

中国科学技术大学软件学院·Android实验讲义

Android应用程序框架

赵振刚 gavin@ustc.edu.cn

Supported by Google & WHU

内部版本·不得外传

- 1、Android应用程序框架
- 2、程序的生命周期
- 3、Activity启动方式
- 4、Manifest的定义
- 5、资源文件设计

Android应用程序架构

- 本课内容先简介Android的应用程序组件，接着依次介绍各应用程序组件的生命周期。
- 何谓生命周期?应用程序组件都具有生命周期—从Android产生回应一个Intent对象开始，到实际被释放为止。
- 一般情况Android应用程序是由以下四种组件所组成的：
 - 活动(Activity)
 - 服务(Service)
 - 广播接收器(Broadcast Receiver)
 - 内容提供者(Content Provider)

Android应用程序架构

- 活动(Activity)

- 一般所指的活动是用户界面。一个应用程序可能有一个或以上的活动存在，每个活动也都会有自己的View。
- 所有的活动在系统里由活动堆栈所管理，当一个新的活动被执行后，它将会被放置到堆栈的最顶端，并且变成“**running activity**”，而先前的活动原则上还是会存在于堆栈中，但它此时不会是在前景的情况，除非新加入的活动离开。

Android应用程序架构

- 服务(Service)

- 服务是在背景长时间运行的应用组件，不和用户直接进行互动。
- 例如：某服务可能在后台播放音乐，而用于在执行其他的操作，或者它通过网络抓取资料或者执行某些计算，将结果提供给活动（Activity）。

Android应用程序架构

- 广播接收器(Broadcast Receiver)
 - 广播接收器负责接受和响应通知，很多通知源自于系统所发送的，例如：发送时区变换的通知，电池电量不足，或用户改变语言设置。
 - 应用程序也可以发出广播通知，举例来说，通知其它应用程序，数据已下载完毕，可供使用。
 - 应用程序可以拥有任意数量的广播接收器来接收任何的通知。另外也可以启动活动（Activity）去响应接收到的通知，或利用通知管理器(NotificationManager)来通知使用者。

Android应用程序架构

- 内容提供者(Content Provider)
 - 内容提供者将应用程序数据组合成特定的集合供其它应用程序使用。数据可以是储存在文件、SQLite数据库，或是其它任何用户可以存取资料的地方。
 - 内容提供者继承於内容提供者基础类别，并实现一组标准的方法，使应用程序可以检索和储存它控制的数据。
 - 应用程序不是直接调用这些实现方法。而是通过内容解析器(ContentResolver)对象调用方法。内容解析器能够通知任何的内容提供者，并可以参与这些内容提供者进程间的管理。

- 1、Android应用程序框架
- 2、程序的生命周期
- 3、Activity启动方式
- 4、Manifest的定义
- 5、资源文件设计

程序的生命周期

- 程序的生命周期(Life Cycle)

- 活动 (active)

- 一个Activity基本上有三个生命状态：

- active或running
 - Paused
 - Stop

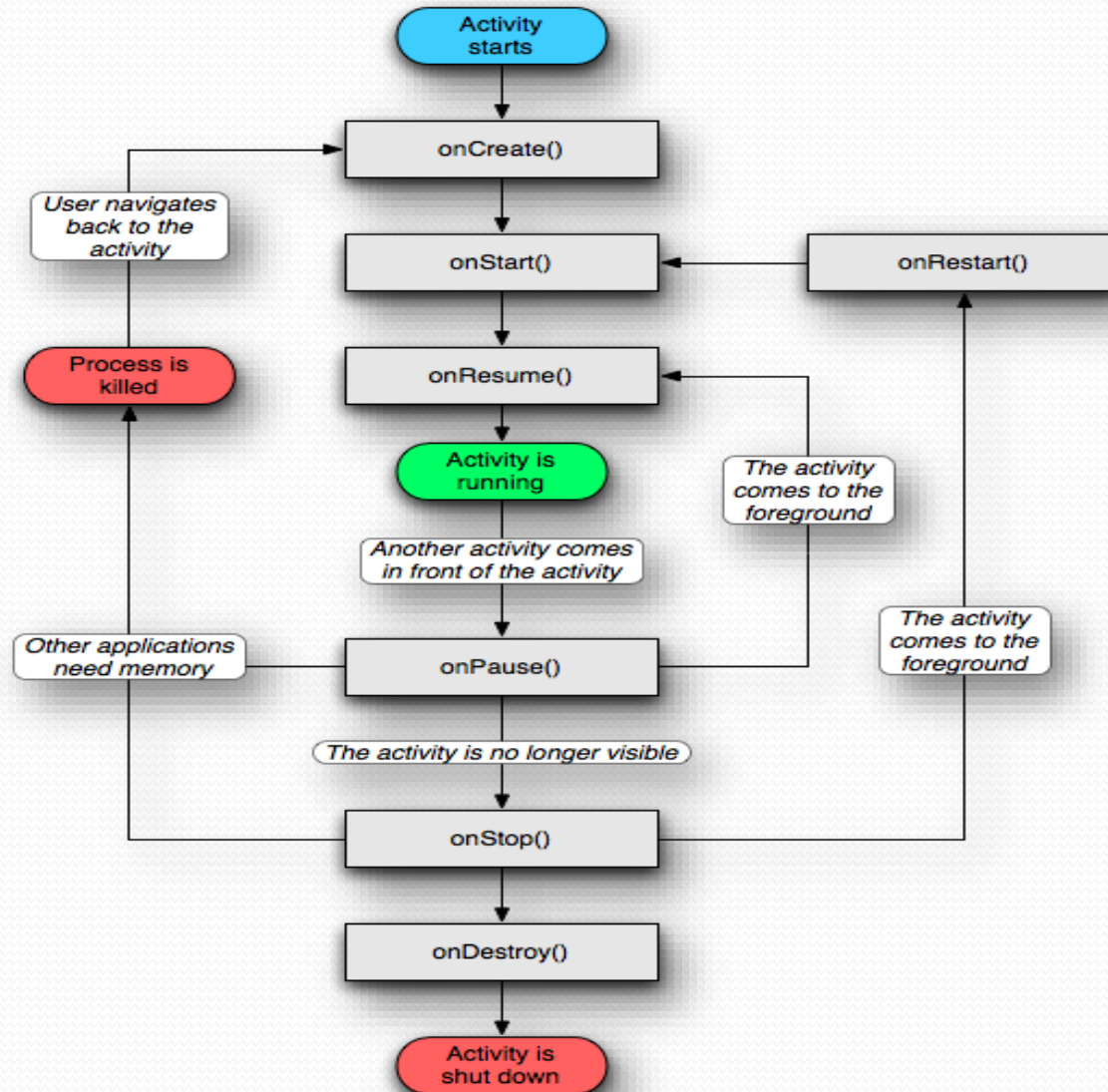
- 当一个Activity处于Pause或Stop的状态时，系统可以要求Activity结束或删除它，当它再度呈现在使用者面前时，要能完整的重新启动及回复先前的状态。

