Desarrollo de Apps para iOS
View Controllers

IWEB 2019-2020
Santiago Pavón

ver: 2019-11-06

La Clase UIViewController

- Es la **clase base** para crear nuestras pantallas *(y más cosas)*.
  - Crearemos **clases derivadas** para añadir la lógica de nuestra aplicación.
    - Sobrescribiendo o añadiendo propiedades y métodos.
- Proporciona la parte controladora (**C**) del patrón MVC.
- Proporciona una vista (**V**) vacía a la que añadiremos nuestra jerarquía de views.
  - La jerarquía de views se puede crear programáticamente, o
  - cargando el GUI desde un fichero storyboard o XIB.
    - Definiremos IBOutlets y IBActions para enganchar las propiedades y métodos entre el código y el diseño gráfico.
- **No** proporciona el modelo (**M**).
- Esta clase base proporciona métodos y propiedades para realizar muchas tareas: gestión de memoria, rotaciones, navegación, transiciones, etc...
  - Sobrescribiremos estos métodos para adaptarlos a nuestras necesidades.

Propiedades y Métodos de UIViewController

view

- De la clase base UIViewController heredamos la propiedad **view**.

```
var view: UIView!
```

  - Apunta a la raíz de nuestra jerarquía de views.

- Normalmente la jerarquía de views se carga desde un fichero storyboard, pero también se puede hacer programáticamente.

- No retener subviews:
  - La propiedad **view** retiene a todas las subviews que forman parte de su jerarquía de subviews.
    - las apuntará directamente o a través de una subview intermedia
  - No es necesario que nuestros outlets también retengan a estas subviews.

```
@IBOutlet weak var label: UILabel!
```

  - Hay que evitar crear bucles de retenciones.

viewDidLoad

- Este método se llama después de cargar la view (y sus subviews) en memoria, y de enganchar los outlets existentes.
  -tanto si la view se cargó desde un storyboard o si se creó en loadView.

```
func viewDidLoad()
```

- Este método se usa típicamente para hacer inicializaciones que no pueden hacerse hasta que vista ya se ha cargado.
  - Pero en este método no pueden hacerse configuraciones que estén relacionadas con la geometría de las vistas.
    - Cuando se invoca este método, aun no se ha calculado la geometría de la vista.

- Ejemplo:
  - Crear un VC que muestre en una `UILabel` el texto guardado en una propiedad.

    - En el inicializador (`init`???) del VC no se puede cambiar el texto de una `UILabel` del GUI.
      - Ya que la `UILabel` aun no existe; todavía no se ha cargado en memoria.

    - Sin embargo, el texto de la `UILabel` si puede cambiarse en el método `viewDidLoad`.
      - Cuando se llama a este método, la vista y sus subvistas ya han sido cargadas.
        - Sobrescribiremos el método `viewDidLoad` para asignar el valor que se desee como texto de la etiqueta.

```
override func viewDidLoad() {

    super.viewDidLoad()

    unaLabelDeLaVista.text = msg
}
```

Muy importante:
No olvidar llamar al padre.

Ponemos como texto de la etiqueta el valor guardado en la propiedad **msg**.

El outlet **unaLabelDeLaVista** es una UILabel que se carga desde un storyboard.
Al crear el objeto VC la view y sus subvistas no se han cargado.
Cuando se llama a **viewDidLoad**, la vista ya ha sido cargada, y **unaLabelDeLaVista** ya existe. Ahora es cuando se puede poner como texto de la etiqueta el valor guardado en la propiedad **msg**.

