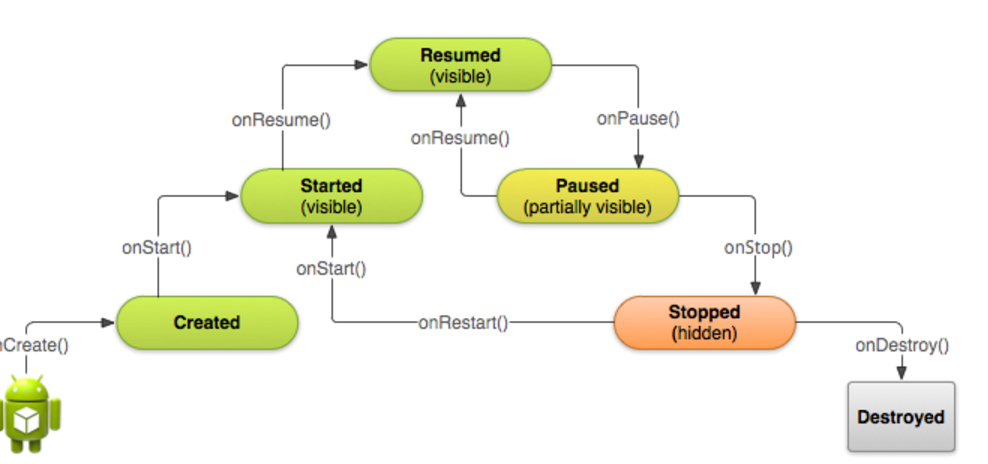
管理 Activity 生命周期：<https://developer.android.com/training/basics/activity-lifecycle/index.html>

**学习有关 Activity 生命周期、用户如何启动您的应用以及如何执行基本 Activity 创建操作的基础知识**。



Activity 只能在三种状态之一下存在很长时间。

继续

在这种状态下，Activity 处于前台，且用户可以与其交互。（有时也称为“运行”状态。）

暂停

在这种状态下，Activity 被在前台中处于半透明状态或者未覆盖整个屏幕的另一个 Activity—部分阻挡。 暂停的 Activity 不会接收用户输入并且无法执行任何代码。

停止

在这种状态下，Activity 被完全隐藏并且对用户不可见；它被视为处于后台。 停止时，Activity 实例及其诸如成员变量等所有状态信息将保留，但它无法执行任何代码。

其他状态（“创建”和“开始”）是瞬态，系统会通过调用下一个生命周期回调方法从这些状态快速移到下一个状态。 也就是说，在系统调用 onCreate() 之后，它会快速调用 onStart()，紧接着快速调用 onResume()。

如果未对您的 Activity 之一声明 [MAIN](https://developer.android.com/reference/android/content/Intent.html#ACTION_MAIN) 操作或 [LAUNCHER](https://developer.android.com/reference/android/content/Intent.html#CATEGORY_LAUNCHER) 类别，那么您的应用图标将不会出现在应用的主屏幕列表中。

您必须实现 [onCreate()](https://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) 方法执行只应在 Activity 整个生命周期出现一次的基本应用启动逻辑。

当 Activity 的第一个生命周期回调是 [onCreate()](https://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) 时，它最近的回调是 [onDestroy()](https://developer.android.com/reference/android/app/Activity.html#onDestroy())。系统会对您的 Activity 调用此方法，作为您的 Activity 实例完全从系统内存移除的最终信号。

大多数应用不需要实现此方法，因为局部类引用与 Activity 一同销毁，并且您的 Activity 应在 [onPause()](https://developer.android.com/reference/android/app/Activity.html#onPause()) 和 [onStop()](https://developer.android.com/reference/android/app/Activity.html#onStop()) 期间执行大多数清理操作。但是，如果您的 Activity 包含您在 [onCreate()](https://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) 期间创建的后台线程或其他如若未正确关闭可能导致内存泄露的长期运行资源，您应在 [onDestroy()](https://developer.android.com/reference/android/app/Activity.html#onDestroy()) 期间终止它们。

