Android移动开发

Android Mobile Application Development

第4讲 Activity活动

吴以凡
计算机学院 一教505
yfwu@hdu.edu.cn

Android组件

Android组件

- 组件是可以调用的基本功能模块，Android应用程序就是由一个或多个基本组件组成的

- Android系统有四个重要的组件，分别是
  - 活动（Activity）
  - 服务（Service）
  - 广播（Broadcast）
  - 内容提供器（ContentProvider）

Android组件

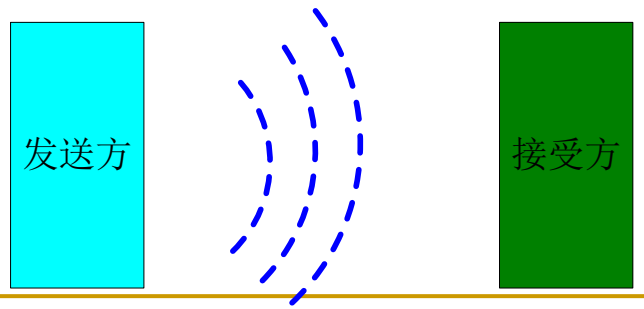
- 活动Activity
  - Activity是Android程序的呈现层，显示可视化的用户界面，并接收与用户交互所产生的界面事件
  - Android应用程序可以包含一个或多个Activity，一般在程序启动后会呈现一个Activity，用于提示用户程序已经正常启动
  - 在界面上的表现形式：全屏窗体，非全屏悬浮窗体，对话框

Android组件

- 服务Service
  - 服务是Android应用程序中具有较长的生命周期但是没有用户界面的代码程序。
    - 在后台运行
    - 跟Activity的级别差不多，但是它不能自己运行，需要通过某一个Activity来调用
  - 典型的例子是媒体播放器：
    - 边看短信，边听音乐
    - 边看新闻，边听音乐

Android组件

- 广播Broadcast
  - Broadcast是用来在组件之间传播数据（Intent）的一种机制
  - 广播将组件分为发送方和接受方
  - 广播的发送者和接收者事先是不需要知道对方的存在的
    - 优点就是系统的各个组件可以松耦合地组织在一起，使得系统具有高度的可扩展性，容易与其它系统进行集成



Android组件

- 内容提供器ContentProvider
  - ContentProvider是Android系统提供的一种共享数据的机制，供多个应用之间进行数据共享。
  - 一个应用程序可以通过实现一个ContentProvider的抽象接口将自己的数据暴露出去。
  - Android系统内部也提供一些内置的ContentProvider，能够为应用程序提供重要的数据信息

活动Activity

Activity基础

- Activity： “活动”，是在Android应用中负责与用户交互的组件
- 通过setContentView(View)来显示布局文件中已经定义的组件
- Activity就像一个界面管理员，用户在界面上的操作是通过Activity来管理的

创建Activity

- 创建一个Activity的具体步骤：
  - 定义一个类继承自android.app.Activity或其子类
  - 在res/layout目录下创建一个xml文件，用于创建Activity的布局
  - 在app/manifests目录下的AndroidManifest.xml清单文件中注册Activity
  - 重写Activity的onCreate()方法，并在该方法中使用setContentView()加载指定的布局文件

创建Activity

- 定义Activity类

android.app.Activity

```
package com.hdu.course.helloworld;
import android.app.Activity;
import android.os.Bundle;

public class SecondActivity extends Activity {
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.second_activity);
   }
}
```

创建Activity

- 定义Activity类（向低版本Android兼容）

android.support.v7.app.AppCompatActivity

```
package com.hdu.course.helloworld;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class SecondActivity extends AppCompatActivity {
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.second_activity);
   }
}
```

创建Activity

- 在app/manifests/AndroidManifest.xml清单文件中注册新建的Activity

```
<activity android:name=“.SecondActivity”>

</activity>
```

Activity的常用事件

- 重写android.app.Activity类的方法从而使自定义的Activity在状态改变时执行用户所期望的操作

```
//响应按键按下事件
public boolean onKeyDown(int keyCode, KeyEvent event){
   Toast.makeText(this,"按键已经按下了！",Toast.LENGTH_SHORT).show();
   return super.onKeyDown(keyCode, event);
}
//响应按键松开事件
public boolean onKeyUp(int keyCode, KeyEvent event){
   Toast.makeText(this,"按键松开了！",Toast.LENGTH_SHORT).show();
   return super.onKeyDown(keyCode, event);
}
//响应屏幕触摸操作
public boolean onTouchEvent(MotionEvent event){
   Toast.makeText(this,"触摸了屏幕！",Toast.LENGTH_SHORT).show();
   return super.onTouchEvent(event);
}
```

启动另一个Activity

- 例子：
  - 假设你想让用户看到Internet上的某个图片。
  - 当前有一个Activity具有打开Internet上的某个图片的功能，那么“宿主Activity”只需将请求信息放到一个Intent对象里面，并把它传递给startActivity()或startActivityForResult()
  - 然后浏览器就会显示指定link的图片。而当用户按下BACK键的时候，宿主Activity又会再一次的显示在屏幕上。
- 启动Activity的为“宿主Activity”，被启动的Activity为“随从Activity”
- 有三种方式来启动另外一个Activity：
  - 启动同一个Application 的Activity
  - 启动不同Application 的Activity
  - 启动系统自带的Application 的Activity
- Activity之间通过Intent来传递消息

