* [首页](https://www.cnblogs.com/)
* [新闻](https://news.cnblogs.com/)
* [博问](https://q.cnblogs.com/)
* [专区](https://brands.cnblogs.com/)
* [闪存](https://ing.cnblogs.com/)
* [班级](https://edu.cnblogs.com/)

窗体顶端



窗体底端

* [注册](https://account.cnblogs.com/signup/)[登录](javascript:void(0);)

[返回主页](https://www.cnblogs.com/little-byte/)

[**little-byte**](https://www.cnblogs.com/little-byte/)

记录我努力的点滴，那是成功的每一级阶梯

* [**博客园**](https://www.cnblogs.com/)
* [**首页**](https://www.cnblogs.com/little-byte/)
* [**新随笔**](https://i.cnblogs.com/EditPosts.aspx?opt=1)
* [**联系**](https://msg.cnblogs.com/send/little-byte)
* [**订阅**](javascript:void(0))
* [**管理**](https://i.cnblogs.com/)

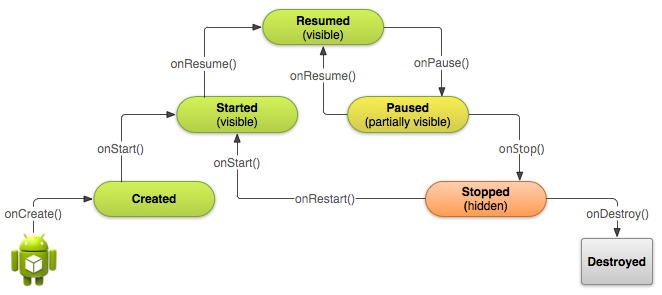
随笔 - 12  文章 - 0  评论 - 0  阅读 - 6465

[**activity生命周期深入理解**](https://www.cnblogs.com/little-byte/p/0639a10c7783d0f1a952e27e1b975c7d.html)

**摘自github上面的Google Android官方培训课程中文版，地址：**[**https://github.com/kesenhoo/android-training-course-in-chinese**](https://github.com/kesenhoo/android-training-course-in-chinese)

**Understand the Lifecycle Callbacks[理解生命周期的回调]**

* 在一个activity的生命周期中，系统会像金字塔模型一样去调用一系列的生命周期回调方法。Activity生命周期的每一个阶段就像金字塔中的台阶。当系统创建了一个新的activity实例，每一个回调函数会向上一阶的移动activity状态。金字塔顶端意味着activity是跑在最前端的并且用户可以与它进行交互。
* 当用户开始离开这个activity,为了卸载这个activity，系统会调用其它方法来向下一阶移动activity状态。在某些情况下，activity会隐藏在金字塔下等待(例如当用户切换到其他app),这个时候activity可以重新回到顶端(如果用户回到这个activity)并且恢复用户离开时的状态。
* Figure 1. 下面这张图讲解了activity的生命周期：*(显然，这个金字塔模型要比之前Dev Guide里面的生命周期图更加容易理解，更加形象)*



* 根据你的activity的复杂度，你也许不需要实现所有的生命周期方法。然而，你需要知道每一个方法的功能并确保你的app能够像用户期望的那样执行。如何实现一个符合用户期待的app，你需要注意下面几点：
  + 当使用你的app的时候，不会因为有来电通话或者切换到其他app而导致程序crash。
  + 当用户没有激活某个组件的时候不要消耗宝贵的系统资源。
  + 当离开你的app并且一段时间后返回，不要丢失用户的使用进度。
  + 当设备发送屏幕旋转的时候，不会crash或者丢失用户的使用进度。
* 在下面的课程中会介绍上图所示的几个生命状态。然而，其中只有三个状态是静态的，这三个状态下activity可以存在一段比较长的时间。*(其它几个状态会很快就切换掉，停留的时间比较短暂)*
  + **Resumed**：在这个状态，activity是在最前端的，用户可以与它进行交互。(通常也被理解为"running" 状态)
  + **Paused**：在这个状态，activity被另外一个activity所遮盖：另外的activity来到最前面，但是半透明的，不会覆盖整个屏幕。被暂停的activity不会再接受用户的输入且不会执行任何代码。
  + **Stopped**：在这个状态, activity完全被隐藏，不被用户可见。可以认为是在后台。当stopped, activity实例与它的所有状态信息都会被保留，但是activity不能执行任何代码。
* 其它状态 (Created and Started)都是短暂的，系统快速的执行那些回调函数并通过执行下一阶段的回调函数移动到下一个状态。也就是说，在系统调用onCreate(), 之后会迅速调用onStart(), 之后再迅速执行onResume()。上面就是基本的activity生命周期。

