06-App如何通过注入动态库的方式实现极速编译调试？

你好，我是戴铭。

在上一篇文章中，我和你分享了链接器的基础知识。今天我们再继续聊聊，动态库链接器的实际应用，也就是编译调试的提速问题。

iOS 原生代码的编译调试，都是通过一遍又一遍地编译重启 App 来进行的。所以，项目代码量越大，编译时间就越长。虽然我们可以通过将部分代码先编译成二进制集成到工程里，来避免每次都全量编译来加快编译速度，但即使这样，每次编译都还是需要重启 App，需要再走一遍调试流程。

对于开发者来说，提高编译调试的速度就是提高生产效率。试想一下，如果上线前一天突然发现了一个严重的bug，每次编译调试都要耗费几十分钟，结果这一天的黄金时间，一晃就过去了。到最后，可能就是上线时间被延误。这个责任可不轻啊。

那么问题来了，原生代码怎样才能够实现动态极速调试，以此来大幅提高编译调试速度呢？在回答这个问题之前，我们先看看有哪些工具是这么玩儿的。了解了它们的玩法，我们也就自然清楚这个问题的答案了。

Swift Playground

说到iOS代码动态极速调试的工具，你首先能想到的估计就是 Playground。它是 Xcode 里集成的一个能够快速、实时调试程序的工具，可以实现所见即所得的效果，如下图所示：