- 应用程序存在与否并非由应用程序所自行决定，而是由Android系统通过运行机制决定。

程序的生命周期(Life Cycle)

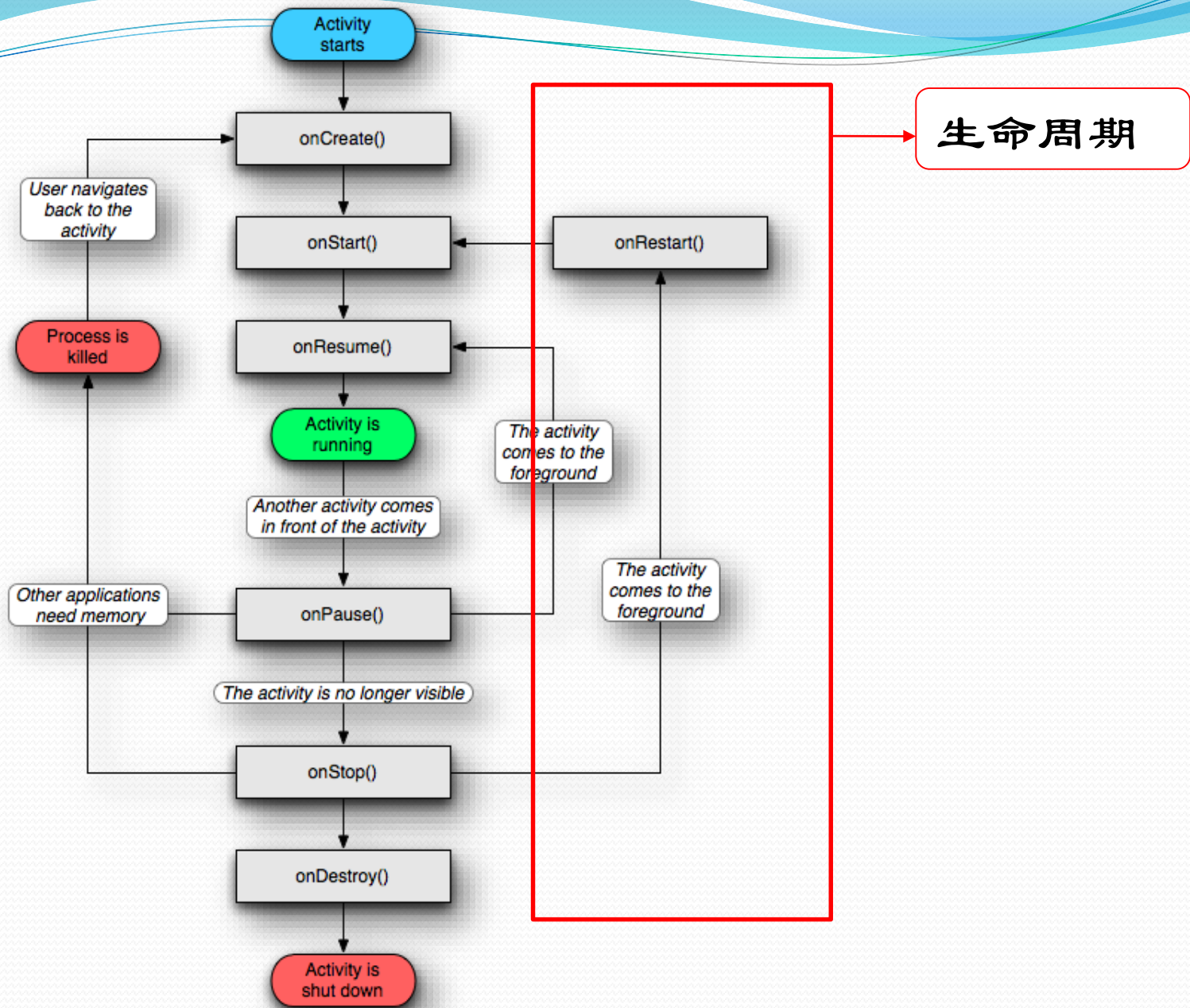
- 活动状态间的切换包含了调用下列几种回调方法：
 - `void onCreate(Bundle savedInstanceState)`
 - `void onStart()`
 - `void onRestart()`
 - `void onResume()`
 - `void onPause()`
 - `void onStop()`
 - `void onDestroy()`

程序的生命周期(Life Cycle)