**Specify Your App's Launcher Activity[指定你的程序首次启动的Activity]**

* 当用户从主界面点击你的程序图标时，系统会调用你的app里面的被声明为"launcher" (or "main") activity中的onCreate()方法。这个Activity被用来当作你的程序的主要进入点。
* 你可以在AndroidManifest.xml中定义哪个activity作为你的主activity。
* 这个main activity必须在manifest使用包括 MAIN action and LAUNCHER category 的 标签来声明。例如：

<activity android:name=".MainActivity" android:label="@string/app\_name">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

* 如果你的程序中没有一个activity声明了MAIN action 或者LAUNCHER category，那么在设备的主界面列表里面不会呈现你的app图标。

**Create a New Instance[创建一个新的实例]**

* 大多数app都包括许多不同的activities，这样使得用户可以执行不同的动作。不论这个activity是创建的主activtiy还是为了响应用户行为而新创建的，系统都会调用新的activity实例中的onCreate()方法。
* 你必须实现onCreate()方法来执行程序启动所需要的基本逻辑。
* 例如：下面的onCreate()方法演示了为了建立一个activity所需要的一些基础操作。如，声明UI元素 ，定义成员变量，配置UI等。*(onCreate里面尽量少做事情，避免程序启动太久都看不到界面)*

[复制代码](javascript:void(0);)

TextView mTextView; // Member variable for text view in the layout

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

// Set the user interface layout for this Activity

// The layout file is defined in the project res/layout/main\_activity.xml

file

setContentView(R.layout.main\_activity);

// Initialize member TextView so we can manipulate it later

mTextView = (TextView) findViewById(R.id.text\_message);

// Make sure we're running on Honeycomb or higher to use ActionBar APIs

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.HONEYCOMB) {

// For the main activity, make sure the app icon in the action bar

// does not behave as a button

ActionBar actionBar = getActionBar();

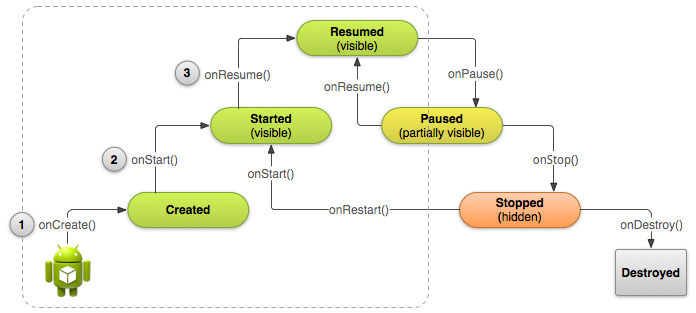
actionBar.setHomeButtonEnabled(false);

}

}

[复制代码](javascript:void(0);)

* 一旦结束onCreate 操作，系统会迅速调用onStart() 与onResume()方法。你的activity不会在Created或者Started状态停留。技术上来说, activity在onStart()被调用后会开始被用户可见，但是 onResume()会迅速被执行使得activity停留在Resumed状态，直到一些因素发生变化才会改变这个状态。例如接受到一个来电，用户切换到另外一个activity，或者是设备屏幕关闭。
* 在后面的课程中，你将看到其他方法是如何使用的，onStart() 与 onResume()在用户从Paused or Stopped状态中恢复的时候非常有用。
* **Note:** onCreate() 方法包含了一个参数叫做savedInstanceState，这将会在后面的课程：重新创建一个activity的时候涉及到。



* Figure 2. 上图显示了onCreate(), onStart(), and onResume()是如何执行的。当这三个顺序执行的回调函数完成后，activity会到达Resumed状态。

**Destroy the Activity[销毁Activity]**

* activity的第一个生命周期回调函数是 onCreate(),它最后一个回调是 onDestroy().系统会执行这个方法作为你 的activity要从系统中完全移除的信号。
* 大多数 apps并不需要实现这个方法，因为局部类的references会被destroyed并且你的activity应该在onPause() 与 onStop()中执行清除的操作。然而，如果你的activity包含了你在onCreate时创建的后台线程，或者是其他有可能导致内存泄漏的资源，你应该在OnDestroy()时杀死他们。

