

 Implementing your activity lifecycle methods properly ensures your app behaves well in several ways, including that it:

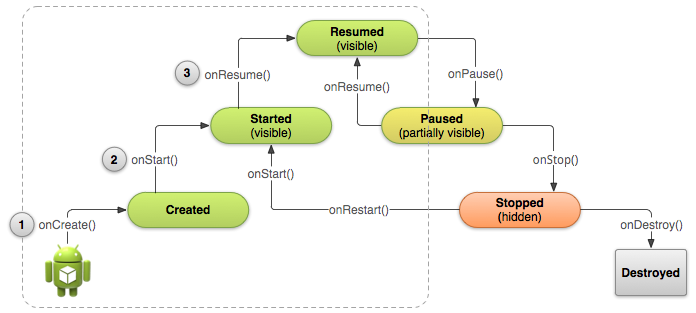
* 当用户接听电话或者切换到另一个app时，当前activity不会崩溃。
* 当用户不再使用该activity时不会占用太多的系统资源
* 当用户切换回来时，不会丢失该用户之前的事物处理进度（看电影、下载文档等）。
* 当用户横竖屏切换时，不会崩溃或者丢失数据。

创造一个实例：

必须实现onCreat()方法，生命用户界面，初始化一些变量。在整个生命周期中onCreate()方法只会被执行一次。

TextView mTextView;   
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main\_activity);  
  
    mTextView = (TextView) findViewById(R.id.text\_message);  
    // 保证该段代码运行在HONEYCOMB版本之上  
    if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.HONEYCOMB) {  
        ActionBar actionBar = getActionBar();  
        actionBar.setHomeButtonEnabled(false);  
    }  
}

[onCreate()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\app\Activity.html#onCreate(android.os.Bundle))执行完毕后，[onStart()](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\app\\Activity.html" \l "onStart()) 和 [onResume()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\app\Activity.html#onResume())很快会被执行。从技术上来说，当onStart()方法被执行后，activity变得可见，但是[onResume()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\app\Activity.html#onResume())方法很快就会被执行，并保持在Resume状态。



**Pausing and Resuming an Activity**

During normal app use, the foreground activity is sometimes obstructed by other visual components that cause the activity to *pause*. For example, when a semi-transparent activity opens (such as one in the style of a dialog), the previous activity pauses. As long as the activity is still partially visible but currently not the activity in focus, it remains paused.

However, once the activity is fully-obstructed and not visible, it *stops* (which is discussed in the next lesson).

（Pause状态和stop状态的唯一区别时activity被部分遮挡还是全部遮挡）

一旦activity进入pause状态，你就可以在onPause()方法中停止正在进行的活动或者保存一些持久化的数据。

You should usually use the [onPause()](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\app\\Activity.html" \l "onPause()) callback to:

* 停止动画及其他消耗cup的动作。
* 提交为保存的变化，但只保存用户需要持久保存的数据，such as草稿。。。
* 释放系统资源（取消注册广播接收者、处理传感器、以及一些既耗电有没用的东西）
* @Override  
  public void onPause() {  
      super.onPause();  //记住首先调用父类方法再做其他操作  
      // 释放相机资源，因为我们不需要它，但其他应用可能需要

if (mCamera != null) {  
        mCamera.release()  
        mCamera = null;  
    }  
 }

无论如何，你应该避免在onPause方法中执行过多消耗CPU的操作，因为他会拖慢可视化过度的过程（你应该在onStop()方法中执行消耗内存的操作）

## Resume Your Activity

当用户从Paused状态切换回Resume状态时，系统回调onResume()方法.activity每次回到前台时都会调用此方法。因此，在此方法中，你应该初始化那些在onPause中释放的组件，以及执行其他每当activity出现时都应该刷新的组件。总之onResume和onPause一一对应。

@Override  
public void onResume() {  
    super.onResume();  // Always call the superclass method first  
  
    // Get the Camera instance as the activity achieves full user focus  
    if (mCamera == null) {  
        initializeCamera(); // Local method to handle camera init  
    }  
}

# Stopping and Restarting an Activity

# E:\android\adt-bundle-windows-x86-20131030\sdk\docs\images\training\basics\basic-lifecycle-stopped.png

# 这两个生命周期方法非常重要，它保证了让用户感觉你的程序一直存在着，并没有失去他们之前的进度。在这两个生命周期中，主要发生了一下情节：

# 当用户打开最近用过的应用界面，从你的app切换到另一个app，当前处于前台的activity就变为stop状态。当用户通过点击应用程序图标或从最近用过的APP中返回你的app时，restart原来的activity。

# 从当前activity启动一个新的activity时activity变为stop，用back键返回时又重新执行restart。

# 当用户接到一个电话时，执行stop，电话结束时，又执行restart。

## Stop Your Activity

# 当activity执行[onStop()](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\app\\Activity.html" \l "onStop())方法时意味着他再也不可见了，你应当释放几乎所有不用的资源，一旦activity被stop，它的实例随时可能被系统回收以恢复内存。在极端情况下，系统可能直接kill掉应用进程而不回调activity的[onDestroy()](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\app\\Activity.html" \l "onDestroy())方法。所以，在onStop()方法中回收资源非常重要，否则可能回引起内存泄漏。

