#9 Widget开发

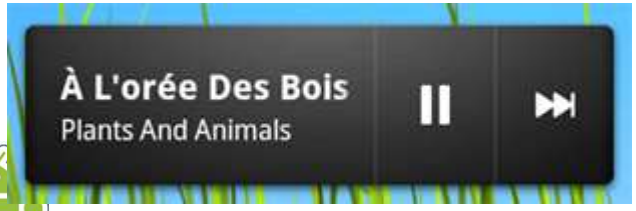
Widget

Category

1.Widget概述
2.Widget的开发
3.Widget的开发实例
4.Widget事件处理
5.从配置活动更新Widget

Widget的概述（1）

- Widget是微型应用程序视图，可以嵌入其他应用程序（如主屏幕）并接收定期更新。这些视图在用户界面中称为小部件。
- 标准的Android系统映像包含了一些示例widgets包括指针时钟、音乐播放器和其他工具如Google搜索栏
- Widget和标准的Apps相比没有太大的区别，更多的是在UI上的处理，逻辑执行设计成服务，具备更稳定和更高的可靠性



Widget的概述（2）

- 官方文档：
  https://developer.android.com/guide/topics/appwidgets/index.html

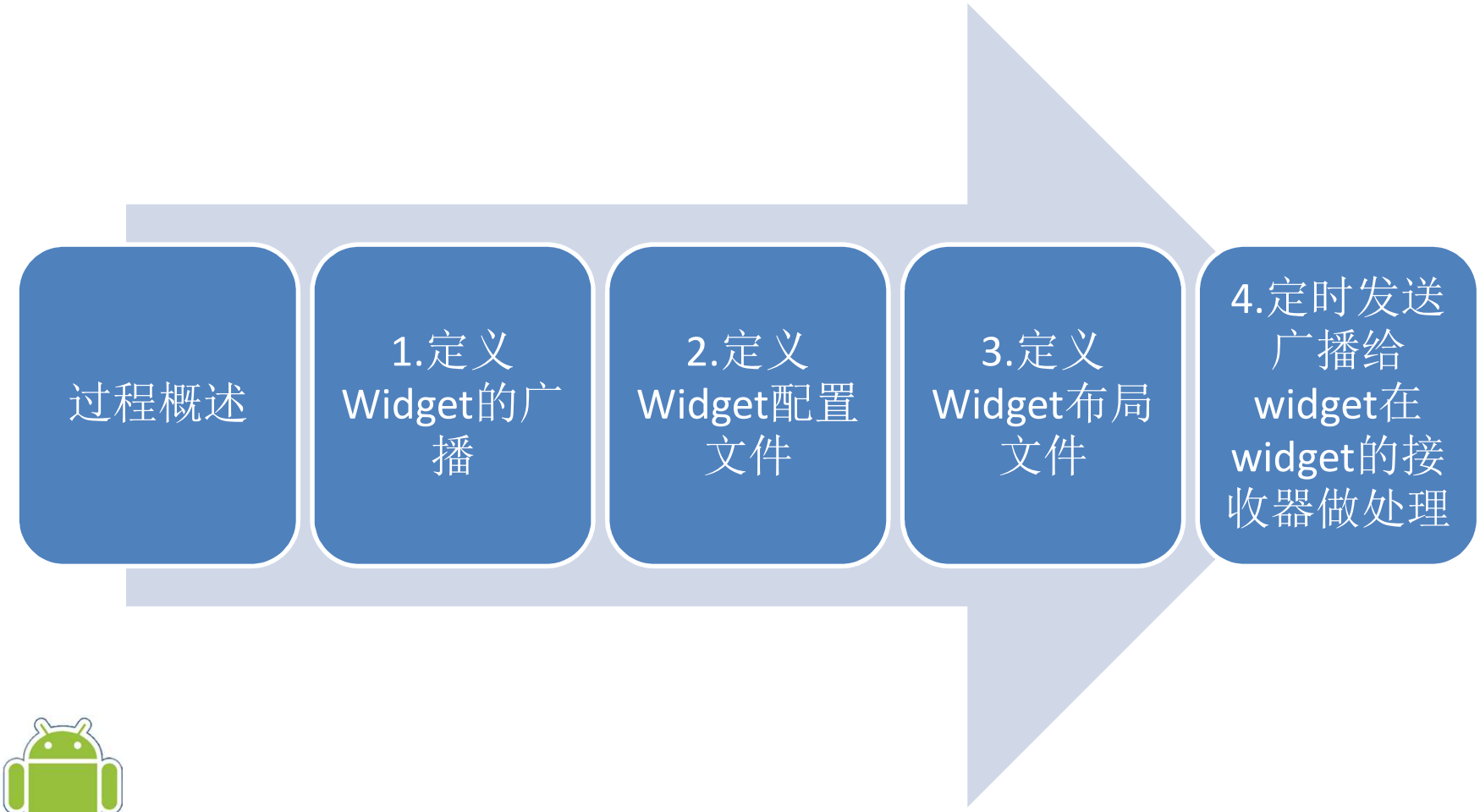
Widget的概述（3）

- 直接显示到桌面上的小控件，定期更新
- 每个Widget就是一个广播接收器
- 显示的内容封装成RemoteViews对象

Widget的概述（4）

- Widget 不是运行在自己进程里，而是宿主进程，所以交互需要处理AppWidget 广播。AppWidgetProvider 只接收和这个App Widget 相关的事件广播，比如这个App Widget 被更新，删除，启用，以及禁用。
- 每个Widget就是一个BroadcastReceiver，它们用XML metadata来描述Widget细节。AppWidget framework通过Broadcastintents 和Widget通信，Widget更新使用RemotesViews来发送。RemotesViews被包装成一个layout和特定的内容来显示到桌面上。

Widget的开发（1）


过程概述
1.定义Widget的广播
2.定义Widget配置文件
3.定义Widget布局文件
4.定时发送广播给widget在widget的接收器做处理

Widget的开发（2）

AppWidgetProviderInfo object 定义一个对象（XML)	• – Describes the metadata for an App Widget, such as the App Widget's layout, update frequency, and the AppWidgetProvider class. This should be defined in XML.
