# 第 14 章 fragment的保留

当前，HelloMoon应用对设备旋转的处理还不够完善。运行HelloMoon应用，播放音频，然后旋转设备。音频播放会嘎然而止。

设备旋转后，HelloMoonActivity随即被销毁。与此同时，负责销毁HelloMoonFragment的FragmentManager立即逐一调用fragment的生命周期方法，即onPause()、onStop()和OnDestroy()方法。我们知道，HelloMoonFragment.onDestroy()方法被调用后，MediaPlayer实例即被释放，结果导致了音频播放的停止。

在本书的第3章，我们通过覆盖Activity.onSaveInstanceState(Bundle)方法，修复了GeoQuiz应用的设备旋转相关问题。设备旋转后，新产生的activity读取保存的数据，然后恢复到旋转前的状态。Fragment具有相同功能的onSaveInstanceState(Bundle)方法。然而，就算保存了MediaPlayer对象的状态并在随后恢复，音频播放仍会中断。这显然会惹恼用户。

## 14.1 保留 fragment 实例

幸运的是，为应对设备配置的变化，可使用fragment的一个特殊方法来确保MediaPlayer实例一直存在。覆盖HelloMoonFragment.onCreate(...)方法并设置fragment的属性值，如代码清单14-1所示。

**代码清单14-1 调用setRetainInstance(true)方法（HelloMoonFragment.java）**

```
...

private Button mPlayButton;
private Button mStopButton;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setRetainInstance(true);
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup parent,
        Bundle savedInstanceState) {
    ...
```



图灵社区会员 zwjjbb(zwjjbb@qq.com) 专享 尊重版权

fragment的retainInstance属性值默认为false。这表明其不会被保留。因此，设备旋转时fragment会随托管activity一起销毁并重建。调用setRetainInstance(true)方法可保留fragment。已保留的fragment不会随activity一起被销毁。相反，它会被一直保留并在需要时原封不动的传递给新的activity。

对于已保留的fragment实例，其全部实例变量（如mPlayButton、MPlayer和mStopButton）值也将保持不变，因此可放心继续使用。

运行HelloMoon应用。播放音频，然后旋转设备，可看到音频的播放丝毫未受影响。

## 14.2　设备旋转与保留的 fragment

我们来看看保留的fragment的工作原理。保留的fragment利用了这样一个事实： 可销毁和重建fragment的视图，但无需销毁fragment自身。

设备配置发生改变时，FragmentManager首先销毁队列中的fragment的视图。在设备配置改变时,总是销毁与重建fragment与activity的视图，都是基于同样的理由：新的配置可能需要新的资源来匹配；当有更合适的匹配资源可以利用时，则需重新创建视图。

紧接着，FragmentManager检查每个fragment的retainInstance属性值。如属性值为false（初始默认值），FragmentManager会立即销毁该fragment实例。随后，为适应新的设备配置，新activity的新FragmentManager会创建一个新的fragment及其视图，如图14-1所示。

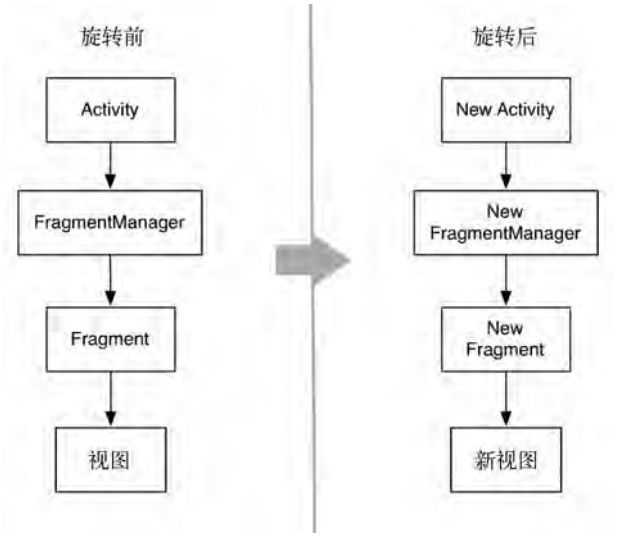


图14-1　设备旋转与默认不保留的UI fragment

如属性值为true，则该fragment的视图立即被销毁，但fragment本身不会被销毁。为适应新的设备配置，当新的activity创建后，新的FragmentManager会找到被保留的fragment，并重新创建它的视图，如图14-2所示。

