SCIENCES SORBONNE UNIVERSITÉ

«µNav2»

Fabrice.Kordon@lip6.fr

LIP6

As an introduction…

Objectives…

- Enrich µNav

What for?

- Better use of WKWebview capabilities
- Use of a UIToolBar
- Use of «popover» on large devices

Demo (small device)

Demo (large device)

ViewController


```
import UIKit

class ViewController: UIViewController {
    private let alertVC = UIAlertController(title: "URL",
                                            message: "Enter the URL you want to reach",
                                            preferredStyle: .alert)
    private let actionVC = UIAlertController(title: "Go to LIP6",
                                             message: "Do you confirm?",
                                             preferredStyle: .actionSheet)
    private var alertVCAnswer : UITextField?

    private let v = MyView(frame: .zero)
    private let isIpad = UIDevice.current.userInterfaceIdiom == .pad
```

ViewController


```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
    self.view = v
    v.drawInFormat(format:UIScreen.main.bounds.size)
    // Build the alert
    alertVC.addTextField(configurationHandler: {f -> Void in
        // Let's customize
        f.textColor = UIColor.blue
        f.text = "https://"
        // Let's associate a known reference
        self.alertVCAnswer = f
    })
    alertVC.addAction(UIAlertAction(title: "OK",
                                    style: .default,
                                    handler: readURL))
    alertVC.addAction(UIAlertAction(title: "Cancel",
                                    style: .cancel,
                                    handler: nil))
    // Build the action
    actionVC.addAction(UIAlertAction(title: "Yes I want",
                                     style: .destructive,
                                     handler: loadLIP6))
    actionVC.addAction(UIAlertAction(title: "No, I changed my mind",
                                     style: .cancel,
                                     handler: nil))
}
```

ViewController


```
override func viewWillTransition(to size: CGSize,
                  with coordinator: UIViewControllerTransitionCoordinator) {
    super.viewWillTransition(to: size, with: coordinator)
    let v = self.view as! MyView
    v.drawInFormat(format:size)
}

@objc func tbButtonAction(sender : UIBarButtonItem) {
    if sender == v.backtb {
        v.backwardInWebView()
    } else {
        v.forwardInWebView()
    }
}

@objc func newURL() {
    self.present(alertVC, animated: true, completion: {})
}

@objc func readURL(sender : UIAlertAction) {
    let v = self.view as! MyView
    v.loadURL(url: (alertVCAnswer?.text!)!)
}
```

ViewController


```
    @objc func goLIP6() {
        if isIpad {
            actionVC.modalPresentationStyle = .popover
            actionVC.popoverPresentationController?.barButtonItem = v.lip6tb
        }
        self.present(actionVC, animated: true, completion: {})
    }

    func loadLIP6(sender : UIAlertAction) {
        v.loadURL(url: "https://lip6.fr")
    }
}
```

MyView


```
import UIKit
import WebKit

class MyView: UIView, WKNavigationDelegate, WKUIDelegate {

    private var wv : WKWebView!
    private let b1 = UIButton(type: .system)
    private let b2 = UIButton(type: .system)
    private let l = UILabel()
    // new in µNav2
    private let tb = UIToolbar()
let backtb =  UIBarButtonItem(barButtonSystemItem: .rewind,
                              target: nil, action: nil)
let fwrdtb =  UIBarButtonItem(barButtonSystemItem: .fastForward,
                              target: nil, action: nil)
let lip6tb =  UIBarButtonItem(barButtonSystemItem: .action,
                              target: nil, action: nil)
let loadtb =  UIBarButtonItem(title: "URL", style: .plain,
                              target: nil, action: nil)
```