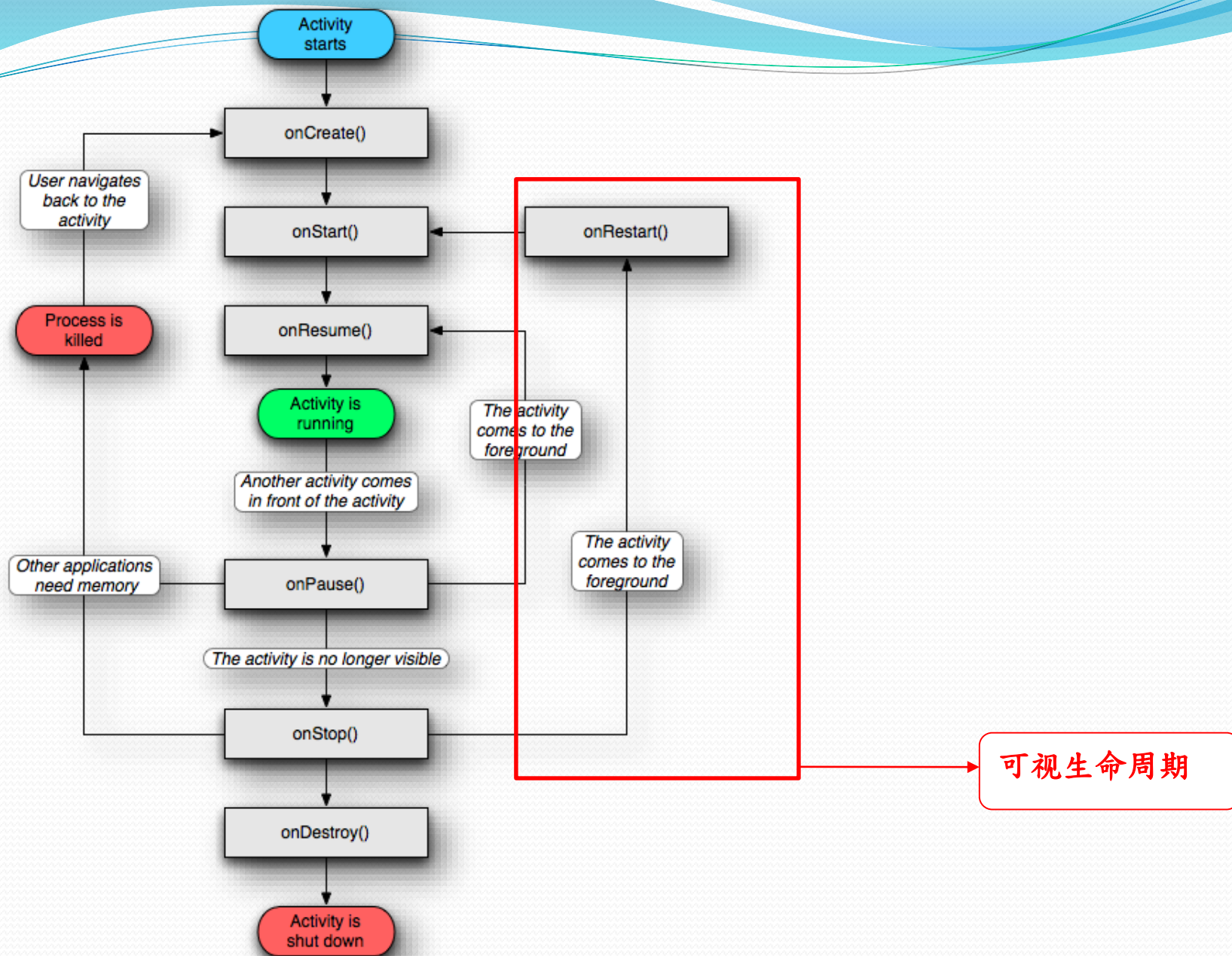
程序的生命周期(Life Cycle)

- 活动的整个生命周期
 - 一个活动的整个生命周期是由onCreate(Bundle)开始，直到onDestroy()结束。
 - 一个活动可以把所有的资源设置写在onCreate中，直到onDestroy()时，再释放出来。



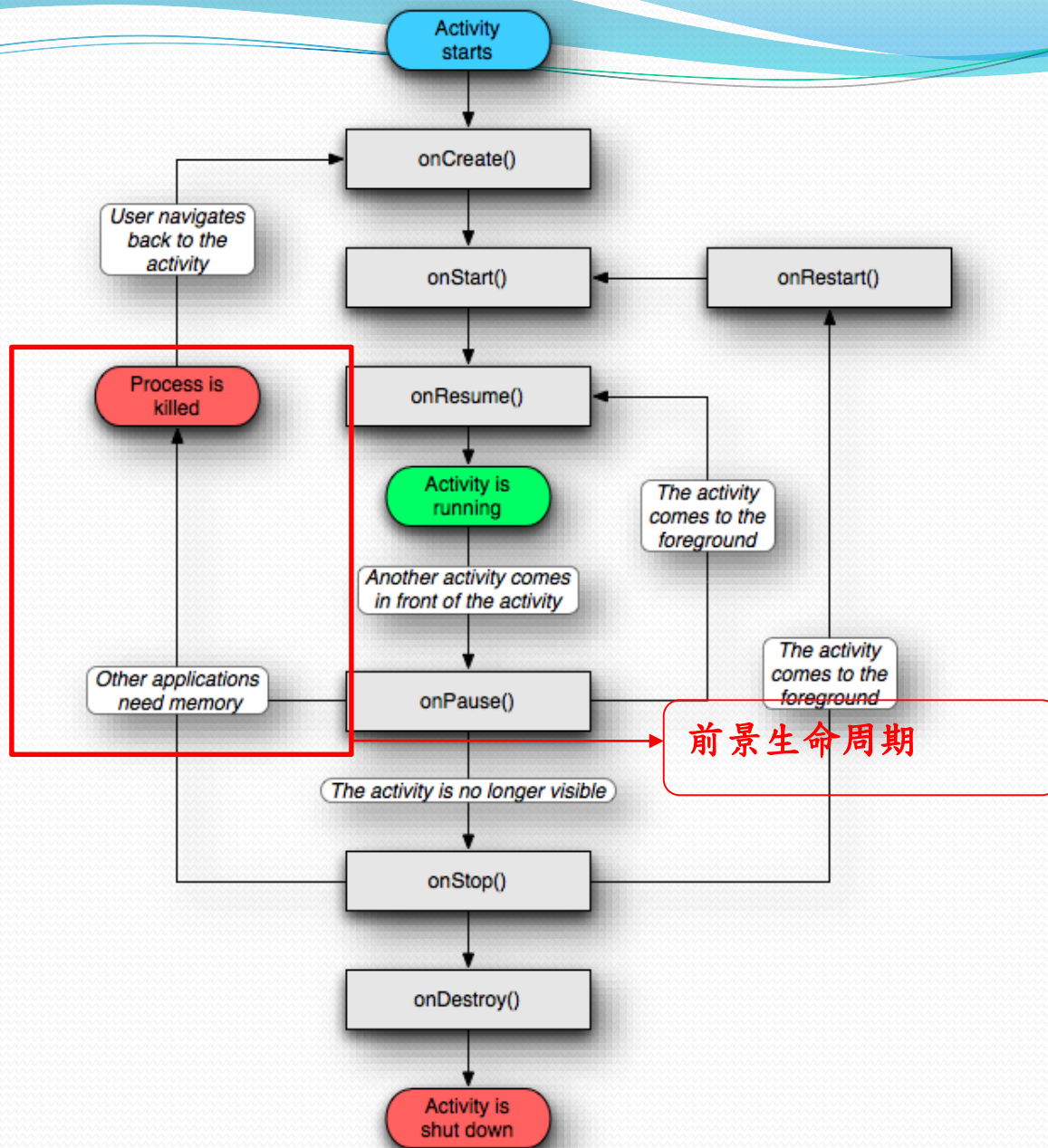
程序的生命周期(Life Cycle)