启动另一个Activity

- 启动同一Application的Activity

```
public class MainActivity extends Activity {
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Intent intent = new Intent();
      intent.setClass(MainActivity.this, SecondActivity.class);
      startActivity(intent);
   }
}
```

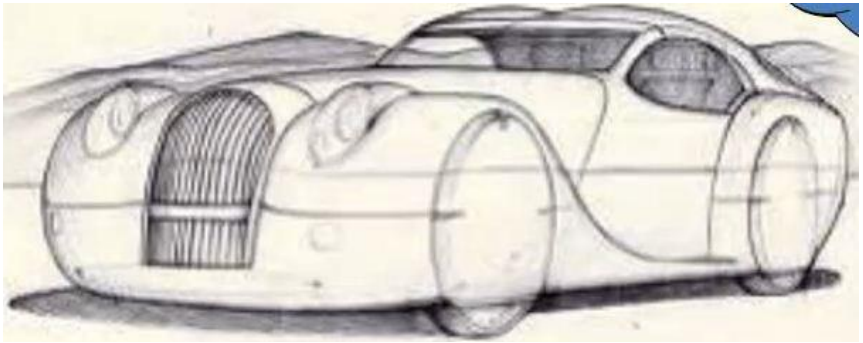
```
public class SecondActivity extends Activity {
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout. second_activity);
   }
}
```

Activity间的消息传递

- Android使用Intent （意图）对象来描述应用中一次操作的动作、数据和附加数据，系统通过该Intent对象的内容来进行调用
- Intent提供了多个Activity之间进行交互的方式，可用于不同的Activity之间传递消息
- 应用程序可通过startActivity()方法指定相应的Intent对象来启动另外一个Activity

“意图” Intent

- Intent
  - 不同组件之间通信的载体，在组件之间传递数据
    - 启动应用中另一个Activity
    - 启动一个Service组件
    - 发送一条广播消息
  - “意图”：Intent对象封装了大量关于目标组件的信息



Activity间的消息传递

- 使用Intent进行数据传递的方法

- 放置数据
  - putExtras(Bundle extras)
  - putExtra(String name, XXX value)
    - putExtra(String name, String value)
    - putExtra(String name, boolean value)
    - putExtra(String name, float value)
    - …
- 接收数据
  - Bundle getExtras()
  - XXX getXXXExtra(String name)
    - String getStringExtra(String name)
    - boolean getBooleanExtra(String name)
    - float getFloatExtra(String name)
    - …

Bundle

- 如果要将当前Activity的数据传递给下一个Activity，可使用Bundle来协助完成
- Bundle对象可被理解成一个哈希表，该映射表建立了键（key,标识）与其值（value,传递的数据）的映射关系
- 通过Bundle类的putXXX(Key, Value)方法将数据封装到Bundle对象中，如putString(String key, String value)
- 通过Bundle的getXXX(String key)方法取得关键字对应的数据
- 可用的函数：
  - get getBoolean getBundle getChar getInt getString ...
  - putAll putBoolean putBundle putChar putInt putString ...

Activity间的消息传递

- “宿主Activity”端的流程
  - 首先创建一个Intent和Bundle对象，其中Bundle用于存储传递的数
  - 然后使用Bundle的put方法输入要传递的数据
  - 将要传递的数据压入Intent中
  - 启动“随从Activity”
- “随从Activity”端的流程
  - 接收“宿主Activity”的Intent
  - 获得传入的Bundle对象
  - 使用Bundle的get方法获取要传递的数据

Activity间的消息传递

```
...
Bundle mybundle = new Bundle();
mybundle.putString("selected_radionbutton", (String) radioButton1.getText());
mybundle.putString("phone","123456");
mybundle.putString("age","18");
myintent.putExtras(mybundle);
MainActivity.this.startActivity(myintent);
...
```

```
...
Intent myintent = this.getIntent();
Bundle mybundle = myintent.getExtras();
String selected_radiobutton = mybundle.getString("selected_radionbutton");
String phone = mybundle.getString(”phone”);
String age= mybundle.getString(”age”);
```

Activity间的消息传递

- putExtras与putExtra的关系

```
Intent intent = new Intent(this,xxx.class);
intent.putExtra("test", true);

startActivity(intent);
```

```
Intent intent = new Intent(this,xxx.class);
Bundle bundle = new Bundle();
bundle.putBoolean("test", true);
intent.putExtras(bundle);

startActivity(intent);
```

生命周期

Android应用程序生命周期

- 程序的生命周期
  - 每一个Android应用程序在运行时，对于底层的Linux Kernel而言都是一个单独的进程
  - 程序的生命周期是在Android系统中进程从启动到终止的所有阶段，也就是Android程序启动到停止的全过程
  - 程序的生命周期由Android系统进行调度和控制的

Android应用程序生命周期

- Android系统中的进程优先级由高到低
  - 前台进程 foreground process
  - 可见进程 visible process
  - 服务进程 service process
  - 缓存进程 cached process

