# hangge.com

当前位置: 首页 > 编程社区 > Swift > Swift - Framework的制作与使用教程1 (纯Swift实现)

### Swift - Framework的制作与使用教程1 (纯Swift实现)

发布: hangge 阅读: 277 2016-11-09 09:00

在开发中我们常常会用到一些第三方 **SDK** 库,使用时只需将 **framework** 文件添加到项目中即可,十分方便。同样地,我们也可以创建自己的 **framework** 框架,用来封装一些常用的工具方法、框架类等。一来不会使源代码完全暴露在外,二来也便于代码复用。

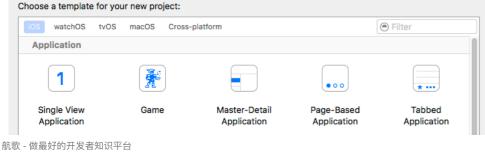
下面演示如何制作一个自定义的图片处理框架,用来实现 Ullmage 的高斯模糊与马塞克化。效果图如下:

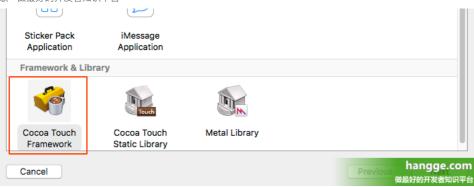


## 一、framework的制作(使用纯Swift)

#### 1, 创建framework工程项目

(1) 新建项目的时候选择"Cocoa Touch Framework"。





(2) 项目名就叫做"HanggeSDK"。

#### 文章类别

■ 首页

■ 编程社区

Swift Flex / Flash
HTML5 / CSS3 Cordova
React / RN 其他

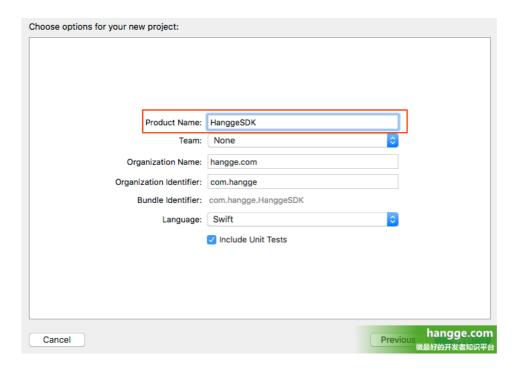
swift 枚举 (多关键字搜索更精确)

■ 奇文共赏



#### 本类相关

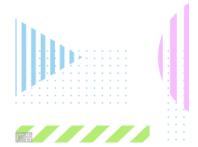
Swift - 导航条(UINavigationBar)的使用
Swift - MJRefresh库的使用详解6(WebView上实
Swift - 真机调试正常,打包成IPA安装后一启动就
Swift - 多行文本输入框(UITextView)的用法
Swift - iOS中各种视图控制器(View Controller))
Swift - 获取日历中所有的事件并显示在表格中(特
Swift - 常用文件目录路径获取(Home目录,文档
Swift - 使用SSZipArchive实现文件的压缩、解压组
Swift - JPush极光推送的使用5(发送通知时附带)
首页 留言