图灵社区会员 zwjjbb(zwjjbb@qq.com) 专享 尊重版权

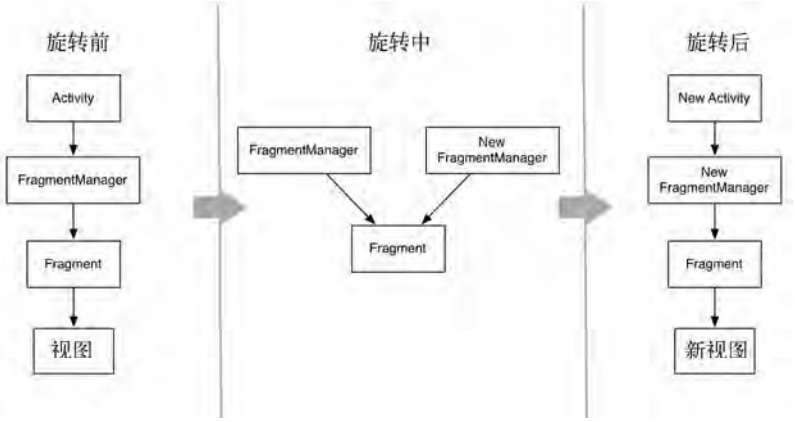


图14-2 设备旋转与保留的UI fragment

虽然保留的fragment没有被销毁，但它已脱离消亡中的activity并处于保留状态。尽管此时的fragment仍然存在，但已没有任何activity在托管它。

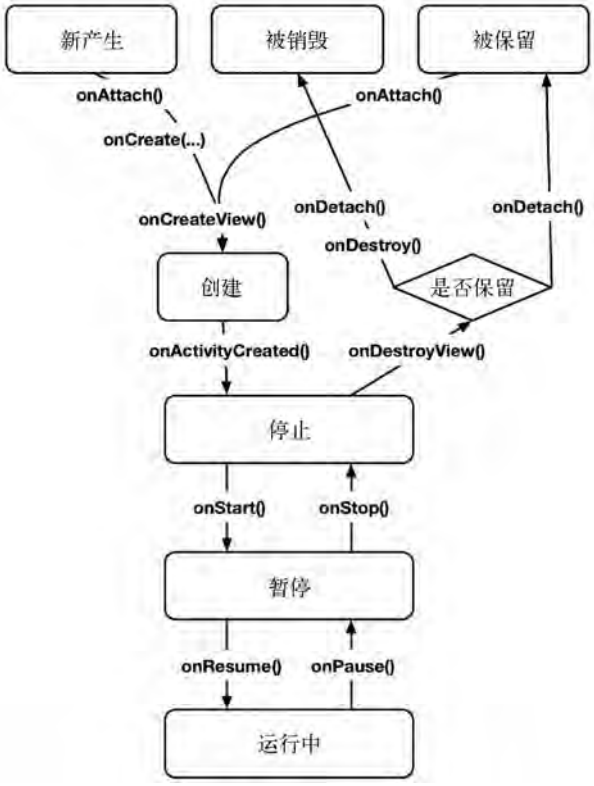


图14-3 fragment的生命周期

图灵社区会员 zwjjbb(zwjjbb@qq.com) 专享 尊重版权

fragment必须同时满足两个条件才能进入保留状态：

- 已调用了fragment的`setRetainInstance(true)`方法
- 因设备配置改变（通常为设备旋转），托管activity正在被销毁

Fragment处于保留状态的时间非常短暂，即fragment脱离旧activity到重新附加给立即创建的新activity之间的一段时间。

## 14.3　保留的 fragment：一切都完美了吗

保留fragment可以说是Android里巧妙的设计，不是吗？没错！它确实给应用开发带来了极大地便利，貌似解决了因设备旋转而销毁activity和fragment所导致的全部问题。当设备配置发生改变时，除了通过创建全新视图获取最合适的资源以外，还可轻松保留原有数据及对象。