- 活动的可视生命周期（Visible Lifetime）：
 - 一个活动的VisibleLifetime则是指在onStart()到onStop()之间，称为“可视生命周期”，在这段时间内，用户可以在屏幕上看见Activity，但这个Activity不见得一定在前景跟使用者直接互动。



程序的生命周期(Life Cycle)

- 活动的前景生命周期（Foreground Lifetime）：
 - ForegroundLifetime则是指onResume()到onPause()之间，在这个时期的活动是在所有的活动的前面，并且直接跟使用者进行互动。
 - 一个活动能很频繁的在Resume及Pause这两个状态切换，所以在onResume()及onPause()中实现的程序应尽量精简。



程序的生命周期(Life Cycle)

- 服务

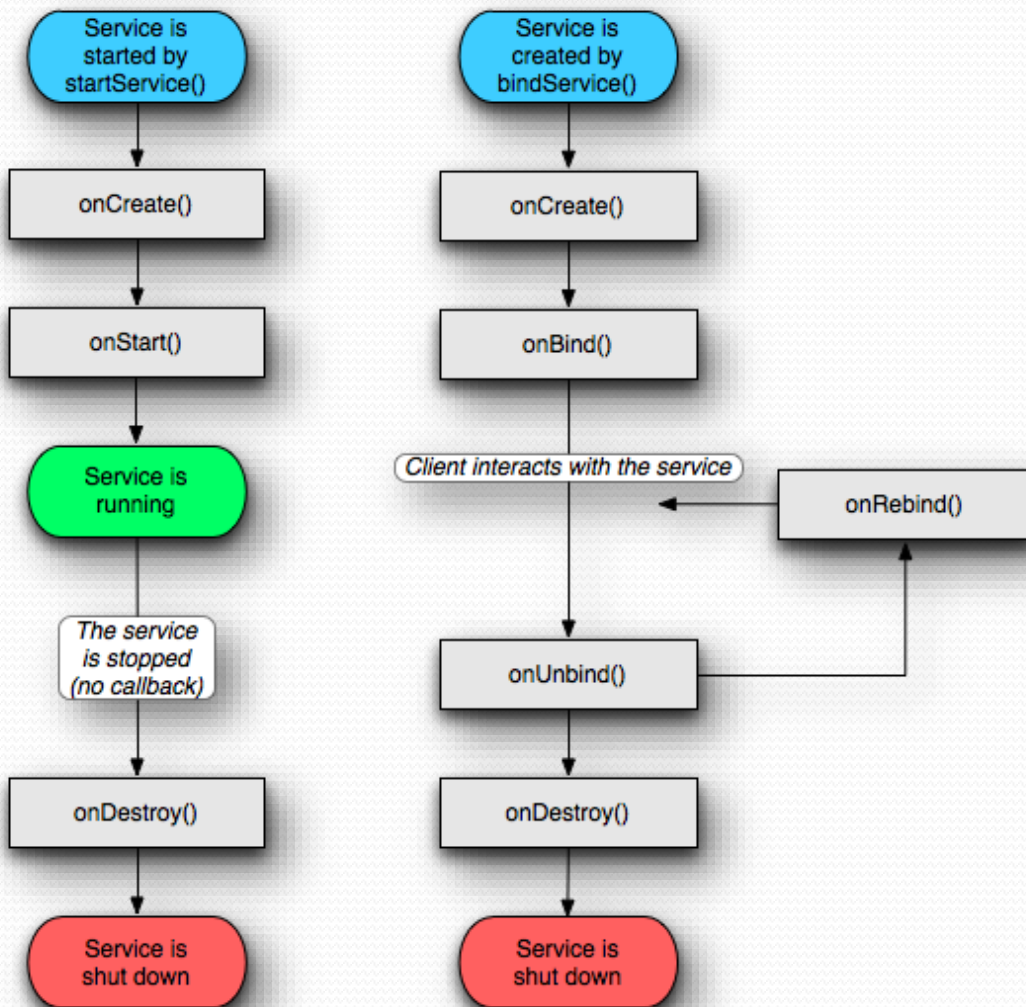
- 在每一个包含服务组件的AndroidManifest.xml文件中，必须有一个相应的<service>声明。
- 服务可以用两种方式调用：
 - 服务自行启动和运行，直到某项操作停止或自行停止时。
 - 它可被提供给对外用户的接口所操作。客户端与服务对象建立连接并调用服务。

程序的生命周期(Life Cycle)

- 服务

- 上述两种模式不是完全分开的，且可以绑定一个被 `startService()` 启动的服务。
- 这种情况下，`stopService()` 不会停止服务，一直到最后一个绑定的连接关闭时。
- 与活动相似，服务也有一些生命周期方法，程序员可以实现它们去改变状态，但方法比活动要少。

程序的生命周期(Life Cycle)



程序的生命周期(Life Cycle)

- 服务的整个生命周期（Entire Lifetime）
 - 服务的整个生命周期是在onCreate()开始，于onDestroy()结束。
- 类似活动（Activity），服务在onCreate()方法中进行初始化，在onDestroy()方法中释放所有系统资源。
- 例如，音乐播放服务可能在onCreate()方法中建立音乐播放线程，在onDestroy()方法中停止线程。

服务的生命周期(Life Cycle)

- 服务的Active Lifetime
 - 在调用startService()方法后，此方法被Intent对象指示去调用onStart()方法。
 - 音乐服務将分析这个Intent来确定播放什么音乐，并开始播放。

程序的生命周期(Life Cycle)

- 广播接收器
 - 广播接收器只有一个生命周期方法：
 - `void onReceive(Context curContext, Intent broadcastMsg)`
 - 当广播消息到达接收者时，Android调用onReceive()方法，并传递保存着讯息的Intent对象。
 - 当调用该方法时，广播接收者被认为是active的。当onReceive()结束时，它就是inactive的。
 - 拥有活动的广播接收者进程，会被保护不被清除。但是只拥有inactive的广播组件时，当系统认为内存应该被其他进程使用时，可以在任何时间被系统清除。

程序的生命周期(Life Cycle)

