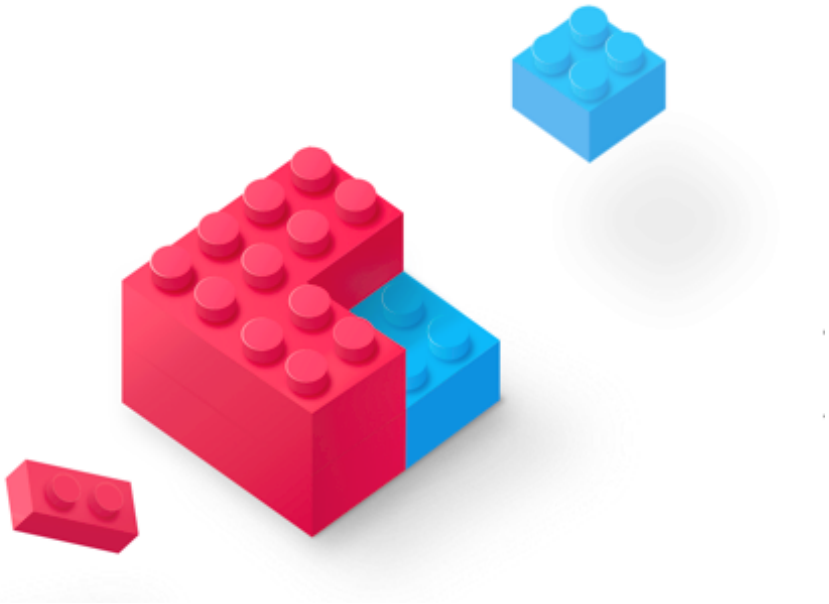
iOS 模块化之 JLRoute 路由示例

基于 JLRoute 实现的模块化示例，包括链接跳转原生页面、WebView页面和ReactNative页面

> 模块化已经成为调剂庞大项目结构的一剂良药，对项目的开发、维护和后续的扩展的好处已经不言而喻。



要求

- iOS 8.0+
- Xcode 7.0+

安装方法

安装

在 *iOS*, 你需要在 Podfile 中添加.

```
source 'https://github.com/CocoaPods/Specs.git'
platform :ios, '9.0'
use_frameworks!

pod 'JLRoutes', '~> 2.0.1'

# 'node_modules'目录一般位于根目录中
# 但是如果你的结构不同，那你就要根据实际路径修改下面的`:path`
pod 'React', :path => './node_modules/react-native', :subspecs => [
    'Core',
    'RCTText',
    'RCTNetwork',
    'RCTWebSocket', # 这个模块是用于调试功能的
    # 在这里继续添加你所需要的模块
]
# 如果你的RN版本 >= 0.42.0，请加入下面这行
pod "Yoga", :path => "./node_modules/react-native/ReactCommon/yoga"
```

启动 ReactNative 环境

1.修改项目ModuleARNPageViewController.m IP 跳转地址

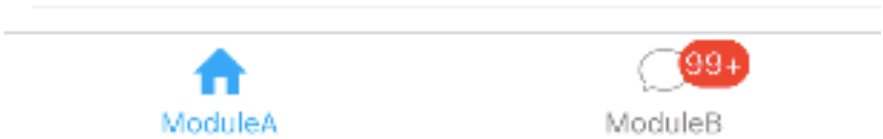
```
- (void)viewDidLoad {
    [super viewDidLoad];

    [self.view setBackgroundColor:[UIColor whiteColor]];
    NSURL *jsCodeLocation;
    jsCodeLocation = [NSURL URLWithString:@"http:  .0.65:8081/index.ios.bundle?platform=ios&dev=true"];
    RCTRootView *rootView = [[RCTRootView alloc] initWithBundleURL:jsCodeLocation
                                                        moduleName:@"SimpleApp"
                                                 initialProperties:nil
                                                     launchOptions:nil];
```

2.进入项目所在目录，运行（新项目需要先 npm install）

```
npm start
```

??JLRoutes的工作流程和原理

单一的Scheme注册过程：



1.调用注册方法（用户注册routePattern，默认优先级0）

```
- (void)addRoute:(NSString *)routePattern handler:(BOOL (^__nullable)(NSDictionary<NSString *, id
```

2.路由解析（这些解析跟我们设置路由的规则有直接关系）

(1)判断接口URL是否设置可选性URL并将对应的URL封装成JLRRouteDefinition对象

(2)将JLRRouteDefinition对象装载进一个可变数组，内存保留了所有的对象！！

（JLRRouteDefinition对象包括有路径，参数解析，block等信息）

单一的Scheme调用过程：

1.调用URL

```
+ (BOOL)routeURL:(NSURL *)URL
```

2.解析URL，将参数，路由信息封装成JLRRouteRequest对象

```
- (instancetype)initWithURL:(NSURL *)URL alwaysTreatsHostAsPathComponent:(BOOL)alwaysTreatsHostAs
```

3.给JLrouteRequest对象和路由数组里的JLRRouteDefinition对象作比对，并且返回JLRRouteResponse 对象抽出参数和URL在数组里

```
JLRRouteResponse *response = [route routeResponseForRequest:request decodePlusSymbols:shouldDecod
```

4.调用JLRRouteResponse 对象里面的回调方法

```
[route callHandlerBlockWithParameters:finalParameters];
```

JLRoutes的URL注册规则：

1.普通注册

```
JLRoutes *routes = [JLRoutes globalRoutes];
[routes addRoute:@"/user/view/:userID" handler:^BOOL(NSDictionary *parameters) {
```

```
NSString *userID = parameters[@"userID"]; // defined in the route by specifying
":userID"
// present UI for viewing user with ID 'userID'
return YES; // return YES to say we have handled the route
}];
```