Cambios en la Visibilidad del VC

- Avisos indicando que el objeto ViewController se va a hacer visible, que ya es visible, que va a ocultarse, o que ya se ha ocultado.

```
func viewWillAppear(_ animated: Bool)
func viewDidAppear(_ animated: Bool)
func viewWillDisappear(_ animated: Bool)
func viewDidDisappear(_ animated: Bool)
```

- Sobrescribir estos métodos si queremos hacer algo en estos instantes.
  - Por ejemplo, gestionar la persistencia de algún valor, refrescar el dato mostrado por alguna view, …
  - No olvidar llamar a la versión tapada de la clase padre (**super.???**).
- Estos métodos se llaman cada vez que cambia la visibilidad del VC.
  - Ocurrirá con frecuencia en las aplicaciones que tienen varias pantallas.
    - Pero cuidado: cuando la aplicación termina no se llama a XXXDisappear.
- Cuando se invoca a **viewWillAppear** la geometría de **view** ya se ha calculado,
  - pero no se han aplicado las restricciones de autolayout a las subviews aun.
  - Cuando se invoca a **viewDidAppear** ya se ha calculado la geometría con la que se mostrarán las subviews.

didReceiveMemoryWarning

- Este método se llama cuando hay problemas de falta de memoria.

```
func didReceiveMemoryWarning()
```

- Solo debe liberarse memoria de un VC si **no se está mostrando** en la pantalla.

```
override func didReceiveMemoryWarning() {

    super.didReceiveMemoryWarning()

    if isViewLoaded && view.window == nil {

        // Liberar memoria que pueda regenerarse otra vez
        // en un futuro en viewDidLoad o viewWillAppear, …
        mi_tabla_de_fotos = nil
    }
}
```

- Aclaraciones sobre el ejemplo anterior:

- Primero debe llamarse a la versión del método tapada en el padre.

```
super.didReceiveMemoryWarning()
```

- La sentencia **if** comprueba que el VC no sea visible en la pantalla en este momento.
  - Primero hay que comprobar que **view** esté cargada en memoria:

```
isViewLoaded
```

    - Esta comprobación es necesaria porque si **view** no está cargada en memoria, se cargaría automáticamente de nuevo al acceder a **view** al evaluar la expresión **view.window**.

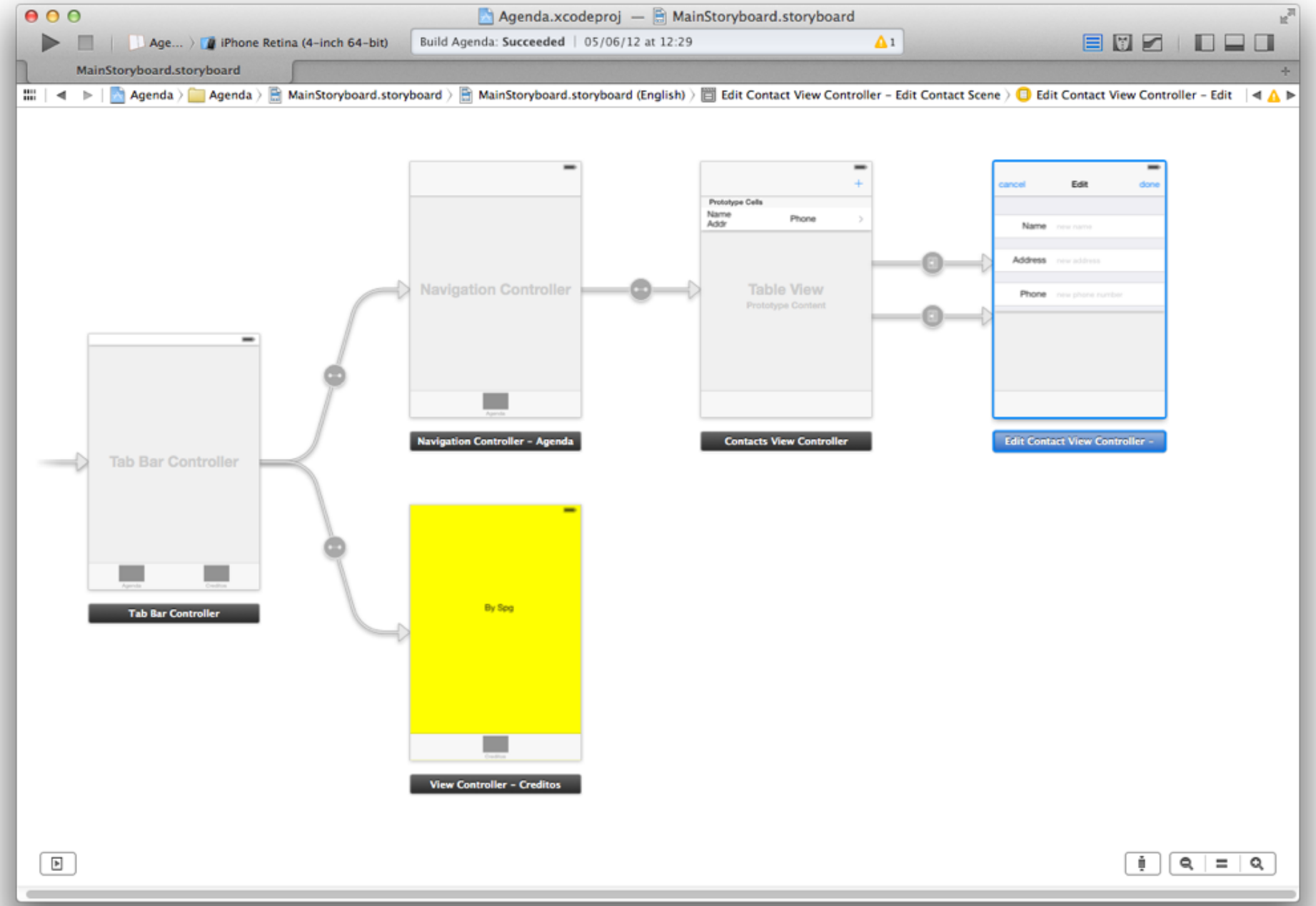
  - Un VC no está visible en pantalla si la propiedad **window** de su propiedad **view** es nil.

```
view.window == nil
```