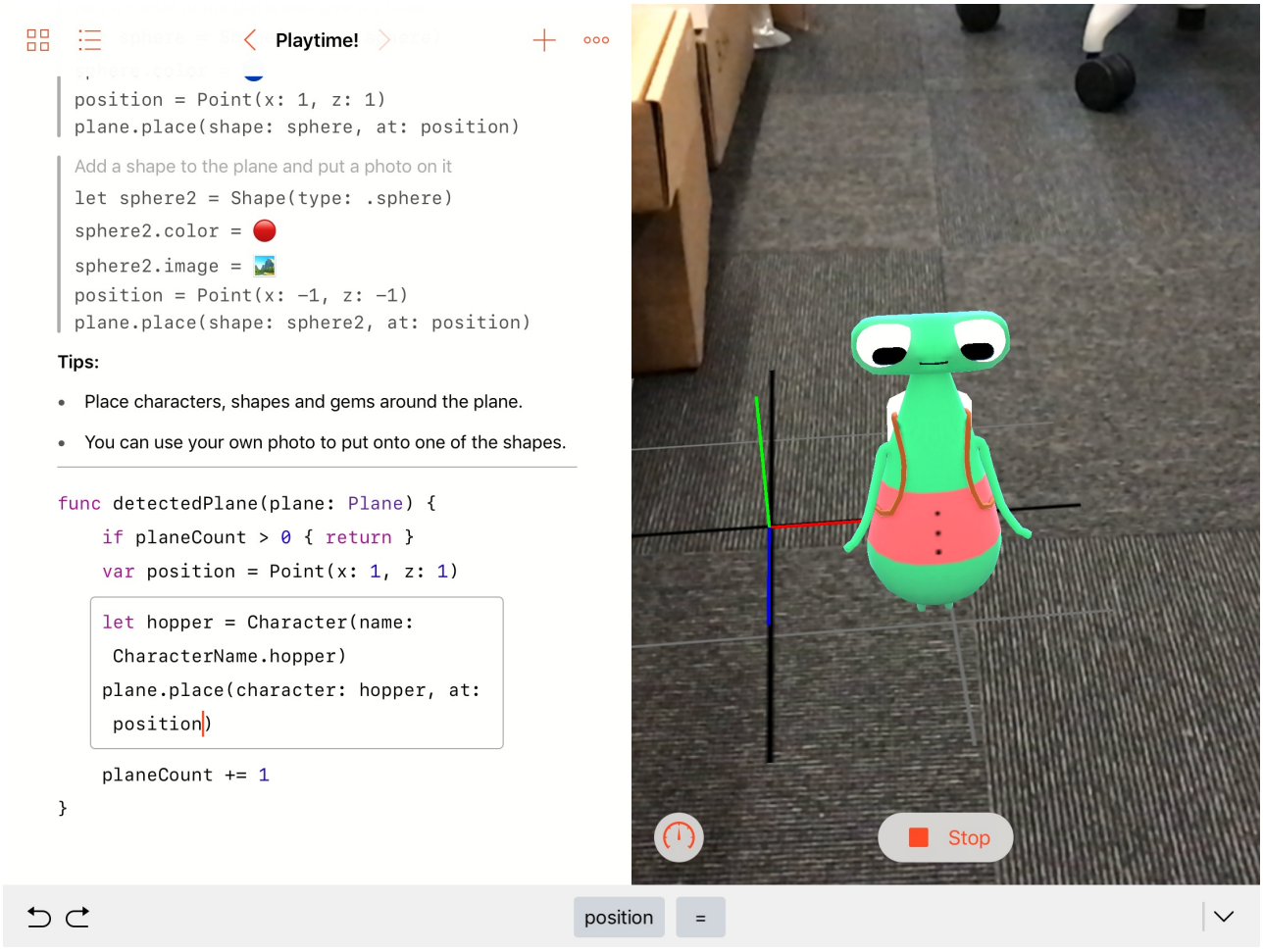

图1 Playground工具实时调试示例

可以看到，任何的代码修改都能够实时地在右侧反馈出来。

Flutter Hot Reload

Flutter 是 Google 开发的一个跨平台开发框架，调试也是快速实时的。官方的效果动画如下：

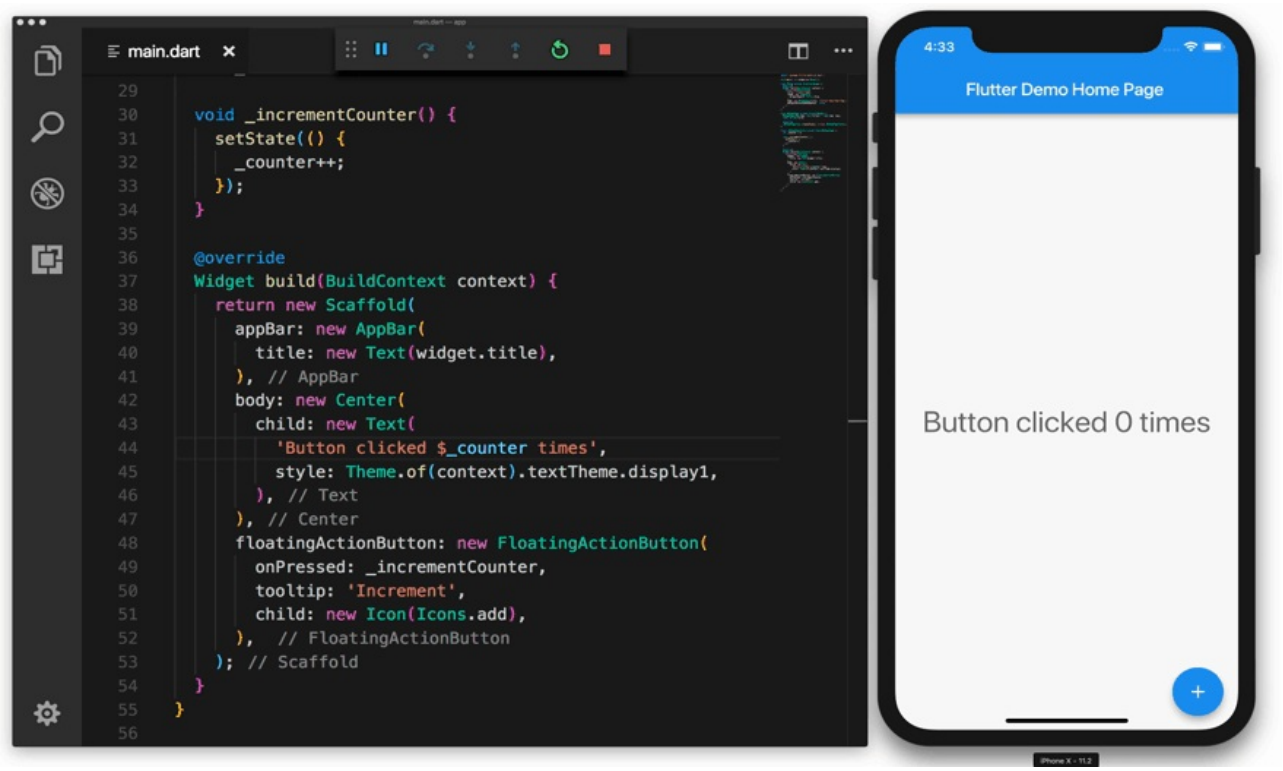

图2 Flutter使用示例

可以看到，在 Flutter 编辑器中修改文字 clicked 为 tapped 后点击 reload，模拟器中的文字立刻就改变了，程序没有重启。同样地，修改按钮图标也会立刻生效。

接下来，我们先看看 Flutter 是怎么实现实时编译的。

Flutter 会在点击 reload 时去查看自上次编译以后改动过的代码，重新编译涉及到的代码库，还包括主库，以及主库的相关联库。所有这些重新编译过的库都会转换成内核文件发到 Dart VM 里，Dart VM 会重新加载新的内核文件，加载后会让 Flutter framework 触发所有的Widgets 和 Render Objects 进行重建、重布局、重绘。

Flutter 为了能够支持跨平台开发，使用了自研的 Dart 语言配合在 App 内集成 Dart VM 的方式运行 Flutter 程序。目前 Flutter 还没有达到 Cocoa 框架那样的普及程度，所以如果你不是使用 Flutter 来开发 iOS 程序的话，想要达到极速调试应该要怎么做呢？

Injection for Xcode

所幸的是，John Holdsworth 开发了一个叫作 Injection 的工具可以动态地将 Swift 或 Objective-C 的代码在已运行的程序中执行，以加快调试速度，同时保证程序不用重启。John Holdsworth 也提供了动画演示效果，如下：

