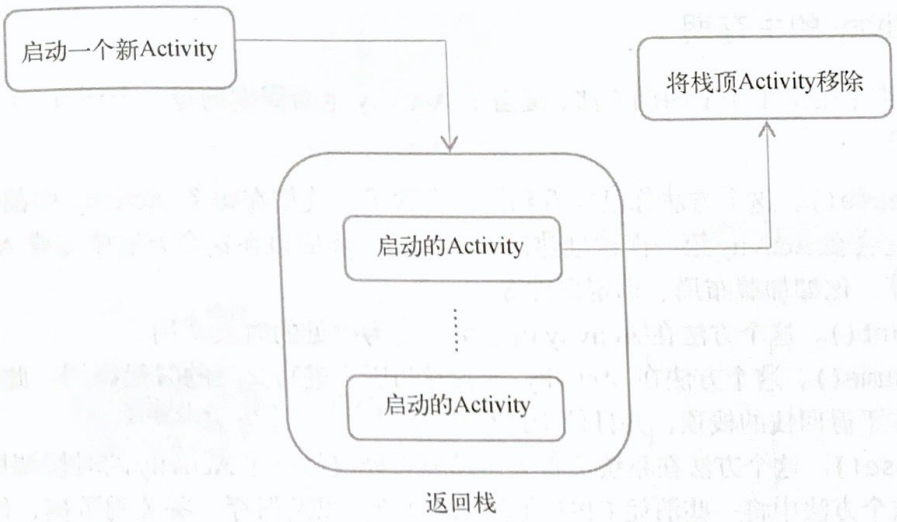

图 3.23 返回栈工作示意图

3.4.2 Activity 状态

每个 Activity 在其生命周期中最多可能会有 4 种状态。

1. 运行状态

当一个 Activity 位于返回栈的栈顶时，Activity 就处于运行状态。系统最不愿意回收的就是处于运行状态的 Activity，因为这会带来非常差的用户体验。

2. 暂停状态

当一个 Activity 不再处于栈顶位置，但仍然可见时，Activity 就进入了暂停状态。你可能会觉得，既然 Activity 已经不在栈顶了，怎么会可见呢？这是因为并不是每一个 Activity 都会占满整个屏幕，比如对话框形式的 Activity 只会占用屏幕中间的部分区域。处于暂停状态的 Activity 仍然是完全存活着的，系统也不愿意回收这种 Activity（因为它还是可见的，回收可见的东西都会在用户体验方面有不好的影响），只有在内存极低的情况下，系统才会去考虑回收这种 Activity。

3. 停止状态

当一个 Activity 不再处于栈顶位置，并且完全不可见的时候，就进入了停止状态。系统仍然会为这种 Activity 保存相应的状态和成员变量，但是这并不是完全可靠的，当其他地方需要内存时，处于停止状态的 Activity 有可能会被系统回收。

4. 销毁状态

一个 Activity 从返回栈中移除后就变成了销毁状态。系统最倾向于回收处于这种状态的 Activity，以保证手机的内存充足。

3.4.3　Activity 的生存期

Activity 类中定义了 7 个回调方法，覆盖了 Activity 生命周期的每一个环节，下面就来一一介绍这 7 个方法。

- onCreate()。这个方法你已经看到过很多次了，我们在每个 Activity 中都重写了这个方法，它会在 Activity 第一次被创建的时候调用。你应该在这个方法中完成 Activity 的初始化操作，比如加载布局、绑定事件等。
- onStart()。这个方法在 Activity 由不可见变为可见的时候调用。
- onResume()。这个方法在 Activity 准备好和用户进行交互的时候调用。此时的 Activity 一定位于返回栈的栈顶，并且处于运行状态。
- onPause()。这个方法在系统准备去启动或者恢复另一个 Activity 的时候调用。我们通常会在这个方法中将一些消耗 CPU 的资源释放掉，以及保存一些关键数据，但这个方法的执行速度一定要快，不然会影响到新的栈顶 Activity 的使用。
- onStop()。这个方法在 Activity 完全不可见的时候调用。它和 onPause()方法的主要区别在于，如果启动的新 Activity 是一个对话框式的 Activity，那么 onPause()方法会得到执行，而 onStop()方法并不会执行。
- onDestroy()。这个方法在 Activity 被销毁之前调用，之后 Activity 的状态将变为销毁状态。
- onRestart()。这个方法在 Activity 由停止状态变为运行状态之前调用，也就是 Activity 被重新启动了。

以上 7 个方法中除了 onRestart()方法，其他都是两两相对的，从而又可以将 Activity 分为以下 3 种生存期。

- **完整生存期**。Activity 在 onCreate()方法和 onDestroy()方法之间所经历的就是完整生存期。一般情况下，一个 Activity 会在 onCreate()方法中完成各种初始化操作，而在 onDestroy()方法中完成释放内存的操作。
- **可见生存期**。Activity 在 onStart()方法和 onStop()方法之间所经历的就是可见生存期。在可见生存期内，Activity 对于用户总是可见的，即便有可能无法和用户进行交互。我们可以通过这两个方法合理地管理那些对用户可见的资源。比如在 onStart()方法中对资源进行加载，而在 onStop()方法中对资源进行释放，从而保证处于停止状态的 Activity 不会占用过多内存。
- **前台生存期**。Activity 在 onResume()方法和 onPause()方法之间所经历的就是前台生存期。在前台生存期内，Activity 总是处于运行状态，此时的 Activity 是可以和用户进行交互的，我们平时看到和接触最多的就是这个状态下的 Activity。