[复制代码](javascript:void(0);)

@Override

public void onDestroy() {

super.onDestroy(); // Always call the superclass

// Stop method tracing that the activity started during onCreate()

android.os.Debug.stopMethodTracing();

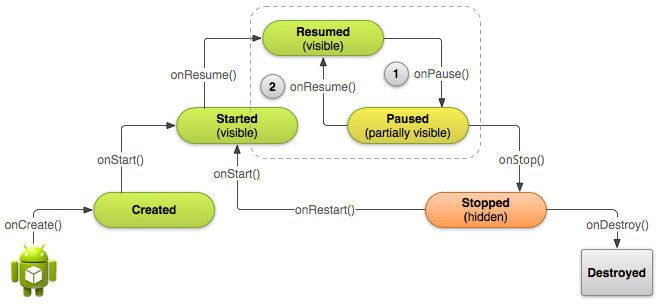
}

[复制代码](javascript:void(0);)

* **Note:** 系统通常是在执行了 onPause() and onStop() 之后再调用onDestroy() ，除非你的程序里面再其他地方调用了finish()方法，这样系统会直接就调用onDestory方法，其它生命周期的方法则不会被执行。

**暂停与恢复Activity**

* 在使用通常的app时，前端的activity有时候会被其他可见的组件而阻塞(obstructed)，这样会导致当前的activity进入Pause状态。例如，当打开一个半透明的activity时(例如以对话框的形式) ，之前的activity会被暂停. 只要这个activity仍然被部分可见，之前的activity则一直处于Paused状态。
* 然而，一旦之前的activity被完全阻塞并不可见，它则会进入Stop状态(将在下一小节讨论)。
* 当你的activity进入paused状态，系统会调用你的activity中的onPause()方法, 这个方法会停止目前正在运行的操作，比如暂停视频播放。它会保存那些有可能需要长期保存的信息。(请注意是在onPause里面去保存那些信息)。如果用户从暂停状态回到你的activity，系统会恢复那些数据并执行onResume()方法。
* **Note:**当你的activity接受到调用onPause()信息 ，那可能意味者activity将被暂停一段时间，并且用户很可能回到你的activity。然而，那也是用户要离开你的activtiy的第一个信号。
* **Figure 1.**下图显示了，当一个半透明的activity阻塞你的activity时，系统会调用onPause()方法并且这个activity会停留在Paused state (1). 如果用户在这个activity还是在Paused State时回到这个activity，系统则会调用它的onResume() (2).



**Pause Your Activity[暂停你的Activity]**

* 当系统调用你的activity中的onPause(),从技术上讲，那意味着你的activity仍然处于部分可见的状态，当时大多数时候，那意味着用户正在离开这个activity并马上会进入Stopped state. 你通常应该在onPause()回调方法里面做下面的事情:
  + 停止动画或者是其他正在运行的操作，那些都会导致CPU的浪费.
  + 提交没有保存的改变，但是仅仅是在用户离开时期待保存的内容(例如邮件草稿).
  + 释放系统资源，例如broadcast receivers, sensors (比如GPS), 或者是其他任何会影响到电量的资源。
  + 例如, 如果你的程序使用Camera,onPause()会是一个比较好的地方去做那些释放资源的操作。

[复制代码](javascript:void(0);)

@Override

public void onPause() {

super.onPause(); // Always call the superclass method first

// Release the Camera because we don't need it when paused

// and other activities might need to use it.

if (mCamera != null) {

mCamera.release()

mCamera = null;

}

}

[复制代码](javascript:void(0);)

* 通常，你**不**应该使用onPause()来保存用户改变的数据 (例如填入表格中的个人信息) 到永久存储上。仅仅当你确认用户期待那些改变能够被自动保存的时候(例如正在撰写邮件草稿)，你可以把那些数据存到永久存储 。然而，你应该避免在onPause()时执行CPU-intensive 的工作，例如写数据到DB，因为它会导致切换到下一个activity变得缓慢(你应该把那些heavy-load的工作放到onStop()去做)。
* 如果你的activity实际上是要被Stop，那么你应该为了切换的顺畅而减少在OnPause()方法里面的工作量。
* **Note:**当你的activity处于暂停状态，[Activity](http://developer.android.com/reference/android/app/Activity.html)实例是驻留在内存中的，并且在activity 恢复的时候重新调用。你不需要在恢复到Resumed状态的一系列回调方法中重新初始化组件。

