ETSIT UPM

dit UPM

Desarrollo de Apps para iOS

Container View Controllers

IWEB 2016-2017
Santiago Pavón

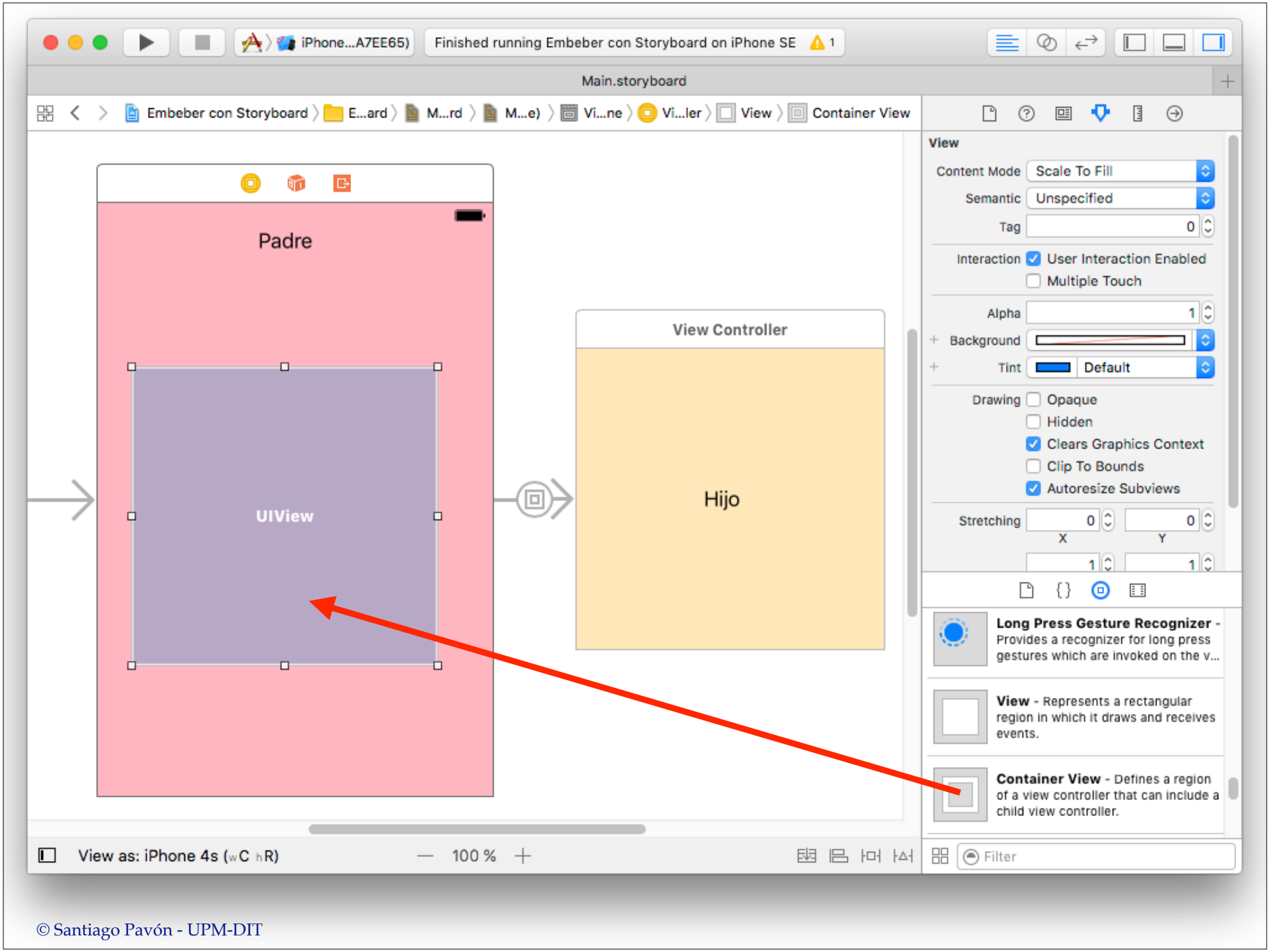
ver: 2016.09.05

¿En qué Consiste?

