WKWebview 和缓存清理

1. 概述：

项目中经常会有显示网页的需求，我想，很容易想到的就是 UIWebView 了。但是，在使用 UIWebView 的时候，有没有发现什么问题？首先，使用它很容易造成内存泄漏；其次，它加载速度慢；再者，它优化困难；最致命的一点：如果加载网页多，还可能因为过量占用内存而给系统 kill 掉。

Ios8 后，苹果针对 UIWebView 的缺陷，推出了一个新的网页加载库 Webkit，提供了替换 UIWebView 的组件 WKWebView。相比 UIWebView，在性能、稳定性、功能方面都有很大提升。如果之前考虑到 ios7 的支持，那么现在大可放心使用了，因为现在市场上的 app 已经普遍默认只支持 ios8 以上了。

2. WKWebview 基础

(1) 加载网页

加载网页或 HTML 代码的方式与 UIWebView 相同，代码示例如下：

```
WKWebView *webView = [[WKWebView alloc]
initWithFrame:self.view.bounds];

[webView loadRequest:[NSURLRequest requestWithURL:[NSURL
URLWithString:@"http://www.baidu.com"]]];

[self.view addSubview:webView];
```

(2) 加载的状态回调 （WKNavigationDelegate）

用来追踪加载过程（页面开始加载、加载完成、加载失败）的方法：

```
// 页面开始加载时调用
- (void)webView:(WKWebView *)webView
didStartProvisionalNavigation:(WKNavigation *)navigation;
// 当内容开始返回时调用
- (void)webView:(WKWebView *)webView
didCommitNavigation:(WKNavigation *)navigation;
// 页面加载完成之后调用
- (void)webView:(WKWebView *)webView
didFinishNavigation:(WKNavigation *)navigation;
// 页面加载失败时调用
```

```
- (void)webView:(WKWebView *)webView
didFailProvisionalNavigation:(WKNavigation *)navigation;
```

(3) 页面跳转的代理方法

```
// 接收到服务器跳转请求之后调用
 - (void)webView:(WKWebView *)webView
didReceiveServerRedirectForProvisionalNavigation:(WKNavigation
*)navigation;
 // 在收到响应后，决定是否跳转
- (void)webView:(WKWebView *)webView
decidePolicyForNavigationResponse:(WKNavigationResponse
*)navigationResponse decisionHandler:(void
(^)(WKNavigationResponsePolicy))decisionHandler;
 // 在发送请求之前，决定是否跳转
- (void)webView:(WKWebView *)webView
decidePolicyForNavigationAction:(WKNavigationAction
*)navigationAction decisionHandler:(void
(^)(WKNavigationActionPolicy))decisionHandler;
```

(4) 新的 WKUIDelegate

这个协议主要用于 WKWebView 处理 web 界面的三种提示框(警告框、确认框、输入框)，下面是警告框的例子:

```
/**
* web 界面中有弹出警告框时调用
*
* @param webView 实现该代理的 webview
* @param message 警告框中的内容
* @param frame 主窗口
* @param completionHandler 警告框消失调用
*/
 - (void)webView:(WKWebView *)webView
runJavaScriptAlertPanelWithMessage:(NSString *)message
initiatedByFrame:(void (^)())completionHandler;
```

3. WKWebview 进阶

(1) 动态加载并运行 JS 代码

用于在客户端内部加入 JS 代码，并加载网页，示例如下:

```
// 图片缩放的js代码
NSString *js = @"var count = document.images.length;for (var i = 0; i < count; i++) {var image = document.images[i];image.style.width=320;};window.alert('找到' + count + '张图');";
// 根据JS字符串初始化WKUserScript对象
WKUserScript *script = [[WKUserScript alloc] initWithSource:js injectionTime:WKUserScriptInjectionTimeAtDocumentEnd forMainFrameOnly:YES];
// 根据生成的WKUserScript对象，初始化WKWebViewConfiguration
WKWebViewConfiguration *config = [[WKWebViewConfiguration alloc] init];
[config.userContentController addUserScript:script];
_webView = [[WKWebView alloc] initWithFrame:self.view.bounds configuration:config];
[(WKWebView *)_webView loadHTMLString:@"<head></head><imgea src='http://www.nsu.edu.cn/v/2014v3/img/background/3.jpg' />"baseURL:nil];
[self.view addSubview:_webView];
```