为什么不保留每个fragment，或者默认设置fragment的retainInstance属性值为true？ 这是因为Android似乎并不鼓励保留fragment。我们尚不明确其具体原因，但需要指出的是，如果哪天Android开发团队不再重视fragment的此项特色，保不准什么时候会出问题。

请记住，只有当activity因设备配置发生改变被销毁时，fragment才会短时间处于被保留状态。如果activity是因操作系统需要回收内存而被销毁，则所有被保留的fragment也会被随之销毁。

## 14.4　设备旋转处理与 onSaveInstanceState(Bundle)方法

`onSaveInstanceState(Bundle)`方法是用于处理设备旋转问题的另一工具。事实上，如果某个应用不存在任何设备旋转相关问题，这还要归功于`onSaveInstanceState(Bundle)`方法的默认工作行为。

CriminalIntent应用就是一个很好的例子。`CrimeFragment`没有被保留，但如果改变某项crime的标题或者切换问题是否解决的状态，则`View`对象的新状态会被自动保存并在设备旋转后得到恢复。这就是`onSaveInstanceState(...)`方法的设计用途——保存并恢复应用的UI状态。

覆盖`Fragment.onSaveInstanceState(...)`方法与保留fragment方法的主要区别在于，数据可以保存多久。如只需短暂保留数据，能应对设备配置改变就可以了，则保留fragment可以很轻松地解决问题。如果是保存对象，则更能体会使用保留fragment的便利。因为我们再也无需操心要保存的对象是否已实现`Serializable`接口了。

如需持久地保存数据，保留fragment的方式就行不通了。用户暂时离开应用后，如系统因回收内存需要销毁activity，则保留的fragment也会被随之销毁。

为更清楚地了解两种数据保存方式的差异，我们再回头看看GeoQuiz应用。当时我们面临的问题是，一旦设备发生旋转，数组中题目的索引值即被重置为零。无论用户当前正在回答哪一道题目，设备旋转后，用户总是会回到第一道题目。保存题目的索引值，然后在设备旋转后重新读取该值，以保证用户能够看到正确的题目。

GeoQuiz应用没有使用fragment。不过，假设以`QuizActivity`托管一个`QuizFragment`的方式，重新设计GeoQuiz应用。对于覆盖`Fragment.onSaveInstanceState(...)`方法保存题目索

图灵社区会员 zwjjbb(zwjjbb@qq.com) 专享 尊重版权

引值，以及通过保留QuizFragment存留变量这两种方式，又该如何选择？

图14-4为三种需处理的不同生命周期：activity对象（包括非保留的fragment）的生命周期，被保留fragment的生命周期以及activity记录的生命周期。