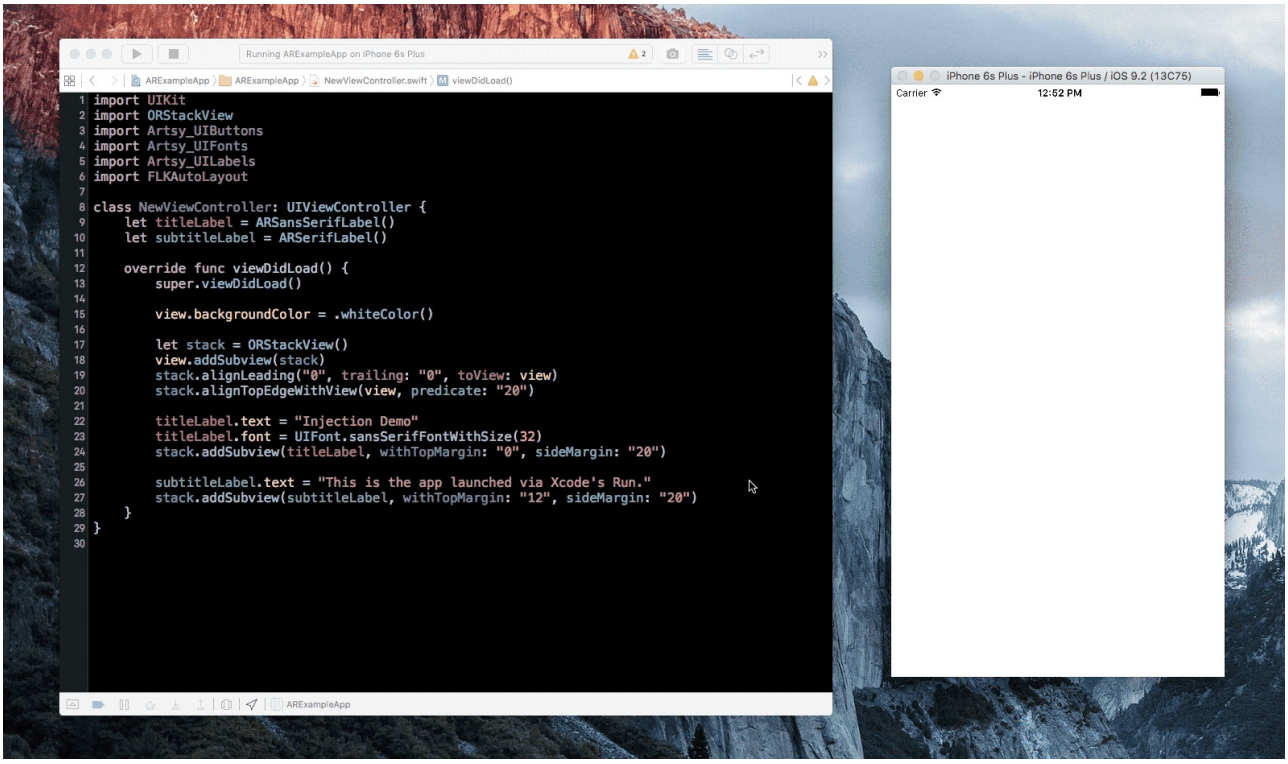

图3 Injection使用示例

作者已经开源了这个工具，地址是https://github.com/johnno1962/InjectionIII 。使用方式就是 clone 下代码，构建 InjectionPluginLite/InjectionPlugin.xcodeproj ；删除方式是，在终端里运行下面这行代码：

```
rm -rf ~/Library/Application\ Support/Developer/Shared/Xcode/Plug-ins/InjectionPlugin.xcplugin
```

构建完成后，我们就可以编译项目。这时添加一个新的方法：

```
- (void)injected
{
    NSLog(@"I've been injected: %@", self);
}
```

然后在这个方法中添加一个断点，按下 ctrl + = ，接下来你会发现程序运行时会停到断点处，这样你的代码就成功地被运行中的 App 执行了。那么，**Injection 是怎么做到的呢？**

Injection 会监听源代码文件的变化，如果文件被改动了，Injection Server 就会执行 rebuildClass 重新进行编译、打包成动态库，也就是 .dylib 文件。编译、打包成动态库后使用 writeSting 方法通过 Socket 通知运行的 App。writeString 的代码如下：

```
- (BOOL)writeString:(NSString *)string {
    const char *utf8 = string.UTF8String;
    uint32_t length = (uint32_t)strlen(utf8);
    if (write(clientSocket, &length, sizeof length) != sizeof length ||
```

```
        write(clientSocket, utf8, length) != length)
        return FALSE;
    return TRUE;
}
```

Server 会在后台发送和监听 Socket 消息，实现逻辑在 `InjectionServer.mm` 的 runInBackground 方法里。Client 也会开启一个后台去发送和监听 Socket 消息，实现逻辑在 `InjectionClient.mm`里的 runInBackground 方法里。

Client 接收到消息后会调用 inject(tmpfile: String) 方法，运行时进行类的动态替换。inject(tmpfile: String) 方法的具体实现代码，你可以点击这个链接查看。

inject(tmpfile: String) 方法的代码大部分都是做新类动态替换旧类。inject(tmpfile: String) 的入参 tmpfile 是动态库的文件路径，那么这个动态库是如何加载到可执行文件里的呢？具体的实现在inject(tmpfile: String) 方法开始里，如下：

```
let newClasses = try SwiftEval.instance.loadAndInject(tmpfile: tmpfile)
```

