38 | 热点问题答疑（四）

戴铭 2019-06-06

00:00 09:06
讲述：冯永吉 大小：8.35M

你好，我是戴铭。今天这篇答疑文章，我要针对近期留言中的热点问题，进行一次集中解答。

目前，我们专栏已经更新完了基础篇、应用开发篇和原理篇 3 大模块的内容。其中，原理篇的内容，因为涉及到的都是底层原理，比如系统内核 XNU、AOP、内存管理和编译等，学习起来会很辛苦。但所谓良药苦口，你只有搞明白了这些最最底层的原理，才可以帮你抓住开发知识的规律，达到融会贯通的效果，进而提升自己造轮子、解决问题的能力。

也正因为这些底层知识比较难啃，需要细细琢磨，所以在这期答疑文章中，我并没有展开这个模块的内容。如果你对这个模块的文章有哪里不理解，或者觉得哪里有问题的话，可以在评论区留下你的观点，我会挑选合适的时机，给你答复。

接下来，我们就看看今天这篇文章要展开讨论的问题吧。

关于监控卡顿

@凡在第 13 篇文章《如何利用 RunLoop 原理去监控卡顿？》后问道：

> 大多数的卡顿监控，都是在主线程上做的。音视频播放以及直播的卡顿，能否使用这种方式来监控呢？另外，我们公司对接的直播都是第三方的库和知识平台，我应该如何把这种监控放到客户端来做呢？

针对这个同学的问题，我想说的是，只有在主线程上卡了，用户才会感知到，而监控卡顿主要就是要监控什么时候会卡。只要我们在发生卡顿的时刻，想办法去收集卡顿信息，就能够定位到问题，找出具体是由谁引起的卡顿。

比如，@凡同学提到的音视频播放卡顿问题，监控到发生卡顿的时刻，通过获取当时方法调用堆栈的方式，就能够确定出具体是哪个方法在调用，从而找到发生卡顿问题的原因。

当然，有些时候只通过各个线程中的方法调用栈来分析问题，可能信息还不太够，这时你还可以捕获各线程卡顿时的 CPU 使用率，进而发现哪个方法占用资源过高。同时，你还能够通过业务场景和环境数据埋点信息，综合分析发生卡顿时，业务场景以及数据是否出现了异常。

关于 SMLogger 的实现

@梁华建在第 9 篇文章《无侵入的埋点方案如何实现？》后留言，想要知道 SMLogger 是如何实现的。

SMLogger，是我对日志记录的一个封装。我在第 9 篇文章中使用 SMLogger 的方式，是这样的：

复制代码

```
[[[[SMLogger create]
   message:[NSString stringWithFormat:@"%@ Appear",NSStringFromClass([self class
  classify:ProjectClassifyOperation]
 save];

```

可以看出，我把 SMLogger 的接口设计成了链式调用的方式。这样的接口接收外部数据后，能够更加灵活地进行组合。

对于日志记录来说，可以设置默认的日志分类和日志级别，简单记录日志描述就只需要一个日志描述数据。

当使用者需要日志库记录一个对象时，就需要增加一个新的接口来支持记录对象。接下来，就会面对外部输入会进行不同组合的情况，比如日志记录对象、日志描述、日志分类、日志级别这四个数据的不同组合。为了满足这些不同的组合，你设置的接口数量也会增加很多。如果都放到一个统一接口中当作不同参数，那么参数的个数就会非常多，导致接口使用起来非常不方便。比如，你每次只需要设置日志描述这个参数，但是使用了多参数的统一接口后，需要手动去设置其他参数值。

使用链式调用的好处就是可以随意组合。而且，当有新的输入类型加入，要和以前接口组合时，也不需要额外工作。我定义的 SMLogger 的链式接口，如下所示：

复制代码

```
// 初始化
+ (SMLogger *)create;
// 可选设置
- (SMLogger *)object:(id)obj;                        //object 对象记录
- (SMLogger *)message:(NSString *)msg;               // 描述
- (SMLogger *)classify:(SMProjectClassify)classify;  // 分类
- (SMLogger *)level:(SMLoggerLevel)level;            // 级别
// 场景记录
- (SMLogger *)scene:(SceneType)scene;

// 最后需要执行这个方法进行保存，什么都不设置也会记录文件名，函数名，行数等信息
- (void)save;

```

可以看出，日志记录对象、日志描述、日志分类、日志级别分别为 object、message、classity、level。当需要在日志记录中增加业务场景数据时，只需要简单增加一个 scene 链式接口，就能够达到组合使用业务场景数据和其他链式接口的目的。

在 SMLogger 中，我还在链式基础上实现了宏的方式，来简化一些常用的日志记录接口调用方式。宏的定义如下：

复制代码

```
// 宏接口
FOUNDATION_EXPORT void SMLoggerDebugFunc(NSUInteger lineNumber, const char *func
// debug 方式打印日志，不会上报
#ifdef DEBUG
    #define SMLoggerDebug(frmt, ...) SMLoggerCustom(SMProjectClassifyNormal,SMLo
#else
    #define SMLoggerDebug(frmt, ...) do {} while (0)
#endif
// 简单的上报日志
#define SMLoggerSimple(classify,frmt, ...) SMLoggerCustom(classify,SMLoggerLevel
// 自定义 classify 和 level 的日志，可上报
#define SMLoggerCustom(classify,level,frmt, ...) \
do { SMLoggerDebugFunc(__LINE__,__FUNCTION__,classify,level,frmt, ##__VA_ARGS__)

