25个增强iOS应用程序性能的提示和技巧 — 高级篇

- 作者 BeyondVincent
- 11 四月, 2013
- 暂无评论

本文由破船译自：raywenderlich
转载请注明出处：BeyondVincent的博客

在开发iOS应用程序时，让程序具有良好的性能是非常关键的。这也是用户所期望的，如果你的程序运行迟钝或缓慢，会招致用户的差评。

然而由于iOS设备的局限性，有时候要想获得良好的性能，是很困难的。在开发过程中，有许多事项需要记住，并且关于性能影响很容易就忘记。

这就是为什么我要写这篇文章！本文收集了25个关于可以提升程序性能的提示和技巧。

目录

我把性能优化技巧分为3个不同的等级：初级、中级和高级：

高级
当且仅当下面这些技巧能够解决问题的时候，才使用它们：

高级性能提升

寻找一些高明的方法，让自己变为一个全代码忍者？下面这些高级的性能优化技巧可以在适当的时候让程序尽可能的高效运行！

22) 加速启动时间

能快速的启动程序非常重要，特别是在用户第一次启动程序时。第一映像对程序来说非常重要！

让程序尽量快速启动的方法就是尽量以异步方式执行任务，例如网络请求，数据访问或解析。

另外，避免使用臃肿的XIBs，因为XIB的加载是在主线程中进行的。但是记住storyboard没有这样的问题——所以如果可以的话就使用storyboard吧！

注意：在利用Xcode进行调试时，watchdog不会运行，所在设备中测试程序启动性能时，不要将设备连接到Xcode。

23) 使用Autorelease Pool

NSAutoreleasePool负责释放一个代码块中的自动释放对象。一般都是由UIKit来创建的。不过有些情况下需要手动创建NSAutoreleasePool。

例如，如果在代码中创建了大量的临时对象，你将注意到内存使用量在增加，直到这些对象被释放。问题是只有当UIKit耗尽了 autorelease pool，这些对象才会被释放，也就是说当不再需要这些对象之后，这些对象还在内存中占据着资源。

不过这个问题完全可以避免：在@autoreleasepool代码块中创建临时对象，如下代码：

```
1. NSArray *urls = &lt;# An array of file URLs #&gt;;
2. for (NSURL *url in urls) {
3.     @autoreleasepool {
4.         NSError *error;
5.         NSString *fileContents = [NSString stringWithContentsOfURL:url
6.                                   encoding:NSUTF8StringEncoding error:&amp;error];
7.         /* Process the string, creating and autoreleasing more objects. */
8.     }
9. }
```

当每次迭代完之后，都会释放所有的autorelease对象。

关于NSAutoreleasePool的更多内容可以阅读苹果的官方文档。

24) 缓存图片 — 或者不缓存

iOS中从程序bundle中加载UIImage一般有两种方法。第一种比较常见：**imageNamed**。第二种方法很少使用：***imageWithContentsOfFile***

为什么有两种方法完成同样的事情呢？

imageNamed的优点在于可以缓存已经加载的图片。苹果的文档中有如下说法：

This method looks in the system caches for an image object with the specified name and returns that object if it exists. If a matching image object is not already in the cache, this method loads the image data from the specified file, caches it, and then returns the resulting object. 这种方法会在系统缓存中根据指定的名字寻找图片，如果找到了就返回。如果没有在缓存中找到图片，该方法会从指定的文件中加载图片数据，并将其缓存起来，然后再把结果返回。

而**imageWithContentsOfFile**方法只是简单的加载图片，并不会将图片缓存起来。

这两个方法的使用方法如下：

```
1. UIImage *img = [UIImage imageNamed:@"myImage"]; // caching
2. // or
3. UIImage *img = [UIImage imageWithContentsOfFile:@"myImage"]; // no caching
```

那么该如何选择呢？

如果加载一张很大的图片，并且只使用一次，那么就不需要缓存这个图片。这种情况**imageWithContentsOfFile**比较合适——系统不会浪费内存来缓存图片。

然而，如果在程序中经常需要重用的图片，那么最好是选择**imageNamed**方法。这种方法可以节省出每次都从磁盘加载图片的时间。

25) 尽量避免Date格式化

如果有许多日期需要使用NSDateFormatter，那么需要小心对待了。如之前（重用花销很大的对象）所提到的，无论什么时候，都应该尽量重用NSDateFormatters。

然而，如果你需要更快的速度，那么应该使用C来直接解析日期，而不是NSDateFormatter。Sam Soffes写了一篇文章，其中提供了一些解析ISO-8601格式日期字符的串代码。你只需要简单的调整一下其中的代码就可以满足自己特殊的需求了。

这听起来不错把——不过你相信这还有更好的一个办法吗？

如果你自己能控制处理日期的格式，那么可以选择 Unix timestamps。Unix timestamps是一个简单的整数，代表了从新纪元时间（epoch）开始到现在已经过了多少秒，通常这个新纪元参考时间是00:00:00 UTC on 1 January 1970。

你可以很容易的见这个时间戳转换为NSDate，如下所示：

```
1. - (NSDate*)dateFromUnixTimestamp:(NSTimeInterval)timestamp {
2.     return [NSDate dateWithTimeIntervalSince1970:timestamp];
3. }
```

上面这个方法比C函数还要快！

注意：许多网络APIs返回的时间戳都是毫秒，因此需要注意的是在将这个时间戳传递给dateFromUnixTimestamp之前需要除以1000。

何去何从？

强烈建议对程序性能优化感兴趣的读者看看下面列出来的WWDC视频。在看视频之前，你需要注册一个Apple ID（只需要注册以此，就可以观看所有WWDC2012的视频）：

- #406: Adopting Automatic Reference Counting
- #238: iOS App Performance: Graphics and Animations
- #242: iOS App Performance: Memory
- #235: iOS App Performance: Responsiveness
- #409: Learning Instruments
- #706: Networking Best Practices
- #514: OpenGL ES Tools and Techniques
- #506: Optimizing 2D Graphics and Animation Performance
- #601: Optimizing Web Content in UIWebViews and Websites on iOS
- #225: Up and Running: Making a Great Impression with Every Launch

下面这些视频来自WWDC 2011 ，也非常有用：

- #308: Blocks and Grand Central Dispatch in Practice
- #323: Introducing Automatic Reference Counting
- #312: iOS Performance and Power Optimization with Instruments
- #105: Polishing Your App: Tips and tricks to improve the responsiveness and performance
- #121: Understanding UIKit Rendering

这里还有更多相关视频，大多数来自iOS 5技术讲座：

- Your iOS App Performance Hitlist
- Optimizing App Performance with Instruments
- Understanding iOS View Compositing

基于 “Your iOS App Performance Hitlist” 视频，Ole Begemann写了一篇文章。

苹果还提供了一篇非常好的文章：性能优化。其中提供的技巧和提示对程序性能提升很有帮助。

25个增强iOS应用程序性能的提示和技巧 — 初级篇
25个增强iOS应用程序性能的提示和技巧 — 中级篇

0

- 支持
- 反对