**注**：在所有情况下，系统在调用 [onPause()](https://developer.android.com/reference/android/app/Activity.html#onPause()) 和 [onStop()](https://developer.android.com/reference/android/app/Activity.html#onStop()) 之后都会调用 [onDestroy()](https://developer.android.com/reference/android/app/Activity.html#onDestroy())，只有一个例外：当您从 [onCreate()](https://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) 方法内调用 [finish()](https://developer.android.com/reference/android/app/Activity.html#finish()) 时。在某些情况下，比如当您的 Activity 作为临时决策工具运行以启动另一个 Activity 时，您可从 [onCreate()](https://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) 内调用 [finish()](https://developer.android.com/reference/android/app/Activity.html#finish()) 来销毁 Activity。在这种情况下，系统会立刻调用 [onDestroy()](https://developer.android.com/reference/android/app/Activity.html#onDestroy())，而不调用任何其他 生命周期方法。

**学习 Activity 暂停时（部分隐藏）和继续时的情况以及您应在这些状态变化期间执行的操作。**

只要 Activity 仍然部分可见但目前又未处于焦点之中，它会一直暂停。但是，一旦 Activity 完全被阻挡并且不可见，它便停止（将在下一堂课中讨论）

当您的 Activity 进入暂停状态时，系统会对您的 [Activity](https://developer.android.com/reference/android/app/Activity.html) 调用 [onPause()](https://developer.android.com/reference/android/app/Activity.html#onPause()) 方法，通过该方法，您可以停止不应在暂停时继续的进行中操作（比如视频），还可保留在用户继续离开应用时应予永久保存的任何信息。如果用户从暂停状态返回到您的 Activity，系统会重新开始该 Activity 并调用 [onResume()](https://developer.android.com/reference/android/app/Activity.html#onResume()) 方法。

当系统为您的 Activity 调用 [onPause()](https://developer.android.com/reference/android/app/Activity.html#onPause()) 时，它从技术角度看意味着您的 Activity 仍然处于部分可见状态，但往往说明用户即将离开 Activity 并且它很快就要进入“停止”状态。 您通常应使用 [onPause()](https://developer.android.com/reference/android/app/Activity.html#onPause()) 回调：

* 检查 Activity 是否可见。如果不可见，请停止动画或其他正在进行的可能消耗 CPU 的操作。记住，从 Android 7.0 开始，暂停的应用可能会在[多窗口模式](https://developer.android.com/guide/topics/ui/multi-window.html)下运行。 在本示例中，您可能不想停止动画或视频播放。
* 提交未保存的更改，但仅当用户离开时希望永久性保存此类更改（比如电子邮件草稿）。
* 释放系统资源，比如广播接收器、传感器手柄（比如 GPS） 或当您的 Activity 暂停且用户不需要它们时仍然可能影响电池寿命的任何其他资源。

例如，如果您的应用使用 [Camera](https://developer.android.com/reference/android/hardware/Camera.html)，[onPause()](https://developer.android.com/reference/android/app/Activity.html#onPause()) 方法是释放它的好位置。

@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;  
    }  
}

一般情况下，您**不得**使用 [onPause()](https://developer.android.com/reference/android/app/Activity.html#onPause()) 永久性存储用户更改（比如输入表格的个人信息）。 只有在您确定用户希望自动保存这些更改的情况（比如，草拟电子邮件时）下，才能在 [onPause()](https://developer.android.com/reference/android/app/Activity.html#onPause()) 中永久性存储用户更改。但您应避免在 [onPause()](https://developer.android.com/reference/android/app/Activity.html#onPause()) 期间执行 CPU 密集型工作，比如向数据库写入信息，因为这会拖慢向下一 Activity 过渡的过程（您应改为在 [onStop()](https://developer.android.com/reference/android/app/Activity.html#onStop()) 间执行高负载关机操作）。

请注意，每当您的 Activity 进入前台时系统便会调用此方法，包括它初次创建之时。 同样地，您应实现 [onResume()](https://developer.android.com/reference/android/app/Activity.html#onResume()) 以初始化您在 [onPause()](https://developer.android.com/reference/android/app/Activity.html#onPause()) 期间释放的组件，并执行每当 Activity 进入“继续”状态时必须进行的任何其他初始化操作（比如开始动画和初始化只在 Activity 具有用户焦点时使用的组件）。

[onResume()](https://developer.android.com/reference/android/app/Activity.html#onResume()) 的以下示例对应于以上的 [onPause()](https://developer.android.com/reference/android/app/Activity.html#onPause()) 示例，因此它初始化 Activity 暂停时释放的照相机。