你先看下 SwiftEval.instance.loadAndInject(tmpfile: tmpfile) 这个方法的代码实现：

```
@objc func loadAndInject(tmpfile: String, oldClass: AnyClass? = nil) throws -> [AnyClass] {

    print("???? Loading .dylib - Ignore any duplicate class warning...")
    // load patched .dylib into process with new version of class
    guard let dl = dlopen("\(tmpfile).dylib", RTLD_NOW) else {
        throw evalError("dlopen() error: \(String(cString: dlerror()))")
    }
    print("???? Loaded .dylib - Ignore any duplicate class warning...")

    if oldClass != nil {
        // find patched version of class using symbol for existing

        var info = Dl_info()
        guard dladdr(unsafeBitCast(oldClass, to: UnsafeRawPointer.self), &info) != 0 else {
            throw evalError("Could not locate class symbol")
        }

        debug(String(cString: info.dli_sname))
        guard let newSymbol = dlsym(dl, info.dli_sname) else {
            throw evalError("Could not locate newly loaded class symbol")
        }

        return [unsafeBitCast(newSymbol, to: AnyClass.self)]
    }
    else {
        // grep out symbols for classes being injected from object file

        try injectGenerics(tmpfile: tmpfile, handle: dl)

        guard shell(command: """
            \(xcodeDev)/Toolchains/XcodeDefault.xctoolchain/usr/bin/nm \(tmpfile).o | grep -E ' S _OBJC_CLA
```

```
            """) else {
            throw evalError("Could not list class symbols")
        }
        guard var symbols = (try? String(contentsOfFile: "\(tmpfile).classes"))?.components(separatedBy: "\
            throw evalError("Could not load class symbol list")
        }
        symbols.removeLast()

        return Set(symbols.flatMap { dlsym(dl, String($0.dropFirst())) }).map { unsafeBitCast($0, to: AnyCl
```

在这段代码中，你是不是看到你所熟悉的动态库加载函数 dlopen 了呢？

```
guard let dl = dlopen("\(tmpfile).dylib", RTLD_NOW) else {
    throw evalError("dlopen() error: \(String(cString: dlerror()))")
}
```

如上代码所示，dlopen 会把 tmpfile 动态库文件载入运行的 App 里，返回指针 dl。接下来，dlsym 会得到 tmpfile 动态库的符号地址，然后就可以处理类的替换工作了。dlsym 调用对应代码如下：

```
guard let newSymbol = dlsym(dl, info.dli_sname) else {
    throw evalError("Could not locate newly loaded class symbol")
}
```