(2) webView 执行 JS 代码

用户调用用 JS 写过的代码，一般指服务端开发的：

```
//javaScriptString是JS方法名，completionHandler是异步回调block
[(WKWebView *)self.webView evaluateJavaScript:@"javaScriptString" completionHandler:completionHandler];
```

(3) JS 调用 App 注册过的方法

1) 在 WKWebView 里面注册供 JS 调用的方法，是通过 WKUserContentController 类下面的方法：

- (void)addScriptMessageHandler:(id <WKScriptMessageHandler>)scriptMessageHandler name:(NSString *)name;

scriptMessageHandler 是代理回调，JS 调用 name 方法后，OC 会调用 scriptMessageHandler 指定的对象。

2) JS 在调用 OC 注册方法的时候要用下面的方式：

window.webkit.messageHandlers.<name>.postMessage(<messageBody>)

注意，name(方法名)是放在中间的，messageBody 只能是一个对象，如果要传多个值，需要封装成数组，或者字典。整个示例如下

```
//OC注册供JS调用的方法
[[(WKWebView *)_webView configuration].userContentController addScriptMessageHandler:self name:@"closeMe"];
//OC在JS调用方法做的处理
- (void)userContentController:(WKUserContentController *)userContentController didReceiveScriptMessage:(WKScriptMessage *)message
    NSLog(@"JS 调用了 %@ 方法，传回参数 %@",message.name,message.body);
}

//JS调用 window.webkit.messageHandlers.closeMe.postMessage(null);
```

如果你在 self 的 dealloc 打个断点，会发现 self 没有释放！这显然是不行的！谷歌后看到一种解决方法，如下：

思路是另外创建一个代理对象，然后通过代理对象回调指定的 self，

```
@interface WeakScriptMessageDelegate : NSObject<WKScriptMessageHandler>

@property (nonatomic, weak) id<WKScriptMessageHandler> scriptDelegate;

- (instancetype)initWithDelegate:(id<WKScriptMessageHandler>)scriptDelegate;

@end

@implementation WeakScriptMessageDelegate

- (instancetype)initWithDelegate:(id<WKScriptMessageHandler>)scriptDelegate
{
    self = [super init];
    if (self) {
        _scriptDelegate = scriptDelegate;
    }
    return self;
}

- (void)userContentController:(WKUserContentController *)userContentController didRe
{
    [self.scriptDelegate userContentController:userContentController didReceiveScrip
}

@end
```

```
WKUserContentController *userContentController =
[[WKUserContentController alloc] init];
[userContentController
addScriptMessageHandler:[[WeakScriptMessageDelegate alloc]
initWithDelegate:self] name:@"closeMe"];
```

运行代码，self 释放了，WeakScriptMessageDelegate 却没有释放啊啊啊！

还需在 self 的 dealloc 里面 添加这样一句代码：

```
[[_webView configuration].userContentController
removeScriptMessageHandlerForName:@"closeMe"];
```