**尽管onPause在onStop之前被调用，应该在onStop方法中执行更耗费资源的操作，例如向数据库写数据等。**

@Override  
protected void onStop() {  
    super.onStop();  //首先调用父类方法  
    // activity停止时保存草稿放置当前进度丢失    ContentValues values = new ContentValues();  
    values.put(NotePad.Notes.COLUMN\_NAME\_NOTE, getCurrentNoteText());  
    values.put(NotePad.Notes.COLUMN\_NAME\_TITLE, getCurrentNoteTitle());  
    getContentResolver().update(  
            mUri,    // The URI for the note to update.  
            values,  // 键值对  
            null, null            );  
}

# 当activity停止时，实例仍保存在内存中。你不必去重新初始化任何在生命周期方法中定义的组件，系统回持续跟踪每一个组件的最新状态，如你在textview中输入文字后打开另一个activity，在回来时之前输入的文字依然被保留着。及时系统销毁了实例，个组件的状态依然被保留在一个Bundle里面，放用户返回时恢复之前的状态。

## Start/Restart Your Activity

# 当你的activity从stop状态回到前台时，回接收到[onRestart()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\app\Activity.html#onRestart())回调函数。系统同样回调用 [onStart()](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\app\\Activity.html" \l "onStart())方法（每次activity可见时都会调用，无论是第一次创建还是重新启动），restart方法只有在stop恢复时才会调用。但此方法一般并不常用。

# Recreating an Activity

# 一些操作回导致一个activity的实例销毁，例如按下back键时或者调用了finish方法。系统也会回收一些处于stop状态的activity（当他们长时间未使用，或为前台activity获取资源时）。

# 当你的activity由于back键或在内部finish时，系统默认该activity以不再需要，所以不会保存任何数据。然而，当因为系统限制而被迫销毁的activity实例，尽管该实力已被销毁，但系统会记录下来，当用户回到该activity时，系统会用之前保存下来的数据重新创建一个实例。这些被保存的数据事被称为“instance state”的一系列键值对。

# 系统默认用Bundle实例来保存当前布局的每一个View对象（包括在TextView中输入的文字）。所以，如果你的activity被销毁和重建，不需要写任何代码即可恢复到之前的状态。至于类似变量这类的数据，还是需要手动保存起来。

# NOTE：为了系统能恢复你的activity中各View的状态，每一个view必须要有一个唯一的资源ID属性（[android:id](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\view\\View.html" \l "attr_android:id)）

# 为了保存额外的数据，你必须重写onSaveInstanceState()这个回调方法。系统会在用户离开时调用此方法，并将Bundle对象保存在异常退出的event中。如果必须重新启动该activity，系统会将该Bundle实例同时传递给[onRestoreInstanceState()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\app\Activity.html#onRestoreInstanceState(android.os.Bundle))和[onCreate()](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\app\\Activity.html" \l "onCreate(android.os.Bundle))方法。

# E:\android\adt-bundle-windows-x86-20131030\sdk\docs\images\training\basics\basic-lifecycle-savestate.png

## Save Your Activity State

# 当你的activity即将变为stop状态时，系统调用onSaveInstanceState()方法,所以你可以保存通过键值对来保存各种状态。该方法默认情况下会保存该activity的各级view的状态（EditText的内容或ListView的滑动条位置）

# 保存额外的信息需要实现onSaveInstanceState()方法：

# 同样要调用父类方法以保存view信息

static final String STATE\_SCORE = "playerScore";  
static final String STATE\_LEVEL = "playerLevel";  
...  
  
@Override  
public void onSaveInstanceState(Bundle savedInstanceState) {  
    // Save the user's current game state  
    savedInstanceState.putInt(STATE\_SCORE, mCurrentScore);  
    savedInstanceState.putInt(STATE\_LEVEL, mCurrentLevel);  
      
    // Always call the superclass so it can save the view hierarchy state  
    super.onSaveInstanceState(savedInstanceState);  
}

## Restore Your Activity State

当你的activity创建之前曾被销毁过时，可从Bundle中恢复。onCreate和onRestoreInstanceState中都又Bundle参数。但在onCreate方法中，由于不知道之前是否被destroy过，所以必须检查Bundle是否为空。

protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState); // 首先调用父类函数     
    // 检查之前是否被销毁过  
    if (savedInstanceState != null) {//恢复数据  
        mCurrentScore = savedInstanceState.getInt(STATE\_SCORE);  
        mCurrentLevel = savedInstanceState.getInt(STATE\_LEVEL);  
    } else {// 如将又新值要赋给当前activity在这里初始化    }  
    ...  
}

[onRestoreInstanceState()](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\reference\android\app\Activity.html#onRestoreInstanceState(android.os.Bundle))可作为用onCreate恢复数据的替代方法，该函数在onStart方法之后调用，系统调用该方法之前一定创建过该activity实例，所以不必检查Bundle对象是否为空。

public void onRestoreInstanceState(Bundle savedInstanceState) {  
    // Always call the superclass so it can restore the view hierarchy  
    super.onRestoreInstanceState(savedInstanceState);  
     
    // Restore state members from saved instance  
    mCurrentScore = savedInstanceState.getInt(STATE\_SCORE);  
    mCurrentLevel = savedInstanceState.getInt(STATE\_LEVEL);  
}

**Caution:** Always call the superclass implementation of [onRestoreInstanceState()](file:///E:\\android\\adt-bundle-windows-x86-20131030\\sdk\\docs\\reference\\android\\app\\Activity.html" \l "onRestoreInstanceState(android.os.Bundle)) so the default implementation can restore the state of the view hierarchy.

To learn more about recreating your activity due to a restart event at runtime (such as when the screen rotates), read [Handling Runtime Changes](file:///E:\android\adt-bundle-windows-x86-20131030\sdk\docs\guide\topics\resources\runtime-changes.html).