Android应用程序生命周期

- 前台进程 foreground process
  - 前台进程是Android系统中最重要的进程，是与用户正在交互的进程，包含以下四种情况
    - 进程中的Activity正在与用户进行交互
    - 进程服务被Activity调用，而且这个Activity正在与用户进行交互
    - 进程服务正在执行生命周期中的回调函数，如onCreate()、onStart()或onDestroy()
    - 进程的BroadcastReceiver正在执行onReceive()函数

Android应用程序生命周期

- 可见进程
  - 可见进程指部分程序界面能够被用户看见，却不在前台与用户交互，不响应界面事件的进程
  - 如果一个进程包含服务，且这个服务正在被用户可见的Activity调用，此进程同样被视为可见进程
  - Android系统一般存在少量的可见进程，只有在特殊的情况下，Android系统才会为保证前台进程的资源而清除可见进程

Android应用程序生命周期

- 服务进程
  - 服务进程是指包含已启动服务的进程
    - 没有用户界面
    - 在后台长期运行
  - Android系统除非不能保证前台进程或可见进程所必要的资源，否则不强行清除服务进程

Android应用程序生命周期

- 缓存进程
  - 缓存进程是不包含任何活跃组件的进程—被用户主动关闭的Activity对应的进程
  - 缓存进程在系统资源紧张时会被首先清除
  - 但为了提高Android系统应用程序的启动速度，Android系统会将缓存进程保存在系统内存用，在用户重新启动该程序时，缓存进程会被重新使用

Android组件的生命周期

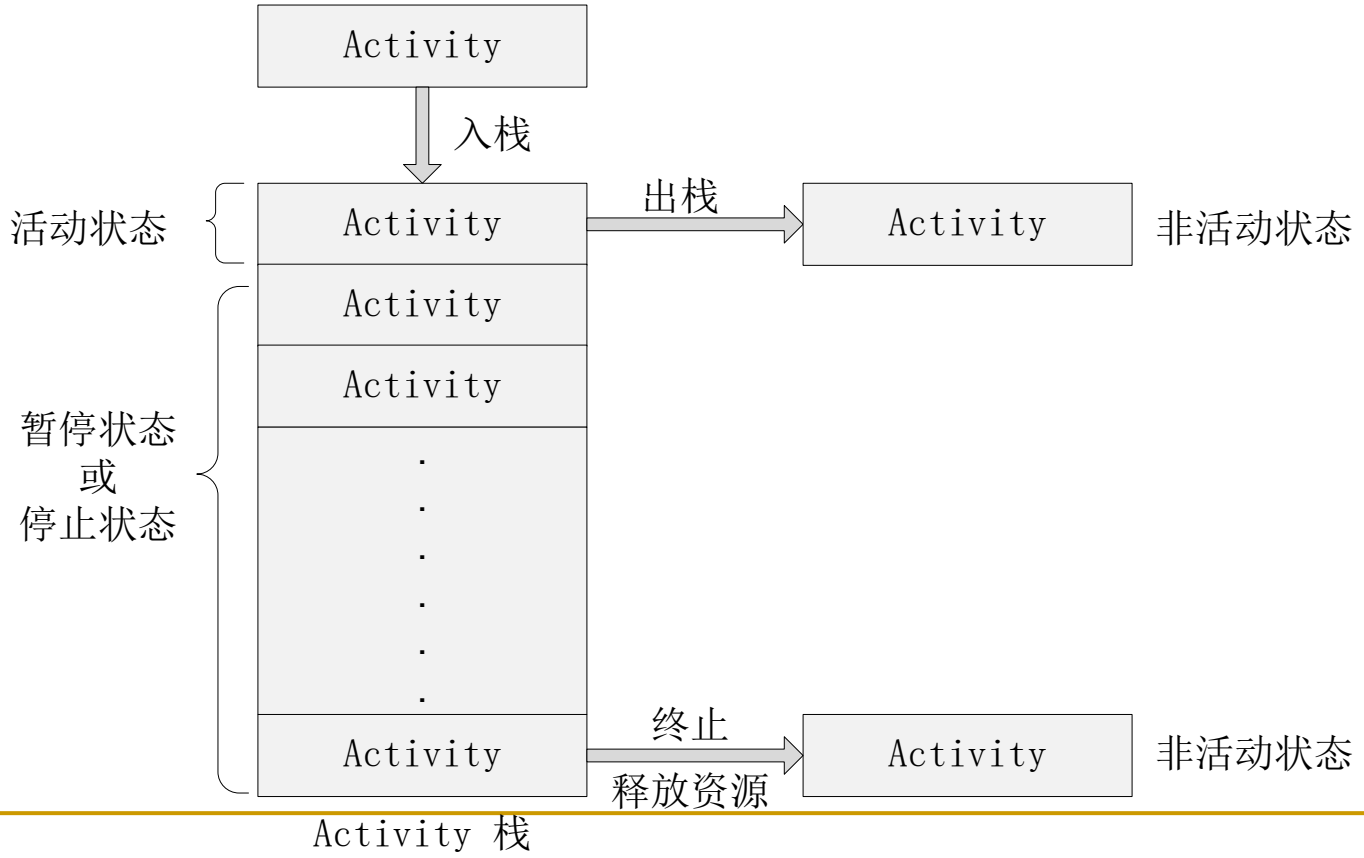
- 组件生命周期
  - 所有Android组件都具有自己的生命周期，是从组件建立到组件销毁的整个过程
  - 在生命周期中，组件会在可见、不可见、活动、非活动等状态中不断变化