```

可以看到，宏定义最终调用的是 SMLoggerDebugFunc 函数，这个函数的实现如下所示：

复制代码

```
void SMLoggerDebugFunc(NSUInteger lineNumber, const char *functionName, SMProjec
    va_list args;
    if (format) {
        va_start(args, format);
        // 输出方法名和行号
        NSString *msg = [[NSString alloc] initWithFormat:format arguments:args];
        msg = [NSString stringWithFormat:@"[%s:%lu]%@",functionName,(unsigned lo
        // SMLogger 链式调用
        [[[[[SMLogger create] message:msg] classify:classify] level:level] save]
        va_end(args);
    }
}

```

通过上面代码可以看到，SMLoggerDebugFunc 在处理完方法名和行号后，最终使用的就是 SMLogger 链式调用。

通过宏的定义，日志记录接口调用起来也会简化很多，使用效果如下：

复制代码

```
// 宏方式使用，会记录具体调用地方的函数名和行数
SMLoggerDebug(@" 此处必改：%@ 此处也必改： %@",arr,dict); // 仅调试，不上报
SMLoggerSimple(SMProjectClassifyNormal,@" 此处必改：%@ 此处也必改： %@",arr,dict);
SMLoggerCustom(SMProjectClassifyNormal,SMLoggerLevelDebug, @" 这两个需要上报 %@%@"

```

NSURLProtocol 相关

@熊在第 28 篇文章《怎么应对各种富文本表现需求？》后留言到：

> WKWebView 对 NSURLProtocol 的支持不太好，我在网上找到的方法都不适用，连 Ajax 请求都不好去拦截。

其实，WKWebView 处理资源缓存的思路和 UIWebView 类似，需要创建一个 WKURLSchemeHandler，然后使用 -[WKWebViewConfiguration setURLSchemeHandler:forURLScheme:] 方法注册到 WKWebView 配置里。

WKURLSchemeHandler 实例可以用来处理对应的 URLScheme 加载的资源，使用它的 webView:startURLSchemeTask 方法可以加载特定资源的数据。这样就能够起到和 NSURLProtocol 同样的效果。

关于 JSON 解析的问题

@大太阳在第 26 篇文章《如何提高 JSON 解析的性能？》中留言到：

> 我现在项目是用 Swift 语言开发的，绝大部分的 JSON 解析用的是 SwiftyJSON，很少一部分用到了 KVC。我想问下，SwiftyJSON 的效率怎么样？我怎么才能评测这个效率？市面上比较出名的第三方库，它们的效率排名是什么样的？

其实，市面上的大多数第三方库，在解析 JSON 时用的都是系统自带的 JSONSerialization。因此，从本质上来看，它们的解析效率并无差别，只是在易用性、容错率、缓存效率上有些许差异。

比如，@大太阳提到的 SwiftyJSON 库，初始化方法如下：

复制代码

```
public init(data: Data, options opt: JSONSerialization.ReadingOptions = []) thro
    let object: Any = try JSONSerialization.jsonObject(with: data, options: opt)
    self.init(jsonObject: object)
}

```

可以看到，SwiftyJSON 库在解析 JSON 时，使用的是 JSONSerialization。你可以点击这个链接，查看 SwiftJSON 的完整代码。

既然 SwiftyJSON 也是使用 JSONSerialization 来解析 JSON 的，那么解析效率就和其他使用 JSONSerialization 解析的第三方库相比，没有本质上的差别。

JSON 案例相关

@徐秀滨在第 23 篇文章《如何构造酷炫的物理效果和过场动画效果？》后留言反馈，对通过 JSON 来控制代码逻辑的能力这块内容，感觉理解起来有些困难。接下来，针对这个问题，我再多说两句，希望能够对你有多帮助。

我在第 26 篇文章《如何提高 JSON 解析的性能？》中，举了个更加具体的例子，使用 JSON 描述了一段 JavaScript 代码逻辑，你可以先看一下这篇文章的相关内容。

对于开发者来说，App 中的任何逻辑都可以通过代码来描述，而代码又能够转换成抽象语法树结构。JSON 作为一种数据结构的表示，同样可以表示代码的抽象语法树，自然也能够具有控制代码逻辑的能力。

总结

今天这篇答疑文章，我和你分享了监控卡顿、SMLogger、NSURLProtocol、JSON 相关的问题。

监控卡顿的方案实际上是通用的，和具体的场景没有关系。卡只是表现在主线程上，根本原因还是需要分析每个线程。

通过 NSURLProtocol 对 WKWebView 支持不好的问题，我们可以看出，苹果公司为了更好地管控 WKWebView 而增加了一层，将资源的加载处理单独提供出来供开发者使用，以满足开发者自定义提速的需求。

最后，JSON 解析效率的提高，还是需要从根本上去解决，封装层解决的是易用性问题，所加缓存也只能解决重复解析的问题。

感谢你的收听，欢迎你在评论区给我留言分享你的观点，也欢迎把它分享给更多的朋友一起阅读。

吴开

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。

Command + Enter 发表 0/2000字 提交留言

精选留言

> 由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。