@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  
    }  
}

**学习用户完全离开您的 Activity 并返回到该 Activity 时发生的情况。**

正确停止和重启 Activity 是 Activity 生命周期中的重要过程，其可确保您的用户知晓应用始终保持活跃状态并且不会丢失进度。Activity 停止和重启的场景主要有以下几种：

* 用户打开“最近应用”窗口并从您的应用切换到另一个应用。您的应用中当前位于前台的 Activity 将停止。 如果用户从主屏幕启动器图标或“最近应用”窗口返回到您的应用，将会重启 Activity。
* 用户在您的应用中执行开始新 Activity 的操作。当第二个 Activity 创建好后，当前 Activity 便停止。 如果用户之后按了*返回*按钮，第一个 Activity 会重新开始。
* 用户在其手机上使用您的应用的同时接听来电。

[Activity](https://developer.android.com/reference/android/app/Activity.html) 类提供两种生命周期方法： [onStop()](https://developer.android.com/reference/android/app/Activity.html#onStop()) 和 [onRestart()](https://developer.android.com/reference/android/app/Activity.html#onRestart())，这些方法允许您专门处理 Activity 句柄停止和重新启动的方式。 不同于识别部分 UI 阻挡的暂停状态，停止状态保证 UI 不再可见，且用户的焦点在另外的 Activity（或完全独立的应用）中。

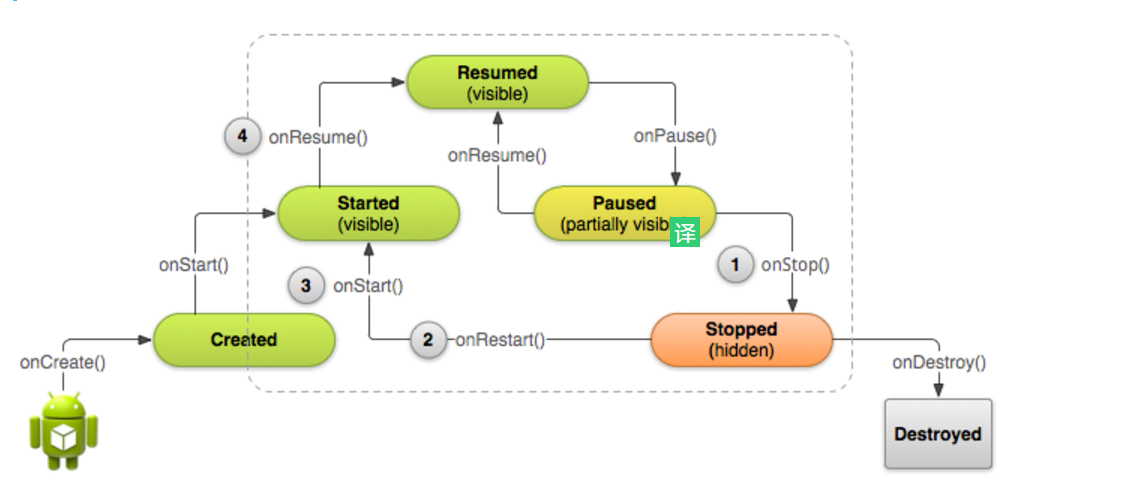
当您的 Activity 收到 [onStop()](https://developer.android.com/reference/android/app/Activity.html#onStop()) 方法的调用时，它不再可见，并且应释放几乎所有用户不使用时不需要的资源。 一旦您的 Activity 停止，如果需要恢复系统内存，系统可能会销毁该实例。 在极端情况下，系统可能会仅终止应用进程，而不会调用 Activity 的最终 [onDestroy()](https://developer.android.com/reference/android/app/Activity.html#onDestroy()) 回调，因此您使用 [onStop()](https://developer.android.com/reference/android/app/Activity.html#onStop()) 释放可能泄露内存的资源非常重要。

尽管 [onPause()](https://developer.android.com/reference/android/app/Activity.html#onPause()) 方法在 [onStop()](https://developer.android.com/reference/android/app/Activity.html#onStop())之前调用，您应使用 [onStop()](https://developer.android.com/reference/android/app/Activity.html#onStop()) 执行更大、占用更多 CPU 的关闭操作，比如向数据库写入信息。