Activity栈管理机制

- 大多数的应用程序根据功能的需要都是由多个界面组成，因此大部分的Android应用中也就必须包含多个Activity类
- Android系统使用栈(activity stack, back stack)来管理多个Activity
- 栈是一种先进后出的数据结构，处于顶端的元素总是被先处理。
  - 当一个新的Activity启动的时候，它首先会被放置在Activity栈顶部并成为运行状态的Activity
  - 只有当这个新的Activity退出以后，之前Activity才能重新回到前台界面。

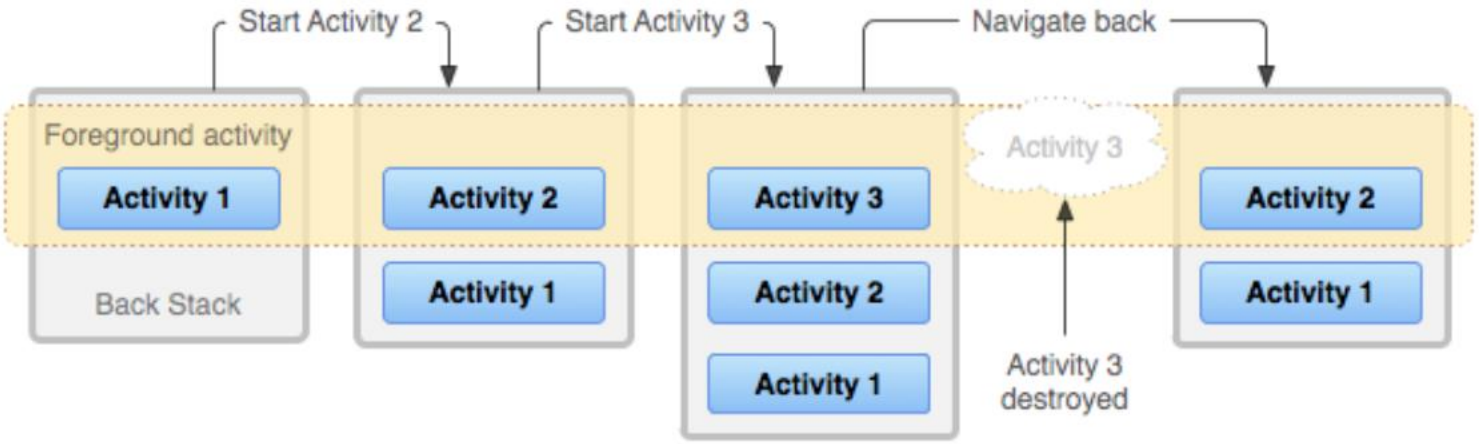
Activity栈管理机制

- Activity栈
  - 遵循“后进先出”的规则---用户退出或被系统终止的Activity可以提前出栈
  - 当前活动的Activity位于栈顶



Activity栈管理机制

- 当前有Activity 1, 处于运行状态，在栈顶
- Activity 2 被启动后， Activity 2处于栈顶， Activity 1处于暂停状态
- Activity 3 被启动后， Activity 3处于栈顶， Activity 1与Activity 2处于暂停状态
- 用户按“后退”按钮后，Activity 3的窗口被关闭，Activity 3被销毁，Activity 2由暂停状态转成运行状态，这时处于运行状态的Activity 2被置于栈顶