- Si la condición del **if** se cumple, se liberan todos aquellos objetos que puedan regenerarse en un futuro en caso de necesidad.
  - Si este VC vuelve a hacerse visible, se llamará otra vez a **viewDidLoad**, donde se regenerarían los objetos liberados aquí.

Crear VC usando Storyboard

- Normalmente se usa un storyboard para diseñar sus pantallas o escenas de las aplicaciones,
  - y se conectan y relacionan entre sí usando segues.
- Estas pantallas o escenas serán:
  - objetos View Controller y
  - controladores de navegación.
- Los objetos VC diseñados en el storyboard se instanciarán **automáticamente** cuando se necesiten.
  - Los segues existentes indican que VC deben instanciarse.
  - y sólo necesitamos configurarlos en **prepare(for:UIStoryboardSegue,sender:Any?)**.
    - o usar **IBSegueAction** (Introducido en Xcode 11)



Aplicaciones con Varias Pantallas

- Las aplicaciones están formadas por varias pantallas por las que se navega usando:
  - Controladores de ViewControllers:
    - Existen VC controladores que manejan otros VC:
      - Crean su vista usando las vistas de otros VC.

➡**`UINavigationController`**

- Maneja una pila de VC.

➡**`UITabBarController`**

- Selección de VC independientes usando pestañas.

➡**`UISplitViewController`**

- VC maestro que controla los detalles mostrados en otro VC.

- etc . . .

  - ViewControllers Modales:
    - Un ViewController puede mostrar de forma modal otro VC.
      - En iPhone los VC modales ocupan toda la pantalla
      - En iPad pueden mostrarse con diferentes estilos.

- Popovers, ...

Relaciones entre Controladores

- Los objetos ViewController poseen propiedades para acceder a los VC con los que están relacionados

```
tabBarController
navigationController
parent
presentingViewController
presentedViewController
splitViewController
popoverPresentationController
presentationController
childViewControllers
```