- 广播接收器

- 当与广播组件频繁互动时，会产生一些问题，因此，某些任务在独立的线程中操作，与管理其他的用户组件的主线程是分开的。
- 如果onReceive()产生线程然后返回，这个完整的进程，包括新产生的线程，将被判断为inactive的，会被标记为将被清除的进程。
- 解决这个问题的办法是为onReceive()启动一个Service，让Service接手这工作，因此系统会认为在进程中仍然有active的工作在进行。

程序的生命周期(Life Cycle)

- 进程与生命周期

- Android系统会尽可能保存各个应用程序进程，但当内存不够用时，系统需要删除较旧的进程。
- 为了判断哪些进程该被保留，哪些进程该被清除，Android把每一个进程放到一个“优先级”列表，组件的运行和状态是基于这个“优先级”列表。

程序的生命周期(Life Cycle)

- **Process and Lifecycle**

- 该优先权分五层。下面的列表显示优先权层次顺序：
 - 前景进程
 - 可视进程
 - 服务进程
 - 背景进程
 - 空进程

程序的生命周期(Life Cycle)

- Android会基于拥有当前活动的进程中，各组件重要性，尽量给予它最高级别。
- 运行服务的线程比运行背景活动的进程级别要高，一个活动需要开始执行一个长时间操作时，应启动一个服务，而不是自行建立一个线程。
- 同样的原因，广播接收器也应该调用服务而不是将一个长时间操作放到自行建立的线程中处理。

- 1、Android应用程序框架
- 2、程序的生命周期
- 3、Activity启动方式
- 4、Manifest的定义
- 5、资源文件设计

Activity的启动方式

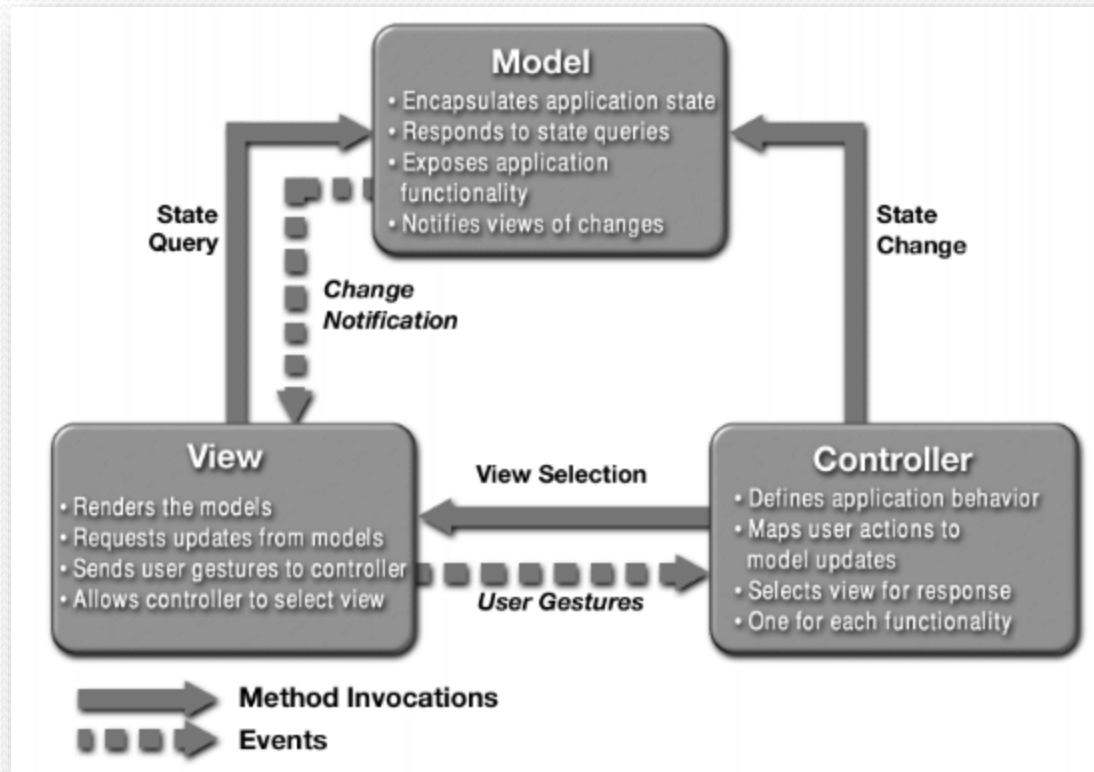
- 在android里，有4种activity的启动模式，分别为：
 - “standard” (默认)
 - “singleTop”
 - “singleTask”
 - “singleInstance”

- 1、Android应用程序框架
- 2、程序的生命周期
- 3、Activity启动方式
- 4、Manifest的定义
- 5、资源文件设计

MVC设计模式的好处

- MVC(Model-View-Controller，模型-检视-控制器模式)用于表示一种软件架构模式
- MVC把软件分为三个基本部分：
 - 模型(Model)
 - 检视(View)
 - 控制器(Controller)

