POLITÉCNICA

ETSIT UPM

dit UPM

Desarrollo de Apps para iOS

Mezclar Swift y Objective-C

IWEB-LSWC 2014-2015
Santiago Pavón

ver: 2014.11.25

Mezclar Swift y Objective-C

- Objective-C sigue siendo un lenguaje válido para desarrollar.
  - No hay que reescribir todo el código Objective-C existente en Swift.
  - El código Objective-C existente se puede mantener.
- Las nuevas funcionalidades se pueden desarrollar con Swift y funcionarán junto con el código Objective-C existente.
  - Objective-C y Swift se ven mutuamente.
  - Los patrones usados en Objective-C siguen siendo válidos con Swift.

- Para añadir un fichero Swift en un proyecto desarrollado con Objective-C:

  `Menú File > New… > File > iOS + Source > Swift File`

- Para añadir un fichero Objective-C en un proyecto desarrollado Swift:

  `Menú File > New… > File > iOS + Source > Objective-C File`

- La primera vez que se añade un fichero de un lenguaje en un proyecto del otro lenguaje, aparece una ventana de diálogo preguntando si se desean crear los ficheros de cabeceras puente necesarios para que Objective-C y Swift se vean entre sí.
  - Hay que aceptar.
  - Se crea el fichero:
    - AppName-Bridging-Header.h
  - Se configuran dos opciones en el ***Build Setting*** del producto que se está desarrollando.
  - ***Install Objective-C Compatibility Header***
    - Se asigna el valor Yes.
  - ***Objective-C Bridging Header***
    - Apunta al fichero AppName-Bridging-Header.h.

- Los elementos desarrollados con Objective-C que se usen en los ficheros Swift se deben importar en el fichero AppName-Bridging-Header.h

```
import AppName-Bridging-Header.h
```

  - El desarrollador debe mantener el contenido de este fichero manualmente.
    - Debe incluir en él las sentencias # **import** necesarias para importar todos ficheros .h que deban verse desde Swift.

- En el siguiente ejemplo:
  - **Person** es una clase desarrollada con Objective-C.
  - La clase **Worker** desarrollada con Swift debe derivar de **Person**.
  - Para que la clase **Person** pueda usarse en el fichero **Worker.swift** hay que editar el fichero **AppName-Bridging-Header.h** para importar **Person.h**.

Person.h

```objc
#import <Foundation/Foundation.h>

@interface Person : NSObject

@property (strong,nonatomic) NSString *name;

@end
```

Worker.swift

```swift
import Foundation

class Worker : Person {
    var job : String = "Driver"
}
```

AppName-Bridging-Header.h

```objc
#import "Person.h"
```

- Todo fichero .m (Objective-C) que use algún elemento definido en Swift, debe importar el fichero:

```
#import "AppName-Swift.h"
```

  - Este fichero lo crea y lo mantiene automáticamente Xcode.
    - Nota: AppName es el nombre del producto (Target).
      - sustituir en AppName los caracteres no alfanuméricos por _.
- Recordad que los elementos declarados en Swift que deban ser visibles en Objective-C deben marcarse con **@objc**.

- En el siguiente ejemplo:
  - **Model** es un modelo desarrollado en Swift que se usa en **ViewController** que está hecho con Objective-C.
    - En el modelo **Model** se crea un observador para que cada vez que cambie el valor de su propiedad avise a su delegado.
    - El delegado es **ViewController**.
  - **Model** adopta **NSObject** para tener acceso a **alloc**.

Model.swift

```
import Foundation

@objc protocol ModelDelegate {
    func model(model: Model, hasChangedTo name: String)
}

@objc class Model : NSObject {

    weak var delegate: ModelDelegate?

    var name: String = "" {
        didSet {
           delegate?.model(self, hasChangedTo: self.name)
        }
    }

    init(name: String) {
        self.name = name
        super.init()
    }
}
```

ViewController.m

```
#import "ViewController.h"
#import "AppName-Swift.h"

@interface ViewController () <ModelDelegate>
@property (weak, nonatomic) IBOutlet UILabel *msg;
@property (strong, nonatomic) Model * model;
@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    self.model = [[Model alloc] initWithName:@"Hola"];
    self.model.delegate = self;
}

- (void) model:(Model *)model hasChangedTo:(NSString *)name {
    self.msg.text = name;
}

@end
```