例如，此处是将草稿笔记内容保存在永久存储中的 [onStop()](https://developer.android.com/reference/android/app/Activity.html#onStop()) 的实现：

@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.  
            );  
}

当您的 Activity 停止时， [Activity](https://developer.android.com/reference/android/app/Activity.html) 对象将驻留在内存中并在 Activity 继续时被再次调用。 您无需重新初始化在任何导致进入“继续”状态的回调方法过程中创建的组件。 系统还会在布局中跟踪每个 [View](https://developer.android.com/reference/android/view/View.html) 的当前状态，如果用户在 [EditText](https://developer.android.com/reference/android/widget/EditText.html) 小部件中输入文本，该内容会保留，因此您无需保存即可恢复它。



当您的 Activity 从停止状态返回前台时，它会接收对 [onRestart()](https://developer.android.com/reference/android/app/Activity.html#onRestart()) 的调用。系统还会在每次您的 Activity 变为可见时调用 [onStart()](https://developer.android.com/reference/android/app/Activity.html#onStart()) 方法（无论是正重新开始还是初次创建）。但是，只会在 Activity 从停止状态继续时调用 [onRestart()](https://developer.android.com/reference/android/app/Activity.html#onRestart()) 方法，因此您可以使用它执行只有在 Activity 之前停止但未销毁的情况下可能必须执行的特殊恢复工作。

应用需要使用 [onRestart()](https://developer.android.com/reference/android/app/Activity.html#onRestart()) 恢复 Activity 状态的情况并不常见，因此没有适用于一般应用群体的任何方法指导原则。 但是，因为您的 [onStop()](https://developer.android.com/reference/android/app/Activity.html#onStop()) 方法应基本清理所有 Activity 的资源，您将需要在 Activity 重新开始时重新实例化它们。 但是，您还需要在您的 Activity 初次创建时重新实例化它们（没有 Activity 的现有实例）。 出于此原因，您应经常使用 [onStart()](https://developer.android.com/reference/android/app/Activity.html#onStart()) 回调方法作为 [onStop()](https://developer.android.com/reference/android/app/Activity.html#onStop()) 方法的对应部分，因为系统会在它创建您的 Activity 以及从停止状态重新启动 Activity 时调用 [onStart()](https://developer.android.com/reference/android/app/Activity.html#onStart())。

例如，因为用户可能在回到它之前已离开应用很长时间，[onStart()](https://developer.android.com/reference/android/app/Activity.html#onStart()) 方法是验证所需系统功能是否已启用的理想选择：

@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  
}

**学习您的 Activity 被销毁时的情况以及您如何能够根据需要重新构建 Activity。**

当您的 Activity 因用户按了返回 或 Activity 自行完成而被销毁时，系统的 [Activity](https://developer.android.com/reference/android/app/Activity.html) 实例概念将永久消失，因为行为指示不再需要 Activity。但是，如果系统因系统局限性（而非正常应用行为）而销毁 Activity，尽管 [Activity](https://developer.android.com/reference/android/app/Activity.html) 实际实例已不在，系统会记住其存在，这样，如果用户导航回实例，系统会使用描述 Activity 被销毁时状态的一组已保存数据创建 Activity 的新实例。 系统用于恢复先前状态的已保存数据被称为“实例状态”，并且是 [Bundle](https://developer.android.com/reference/android/os/Bundle.html)对象中存储的键值对集合。

要保存有关 Activity 状态的其他数据，您必须替代 [onSaveInstanceState()](https://developer.android.com/reference/android/app/Activity.html#onSaveInstanceState(android.os.Bundle)) 回调方法。当用户要离开 Activity 并在 Activity 意外销毁时向其传递将保存的 [Bundle](https://developer.android.com/reference/android/os/Bundle.html) 对象时，系统会调用此方法。 如果系统必须稍后重新创建 Activity 实例，它会将相同的 [Bundle](https://developer.android.com/reference/android/os/Bundle.html) 对象同时传递给 [onRestoreInstanceState()](https://developer.android.com/reference/android/app/Activity.html#onRestoreInstanceState(android.os.Bundle)) 和 [onCreate()](https://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) 方法。