**Resume Your Activity[恢复你的activity]**

* 当用户从Paused状态恢复你的activity时，系统会调用onResume()方法。
* 请注意，系统每次调用这个方法时，activity都处于最前台，包括第一次创建的时候。所以，你应该实现onResume()来初始化那些你在onPause方法里面释放掉的组件，并执行那些activity每次进入Resumed state都需要的初始化动作 (例如开始动画与初始化那些只有在获取用户焦点时才需要的组件)
* 下面的onResume()的例子是与上面的onPause()例子相对应的。

[复制代码](javascript:void(0);)

@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

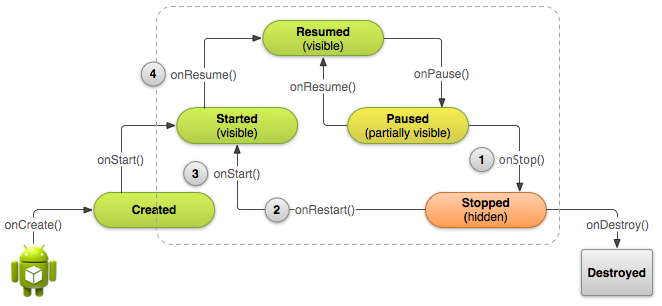
}

}

[复制代码](javascript:void(0);)

**停止与重启Activity**

* 恰当的停止与重启你的activity是很重要的，在activity生命周期中，他们能确保用户感知到程序的存在并不会丢失他们的进度。在下面一些关键的场景中会涉及到停止与重启：
  + 用户打开最近使用app的菜单并切换你的app到另外一个app，这个时候你的app是被停止的。如果用户通过统一的办法回到你的app，那么你的activity会重启.
  + 用户在你的app里面执行启动一个新的activity的操作，当前activity会在第二个activity被创建后停止。如果用户点击back按钮，第一个activtiy会被重启.
  + 用户在使用你的app时接受到一个来电通话.
* [Activity](http://developer.android.com/reference/android/app/Activity.html)类提供了onStop()与onRestart(), 方法来允许你在activity停止与重启时进行调用。不像暂停状态是部分阻塞UI，停止状态是UI不在可见并且用户的焦点转移到另一个activity中.
* **Note:**因为系统在activity停止时会在内存中保存了Activity实例。有些时候你不需要实现onStop(),onRestart()甚至是onStart()方法. 因为大多数的activity相对比较简单，activity会自己停止与重启，你只需要使用onPause()来停止正在运行的动作并断开系统资源链接。



* **Figure 1.**上图显示：当用户离开你的activity，系统会调用onStop()来停止activity (1). 这个时候如果用户返回，系统会调用onRestart()(2), 之后会迅速调用[onStart()](http://hukai.me/android-training-course-in-chinese/basics/activity-lifecycle/3) 与onResume()(4). 请注意：无论什么原因导致activity停止，系统总是会在onStop()之前调用onPause()方法。

**Stop Your Activity[停止你的activity]**

* 当你的activity调用onStop()方法, activity不再可见，并且应该释放那些不再需要的所有资源。一旦你的activity停止了，系统会在不再需要这个activity时摧毁它的实例。在极端情况下，系统会直接杀死你的app进程，并且不执行activity的onDestroy()回调方法, 因此你需要使用onStop()来释放资源，从而避免内存泄漏。*(这点需要注意)*
* 尽管onPause()方法是在onStop()之前调用，你应该使用onStop()来执行那些CPU intensive的shut-down操作，例如writing information to a database.
* 例如，下面是一个在onStop()的方法里面保存笔记草稿到persistent storage的示例:

[复制代码](javascript:void(0);)

@Override

protected void onStop() {

super.onStop(); // Always call the superclass method first

// Save the note's current draft, because the activity is stopping

// and we want to be sure the current note progress isn't lost.

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, // The map of column names and new values to apply to them.

null, // No SELECT criteria are used.

null // No WHERE columns are used.