当类的方法都被替换后，我们就可以开始重新绘制界面了。整个过程无需重新编译和重启 App，至此使用动态库方式极速调试的目的就达成了。

我把Injection的工作原理用一张图表示了出来，如下所示：

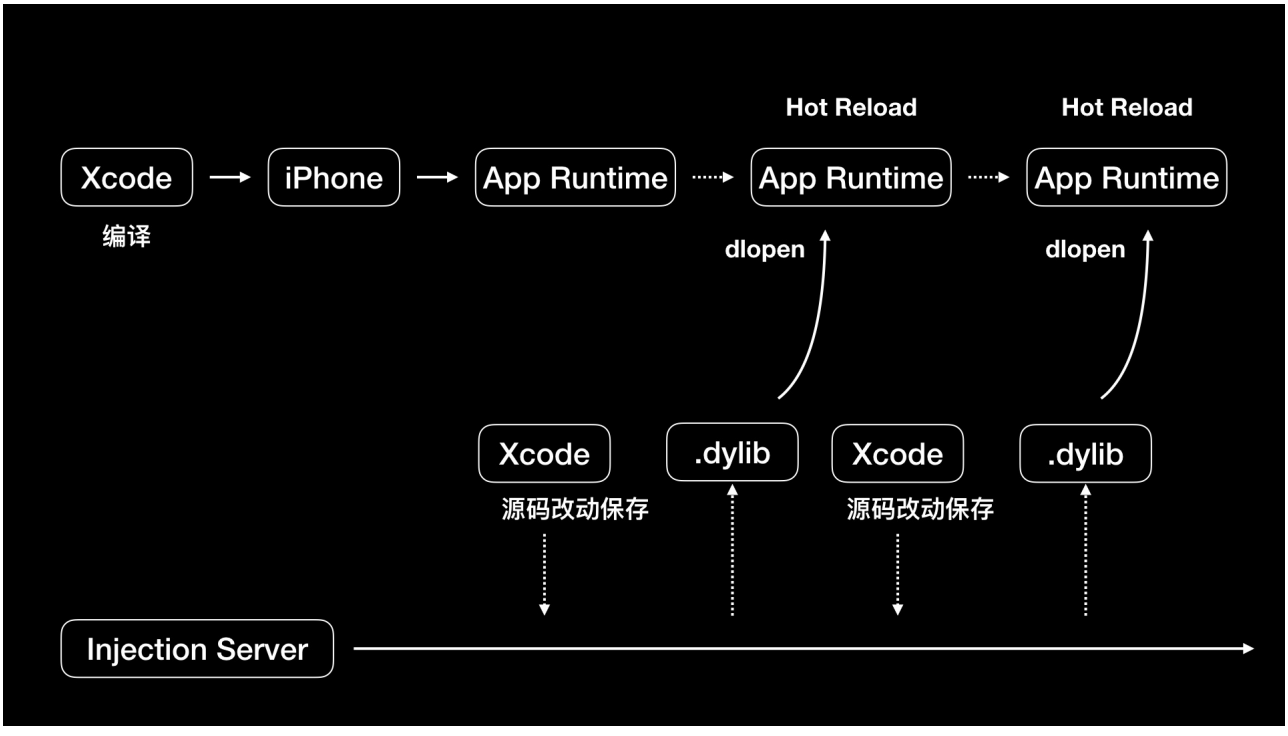

图4 Injection的工作原理示意图

小结

今天这篇文章，我和你详细分享了动态库链接器的一个非常实用的应用场景：如何使用动态库加载方式进行极速调试。由此我们可以看出，类似链接器这样的底层知识是非常重要的。

当然了，这只是一个场景，还有更多的场景等待着我们去发掘。比如把 Injection 技术扩展开想，每当你修改了另一个人负责的代码就给那个人发条消息，同时将修改的代码编译、打包成动态库直接让对方看到修改的情况，这样不仅是提高了自己的效率，还提高了整个团队的沟通效率。怎么样？是不是有种想立刻尝试的感觉，心动不如行动，动手写起来吧。

所以，打好了底层知识的基础以后，我们才可以利用它们去提高开发效率，为用户提供更稳定、性能更好的 App 。

今天这篇文章最后，我留给你的一个小作业是，思考一下底层知识还有哪些运用场景，并在评论区分享出来吧。

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。

精选留言：

- drunkenMouse 2019-03-23 14:01:32
  Clone一下代码弄得有点懵，不知道怎么克隆。然后我用了另一种方式使用了injection

  1.在App Store下载InjectionIII, 打开。
  2.选择项目的根目录
  3.项目的Appdelegate加入：
  #if DEBUG
  // iOS
  [[NSBundle bundleWithPath:@"/Applications/InjectionIII.app/Contents/Resources/iOSInjection.bundle"] load];
  #endif

XCode10 是这个
#if DEBUG
// iOS
[[NSBundle bundleWithPath:@"/Applications/InjectionIII.app/Contents/Resources/iOSInjection10.bundle"] load];
#endif

而后启动，修改，保存，就会卡到断点位置了。 [13赞]

- mɑnajay 2019-03-23 14:52:48
  InjectionIII 上面有个 issue 是解决 pod 组件引入修改源码无法进行注入的问题 https://github.com/johnno1962/InjectionIII/issues/34, https://github.com/johnno1962/InjectionIII/issues/53 , 使用组件后接入还是有点麻烦 [6赞]