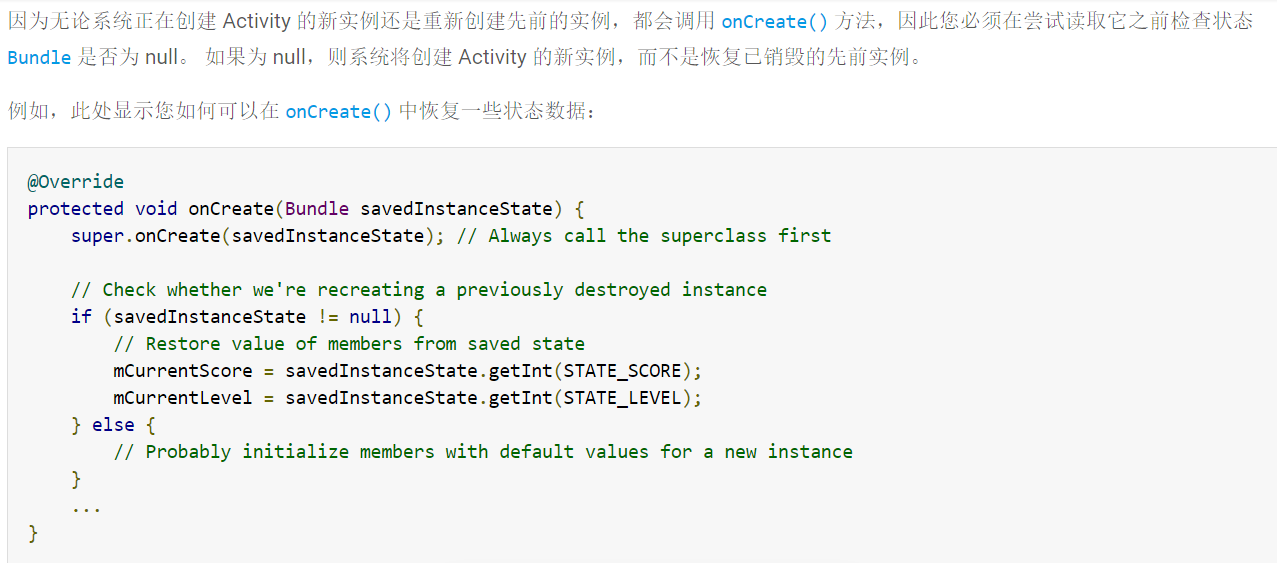


保存 Activity 状态



## 恢复 Activity 状态

您的 Activity 在先前销毁之后重新创建时，您可以从系统向 Activity 传递的 [Bundle](https://developer.android.com/reference/android/os/Bundle.html) 恢复已保存的状态。[onCreate()](https://developer.android.com/reference/android/app/Activity.html#onCreate(android.os.Bundle)) 和 [onRestoreInstanceState()](https://developer.android.com/reference/android/app/Activity.html#onRestoreInstanceState(android.os.Bundle)) 回调方法均接收包含实例状态信息的相同 [Bundle](https://developer.android.com/reference/android/os/Bundle.html)。。





[**https://developer.android.com/training/basics/data-storage/index.html**](https://developer.android.com/training/basics/data-storage/index.html) **保存数据**



1. 保存键值集

如果您有想要保存的相对较小键值集合，您应使用 [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html)

[SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html) 对象指向包含键值对的文件并提供读写这些文件的简单方法。 每个[SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html) 文件由框架进行管理并且可以专用或共享。

**注**：[SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html) API 仅用于读写键值对，您不得将其与 [Preference](https://developer.android.com/reference/android/preference/Preference.html) API 混淆，后者帮助您为您的应用设置构建用户界面（尽管它们使用 [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html) 作为其实现以保存应用设置）。 有关使用 [Preference](https://developer.android.com/reference/android/preference/Preference.html) API 的信息，请参阅[设置](https://developer.android.com/guide/topics/ui/settings.html)指南。





1. 保存文件https://developer.android.com/training/basics/data-storage/files.html

Android 使用与其他平台上基于磁盘的文件系统类似的文件系统。 本课程讲述如何使用 Android 文件系统通过 [File](https://developer.android.com/reference/java/io/File.html) API 读取和写入文件。

[File](https://developer.android.com/reference/java/io/File.html) 对象适合按照**从开始到结束的顺序不跳过地读取或写入大量数据**。 例如，它适合于图片文件或通过网络交换的任何内容。