Activity生命周期

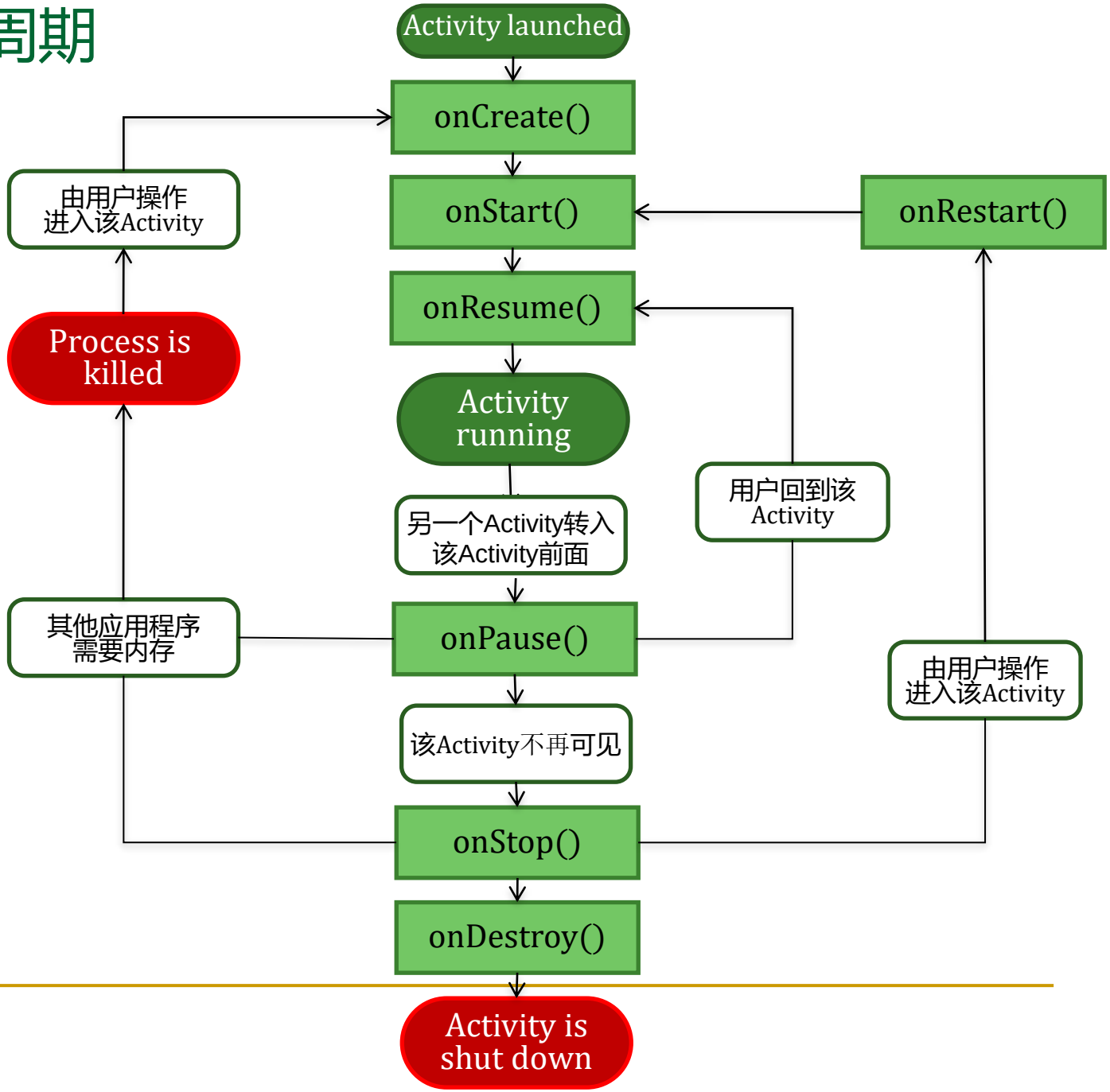
- Activity生命周期
  - Activity生命周期指Activity从启动到销毁的过程
  - Activity表现为四种状态
    - **活动状态（运行状态）：**Activity在用户界面中处于最上层（前台），完全能被用户看到，能够与用户进行交互（获得焦点）
      - 有且只有1个Activity处于运行状态
    - **暂停状态：**Activity在界面上被部分遮挡，该Activity不再处于用户界面的最上层，且不能够与用户进行交互（失去焦点）
    - **停止状态：**Activity在界面上完全不能被用户看到，也就是说这个Activity被其他Activity全部遮挡
    - **销毁状态（终止状态）：**该Activity结束，或Activity所在的Dalvik进程被结束

Activity生命周期

- Activity一种状态变到另一种状态时会经过一系列Activity类的回调方法(callback)
  - onCreate(Bundle savedInstanceState):该方法在Activity的实例被Android系统创建后第一个被调用。通常在该方法中设置显示屏幕的布局、初始化数据、设置控件被点击的事件响应代码。
  - onStart():在Activity可见时执行。
  - onRestart():回到最上边的界面，再次可见时执行。
  - onResume():Activity获取焦点时执行。
  - onPause():Activity失去焦点时执行。
  - onStop():用户不可见、进入后台时执行。
  - onDestroy():Activity销毁时执行。

