SCIENCES SORBONNE UNIVERSITÉ

«Memory»

Fabrice.Kordon@lip6.fr

LIP6

Goal of the example

Quite simple!

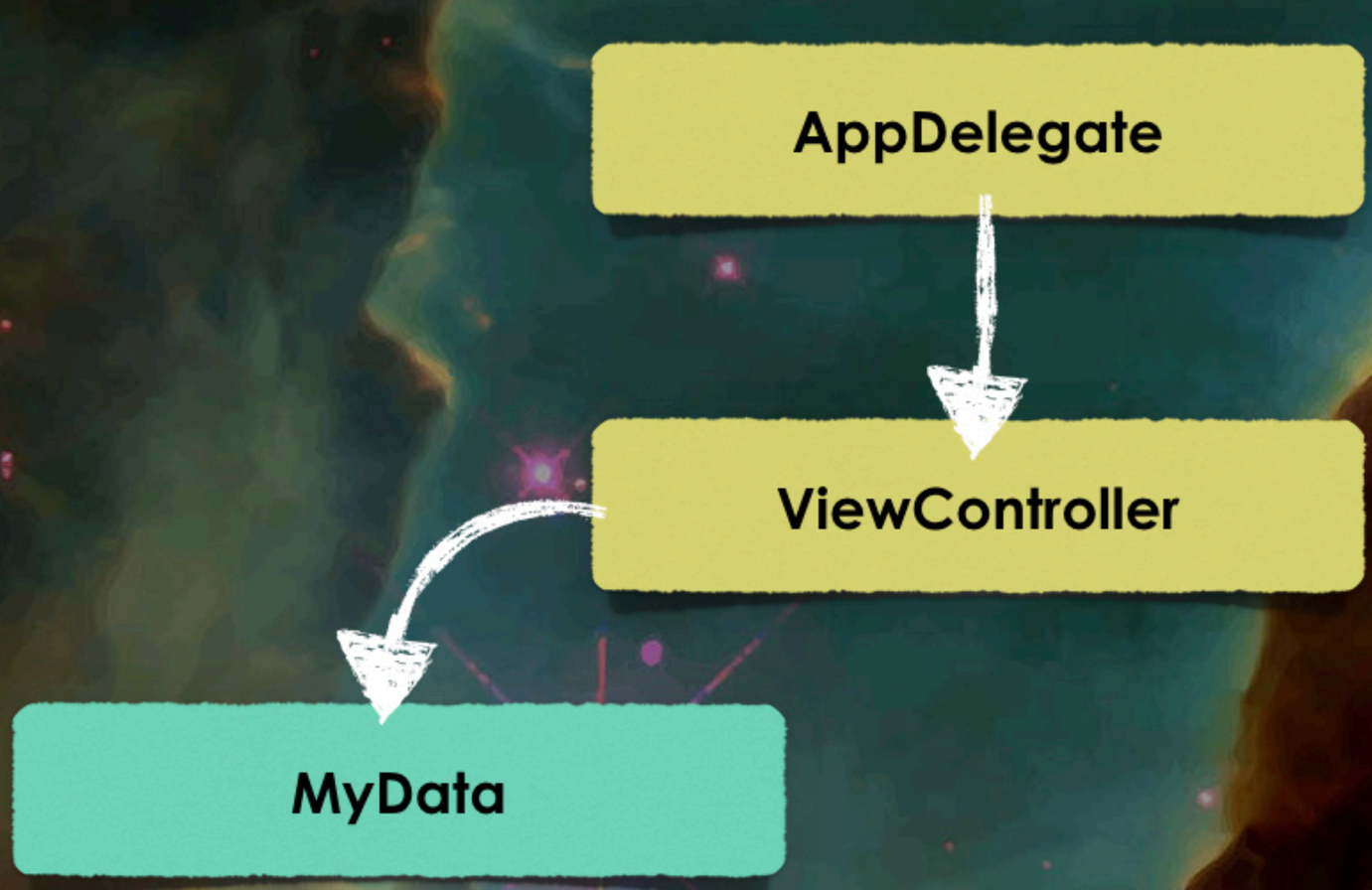
- Read saved data
  - **Check for the file's existence**
  - **Decode data**
- Allow the user to type data
  - **UITextView**
- Save these data
  - **Standard encoding of a class…**
  - **.. even it it contains only a String**

Demo

Architecture of the Application

A class to store data to be saved

- Here, a String only
- Use of NSCoding (technique 1)