AppWidgetProvider class implementation 实现一个类 (java)	• – Defines the basic methods that allow you to programmatically interface with the App Widget, basedon broadcast events. Through it, you will receive broadcasts when the App Widget is updated,enabled, disabled and deleted.
View layout 定义初始布局 (XML)	• – Defines the initial layout for the App Widget, defined in XML. • – Additionally, you can implement an App Widget configuration Activity. This is an optional Activity thatlaunches when the user adds your App Widget and allows him or her to modify App Widget settingsat create-time.

Widget的开发（3）

- **AppWidgetProvider**

  继承自**BroadcastRecevier**，这些广播事件发生时,**AppWidgetProvider**将通过自己的方法来处理，这些方法包括：**update**、**enable**、**disable**和**delete**时接收通知。其中，**onUpdate**、**onReceive**是最常用到的方法。

➡ **onReceive**

接收到每个广播时都会被调用，而且在上面的回调函数之前。

➡ **onUpdate**

间隔性更新App Widget，间隔时间在AppWidgetProviderInfo里的updatePeriodMillis属性定义。该方法也会在添加App Widget时被调用，进行widget配置。

Widget的开发（4）

- **onDisabled**

  当App Widget的最后一个实例被从宿主中删除时被调用。例如在onDisabled中做一些清理工作，比如关掉后台服务。

- **onDeleted**

  当App Widget从宿主中删除时被调用。

- **onEnabled**

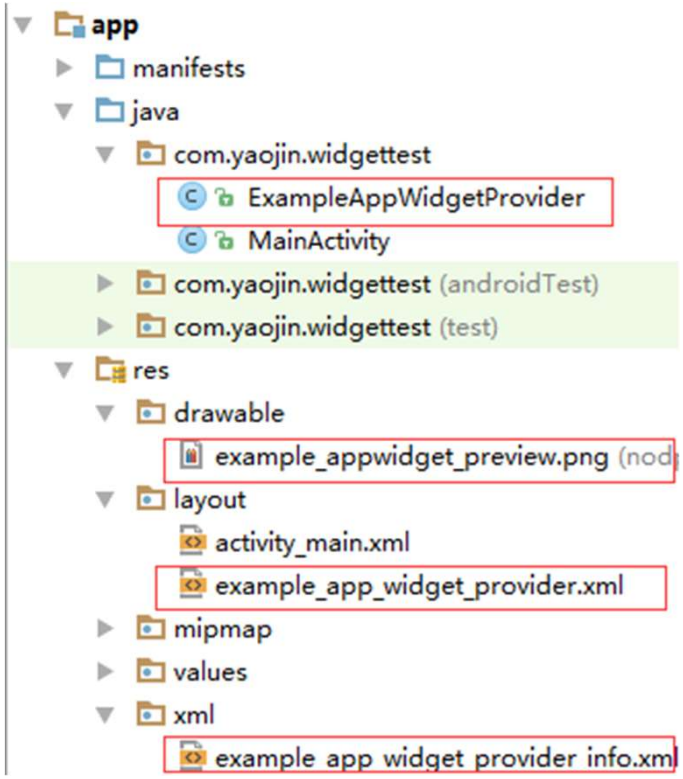
  当Widget实例第一次创建时被调用。若用户添加两个同一个App Widget实例，只在第一次被调用。适用于需要打开一个新的数据库或者执行其他对于所有的App Widget实例只需要发生一次的处理