MyView


```
override init(frame: CGRect) {
    b1.setTitle("Page 1", for: .normal)
    b2.setTitle("Back", for: .normal)
    l.font = UIFont.systemFont(ofSize: 12.0)
    l.textColor = .red
    l.textAlignment = .center
    // New in µNav2
    let smallSpace = UIBarButtonItem(barButtonSystemItem: .fixedSpace,
                                     target: nil, action: nil)
    smallSpace.width = 20
    let varSpace = UIBarButtonItem(barButtonSystemItem: .flexibleSpace,
                                   target: nil, action: nil)
tb.items = [varSpace, backtb, smallSpace, loadtb, smallSpace,
            fwrdtb, varSpace, lip6tb]
// handling the web part
wv = WKWebView(frame: .zero, configuration: WKWebViewConfiguration())
super.init(frame: frame) // then we will need self
wv.navigationDelegate = self
wv.uiDelegate = self
self.addSubview(wv)
b1.addTarget(self, action: #selector(goThere), for: .touchDown)
b2.addTarget(self, action: #selector(goBack), for: .touchDown)
self.backgroundColor = UIColor(red:0.7, green:0.8, blue:0.9, alpha:1)
self.addSubview(wv)
self.addSubview(b1)
self.addSubview(b2)
self.addSubview(l)
```

MyView


```
        // New in µNav2
        self.addSubview(tb)
        // Another way to deal with UIBarButton's target
        backtb.target = self.superview
        fwrdtb.target = self.superview
        lip6tb.target = self.superview
        loadtb.target = self.superview
        backtb.action = #selector(ViewController.tbButtonAction)
        fwrdtb.action = #selector(ViewController.tbButtonAction)
        lip6tb.action = #selector(ViewController.goLIP6)
        loadtb.action = #selector(ViewController.newURL)
        // Loading an URL
        self.loadURL(url: "https://sorbonne-universite.fr")
        // set-up the display
        self.drawInFormat(format: frame.size)
    }

    required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }
```

MyView


```
func loadURL(url: String) {
    wv.load(URLRequest(url:URL(string: url)!))
}

func checkNavButtons() {
    if wv.canGoForward {
        fwrdtb.isEnabled = true
    } else {
        fwrdtb.isEnabled = false
    }
    if wv.canGoBack {
        backtb.isEnabled = true
    } else {
        backtb.isEnabled = false
    }
}
```

MyView


```
@objc func goThere () {
    let crtURL : URL?
    if b1.titleLabel?.text == "Page 1" {
        crtURL = URL(fileURLWithPath:
            Bundle.main.path(forResource: "page-1", ofType: "html")!)
        b1.setTitle("Page 2", for: .normal)
    } else {
        crtURL = URL(fileURLWithPath:
            Bundle.main.path(forResource: "page-2", ofType: "html")!)
        b1.setTitle("Page 1", for: .normal)
    }
    let r = URLRequest(url: crtURL!)
    wv.load(r)
}

@objc func goBack () {
    wv.goBack()
    self.setNeedsDisplay() // to refrech
}
```

MyView


```
func drawInFormat (format: CGSize) {
    var top = 20;
    if UIDevice.current.userInterfaceIdiom == .phone &&
        format.height >= 812 {
        top = 30
    } else if UIDevice.current.userInterfaceIdiom == .phone &&
        format.width > format.height {
        top = 0
    }
    let tBar = 44 // typical size of a toolBar
    let sBar = 40
    wv.frame = CGRect(x:0, y:top + tBar + sBar,
                      width:Int(format.width),
                      height:Int(format.height - 50))
    b1.frame = CGRect(x:10, y:top + tBar + 2, width:70, height:20)
    b2.frame = CGRect(x:Int(format.width) - 80,
                      y:top + tBar + 2, width:70, height:20)
    l.frame = CGRect(x:10, y:top + tBar + 20,
                     width:Int(format.width)-20, height:20)
    tb.frame = CGRect(x:0, y:top,
                      width:Int(format.width), height:tBar )
}

func forwardInWebView() {
    wv.goForward()
}
```

MyView


```
func backwardInWebView() {
    wv.goBack()
}

// WKNavigationDelegate delegate protocol

func webView(_ webView: WKWebView,
             didFailProvisionalNavigation navigation: WKNavigation!,
             withError error: Error) {
    // avoid this stupid "NSURLErrorDomain error -999"
    if error.localizedDescription.range(of: "error -999") == nil {
        let errVC = UIAlertController(title: "Error",
                                      message: error.localizedDescription,
                                      preferredStyle: .alert)
        errVC.addAction(UIAlertAction(title: "OK",
                                      style: .default,
                                      handler: nil))
        // I have no access to my UIViewController
        let vc = UIApplication.shared.windows[0].rootViewController
        vc?.present(errVC, animated: true, completion: nil)
    }
}
```