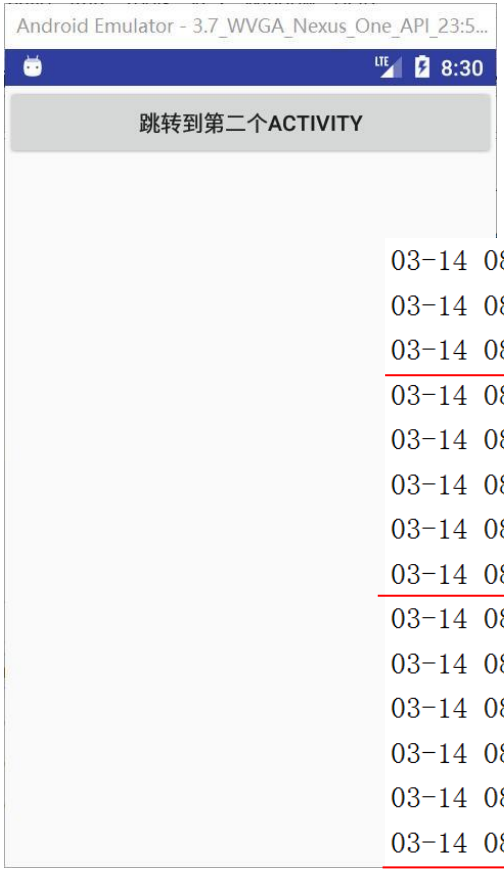
Activity生命周期

- Activity生命周期及相关回调方法



Activity生命周期

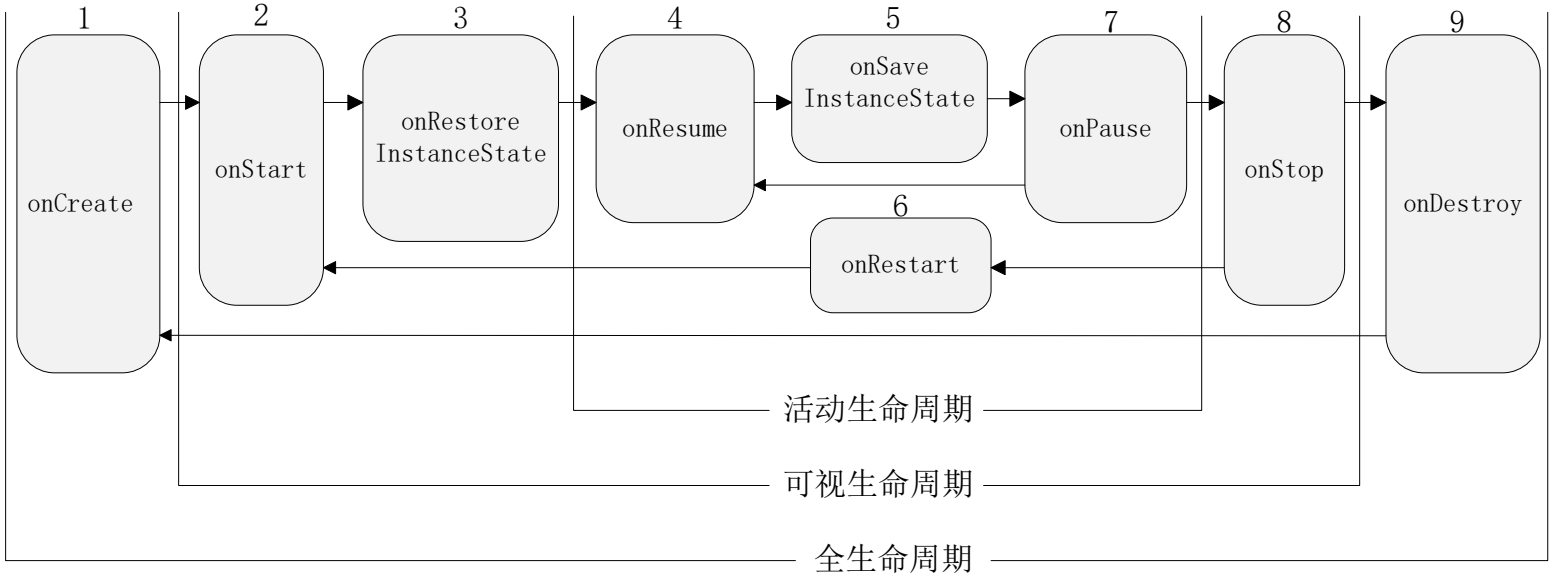
- 示例




```
03-14 08:48:42.544 5825-5825/com.example.lenovo.myapplication I/Activity1: onCreate()
03-14 08:48:42.548 5825-5825/com.example.lenovo.myapplication I/Activity1: onStart()
03-14 08:48:42.550 5825-5825/com.example.lenovo.myapplication I/Activity1: onResume()
03-14 08:48:46.651 5825-5825/com.example.lenovo.myapplication I/Activity1: onPause()
03-14 08:48:46.695 5825-5825/com.example.lenovo.myapplication I/Activity2: onCreate()
03-14 08:48:46.695 5825-5825/com.example.lenovo.myapplication I/Activity2: onStart()
03-14 08:48:46.695 5825-5825/com.example.lenovo.myapplication I/Activity2: onResume()
03-14 08:48:47.179 5825-5825/com.example.lenovo.myapplication I/Activity1: onStop()
03-14 08:48:49.959 5825-5825/com.example.lenovo.myapplication I/Activity2: onPause()
03-14 08:48:49.961 5825-5825/com.example.lenovo.myapplication I/Activity1: onReStart()
03-14 08:48:49.962 5825-5825/com.example.lenovo.myapplication I/Activity1: onStart()
03-14 08:48:49.962 5825-5825/com.example.lenovo.myapplication I/Activity1: onResume()
03-14 08:48:50.294 5825-5825/com.example.lenovo.myapplication I/Activity2: onStop()
03-14 08:48:50.294 5825-5825/com.example.lenovo.myapplication I/Activity2: onDestroy()
03-14 08:48:52.211 5825-5825/com.example.lenovo.myapplication I/Activity1: onPause()
03-14 08:48:52.220 5825-5825/com.example.lenovo.myapplication I/Activity1: onStop()
03-14 08:48:52.220 5825-5825/com.example.lenovo.myapplication I/Activity1: onDestroy()
```