OK，圆满解决问题！

4. WKWebView 封装

```
#import <WebKit/WebKit.h>

@protocol MyWebViewDelegate;

@interface MyWebView : UIView

@property (nonatomic,assign) id <MyWebViewDelegate> delegate;

@property(strong,nonatomic)WKWebView *webView;

/**
 加载网页
 */
-(void)loadUrlString:(NSString *)urlString;


@end
```

```
@protocol MyWebViewDelegate <NSObject>

@optional

/**
 开始加载
 */
- (void)didStartWebView:(MyWebView *)webView;
/**
 完成加载
 */
- (void)didFinishWebView:(MyWebView *)webView;
/**
 加载失败
 */
- (void)didFailWebView:(MyWebView *)webView;

/**
 获取到标题
 */
- (void)didGetTitle:(NSString *)title;


@end

#import "MyWebView.h"
#import "IndicatorView.h"

@interface MyWebView()<WKNavigationDelegate,MyWebViewDelegate>

@property (strong ,nonatomic) UIProgressView *progress;
@property (strong ,nonatomic) IndicatorView *loading;


@end

@implementation MyWebView


- (instancetype)initWithFrame:(CGRect)frame
{
    self = [super initWithFrame:frame];
    if (self) {
        self.backgroundColor = [UIColor clearColor];
        [self addSubview:self.webView];
        [self insertSubview:self.progress aboveSubview:self.webView];
        [self.webView addSubview:self.loading];
    }
    return self;
}

-(WKWebView *)webView{
    if (_webView==nil) {
        _webView = [[WKWebView alloc] initWithFrame:self.bounds];
        _webView.navigationDelegate = self;
        _webView.backgroundColor = [UIColor clearColor];
        _webView.opaque = NO;
        _webView.allowsBackForwardNavigationGestures = YES;
        [_webView goBack];
        [_webView addObserver:self forKeyPath:@"title" options:NSKeyValueObservingOptionNew context:NULL];
        [_webView addObserver:self forKeyPath:@"estimatedProgress" options:NSKeyValueObservingOptionNew context:NULL];
    }
    return _webView;
}
```

```
-(UIProgressView *)progress{
    if (_progress==nil) {
        _progress = [[UIProgressView alloc] initWithProgressViewStyle:UIProgressViewStyleBar];
        _progress.trackTintColor = [UIColor lightGrayColor];
        _progress.progressTintColor = DropColor;
        _progress.frame = CGRectMake(0, 0, self.bounds.size.width, 2);
    }
    return _progress;
}

-(IndicatorView *)loading{
    if (_loading==nil) {
        _loading = [[IndicatorView alloc] initWithType:IndicatorTypeMusic1 tintColor:RiseColor];
        _loading.center = CGPointMake(self.webView.bounds.size.width/2.0, self.webView.bounds.size.height/2.0);
    }
    return _loading;
}
#pragma mark - <************************ 代理 ************************>
/// 开始加载
- (void)webView:(WKWebView *)webView didStartProvisionalNavigation:(WKNavigation *)navigation{
    DLog(@"url:%@",webView.URL.absoluteString);
    [self.loading startAnimating];
    if (self.delegate&&[self.delegate respondsToSelector:@selector(didStartWebView:)]) {
        [self.delegate didStartWebView:self];
    }
}
/// 获取到网页内容
- (void)webView:(WKWebView *)webView didCommitNavigation:(WKNavigation *)navigation{
    NSLog(@"获取到内容");
}
/// 加载完成
- (void)webView:(WKWebView *)webView didFinishNavigation:(WKNavigation *)navigation{
    NSLog(@"加载完成");
    [self.loading stopAnimating];
    if (self.delegate&&[self.delegate respondsToSelector:@selector(didFinishWebView:)]) {
        [self.delegate didFinishWebView:self];
    }
}
/// 加载失败
- (void)webView:(WKWebView *)webView didFailProvisionalNavigation:(WKNavigation *)navigation{
    NSLog(@"加载失败");
    [self.loading stopAnimating];
    if (self.delegate&&[self.delegate respondsToSelector:@selector(didFailWebView:)]) {
        [self.delegate didFailWebView:self];
    }
}

#pragma mark - <************************ kvo监听 ************************>
- (void)observeValueForKeyPath:(NSString *)keyPath ofObject:(id)object change:(NSDictionary *)change context:(void *)context {
    // 监听标题
    if ([keyPath isEqualToString:@"title"]){
        if (object == self.webView){
            if (self.delegate&&[self.delegate respondsToSelector:@selector(didGetTitle:)]) {
                [self.delegate didGetTitle:self.webView.title];
            }
        }
        else
            [super observeValueForKeyPath:keyPath ofObject:object change:change context:context];
    }

    // 监听进度
    else if ([keyPath isEqualToString:@"estimatedProgress"]) {
        if (object == self.webView) {
            NSLog(@"%f",self.webView.estimatedProgress);
            [self.progress setProgress:self.webView.estimatedProgress animated:YES];
            self.progress.hidden = self.progress.progress==1?YES:NO;
            self.progress.progress = self.progress.progress==1?0:self.progress.progress;
        }
        else{
            [super observeValueForKeyPath:keyPath ofObject:object change:change context:context];
        }
    }
    else
        [super observeValueForKeyPath:keyPath ofObject:object change:change context:context];
}
```

```
#pragma mark - <************************* dealloc *************************>
-(void)dealloc{
    [self.webView removeObserver:self forKeyPath:@"title"];
    [self.webView removeObserver:self forKeyPath:@"estimatedProgress"];
    [self deleteWebCache];
}


- (void)deleteWebCache {
    if ([[UIDevice currentDevice].systemVersion floatValue] >= 9.0) {
        NSSet *websiteDataTypes = [NSSet setWithArray:@[
                                        WKWebsiteDataTypeDiskCache,
                                        WKWebsiteDataTypeOfflineWebApplicationCache,
                                        WKWebsiteDataTypeMemoryCache,
                                        WKWebsiteDataTypeLocalStorage,
                                        WKWebsiteDataTypeCookies,
                                        WKWebsiteDataTypeSessionStorage,
                                        WKWebsiteDataTypeIndexedDBDatabases,
                                        WKWebsiteDataTypeWebSQLDatabases
                                        ]];

        NSDate *dateFrom = [NSDate dateWithTimeIntervalSince1970:0];
        [[WKWebsiteDataStore defaultDataStore] removeDataOfTypes:websiteDataTypes modifiedSince:dateFrom completionHandler:^{
        }];
    } else {
        NSString *libraryPath = [NSSearchPathForDirectoriesInDomains(NSLibraryDirectory, NSUserDomainMask, YES) objectAtIndex:0];
        NSString *cookiesFolderPath = [libraryPath stringByAppendingString:@"/Cookies"];
        NSError *errors;
        [[NSFileManager defaultManager] removeItemAtPath:cookiesFolderPath error:&errors];
    }
}
```