URL里，分号表示这个是参数

另外一种注册方式，下标注册法

```
JLRoutes.globalRoutes[@"/route/:param"] = ^BOOL(NSDictionary *parameters) {
// ...
};
```

如何按照以上的方式注册，在任何时刻（包括在其它的APP）你都可以调用这个URL。

```
NSURL *viewUserURL = [NSURL URLWithString:@"myapp://user/view/joeldev"];
[[UIApplication sharedApplication] openURL:viewUserURL];
```

在这个例子中，在parmameters字典里面的userID会传给block，它是一个键值对。”userID”:“joeldev”。给UI层或者任何需要它的地方用的。

字典参数：

字典参数总包括至少一下3个键：

```
{
"JLRouteURL":  "(the NSURL that caused this block to be fired)",
"JLRoutePattern": "(the actual route pattern string)",
"JLRouteScheme": "(the route scheme, defaults to JLRoutesGlobalRoutesScheme)"
}
```

处理Block

你会发现，每个注册的block都会返回一个YES。这个值，如果你返回NO，JLRoutes会跳过这个匹配，然后继续去匹配其它的。

如果你的block设置成nil，它会默认返回YES。

2.复杂注册

```
[[JLRoutes globalRoutes] addRoute:@"/:object/:action/:primaryKey" handler:^BOOL(NS
Dictionary *parameters) {
NSString *object = parameters[@"object"];
NSString *action = parameters[@"action"];
NSString *primaryKey = parameters[@"primaryKey"];
// stuff
return YES;
}];
```

这个地址会被匹配很多URL，如/user/view/joeldev or /post/edit/123。这些URL上的是参数。

```
NSURL *editPost = [NSURL URLWithString:@"myapp://post/edit/123?
debug=true&foo=bar"];
[[UIApplication sharedApplication] openURL:editPost];
```

这时，pramater字典就会是以下这样的（传参）

```
{
"object": "post",
"action": "edit",
"primaryKey": "123",
"debug": "true",
"foo": "bar",
"JLRouteURL": "myapp://post/edit/123?debug=true&foo=bar",
"JLRoutePattern": "/:object/:action/:primaryKey",
"JLRouteScheme": "JLRoutesGlobalRoutesScheme"
}
```

3.Scheme（有没有多态的感觉）

JLRoutes支持用指定的URL scheme来创建路由。相同的scheme才能被匹配。默认地，所有的URL会设置进global scheme。

```
[[JLRoutes globalRoutes] addRoute:@"/
foo" handler:^BOOL(NSDictionary *parameters) {
// This block is called if the scheme is not 'thing' or 'stuff' (see below)
return YES;
}];
[[JLRoutes routesForScheme:@"thing"] addRoute:@"/
foo" handler:^BOOL(NSDictionary *parameters) {
// This block is called for thing://foo
return YES;
}];
[[JLRoutes routesForScheme:@"stuff"] addRoute:@"/
foo" handler:^BOOL(NSDictionary *parameters) {
// This block is called for stuff://foo
return YES;
}];
```

如果你调用的使用，是这样调用的

```
[[JLRoutes globalRoutes] addRoute:@"/
global" handler:^BOOL(NSDictionary *parameters) {
return YES;
}];
```

它只会调用global scheme的对应的URL。不会调用ting scheme里面对应的URL。

当然，你可以设置，如果指定的scheme没有这个URL，去查询global scheme 有没有。你需要设置一个属性。

```
[JLRoutes routesForScheme:@"thing"].shouldFallbackToGlobalRoutes = YES;
```

3.通配符的设置URL的方式

通配符为：*

通配符符后面所有的URL上的参数都会以一个数组保存在parameters字典里面的JLRouteWildcardComponentsKey对应的value里。

例如，如果你注册URL如下:

```
[[JLRoutes globalRoutes] addRoute:@"/wildcard/
*" handler:^BOOL(NSDictionary *parameters) {
NSArray *pathComponents = parameters[JLRouteWildcardComponentsKey];
if ([pathComponents count] > 0 && [pathComponents[0] isEqualToString:@"joker"]) {
// the route matched; do stuff
return YES;
}
// not interested unless the joker's in it
return NO;
}];
```

如果调用的URL开始是 / wildcard，这个路由就可能被触发！！如果第一个参数是joker，就被触发，如果不是，就被拒绝触发。。。

4.选择性路由

如果路由地址设置样式有括号，如：/the(/foo/:a)(/bar/:b)，其实它代表的URL有如下：

```
/the/foo/:a/bar/:b
/the/foo/:a
/the/bar/:b
/the
```

5.查询 Routes

下面的方式，你可以查看Routes里所有注册的URL Routes。

```
/// All registered routes, keyed by scheme
+ (NSDictionary <NSString *, NSArray <JLRRouteDefinition *> *> *)allRoutes;
/// Return all registered routes in the receiving scheme namespace.
- (NSArray <JLRRouteDefinition *> *)routes;
```

自定义路由解析 如果你想自己定制一个路由编辑，你可以继承JLRouteDefinition并且用 addRoute：方法去添加你自定义类的对象。

调研来源

JLRoutes ： https://github.com/joeldev/JLRoutes

JLRoutes 资料博客：https://www.varsiri.com/archives/305

联系

- 微信：WhatsXie
- 邮件：ReverseScale@iCloud.com
- 博客：https://reversescale.github.io

- 源码：https://github.com/ReverseScale/JLRouteDemo