Activity生命周期

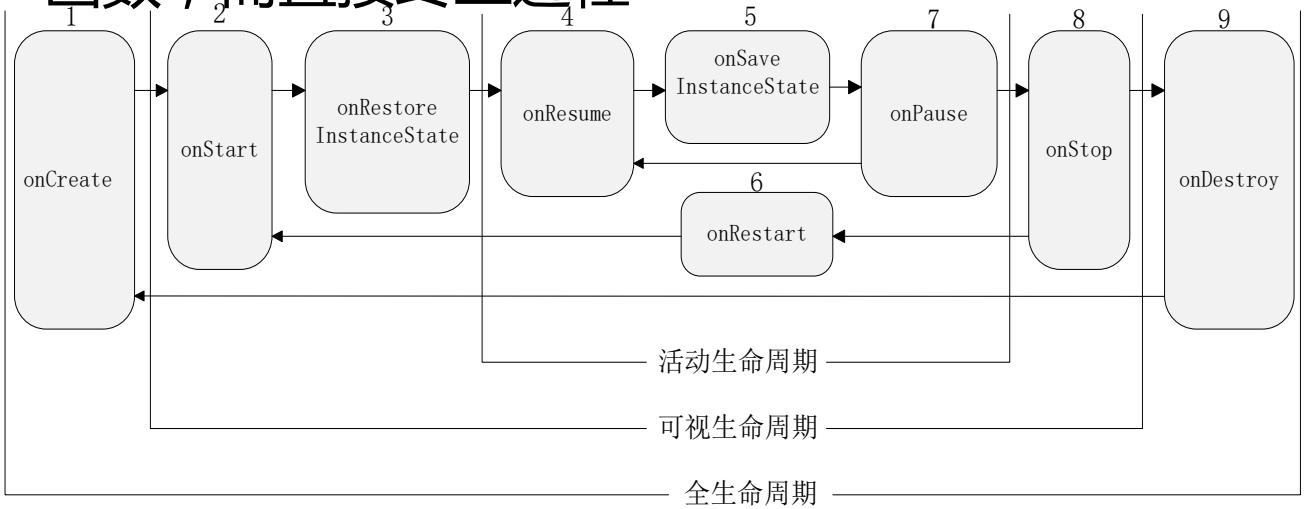
- Activity事件回调函数的调用顺序



- Activity生命周期分类
  - Activity的生命周期可分为全生命周期、可视生命周期和活动生命周期
  - 每种生命周期中包含不同的事件回调函数

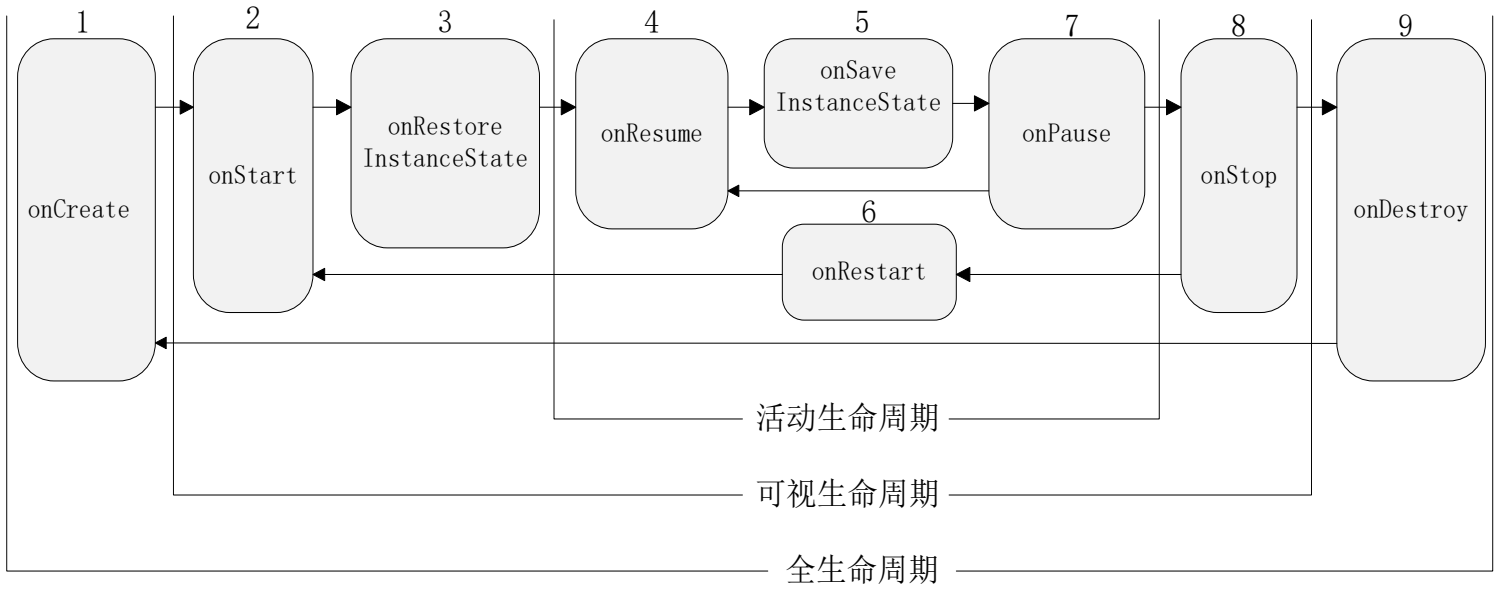
Activity生命周期

- 全生命周期
  - 全生命周期是从Activity建立到销毁的全部过程，始于onCreate()，结束于onDestroy()
    - 使用者通常在onCreate()中初始化Activity所能使用的全局资源和状态，并在onDestroy()中释放这些资源
    - 在一些极端的情况下，Android系统会不调用onDestroy()函数，而直接终止进程