MyView


```
func webView(_ webView: WKWebView,
             didReceiveServerRedirectForProvisionalNavigation
                 navigation: WKNavigation!) {
    // Let's get the new redirected URL
    let url = URL(string: webView.url!.absoluteString)
    if url != nil {
        let request = URLRequest(url: url!)
        wv.load(request)
    } else {
        l.text = "Bad redirection!!!"
    }
}

func webView(_ webView: WKWebView, didFinish navigation: WKNavigation!) {
    let t = webView.title
    if t != nil {
        l.text = "«"+t!+"» loaded"
    }
    self.checkNavButtons()
}
```

MyView

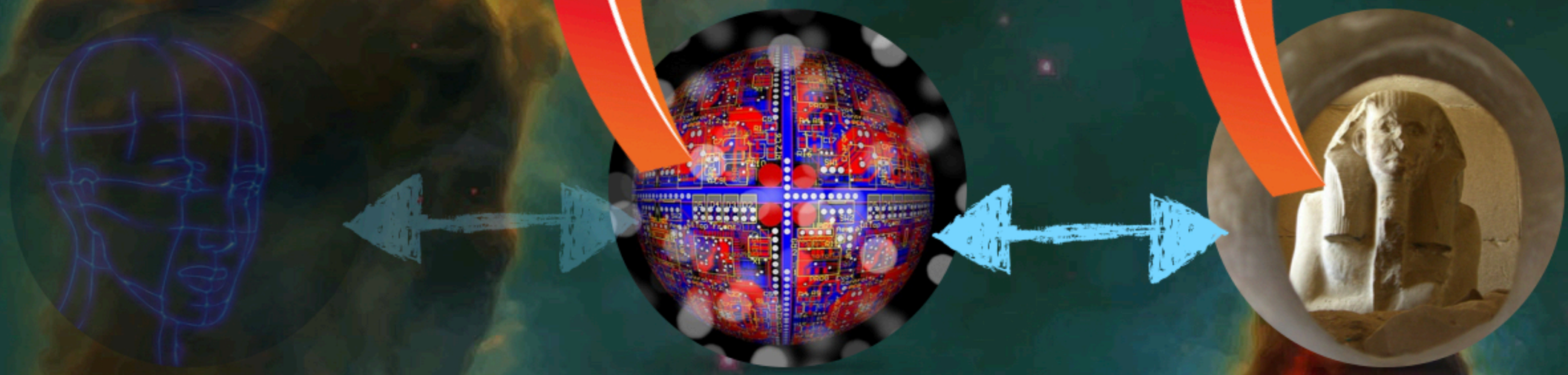

```
// WKUIDelegate delegate protocol

func webView(_ webView: WKWebView,
             runJavaScriptAlertPanelWithMessage message: String,
             initiatedByFrame frame: WKFrameInfo,
             completionHandler: (() -> Void)) {
    let a = UIAlertController(title: "Alert",
                              message: message,
                              preferredStyle: .alert)
    a.addAction(UIAlertAction(title: "OK", style: .default, handler: nil))
    let vc = UIApplication.shared.windows[0].rootViewController
        vc?.present(a, animated:true,completion:nil)
        completionHandler() // MUST be called (defined with @escaping)
    }
}
```

What About MVC?

- Orientation
- barButton's actions

- GoThere
- goback
- delegation



ViewController

MyView

In the ViewController

- interaction with the view
  - **Calls to MyView's methode generate call-backs handled by MyView**

In MyView

- All local actions
  - **Those relative to the management of the WKWebView**

As a conclusion…

We got it!

- Lots of delegation
  - **Even over several objects**
  - **Cascades of call-backs**
- Adapted behavior (depends of the device)

You start to see what it looks like!

- You have to be careful!