- Love mi 2019-03-28 11:28:03
  1.在App Store下载InjectionIII, 打开。
  2.选择项目的根目录
  3.项目的Appdelegate加入：
  [[NSBundle bundleWithPath:@"/Applications/InjectionIII.app/Contents/Resources/iOSInjection.bundle"] load];
  运行报
  Error loading /Applications/InjectionIII.app/Contents/Resources/iOSInjection.bundle/iOSInjection: dlopen(/Applications/InjectionIII.app/Contents/Resources/iOSInjection.bundle/iOSInjection, 265): Symbol not found: _$s19ArrayLiteralElements013ExpressibleByaB0PTl
  Referenced from: /Applications/InjectionIII.app/Contents/Resources/iOSInjection.bundle/iOSInjection
  Expected in: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/lib/swift/iphonesimulator/libswiftCore.dylib
  in /Applications/InjectionIII.app/Contents/Resources/iOSInjection.bundle/iOSInjection [4赞]

- Hyman 2019-03-28 10:11:21
  command+s就没有人报错误吗？
  Loading .dylib ...
  objc[4708]: Class ViewController is implemented in both /Users/hyman/Library/Developer/CoreSimulator/Devices/7C19EFEC-F0B4-4405-979A-358FA75F4A4F/data/Containers/Bundle/Application/A1A3313C-9092-4BBA-B2EF-604ECAD508D3/MTInjectionOC.app/MTInjectionOC (0x10f488d10) and /Users/hyman/Library/Containers/com.johnholdsworth.InjectionIII/Data/eval104.dylib (0x12d168138). One of the two will be used. Which one is undefined.
  🔪 Loaded .dylib - Ignore any duplicate class warning ^ [4赞]

- 景迪 2019-03-23 05:20:36
  起来第一件事就是看看有没有更新😊 [4赞]

- SLY 2019-03-25 15:05:54
  @Link 回复楼上那位同学，把 file -> Workspace Setting -> Build System， 改为Legacy Build System模式，默认的New Build System(Default)模式，是不会编译pod 里面的改动的 [3赞]

- 亡命之徒 2019-03-29 14:13:10
  项目使用了cocopods、这个插件用不了呀，保存的时候报错😨，新建的demo使用就没问题,麻烦老师解答下 [2赞]

- Melvins 2019-03-28 10:37:25
  @drunkenMouse，用你的方法在xcode 10.2上会出现👇错误提示，求教！！！

  Error loading /Applications/InjectionIII.app/Contents/Resources/iOSInjection10.bundle/iOSInjection10: dlopen(/Applications/InjectionIII.app/Contents/Resources/iOSInjection10.bundle/iOSInjection10, 265): Symbol not found: _$SBOWV
  Referenced from: /Applications/InjectionIII.app/Contents/Resources/iOSInjection10.bundle/iOSInjection10
  Expected in: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/lib/swift/iphonesimulator/libswiftCore.dylib
  in /Applications/InjectionIII.app/Contents/Resources/iOSInjection10.bundle/iOSInjection10 [2赞]

- mqhong 2019-03-23 11:32:20
  讲的有点深了 有没有推介的gnustep源码必读的类实现 和阅读的方法😁 [2赞]

  作者回复2019-03-23 13:18:04
  我觉得你在使用库有疑问时，或者看到分析有疑点时，带着问题去看，效果会很好。

- 21zz 2019-04-09 12:00:07
  - (void)injected
  + (void)injected
  INJECTION_BUNDLE_NOTIFICATION
  我这边所有的修改只有写在这3个方法里才会刷新，并不能像图片上那样随修随变，并一直报dylib 重复警告

  inject(tmpfile: String) 方法的代码大都是新类替换旧类，但是当我在- (void)injected里写入[self viewDidLoad];后，会发现并没有替换，而是重新创建了一遍新的UI,有请踩过坑的老哥发言😁