Activity生命周期

- 可视生命周期
  - 可视生命周期是Activity在界面上从可见到不可见的过程，开始于onStart()，结束于onStop()



Activity生命周期

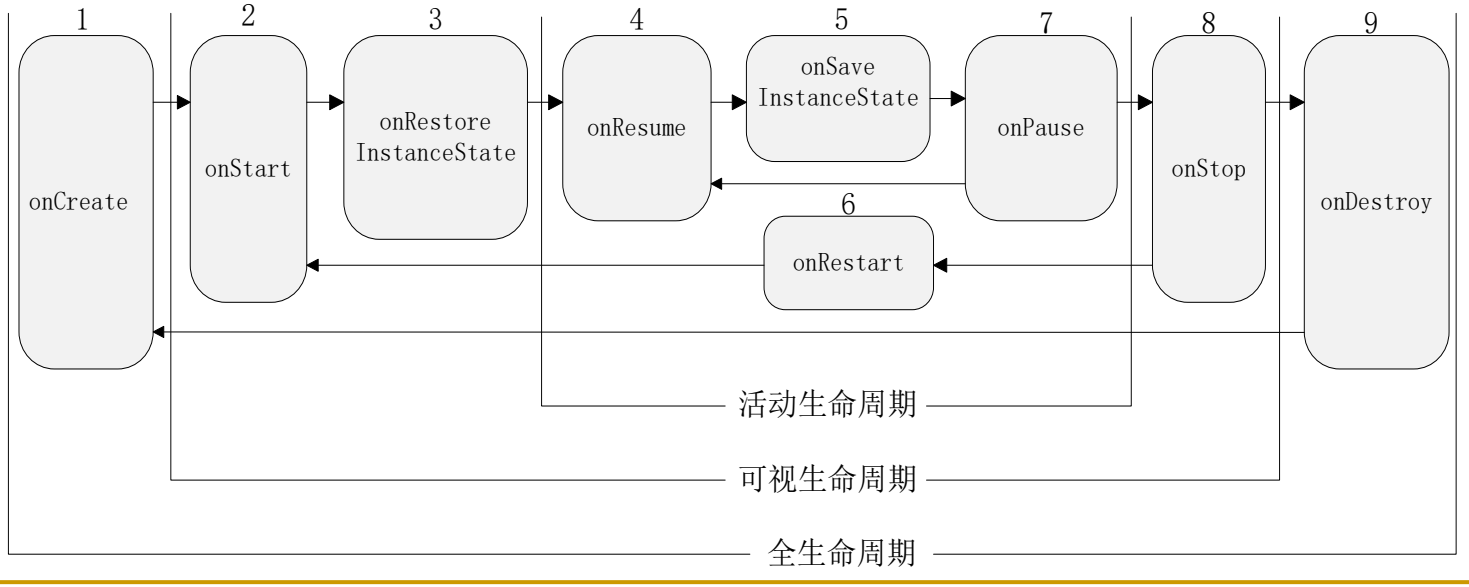
- 可视生命周期
  - onStart()一般用来初始化或启动与更新界面相关的资源
  - onStop()一般用来暂停或停止一切与更新用户界面相关的线程、计时器和服务
  - onRestart()函数在onStart()前被调用，用来在Activity从不可见变为可见的过程中，进行一些特定的处理过程
  - onStart()和onStop()会被多次调用
  - onStart()和onStop()也经常被用来注册和注销BroadcastReceiver

Activity生命周期

- 活动生命周期
  - 活动生命周期是Activity在屏幕的最上层，并能够与用户交互的阶段，开始于onResume()，结束于onPause()
  - 在Activity的状态变换过程中onResume()和onPause()经常被调用，因此这两个函数中应使用更为简单、高效的代码

Activity生命周期

- 活动生命周期
  - onPause()是第一个被标识为“可终止”的函数
  - 在onPause()返回后，onStop()和onDestroy()随时能被Android系统终止
  - onPause()常用来保存持久数据，如界面上的用户的输入信息等



Activity生命周期

- onPause()和onSaveInstanceState()的区别

  这两个函数都可以用来保存界面的用户输入数据，他们的区别在于

  - onPause()一般用于保存持久性数据，并将数据保存在存储设备上的文件系统或数据库系统中的
  - onSaveInstanceState()主要用来保存动态的状态信息，信息一般保存在Bundle中
    - Bundle是能够保存多种格式数据的对象
    - 在onSaveInstanceState()保存在Bundle中的数据，系统在调用onRestoreInstanceState()和onCreate()时，会同样利用Bundle将数据传递给函数

练习

- 用户注册


7:26
用户注册
手机号：
请输入手机号
密　码：
请输入密码
男
女
读书
打球
听音乐
北京
注 册


6:18
手机号为：13900005678
密码为：123
性别是：女性
爱好是：读书打球
城市是：上海

练习

- 用户注册
  - 按钮单击事件

```
<!--布局文件中添加点击事件为其制定方法名-->
<Button android:onClick="myClick">
```

```
public void myClick(View view){
    Intent intent=new Intent(MainActivity.this, SecondActivity.class);
    startActivity(intent);
}
```