- Un View Controller Contenedor es un View Controller que muestra dentro de su vista las vistas de otros View Controllers.
  - Es interesante para evitar que el código de un VC sea muy grande/ complejo, para facilitar la reutilización, …
  - Se trocean las funcionalidades en varios View Controllers y luego se incluyen sus vistas en la vista del contenedor.
- Crear jerarquía de View Controllers:
  - Además de construir la jerarquía completa de UIViews, es necesario construir la jerarquía de UIViewControllers para que los métodos del ciclo de vida de los View Controllers incluidos se llamen adecuadamente:
    - al rotar el terminal, cuando aparece y desaparece el VC, al recalcular layouts, …

Con Storyboard - Interface Builder

- Se usan Container Views
  - Son views cuyo contenido es la view de otro VC.
- Arrastrarlas desde la biblioteca de objetos hasta el Storyboard.
- Un Container View se enlaza con el View Controller que mostrará con **segues** de tipo **embed**.
  - Con los segues embed también puede sobrescribirse el método **prepare(for:sender:)** del VC contenedor para configurar el VC embebido.
    - No olvidar asignar un identificador único al segue.



Programáticamente

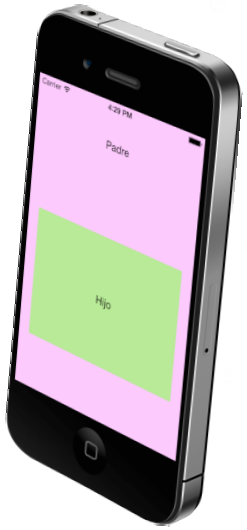
- Crear la **jerarquía de UIViews**, que será una mezcla de las vistas de varios VC.
- Crear la **jerarquía de VC**.
  - Para añadir/eliminar un UIViewController como hijo:

```
func addChildViewController(_ childController: UIViewController)
func removeFromParentViewController()
```

  - Antes de añadir o eliminar un VC hijo:

```
func willMove(toParentViewController parent: UIViewController?)
```

    - Este método ya se llama automáticamente desde addChildViewController.
    - Debemos llamarlo después de removeFromViewController.



  - Para hacer transiciones entre VC hijos:

```
func transition(from fromViewController: UIViewController,
                to toViewController: UIViewController,
                duration: TimeInterval,
                options: UIViewAnimationOptions = [],
                animations: (@escaping () -> Void)?,
                completion: (@escaping (Bool) -> Void)? = nil)
```

  - Después de añadir o eliminar un VC hijo:

```
func didMove(toParentViewController parent: UIViewController?)
```

    - Este método ya se llama automáticamente desde removeFromViewController.
    - Debemos llamarlo después de addChildViewController.
      - y una vez hayan terminado las animaciones de las transiciones.

- Gestionar Trait Collection y cambios de tamaño:
  - Cambiar el TraitCollection de un View Controller hijo:

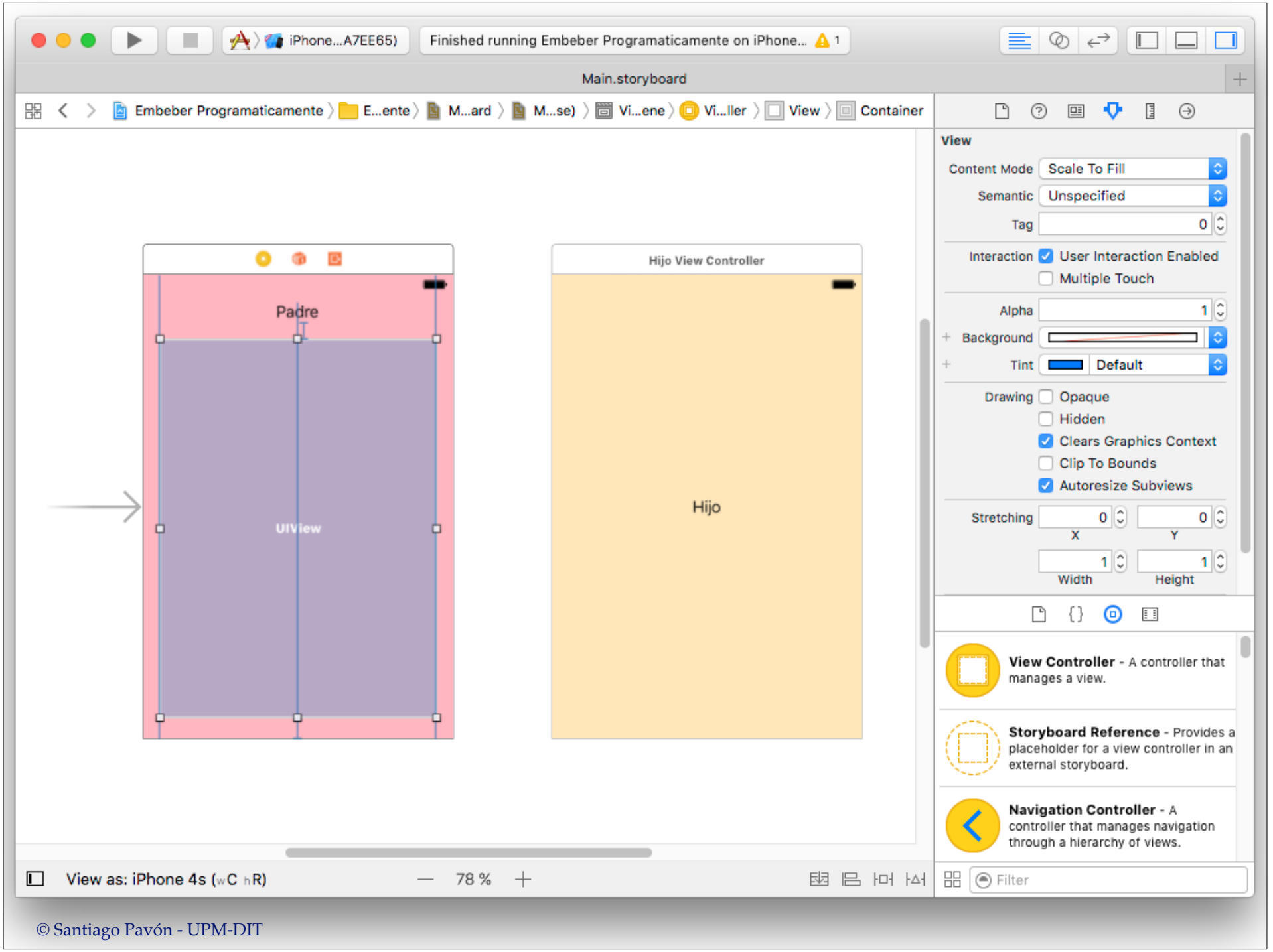
```
func setOverrideTraitCollection(_: UITraitCollection?,
                forChildViewController: UIViewController)
```

  - Especificar el tamaño preferido de la view de un View Controller:

```
var preferredContentSize: CGSize
```

  - Informar sobre cambios de tamaño:

```
func preferredContentSizeDidChange(
            forChildContentContainer: UIContentContainer)
```



```
class ViewController: UIViewController {

    @IBOutlet weak var container: UIView!

    var ivc: HijoViewController?

    override func viewDidLoad() {
        super.viewDidLoad()

        // Creo el objeto VC hijo:
        ivc = storyboard?.instantiateViewController(withIdentifier("Hijo VC")
                                                    as? HijoViewController
        if let ivc = self.ivc {

            // Construir jerarquia de views
            container.addSubview(ivc.view)

            // Construir jerarquia de View Controllers
            addChildViewController(ivc)
            ivc.didMove(toParentViewController: self)
        }
    }

    override func viewDidLayoutSubviews() {
        super.viewDidLayoutSubviews()

        // Ajusto geometrias
        ivc?.view.frame = container.bounds
    }
}
```