MVC设计模式的好处

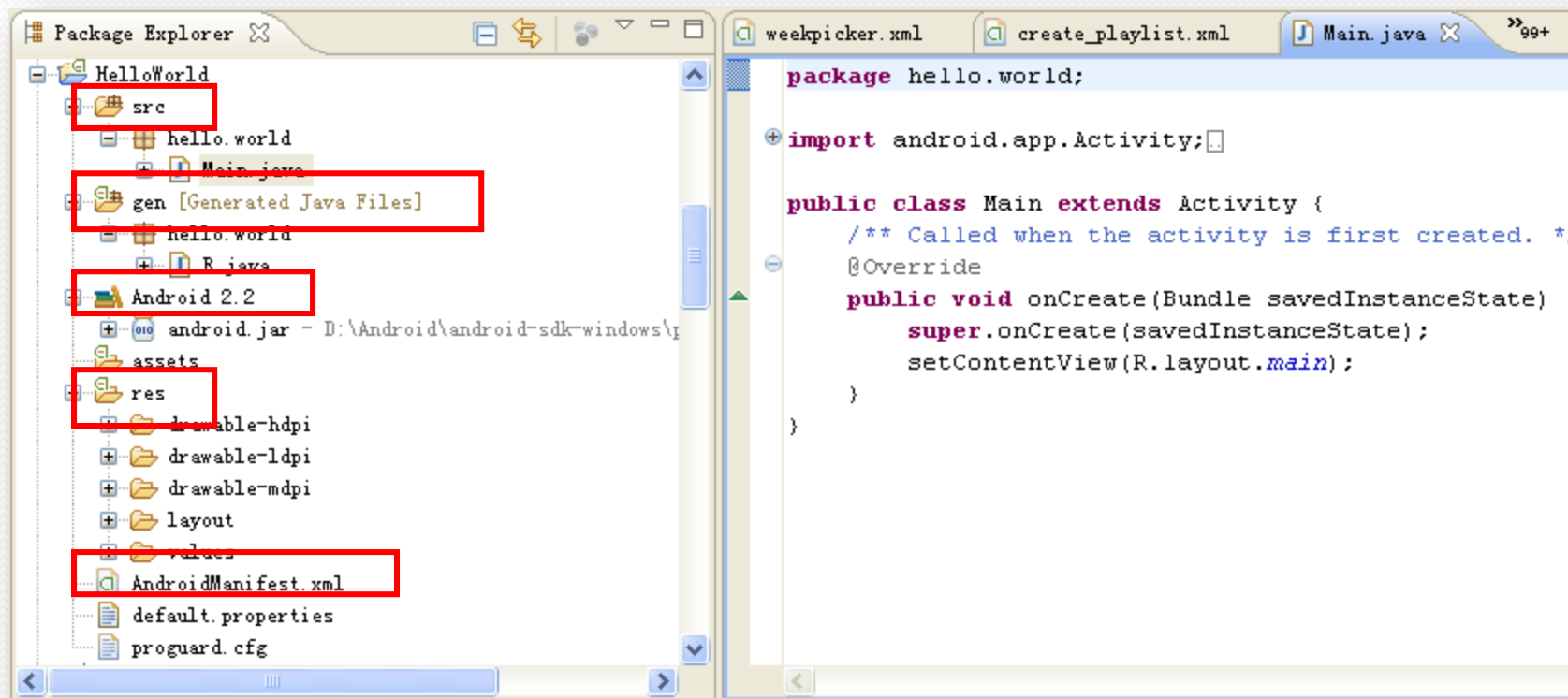


MVC模式

MVC设计模式的好处

- MVC架构起始于一个GUI(graphical user interface design patter)原型。
- 其目的是实现动态程序设计，使日后对于程序修改及扩展更加便利，并使某些程序代码可重复利用。
- 另外通过对复杂度的简化，使程序结构更加直觉。

应用程序架构

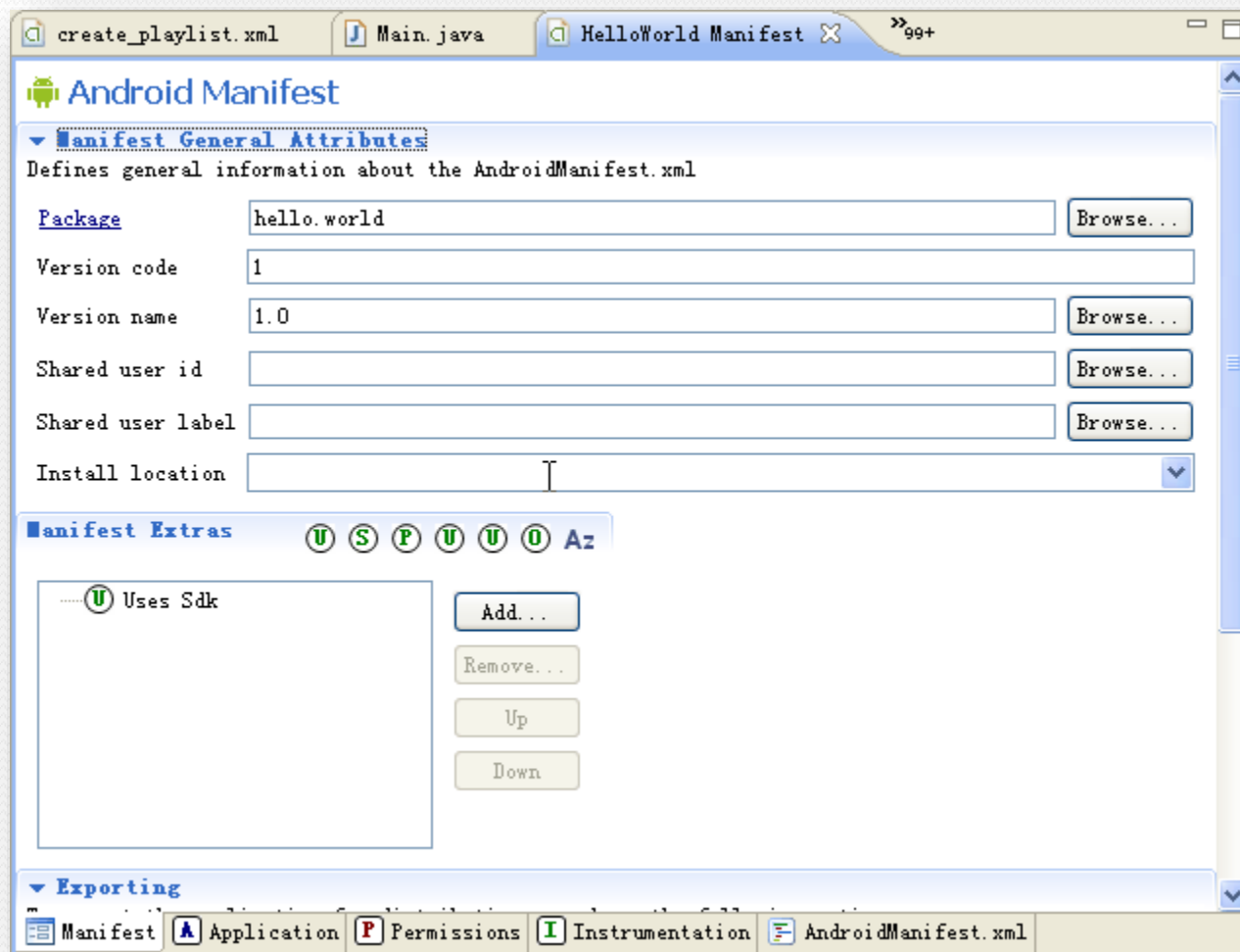


- 1、Android应用程序框架
- 2、程序的生命周期
- 3、Activity启动方式
- 4、Manifest的定义
- 5、资源文件设计

AndroidManifest定义

- AndroidManifest定义文件是一个用来描述应用程序「整体信息」的配置文件。
- 每个应用程序都需要AndroidManifest.xml，它提供了应用程序的必要信息给Android系统使用。