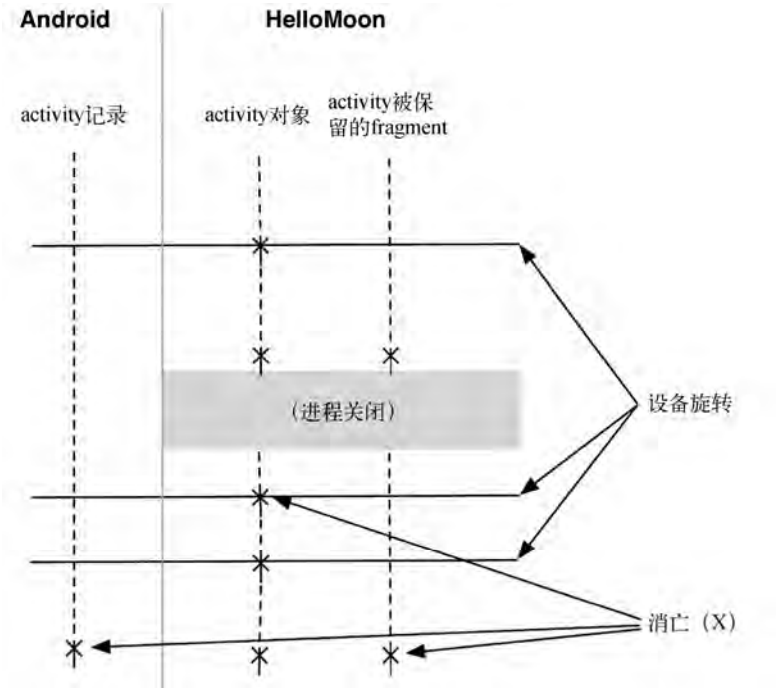


图14-4 三种不同的生命周期

activity对象的生命周期最短。这也是设备旋转问题的根源。题目索引值保留的时间需长于activity对象的生命周期。

如保留了QuizFragment，则题目索引值存留的时间也就是被保留fragment的生命周期。GeoQuiz应用只包含5道题目，因此选择保留QuizFragment的方式来处理设备旋转问题，相对要容易一些，且所需编写的代码量也不多。只需先初始化题目索引成员变量，然后在QuizFragment.onCreate(...)方法中调用setRetainInstance(true)方法即可，如代码清单14-2所示。

**代码清单14-2 保留假想的QuizFragment**

```
public class QuizFragment extends Fragment {

    ...

    private int mCurrentIndex = 0;

    ...

    @Override
```

图灵社区会员 zwjjbb(zwjjbb@qq.com) 专享 尊重版权

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRetainInstance(true);
    }

    ...
}
```

通过将题目索引变量同被保留fragment的生命周期绑定同步，则题目索引变量可不受activity对象销毁的影响而保留下来，从而解决了设备旋转导致的索引变量值重置的问题。然而，如图14-4所示，进程关闭时，被保留QuizFragment中的索引变量值也会被销毁。进程的关闭通常可能发生在用户暂时离开当前应用，系统为回收内存而销毁activity以及保留fragment的时候。

应用当前只包含五道题目，用户被要求从头来过勉强可以接受。但如果GeoQuiz应用含有100道题目呢？返回应用后发现又要从第一道题目重新开始，用户不疯掉才怪！显然，需将题目索引变量的存留时间与activity记录的生命周期保持同步。因此，应在onSaveInstance State(...)方法中将题目索引变量值保存下来。这样，用户在暂时离开应用再返回时，仍可接上题继续开始答题。

因此，如果activity或fragment中有需要长久保存的东西，则应覆盖onSaveInstance State(Bundle)方法，将其状态保存下来。这样，由于同activity记录的生命周期保持了同步，后续可在需要时对其进行恢复。

## 14.5　深入学习：fragment 引入前的设备旋转问题

fragment引入自Honeycomb系统版本，只适用于Ice Cream Sandwich系统版本以上的设备。然而，设备旋转问题却早已有之。

了解到fragment引入前Android是如何解决设备旋转问题的，我们会更加欣赏保留fragment易于使用的神奇魅力。可以说，fragment引入前的设备旋转问题处理复杂到令人瞠目结舌。

- 需要在设备配置发生改变时保留对象吗？覆盖onRetainNonConfigurationInstance()方法，返回需保留的对象。然后，在需要取回的时候，再去调用getLastNonConfiguratio-nInstance()方法。
- 需要保留一个activity中的多个对象？对不起，一个activity只能保留一个对象。如需保留多个对象，需设法将要保存的对象打包在一起。
- 得到一个类似MediaPlayer的对象，在activity因设备发生旋转而被销毁时，需对其进行清理？首先，必须确认onRetainNonConfigurationInstance()方法未被调用，然后再调用onDestroy()方法完成任务。

如今，onRetainNonConfigurationInstance()方法已被废弃。设备旋转时需要保存并恢复的对象几乎都是利用保留fragment完成的。习惯使用fragment后，我们会发现，相比以前设备旋转问题的处理方式，这种方法要简单轻松得多。

图灵社区会员 zwjjbb(zwjjbb@qq.com) 专享 尊重版权