- **onAppWidgetOptionsChange**

  第一次放置小部件和调整小部件的大小被调用这个。您可以使用此回调来根据窗口小部件的大小范围显示或隐藏内容

Widget的开发实例（1）

- 典型的Android Widget有三个主要组件，一个边框、一个框架和图形控件以及其他元素。在Android Studio中创建Widget类后，会直接生成相关文件。创建之后文件如图：



Widget的开发实例（2）

1. Widget内容提供者文件example_app_widget_provider_info.xml，初始代码如下，可以根据情况进行对应修改

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/
    android:initialKeyguardLayout="@layout/example_app_widget_provider"
    android:initialLayout="@layout/example_app_widget_provider"
    android:minHeight="40dp"
    android:minWidth="40dp"
    android:previewImage="@drawable/example_appwidget_preview"
    android:resizeMode="horizontal|vertical"
    android:updatePeriodMillis="86400000"
    android:widgetCategory="home_screen">

</appwidget-provider>
```

其中android:updatePeriodMillis=“86400000”是指自动更新的时间间隔。

Note: Updates requested with updatePeriodMillis will not be delivered more than once every 30 minutes.

Widget的开发实例（3）

<appwidget-provider> 部分属性总结:

minWidth和minHeight属性的值指定App Widget默认消耗的最小空间量。

minResizeWidth和minResizeHeight属性指定App Widget的绝对最小宽度和高度。

updatePeriodMillis属性定义AppWidget框架通过调用onUpdate（）回调方法从AppWidgetProvider请求更新的频率。

initialLayout属性指向定义App Widget布局的布局资源。

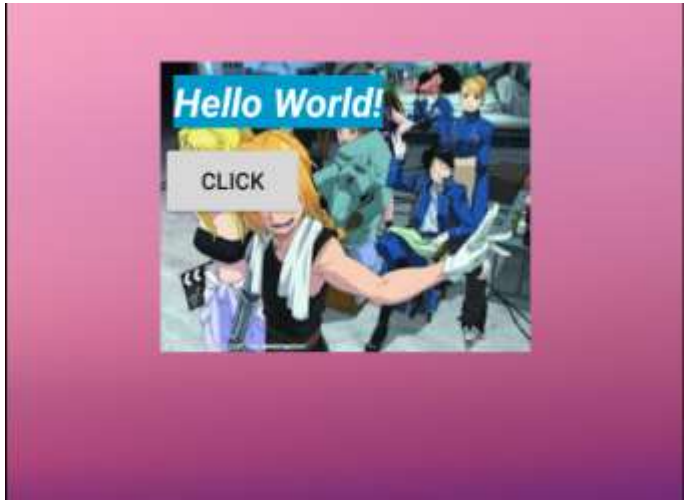
configure属性定义当用户添加App Widget时要启动的Activity，以便他或她配置App Widget属性。

previewImage属性指定在配置应用程序小部件后，用户在选择应用程序小部件时看到的内容。如果没有提供，用户会看到您的应用程序的启动器图标。

Widget的开发实例（4）

2.布局文件example_app_widget_provider.xml，默认情况下仅有1个textView，本实例中添加了一个button按钮并添加了背景，最后效果如图:

```
<TextView
    android:id="@+id/appwidget_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="8dp"
    android:background="#09C"
    android:contentDescription="@string/appwidget_text"
    android:text="@string/appwidget_text"
    android:textColor="#ffffff"
    android:textSize="24sp"
    android:textStyle="bold|italic" />
<Button
    android:id="@+id/appwidget_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/Click"
    android:layout_below="@id/appwidget_text"/>
```



Widget的开发实例（4）

3.修改ExampleAppWidgetProvider.java代码

```
public class ExampleAppWidgetProvider extends AppWidgetProvider {

    static void updateAppWidget(Context context, AppWidgetManager appWidgetManager,
                                int appWidgetId) {
    }

    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds) {
        // There may be multiple widgets active, so update all of them
        for (int appWidgetId : appWidgetIds) {
            updateAppWidget(context, appWidgetManager, appWidgetId);
        }
    }
```

4、修改配置文件AndroidManifest.xml

5、添加到桌面

问题：如何处理*Widget*事件？

Working with remote views

- 主屏幕实际上是一个系统程序，因为安全原因，程序员不容易直接修改其运行代码
- Android提供给用户程序访问主屏幕和修改特定区域内容的方法：RemoteView架构
- RemoteView架构允许用户程序更新主屏幕的View

➡ 点击Widget激活点击事件

➡ Android会将其转发给用户程序，由AppWidgetProviders类处理

➡ 用户程序可更新主屏幕Widget.

Widget事件处理(1)

- 配置文件添加:

```
<receiver android:name=".ExampleAppWidgetProvider">
    <intent-filter>
        <action android:name="com.yaojin.widgettest.CLICK"/>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
    </intent-filter>

    <meta-data
        android:name="android.appwidget.provider"
        android:resource="@xml/example_app_widget_provider_info" />
</receiver>
```

Widget事件处理(2)

- 在onUpdate添加代码定义并发送事件

```
public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds) {
    RemoteViews updateViews=new RemoteViews(context.getPackageName(),
       R.layout.example_app_widget_provider);//实例化RemoteView,其对应相应的Widget布局
    Intent i=new Intent("com.yaojin.widgettest.CLICK");
    PendingIntent pi=PendingIntent.getBroadcast(context,0,i,PendingIntent.FLAG_UPDATE_CURRENT);
    //PendingIntent pi=PendingIntent.getActivity(context,0,i,0);
    updateViews.setOnClickPendingIntent(R.id.appwidget_button,pi);//RemoteView上的Button设置按钮事件
    ComponentName me=new ComponentName(context,ExampleAppWidgetProvider.class);
    appWidgetManager.updateAppWidget(me,updateViews);
}
```

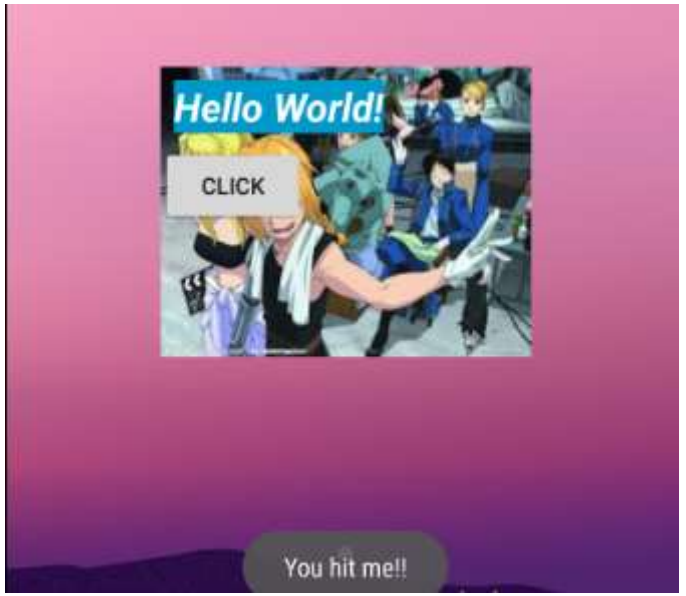
- 添加onReceive()处理事件

```
public void onReceive(Context context,Intent intent)
{
    super.onReceive(context,intent);
    if(intent.getAction().equals("com.yaojin.widgettest.CLICK"))
    {
        Toast.makeText(context,"You hit me!!",Toast.LENGTH_LONG).show();
    }
}
```

Widget事件处理(3)

- 最终效果。

  点击CLICK按钮后Toast显示：



Configuration Activity(1)

- configure属性定义当用户添加App Widget时要启动的Activity，以便他或她配置App Widget属性。

1、<appwidget-provider>添加configure属性

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
    android:initialKeyguardLayout="@layout/example_app_widget_provider"
    android:initialLayout="@layout/example_app_widget_provider"
    android:minHeight="160dp"
    android:minWidth="160dp"
    android:previewImage="@drawable/example_appwidget_preview"
    android:resizeMode="horizontal|vertical"
    android:updatePeriodMillis="86400000"
    android:widgetCategory="home_screen"
    android:configure="com.yaojin.widgettest.MainActivity">

</appwidget-provider>
```

Configuration Activity(2)

2、在configuration activity对widget进行配置:

```
Bundle extras = getIntent().getExtras();
if (extras != null) {
    mAppWidgetId = extras.getInt(
            AppWidgetManager.EXTRA_APPWIDGET_ID,
            AppWidgetManager.INVALID_APPWIDGET_ID);
}
AppWidgetManager appWidgetManager =
AppWidgetManager.getInstance(MainActivity.this);
RemoteViews views = new RemoteViews(getPackageName(),
        R.layout.example_app_widget_provider);
//对view进行配置
views.setTextViewText(R.id.appwidget_text,"changed by activity");
appWidgetManager.updateAppWidget(mAppWidgetId, views);
Intent resultValue = new Intent();
resultValue.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, mAppWidgetId);
setResult(RESULT_OK, resultValue);
```

Configuration Activity(3)

3、最终效果:

Questions?