AndroidManifest定义



AndroidManifest定义文件

- 此定义文件的重点就是其中的目标过滤器(Intent Filters)，这些过滤器描述了什么时间及情况下让Activity启动。
- 除了描述应用程序的活动、内容提供者、服务和Intent接收器，也可以在AndroidManifest.xml档中指定权限和安全控制测试。

AndroidManifest定义

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="hello.world"
    android:versionCode="1"
    android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Main"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

AndroidManifest定义

- <manifest>为文件根节点，描述了程序的所有内容，在其节点下面内可放置各种标签：
 - <uses-permission>
 - <permission>
 - <instrumentation>
 - <application>

AndroidManifest定义

- 在<application>中可以有零个或多个以下的组件：
 - <activity>
 - <intent-filter>
 - <action>
 - <category>
 - <data>
 - <meta-data>
 - <receiver>
 - <service>
 - <provider>

- 1、Android应用程序框架
- 2、程序的生命周期
- 3、Activity启动方式
- 4、Manifest的定义
- 5、资源文件设计

Android资源文件设计

- 文字资源文件- strings.xml
- 颜色设置资源文件 - colors.xml
- 尺寸定义资源文件 - dimens.xml
- 样式资源文件 - styles.xml
- 窗口布局资源文件 - layout\main.xml
- 动画资源文件 - anim.xml
- 图文件资源目录 - drawable

文字资源文件

- 文字资源文件(strings.xml)
 - 开始从最常使用的文字资源文件strings.xml学习

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<resources>
```

```
    <string name="hello">Hello World, Main!</string>
```

```
    <string name="app_name">HelloWorld</string>
```

```
</resources>
```

Android资源文件设计

- 可以在JAVA原始文件中使用这些变量：
 - 用法：R.string.字符串名称
 - 范例：String hello = this.getString(R.string.hello);

```
package hello.world;
import android.app.Activity;
import android.os.Bundle;

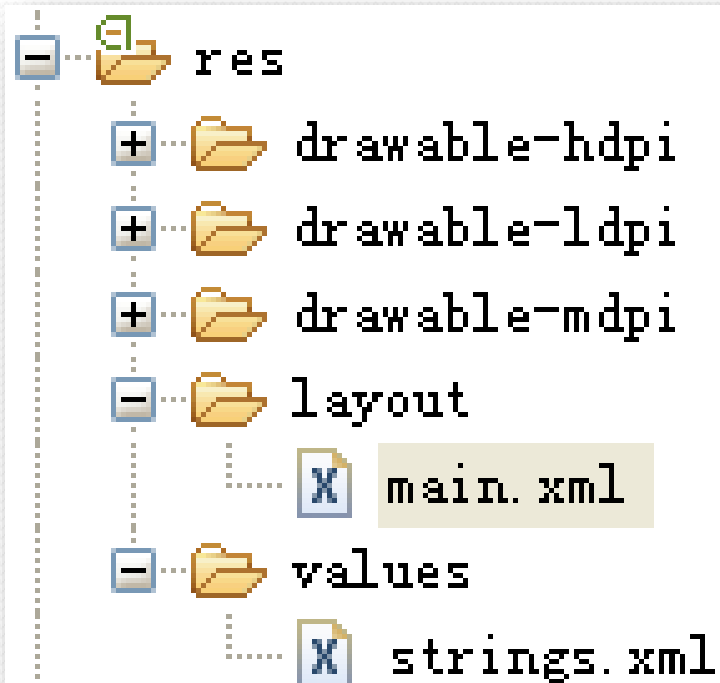
public class Main extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        String hello = getString(R.string.hello);
    }
}
```

Android资源文件设计

- 也可以让XML资源文件使用字符串资源：
 - 用法：@string/字符串名称
 - 范例：android:app_name="@string/hello"

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  >
  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
  />
</LinearLayout>
```

Android资源文件设计






































颜色设置资源文件

- 颜色设置资源文件 (colors.xml)
 - 在Android中的颜色代码类似网页中的颜色代码，都是采用16进位的方式
 - Android支持的颜色语法有：#RGB、#ARGB、#RRGGBB、#AARRGGBB四种

颜色设置资源文件

以#AARRGGBB作为范例:

 Black	#FF000000	 HotPink	#FFFF69B4	 Orchid	#FFDA70D6
 BlanchedAlmond	#FFFEBBCD	 IndianRed	#FFCD5C5C	 PaleGoldenrod	#FFEE8AA
 Blue	#FF0000FF	 Indigo	#FF4B0082	 PaleGreen	#FF98FB98
 BlueViolet	#FF8A2BE2	 Ivory	#FFFFFFF0	 PaleTurquoise	#FFAFEEEE
 Brown	#FFA52A2A	 Khaki	#FFF0E68C	 PaleVioletRed	#FFD87093
 BurlyWood	#FFDEB887	 Lavender	#FFE6E6FA	 PapayaWhip	#FFF9FD5
 CadetBlue	#FF5F9EA0	 LavenderBlush	#FFFFFF0F5	 PeachPuff	#FFFDAB9
 Chartreuse	#FF7FFF00	 LawnGreen	#FF7CFC00	 Peru	#FFCD853F
 Chocolate	#FFD2691E	 LemonChiffon	#FFFFACD	 Pink	#FFFC0CB
 Coral	#FFFF7F50	 LightBlue	#FFADD8E6	 Plum	#FFDDA0DD
 CornflowerBlue	#FF6495ED	 LightCoral	#FFF08080	 PowderBlue	#FFB0E0E6
 Cornsilk	#FFFFF8DC	 LightCyan	#FFE0FFFF	 Purple	#FF800080
 Crimson	#FFDC143C	 LightGoldenrodYellow	#FFFAFAD2	 Red	#FFFF0000
 Cyan	#FF00FFFF	 LightGray	#FFD3D3D3	 RosyBrown	#FFBC8F8F
 DarkBlue	#FF00008B	 LightGreen	#FF90EE90	 RoyalBlue	#FF4169E1
 DarkCyan	#FF008B8B	 LightPink	#FFFB6C1	 SaddleBrown	#FF8B4513
 DarkGoldenrod	#FFB8860B	 LightSalmon	#FFFA07A	 Salmon	#FFFA8072
 DarkGray	#FFA9A9A9	 LightSeaGreen	#FF20B2AA	 SandyBrown	#FFFA4A60
 DarkGreen	#FF006400	 LightSkyBlue	#FF87CEFA	 SeaGreen	#FF2E8B57
 DarkKhaki	#FFBDB76B	 LightSlateGray	#FF778899	 SeaShell	#FFFFFF5EE