);

}

[复制代码](javascript:void(0);)

* 当你的activity已经停止,[Activity](http://developer.android.com/reference/android/app/Activity.html)对象会保存在内存中，并且在activity resume的时候重新被调用到。你不需要在恢复到Resumed state状态前重新初始化那些被保存在内存中的组件。系统同样保存了每一个在布局中的视图的当前状态，如果用户在EditText组件中输入了text，它会被保存，因此不需要保存与恢复它。
* **Note:**即时系统会在activity stop的时候销毁这个activity，它仍然会保存View objects (such as text in an [EditText](http://developer.android.com/reference/android/widget/EditText.html)) 到一个[Bundle](http://developer.android.com/reference/android/os/Bundle.html)中，并且在用户返回这个activity时恢复他们(下一个会介绍在activity销毁与重新建立时如何使用[Bundle](http://developer.android.com/reference/android/os/Bundle.html)来保存其他数据的状态).

**Start/Restart Your Activity[启动与重启你的activity]**

* 当你的activity从Stopped状态回到前台时，它会调用onRestart().系统再调用onStart()方法，onStart()方法会在每次你的activity可见时都会被调用。onRestart()方法则是只在activity从stopped状态恢复时才会被调用，因此你可以使用它来执行一些特殊的恢复(restoration)工作，请注意之前是被stopped而不是destrory。
* 使用onRestart()来恢复activity状态是不太常见的，因此对于这个方法如何使用没有任何的guidelines。然而，因此你的onStop()方法应该做清除所有activity资源的操作，你将会在重新启动activtiy时re-instantiate那些被清除的资源，同样, 你也需要在activity第一次创建时instantiate那些资源。介于上面的原因，你应该使用onStart()作为onStop()所对应方法。因为系统会在创建activity与从停止状态重启activity时都会调用onStart().*(这个地方的意思应该是说你在onStop里面做了哪些清除的操作就应该在onStart里面重新把那些清除掉的资源重新创建出来)*
* 例如：因为用户很可能在回到这个activity之前需要过一段时间，所以onStart()方法是一个比较好的地方用来验证某些必须的功能是否已经Ready。

[复制代码](javascript:void(0);)

@Override

protected void onStart() {

super.onStart(); // Always call the superclass method first

// The activity is either being restarted or started for the first time

// so this is where we should make sure that GPS is enabled

LocationManager locationManager =

(LocationManager) getSystemService(Context.LOCATION\_SERVICE);

boolean gpsEnabled = locationManager.isProviderEnabled(LocationManager.GPS\_PROVIDER);

if (!gpsEnabled) {

// Create a dialog here that requests the user to enable GPS, and use an intent

// with the android.provider.Settings.ACTION\_LOCATION\_SOURCE\_SETTINGS action

// to take the user to the Settings screen to enable GPS when they click "OK"

}

}

@Override

protected void onRestart() {

super.onRestart(); // Always call the superclass method first

// Activity being restarted from stopped state

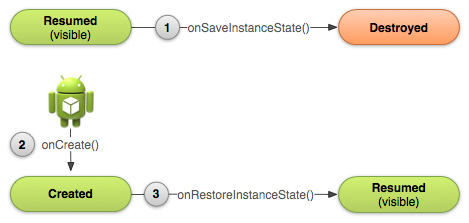
}

[复制代码](javascript:void(0);)

* 当系统Destory你的activity，它会为你的activity调用onDestroy()方法。因为我们会在onStop方法里面做释放资源的操作，那么onDestory方法则是你最后去清除那些可能导致内存泄漏的地方。因此你需要确保那些线程都被destroyed并且所有的操作都被停止。