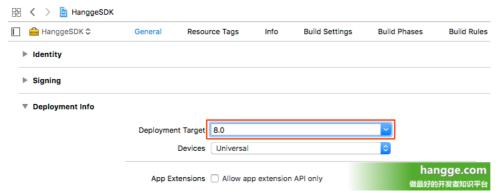
## 短信低至 0.036元/条

免费测试 按需付费



(3) 为了让制作出的 framework 在低版本的系统上也能使用,可以在"General"->"Deployment Info"里设置个较低的发布版本





(4) 创建一个功能实现类(ImageProcessor.swift),代码如下。

```
1
     import Foundation
 2
     public class ImageProcessor {
 4
        //保存原始图片
 5
         var image:UIImage?
 6
         lazy var context: CIContext = {
 7
 8
            return CIContext(options: nil)
 9
         }()
10
11
         //初始化
         public init(image:UIImage?) {
13
            self.image = image
14
15
         //返回像素化后的图片
16
         public func pixellated(scale:Int = 30) -> UIImage? {
18
            if image == nil {
19
                return nil
20
21
            //使用像素化滤镜
            let filter = CIFilter(name: "CIPixellate")!
22
23
            let inputImage = CIImage(image: image!)
24
            filter.setValue(inputImage, forKey: kCIInputImageKey)
            //设置像素版半径,值越大马赛克就越大
25
26
            filter.setValue(scale, forKey: kCIInputScaleKey)
27
            let fullPixellatedImage = filter.outputImage
28
```

```
29
            let cgImage = context.createCGImage(fullPixellatedImage!,
30
                                                from: fullPixellatedImage!.extent)
31
             return UIImage(cgImage: cgImage!)
32
         }
33
         //返回高斯模糊后的图片
34
35
         public func blured(radius:Int = 40) -> UIImage? {
36
            if image == nil {
37
                return nil
38
            //使用高斯模糊滤镜
39
40
             let filter = CIFilter(name: "CIGaussianBlur")!
41
            let inputImage = CIImage(image: image!)
42
             filter.setValue(inputImage, forKey: kCIInputImageKey)
43
             //设置模糊半径值(越大越模糊)
             filter.setValue(radius, forKey: kCIInputRadiusKey)
44
45
            let outputCIImage = filter.outputImage
46
             let rect = CGRect(origin: CGPoint.zero, size: image!.size)
47
             let cgImage = context.createCGImage(outputCIImage!, from: rect)
48
             return UIImage(cgImage: cgImage!)
49
        }
50
    }
```

注意:对于那些需要暴露出来,即在框架外部也能访问使用的类、方便、变量前面需要加上关键字 Public。如果还允许 overrid e 和继承的话,可以使用 open 关键字。(关于访问控制的详细说明,可以参考我之前的这篇文章: Swift - 访问控制(fileprivat e, private, internal, public, open))

#### 2, 生成framework库文件

生成的 framework 文件是分为模拟器使用和真机使用这两种。

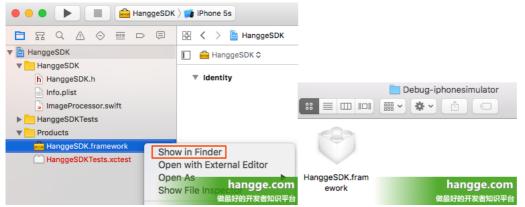
(1) 发布编译目标选择"Generic iOS Device"后,使用快捷键 command+B 或者点击菜单 Product > Build 进行编译。这时生成的是真机调试使用的 framework。



(2) 如果发布编译目标选择的是模拟器,那么编译出来的模拟器使用的 framework。



(3) 编译后右键点击项目中生成的 framework,选择"Show in Finder",即可打开 framework 所在的文件夹。



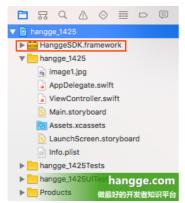
(4) 访问上级文件夹,可以看到两种类型的 framework 分别放在两个不同的文件夹下。



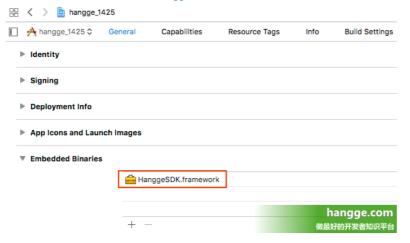
## 二、framework的使用

#### 1, 引入framework

(1) 将生成的 HanggeSDK.framework 添加到项目中来。(注意:要根据你是使用真机调试还是模拟器调试选择对应的 framework)