这样一个 webview 就封装完成了，接下来使用就很方便了

5. WKWebView 封装的应用

```
#import <UIKit/UIKit.h>

@interface MyWebViewController : UIViewController

/**
 网页地址
 */
@property (copy ,nonatomic) NSString *urlString;

@end
```

```
#import "MyWebViewController.h"
#import "MyWebView.h"

@interface MyWebViewController ()
// !!!: WebView
@property (strong ,nonatomic) MyWebView *webView;

@end

@implementation MyWebViewController

- (void)viewDidLoad {
    [super viewDidLoad];

    // 初始化视图
    [self initUI];
}
```

```
#pragma mark - <************************* 配置视图 *************************>
// !!!: 配置视图
-(void)initUI{
    self.view.backgroundColor = [UIColor groupTableViewBackgroundColor];

    // navBar
    {
        [self.navigationBar setTitle:self.title leftBtnImage:@"zhiboLeft" rightBtnImage:nil];
    }

    // webView
    {
        [self.view addSubview:self.webView];
        [self.webView loadUrlString:self.urlString];
    }

    // 返回手势
    {
        self.isNeedGoBack = YES;
    }
}

#pragma mark - <********************** 初始化控件/数据 **********************>
-(MyWebView *)webView{
    if (_webView==nil) {
        _webView = [[MyWebView alloc] initWithFrame:CGRectMake(0, self.navigationBar.bottom, kScreenWidth, kScreenHeight-self.navigationBar.bottom)];
    }
    return _webView;
}


#pragma mark - <************************* 代理方法 *************************>
-(void)navigationViewLeftDlegate{
    [self.navigationController popViewControllerAnimated:YES];
}


#pragma mark - <************************* 私有方法 *************************>


#pragma mark - <************************* 检测释放 *************************>
- (void)dealloc{
    DLog(@"%@释放掉",[self class]);
}


- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
```