**重新创建Activity**

* 有几个场景中,Activity是由于正常的程序行为而被Destory的，例如当用户点击返回按钮或者是你的Activity通过调用finish()来发出停止信号。系统也有可能会在你的Activity处于stop状态且长时间不被使用，或者是在前台activity需要更多系统资源的时候把关闭后台进程，这样来获取更多的内存。
* 当你的Activity是因为用户点击Back按钮或者是activity通过调用finish()结束自己时，系统就丢失了Activity实例这个概念，因为前面的行为意味着不再需要这个activity了。然而，如果因为系统资源紧张而导致Activity的Destory, 系统会在用户回到这个Activity时有这个Activity存在过的记录，系统会使用那些保存的记录数据（描述了当Activity被Destory时的状态）来重新创建一个新的Account实例。那些被系统用来恢复之前状态而保存的数据被叫做 "instance state" ，它是一些存放在 Bundle 对象中的key-value pairs.
* **Caution:**你的Activity会在每次旋转屏幕时被destroyed与recreated。当屏幕改变方向时，系统会Destory与Recreate前台的activity，因为屏幕配置被改变，你的Activity可能需要加载一些alternative的资源(例如layout).
* 默认情况下, 系统使用 Bundle 实例来保存每一个视图对象中的信息(例如输入EditText 中的文本内容)。因此，如果你的Activity被destroyed与recreated, 那么layout的状态信息会自动恢复到之前的状态。然而，你的activity也许存在更多你想要恢复的状态信息，例如记录用户Progress的成员变量(member variables).。
* **请注意**为了让你可以保存额外更多的数据到saved instance state。在Activity的声明周期里面存在一个添加的回调函数。这个回调函数并没有在前面课程的图片示例中显示。这个方法是 onSaveInstanceState() ，当用户离开你的Activity时，系统会调用它。当系统调用这个函数时，系统会在你的Activity被异常Destory时传递 Bundle 对象，这样你可以增加额外的信息到Bundle中并保存与系统中。然后如果系统在Activity被Destory之后想重新创建这个Activity实例时，之前的那个Bundle对象会(系统)被传递到你的activity的 onRestoreInstanceState() 方法与 onCreate() 方法中。



上图：当系统开始停止你的Activity时，会调用到[onSaveInstanceState()](http://developer.android.com/reference/android/app/Activity.html#onSaveInstanceState(android.os.Bundle)) (1) ，因此你可以在Activity实例需要重新创建的情况下，指定特定的附加状态数据到Bunde中。如果这个Activity被destroyed而且同样的实例被重新创建，系统会传递在 (1) 中的状态数据到 onCreate() (2) 与 [onRestoreInstanceState()](http://developer.android.com/reference/android/app/Activity.html#onRestoreInstanceState(android.os.Bundle))(3).

**Save Your Activity State[保存Activity状态]**

* 当你的activity开始Stop，系统会调用 onSaveInstanceState() ，因此你的Activity可以用键值对的集合来保存状态信息。这个方法会默认保存Activity视图的状态信息，例如在 EditText 组件中的文本或者是 ListView 的滑动位置。
* 为了给Activity保存额外的状态信息，你必须实现onSaveInstanceState() 并增加key-value pairs到 Bundle 对象中，例如：

[复制代码](javascript:void(0);)

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);

}

[复制代码](javascript:void(0);)

* Caution: 总是需要调用 onSaveInstanceState() 方法的父类实现，这样默认的父类实现才能保存视图状态的信息。

**Restore Your Activity State [恢复Activity状态]**

* 当你的Activity从Destory中重建。你可以从系统传递给你的Activity的Bundle中恢复保存的状态。 onCreate() 与 onRestoreInstanceState() 回调方法都接收到了同样的Bundle，里面包含了同样的实例状态信息。
* 因为 onCreate() 方法会在第一次创建新的Activity实例与重新创建之前被Destory的实例时都被调用，你必须在你尝试读取 Bundle 对象前Check它是否为null。如果它为Null，系统则是创建一个新的Activity instance，而不是恢复之前被Destory的Activity。
* 下面是一个示例：演示在onCreate方法里面恢复一些数据：

[复制代码](javascript:void(0);)

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState); // Always call the superclass first

// Check whether we're recreating a previously destroyed instance

if (savedInstanceState != null) {

// Restore value of members from saved state

mCurrentScore = savedInstanceState.getInt(STATE\_SCORE);

mCurrentLevel = savedInstanceState.getInt(STATE\_LEVEL);

} else {

// Probably initialize members with default values for a new instance

}

...

}

[复制代码](javascript:void(0);)

* 你也可以选择实现 onRestoreInstanceState() ，而是不是在onCreate方法里面恢复数据。 onRestoreInstanceState()方法会在 onStart() 方法之后执行. 系统仅仅会在存在需要恢复的状态信息时才会调用 onRestoreInstanceState() ，因此你不需要检查 Bundle 是否为Null:

[复制代码](javascript:void(0);)

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);

}

[复制代码](javascript:void(0);)

* Caution: 与上面保存一样，总是需要调用onRestoreInstanceState()方法的父类实现，这样默认的父类实现才能保存视图状态的信息。如果想了解更多关于运行时状态改变引起的recreate你的activity。请参考[Handling Runtime Changes](http://developer.android.com/guide/topics/resources/runtime-changes.html).