(2) 接着在"General"->"Embedded Binaries"中把 HanggeSDK.framework 添加进来。



#### 2, 使用样例

```
import UIKit
 1
     import HanggeSDK
 3
 4
     class ViewController: UIViewController {
 5
 6
         @IBOutlet weak var imageView: UIImageView!
 7
 8
         var defaultImage = UIImage(named: "image1.jpg")
 9
10
         override func viewDidLoad() {
11
12
             super.viewDidLoad()
13
14
         //显示原图按钮点击
15
         @IBAction func btn1Click(_ sender: AnyObject) {
16
```

```
17
             imageView.image = defaultImage
18
19
         //显示像素化图片按钮点击
20
21
         @IBAction func btn2Click(_ sender: AnyObject) {
             imageView.image = ImageProcessor(image: defaultImage).pixellated()
23
24
         //显示高斯模糊图片按钮点击
25
         @IBAction func btn3Click(_ sender: AnyObject) {
26
             imageView.image = ImageProcessor(image: defaultImage).blured()
28
29
30
         override func didReceiveMemoryWarning() {
31
             super.didReceiveMemoryWarning()
32
33
```

#### 三、功能改进

上面样例中我们自定义库中的图像工具类(ImageProcessor)是初始化的时候传入一个 Ullmage,然后调用方法返回处理后的 图片。我们也可以换种方式实现,改成扩展 Ullmage 类,在其之上添加两个新的处理方法。

#### 1, ImageProcessor.swift代码

```
1
     import Foundation
 2
     extension UIImage {
 3
 4
 5
         //返回像素化后的图片
 6
         public func pixellated(scale:Int = 30) -> UIImage {
            //使用像素化滤镜
            let filter = CIFilter(name: "CIPixellate")!
 8
 9
            let inputImage = CIImage(image: self)
10
            filter.setValue(inputImage, forKey: kCIInputImageKey)
            //设置像素版半径,值越大马赛克就越大
11
12
             filter.setValue(scale, forKey: kCIInputScaleKey)
            let fullPixellatedImage = filter.outputImage
13
14
            let cgImage = CIContext().createCGImage(fullPixellatedImage!,
15
                                               from: fullPixellatedImage!.extent)
            return UIImage(cgImage: cgImage!)
16
17
18
         //返回高斯模糊后的图片
19
20
         public func blured(radius:Int = 40) -> UIImage {
            //使用高斯模糊滤镜
21
22
            let filter = CIFilter(name: "CIGaussianBlur")!
            let inputImage = CIImage(image: self)
23
24
            filter.setValue(inputImage, forKey: kCIInputImageKey)
            //设置模糊半径值(越大越模糊)
25
26
            filter.setValue(radius, forKey: kCIInputRadiusKey)
27
            let outputCIImage = filter.outputImage
            let rect = CGRect(origin: CGPoint.zero, size: self.size)
28
29
            let cgImage = CIContext().createCGImage(outputCIImage!, from: rect)
30
            return UIImage(cgImage: cgImage!)
31
        }
32
    }
```

#### 2,使用样例

```
import UIKit
import HanggeSDK

class ViewController: UIViewController {
    @IBOutlet weak var imageView: UIImageView!
}
```

```
//原图
 8
 9
         var defaultImage = UIImage(named: "image1.jpg")
10
11
         override func viewDidLoad() {
12
            super.viewDidLoad()
13
14
15
         //显示原图按钮点击
16
         @IBAction func btn1Click(_ sender: AnyObject) {
17
            imageView.image = defaultImage
18
19
         //显示像素化图片按钮点击
20
         @IBAction func btn2Click(_ sender: AnyObject) {
21
22
            imageView.image = defaultImage?.pixellated()
23
24
         //显示高斯模糊图片按钮点击
25
         @IBAction func btn3Click(_ sender: AnyObject) {
26
27
            imageView.image = defaultImage?.blured()
28
29
30
         override func didReceiveMemoryWarning() {
31
            super.didReceiveMemoryWarning()
32
33
    }
```

上一篇 Swift - 实现图片的模糊效果 (高斯模糊滤镜)

下一篇 Swift - Framework的制作与使用教程2(引用第三方库)







评论(1)



chen.Jie

2016-11-10 15:35

1楼 楼主 能写个二维码的扫描 及扫描的处理结果的案列吗 跪求啊

站长回复: 二维码扫描我原来有写过: Swift - 二维码QRCode的读取(从图片读取,或通过摄像头扫描)

现在有 192 位访客在线

移动版 给我留言 

友情链接(申请)

GTmetrix

© copyright 2006-2016 航歌. All rights reserved. 苏ICP备14040777号