颜色设置资源文件

- 在res/values/目录下新增colors.xml，就可以编辑并使用<color>标签设定资源文件。
- 先设定<color>标签变量名称，接在设定black的颜色代码。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<color name="black">
#ff000000
</color>
</resources>
```

颜色设置资源文件

- 同样地，当设置完资源文件后，如果要被拿来使用的话，可使用与文字资源文件一样的方法来调用。
 - 在JAVA源文件中使用
 - 在XML文件中使用

颜色设置资源文件

- 在JAVA源文件中使用
 - 用法：R.color.颜色常数名称
 - 范例：`getResources().getColor(R.color.black);`

颜色设置资源文件

- 在XML文件中使用
 - 用法：@color/颜色变量名称
 - 范例：android:textColor="@color/black"

```
<TextView android:layout_width="fill_parent"  
android:layout_height="wrap_content"  
android:text="@string/hello"  
android:textColor="@color/black" />
```

颜色设置资源文件

- 另外也可在colors.xml中使用<drawable>图形颜色标签。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#ff000000</color>
    <drawable name="grey">#ffffff00</drawable>
</resources>
```

颜色设置资源文件

- 在JAVA源文件中使用
 - 用法：R.drawable.颜色变量名称
 - 范例：Drawable blueColor =
getResources.getDrawable(R.drawable.grey);

颜色设置资源文件

- 在XML文件中使用
 - 用法：@drawable/颜色变量名称
 - 范例：android:background="@drawable/blue"

尺寸定义资源文件

- 尺寸定义资源文件 - `dimens.xml`
 - 首先于`res/values`中新增`dimens.xml`，此文件可针对字符串个别设定字号，像是`px`、`in`、`mm`、`pt`、`dp`、`dip`、`sp`等等尺寸。

尺寸定义资源文件

尺寸定义资源文件(dimens.xml)

px(Pixel)	以画面真实的像素做为单位
mm(Millimeter)	以画面的毫米为单位
in(inches)	以画面的英吋作为单位
pt(Points)	一点的单位为1/72英吋
dp&dip	相对于160dpi的屏幕中的一个像素
sp	随着屏幕大小改变的一个像素

尺寸定义资源文件

- 在JAVA源文件中使用
 - 用法：R.dimen.尺寸变量名称
 - 范例：`float dimen = getResource().getDiemnsion(R.dimne.px);`

尺寸定义资源文件

- 在XML文件中使用
 - 用法：@dimen/尺寸变量名称
 - 范例：android:textSize="@dimen/px"

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    android:textColor="@color/black"
    android:background="@drawable/blue"
    android:textSize="@dimen/px"
/>
```

样式资源文件

- 样式资源文件 - styles.xml
 - Style资源文件是类似一般手机上可套用的主题，它可以整合许多属性，并提供给系统使用。
 - 在此文件中主要是使用<style>定义手机程序布局，并加入<item>标签作细项设定。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<style name="music_style">
    <item name="android:background">#ffff0000</item>
    <item name="android:textColor">#ff00ff00</item>
</style>
</resources>
```

样式资源文件

- 在JAVA源文件中使用
 - 用法：R.style.样式变量名称
 - 范例：setTheme(R.style.music_style);

```
/** Called when the activity is first created. */  
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    setTheme(R.style.music_style);  
}
```

样式资源文件

- 在XML文件中使用
 - 用法：@style/样式变量名称
 - 范例：android:theme="@style/music_style"

窗口布局资源文件

- 窗口布局资源文件 - layout\xxx.xml
 - 在Android平台里，用户界面都是通过ViewGroup或View类别来显示，ViewGroup和View是Android平台上最基本的用户界面组件。
 - 我们可以通过程序直接调用的方法，或是使用XML文件，来描述用户界面。

窗口布局资源文件

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
    <TextView android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        android:textColor="@color/black" />
</LinearLayout>
```

窗口布局资源文件

- 在JAVA中使用：
 - 用法：R.layout.布局文件名
 - 范例：setContentView(R.layout.main);

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    setTheme(R.style.music_style);
}
```

动画资源文件

- 动画资源文件 - anim.xml
 - 首先在res底下建立anim文件夹，并在文件夹中建立xxx.xml，通过这个资源文件可以达成程序里面的一些动画效果
- Animation主要有两种动画模式：一种是补间动画，另一种是帧动画。
 - 补间动画又分为Alpha Animation（透明度转换）、RotateAnimation（旋转转换）、ScaleAnimation（缩放转换）、TranslateAnimation（位置转换）。

动画资源文件定义

anim1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- 淡出 -->
<set
xmlns:android="http://schemas.android.com/apk/res/android">
<alpha android:fromAlpha="1.0" android:toAlpha="0.0"
android:duration="500"/>
</set>
```

anim2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- 淡入 -->
<set
xmlns:android="http://schemas.android.com/apk/res/android">
<alpha
android:fromAlpha="0.0" android:toAlpha="1.0"
android:duration="500"/>
</set>
```

动画资源文件

- 动画资源的使用方法：
 - 首先要引入相关的Package，加入
`android.view.animation.AnimationUtils;`

```
import android.app.Activity;  
import android.os.Bundle;  
import android.view.animation.AnimationUtils;  
import android.graphics.drawable.Drawable;  
import java.lang.CharSequence;
```

动画资源文件

- 在JAVA中使用：
 - 用法：R.anim.动画资源文件名称
 - 范例：
 - `Animation ani = AnimationUtils.loadAnimation(this, R.anim.anim);`
`View.startAnimation(ani);`
 - `this.overridePendingTransition(R.anim.anim2, R.anim.anim1);` //淡入淡出

图文件资源目录

- 图文件资源目录 (drawable)
 - 所有程序的图标、背景图片等等，皆需放在drawable目录下
 - Android会为每个放置在res/drawable目录下的图片文件产生一变量，变量名称就是这个图片的文件名(不包含扩展名)，可在R.java文件中的drawable中查询。

图文件资源目录

- 在JAVA源文件中使用
 - 用法：R.drawable.图片名称
 - 范例：`Drawable bitmap = getResources().getDrawable(R.drawable.icon);`

图文件资源目录

- 在XML文件中使用
 - 用法：@drawable/图片名称
 - 范例：android:background="@drawable/icon"

Q&A



谢谢！

