具”，结果在“选择窗口小部件 ”列表对话框没有将安装到 SD 卡的程序带有的 widget 显示出来。

PDF-XChange Click to buy NOW! www.docu-track.com

而一旦在应用程序管理中将程序移动到手机内存，则在上述“选择窗口小部件”列表对话框中就会显示该程序带有的 appWidget 部件了。

Android 中除了 appWidget 外，还有一些应用是不能安装到 SD 卡的，比如 Live Folders（活动文件夹）、Input Method Engines（输入法引擎）、Alarm Services（闹铃提醒服务）。在 SDK 中没有找到相关信息，不过网上有篇博文提到类似信息，感兴趣可以打开如下链接：http://blog.csdn.net/raindrophust/article/details/6369810

程序覆盖安装后，会对所有以加载的部件调用一次 onUpdate 操作

因为用户可以在桌面上添加多个同类窗口部件，为此我们最好做到每个窗口部件独立控制，而不是在一个窗口部件上点击，导致其他所有同类窗口部件跟着变化，假如代码起初按照二中的代码所写，那么不遭遇程序覆盖安装，是不会有问题的，但是程序一旦重新覆盖安装，那么安装成功后，桌面会调一次 onUpdate 函数，其中的第三个参数 appWidgetIds 是所有该类部件的实例 ID 数组，而不是像添加部件时一样是单个实例 ID，所以会发生重新覆盖安装后，部件又跟着联动的现象。

为了避免上述问题，最简单的是修改 onUpdate 函数和 onReceive 函数，具体如下

```
/**
 * 到达指定的更新时间或者当用户向桌面添加 AppWidget 时被调用
 * */
@Override
public void onUpdate(Context context, AppWidgetManager appWidgetManager,
        int[] appWidgetIds){
    if (appWidgetIds != null && appWidgetIds.length > 0) {
        Mylog("onUpdate appWidgetIds length" + appWidgetIds.length);
        for(int appWidgetId:appWidgetIds){
            //创建一个 Intent 对象
            Intent intent = new Intent();
            intent.setAction(broadCastString);
```

PDF-XChange Click to buy NOW! www.docu-track.com

```
                    intent.putExtra(extr, 0);
                    intent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID,
appWidgetId);

                    //这里是 getBroadcast，当然也可以是 Activity、Receiver 等
                    PendingIntent pendingIntent = PendingIntent.getBroadcast(context,
appWidgetId, intent, PendingIntent.FLAG_UPDATE_CURRENT);

                    //通过 RemoteViews 添加单击事件
                    RemoteViews remoteViews = new
RemoteViews(context.getPackageName(),R.layout.appwidgetlayout);
                    remoteViews.setOnClickPendingIntent(R.id.btnSend, pendingIntent);

                    //更新 Appwidget
                    appWidgetManager.updateAppWidget(appWidgetId, remoteViews);
                }
            }
        }

        /**
         * 接受广播事件
         * */
        @Override
        public void onReceive(Context context, Intent intent){
            if (intent.getAction().equals(broadCastString)){
                RemoteViews remoteViews = new
RemoteViews(context.getPackageName(),R.layout.appwidgetlayout);
                //获得 appwidget 管理实例，用于管理 appwidget 以便进行更新操作
                AppWidgetManager appWidgetManager =
AppWidgetManager.getInstance(context);

                int mTemp = intent.getIntExtra(extr, 0);
                int appWidgetId =
intent.getIntExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, -1);
                if(mTemp == 0){
                    if (appWidgetId != -1) {
                        //创建一个 Intent 对象
                        Intent vintent = new Intent();
                        vintent.setAction(broadCastString);
                        vintent.putExtra(extr, 1);
                        vintent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID,
appWidgetId);

                        PendingIntent pendingIntent = PendingIntent.getBroadcast(context,
```

PDF-XChange Click to buy NOW! www.docu-track.com

```
appWidgetId, vintent, PendingIntent.FLAG_UPDATE_CURRENT);
                                remoteViews.setOnClickPendingIntent(R.id.btnSend,
pendingIntent);
                                remoteViews.setTextViewText(R.id.btnSend, "hihi");

                                //更新 appwidget
                                appWidgetManager.updateAppWidget(appWidgetId, remoteViews);
                            }
                        }else{
                            if (appWidgetId != -1) {
                                //创建一个 Intent 对象
                                Intent vintent = new Intent();
                                vintent.setAction(broadCastString);
                                vintent.putExtra(extr, 0);
                                vintent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID,
appWidgetId);

                                PendingIntent pendingIntent = PendingIntent.getBroadcast(context,
appWidgetId, vintent, PendingIntent.FLAG_UPDATE_CURRENT);
                                remoteViews.setOnClickPendingIntent(R.id.btnSend,
pendingIntent);
                                remoteViews.setTextViewText(R.id.btnSend, "send");

                                //更新 appwidget
                                appWidgetManager.updateAppWidget(appWidgetId, remoteViews);
                            }
                        }

                    }
                    super.onReceive(context, intent);
                }
```

经过上述修改后，就不会有覆盖安装后，在一个桌面部件上点击，其它同类部件跟着联动的现象了。

程序卸载后，已创建的桌面部件会留下痕迹

我在程序使用过程中，在桌面上添加了如下左图的所示的多个桌面部件，待程序卸载后，桌面上会出现类似如下右图所示的多个坑或者没留坑但是在部件上进行点击等操作就没有反应了。这些坑可能会在桌面重启时消失，也有可能重启后也继续在那里留坑。

PDF-XChange Click to buy NOW! www.docu-track.com

还有一种情况是我用 ADW 桌面和点心桌面做了个测试，即添加桌面部件到两个桌面，不删除程序的情况下，让手机重启，结果重启后如果 ADW 桌面装在手机内存中，那么就一切正常；如果装在 SD 卡上则之前添加部件的桌面位置会留了几个类似上面所示的坑；而装在 SD 卡上的点心桌面连坑都没有留下，被清除的一干二净；当然装在手机内存中的话，

这些问题，其实是跟加载 AppWidget 的具体的桌面应用程序相关了，不属于 AppWidget 开发的范畴，所以对于这个问题也没有什么好的解决方法。

关于 AppWidget 暂时小结到这里，对于高阶地集合使用等未作尝试和整理。

另外再添加几篇关于开发 appWidget 的博文，仅做扩展了解

http://blog.csdn.net/lyfadd85/article/details/6618004

http://blog.csdn.net/lyfadd85/article/details/6686831

http://blog.csdn.net/lyfadd85/article/details/6747546

http://blog.csdn.net/lyfadd85/article/details/6772772

http://www.cnblogs.com/TerryBlog/archive/2010/07/29/1788319.html

http://www.cnblogs.com/TerryBlog/archive/2010/08/10/1796896.html