为了帮助你更好地理解，Android 官方提供了一张 Activity 生命周期的示意图，如图 3.24 所示。

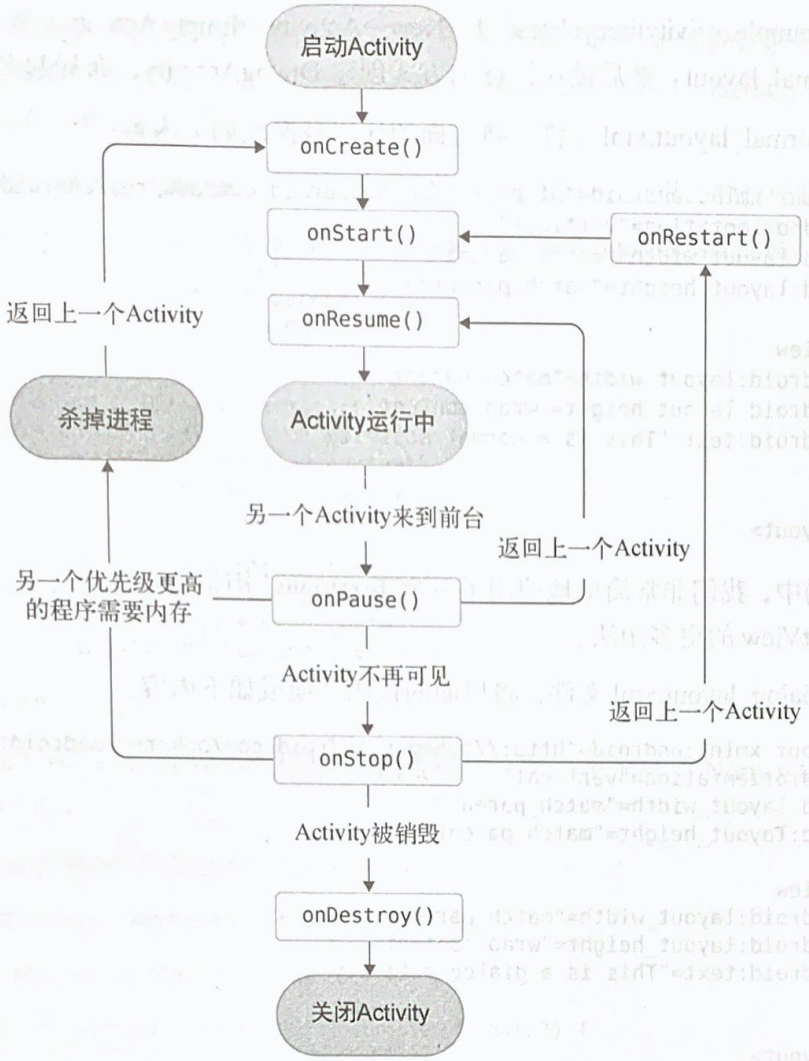

图 3.24 Activity 的生命周期

3.4.4 体验 Activity 的生命周期

讲了这么多理论知识，是时候进行实战了。下面我们将通过一个实例，让你可以更加直观地体验 Activity 的生命周期。

这次我们不准备在 ActivityTest 这个项目的基础上修改了，而是新建一个项目。因此，首先关闭 ActivityTest 项目，点击导航栏 File→Close Project。然后新建一个 ActivityLifeCycleTest 项目，新建项目的过程你应该已经非常清楚了，不需要我再进行赘述，这次我们允许 Android Studio 帮我们自动创建 Activity 和布局，这样可以省去不少工作，创建的 Activity 名和布局名都使用默认值。

这样主 Activity 就创建完成了，我们还需要分别再创建两个子 Activity——NormalActivity 和 DialogActivity，下面一步步来实现。

右击 com.example.activitylifecycletest 包→New→Activity→Empty Activity，新建 NormalActivity，布局起名为 normal_layout。然后使用同样的方式创建 DialogActivity，布局起名为 dialog_layout。

现在编辑 normal_layout.xml 文件，将里面的代码替换成如下内容：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="This is a normal activity"
    />

</LinearLayout>
```

在这个布局中，我们非常简单地使用了一个 TextView，用于显示一行文字，在下一章中你将会学到关于 TextView 的更多用法。

然后编辑 dialog_layout.xml 文件，将里面的代码替换成如下内容：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="This is a dialog activity"
    />

</LinearLayout>
```

两个布局文件的代码几乎没有区别，只是显示的文字不同而已。

NormalActivity 和 DialogActivity 中的代码我们保持默认就好，不需要改动。

其实从名字上就可以看出，这两个 Activity 一个是普通的 Activity，一个是对话框式的 Activity。可是我们并没有修改 Activity 的任何代码，两个 Activity 的代码应该几乎是一模一样的，那么是在哪里将 Activity 设成对话框式的呢？别着急，下面我们马上开始设置。修改 AndroidManifest.xml 的 <activity> 标签的配置，如下所示：

```
<activity android:name=".DialogActivity"
    android:theme="@style/Theme.AppCompat.Dialog">
</activity>
<activity android:name=".NormalActivity">
</activity>
```

这里是两个 Activity 的注册代码，但是 DialogActivity 的代码有些不同，我们给它使用了一