  [1赞]

- 以 2019-03-28 18:06:32
  @Hyman 出现的问题怎么解决的？
  🔪 Loading .dylib ...
  objc[10390]: Class ViewController is implemented in both /Users/lutaohua/Library/Developer/CoreSimulator/Devices/37453FFB-A4BA-4761-B590-BFAEB3C54A5D/data/Containers/Bundle/Application/342F3B0F-D638-4A99-89C6-3C8A04A25135/Mach_O_test.app/Mach_O_test (0x100b65ea8) and /Users/lutaohua/Library/Containers/com.johnholdsworth.InjectionIII/Data/eval101.dylib (0x11e8e3218). One of the two will be used. Which one is undefined.
  🔪 Loaded .dylib - Ignore any duplicate class warning ^ [1赞]

- Tim叔 2019-03-27 16:54:08
  xocde10.1下编译错误open /~/bin/unhide no such file or [1赞]

- 徐秀滨 2019-03-26 16:55:10
  请问为啥我保存后，断点执行到了，但是过了之后就闪了，一脸懵逼。。。 [1赞]

- bubble 2019-03-25 11:40:21
  想问下老师 为什么只有模拟器可以 而真机不可以呢？ [1赞]

作者回复2019-03-25 13:23:27
针对 arm64 芯片写的

- Chouee 2019-03-24 22:50:27
之前总抱怨Xcode编译太慢。。。原来是自己的见识太少 [1赞]

- 政 2019-03-24 15:09:03
这个工具去年下来试过，但不知道为啥有时候可以，有时候不行 [1赞]

- invisible-single 2019-03-23 08:14:59
沙发没抢到，有个小板凳也好 [1赞]

- 菜鸟 2019-04-17 15:27:55
您好，老师，xcode版本是10.2最新的。InjectionIII 是1.5 ，更新日志上已经支持10.2
在项目的Appdelegate加入： [[NSBundle bundleWithPath:@"/Applications/InjectionIII.app/Contents/Resources/iOSInjection.bundle"] load];，然后- (void)injected 方法使用时，会直接崩溃掉。报错：Fatal error: 'try!' expression unexpectedly raised an error: Error Domain=NSCocoaErrorDomain Code=260 "The file “eval101.sh” couldn’t be opened because there is no such file."

如果改成 [[NSBundle bundleWithPath:@"/Applications/InjectionIII.app/Contents/Resources/iOSInjection10.bundle"] load]; ，就报dlopen(/Applications/InjectionIII.app/Contents/Resources/iOSInjection10.bundle/iOSInjection10, 265): Symbol not found: _$SBOWV

- 爱神 2019-04-17 15:22:36
Error loading /Applications/InjectionIII.app/Contents/Resources/iOSInjection.bundle/iOSInjection: dlopen(/Applications/InjectionIII.app/Contents/Resources/iOSInjection.bundle/iOSInjection, 265): Symbol not found:
github下载的 app 报错， 从 AppStore 安装没问题。

- Code2Morning 2019-04-14 21:36:35
1.在App Store下载InjectionIII, 打开。
2.选择项目的根目录
3.项目的Appdelegate加入：
[[NSBundle bundleWithPath:@"/Applications/InjectionIII.app/Contents/Resources/iOSInjection.bundle"] load];
运行报错
Error loading /Applications/InjectionIII.app/Contents/Resources/iOSInjection.bundle/iOSInjection: dlopen(/Applications/InjectionIII.app/Contents/Resources/iOSInjection.bundle/iOSInjection, 265): Symbol not found: _$s19ArrayLiteralElements013ExpressibleByaB0PTl
Referenced from: /Applications/InjectionIII.app/Contents/Resources/iOSInjection.bundle/iOSInjection
Expected in: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/lib/swift/iphonesimulator/libswiftCore.dylib
in /Applications/InjectionIII.app/Contents/Resources/iOSInjection.bundle/iOSInjection