MyData


```swift
import UIKit

class MyData: NSObject, NSCoding {

    var aString = ""

    init(s : String) {
        super.init()
        aString = s
    }

    //  NSCoding protocol

    required init?(coder aDecoder: NSCoder) {
        // if you inherit of an object implementing NSCoding
        // super.init(coder: aDecoder)
        // otherwise
        super.init()
        aString = aDecoder.decodeObject(forKey: "the_text") as! String
    }

    func encode(with aCoder: NSCoder) {
        // To do for each enclosed property/attribute
        aCoder.encode(aString, forKey: "the_text")
    }
}
```

ViewController


```
import UIKit

class ViewController: UIViewController, UITextViewDelegate {

    private let s = UIScreen.main.bounds.size
    private let textArea = UITextView()
    private let backg = UIImageView(image: UIImage(named: "background"))
    private let b = UIButton(type: .system)
    private let rep = NSSearchPathForDirectoriesInDomains(.documentDirectory,
                                                          .userDomainMask, true)

    override var preferredStatusBarStyle: UIStatusBarStyle {
        return .lightContent
    }
```

ViewController


```
override func viewDidLoad() {
        super.viewDidLoad()
        self.view = UIView()
        backg.center = CGPoint(x: s.width / 2, y: s.height / 2)
        self.view.addSubview(backg)
        b.frame = CGRect(x: s.width / 2 - 80, y: 260, width: 160, height: 30)
        b.setTitle("Serialize text", for: .normal)
        b.tintColor = UIColor.white
        b.addTarget(self, action: #selector(doSave), for: .touchDown)
        self.view.addSubview(b)
        textArea.frame = CGRect(x: 20, y: 60, width: s.width - 40, height: 150)
        textArea.backgroundColor = UIColor(red: 1, green: 1, blue: 1, alpha: 0.5)
        textArea.font = UIFont(name: "Helvetica-bold", size: 16)
        textArea.returnKeyType = .done
        textArea.delegate = self
        self.view.addSubview(textArea)
        // Fetch data...
        let thePath = rep[0] + "/backup"
        if FileManager.default.fileExists(atPath: thePath) {
            let data = FileManager.default.contents(atPath: thePath)
            if data != nil {
                do {
                    let decoder = try NSKeyedUnarchiver(forReadingFrom: data!)
                    decoder.requiresSecureCoding = false
                    let d = try
                            NSKeyedUnarchiver.unarchiveTopLevelObjectWithData(data!)
                                as? MyData
                    textArea.text = d?.aString
                } catch {
                    print("decoding failed -- should no happend")
                }
            }
        }
}
```

ViewController


```
@objc func doSave() {
    let d = MyData(s: textArea.text)
    let thePath = rep[0] + "/backup"
    let coder = NSKeyedArchiver(requiringSecureCoding: false)
    coder.encode(d, forKey: NSKeyedArchiveRootObjectKey)
    FileManager.default.createFile(atPath: thePath,
                                   contents: coder.encodedData,
                                   attributes: [:])
}

// UITextViewDelegate Protocol
    func textView(_ textView: UITextView,
                  shouldChangeTextIn range: NSRange,
                  replacementText text: String) -> Bool {
        if text == "\n" {
            textArea.resignFirstResponder()
            return false // no CR
        }
        return true // yes if not CR
    }

    func textViewShouldEndEditing(_ textView: UITextView) -> Bool {
        return textArea.hasText
    }
}
```

Content of the «backup» file


```
// !!! BINARY PROPERTY LIST WARNING !!!
//
// The pretty-printed property list below has been created
// from a binary version on disk and should not be saved as
// the ASCII format is a subset of the binary representation!
//
{    "$archiver" = "NSKeyedArchiver";
    "$objects" = (
        "$null",
        {    "$class" = :false;
            the_text = :false;
    },
        "Toto est content",
        {    "$classes" = ( "Memory.MyData", "NSObject" );
            "$classname" = "Memory.MyData";
    },
    );
    "$top" = { root = :false; };
    "$version" = 100000;
}
```

Oups!!!

Let's iOS handle this…

Before iOS12

In the viewDidLoad()

```
...
// Fetch data...
let thePath = rep[0] + "/backup"
if FileManager.default.fileExists(atPath: thePath) {
    let d = NSKeyedUnarchiver.unarchiveObject(withFile: thePath) as! MyData?
    textArea.text = d?.aString
}
...
```

doSave()

```
@objc func doSave() {
    let d = MyData(s: textArea.text)
    let thePath = rep[0] + "/backup"
    let res = NSKeyedArchiver.archiveRootObject(d, toFile: thePath)
    if !res {
        let a = UIAlertController(title: "Error",
                                  message: "Backup could not be performed",
                                  preferredStyle: .alert)
        a.addAction(UIAlertAction(title: "OK", style: .default, handler: nil))
        self.present(a, animated: true, completion: nil)
    }
}
```

As a conclusion…

- Easy is'n't it?
- You can now store & fetch data
- For the rest…
  - The Fantastic Manual
- Ready to play?