iPad + Aplicaciones (iPhone + iPad) Universales

Creación de aplicaciones MonoTouch que se ejecutan en el iPad y el iPhone

http://www.monocode.net

BREVE INTRODUCCIÓN

Esta es una guía final de la Introducción a la serie MonoTouch.Se procede a través de la creación de una aplicación para iPad, y examina como pueden ser creadas aplicaciones universales diseñadas para trabajar con ambos, el iPad y el Iphone. A continuación,prosigue a través de la creación de una aplicación universal con la plantilla de la aplicación del Proyecto Universal, y por último, se introduce una personalización de la plantilla que permite aplicaciones más complejas que se destinarán a ambos dispositivos.

Tiempo de desarrollo:

1 hora y 30 minutos.

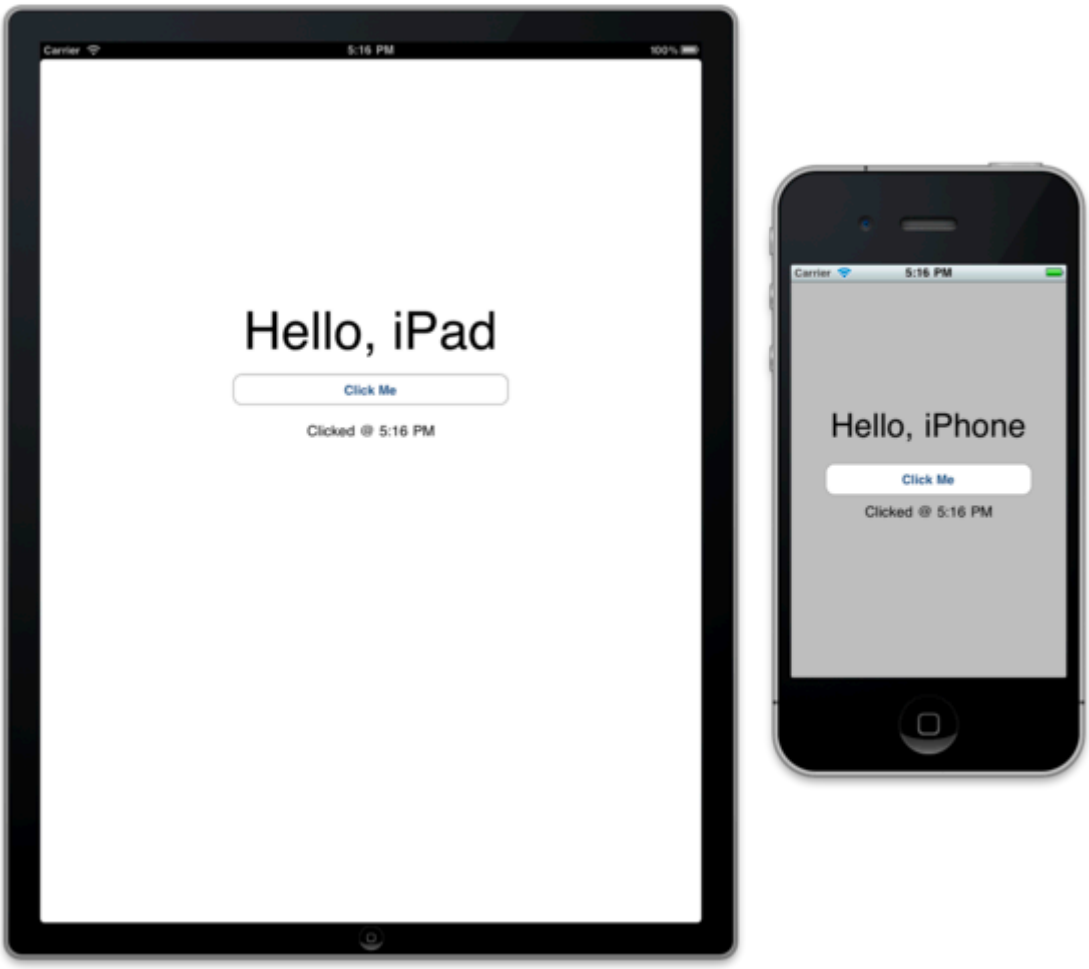
Código fuente del Ejemplo:

Hello, iPad
Hello, Aplicación Universal
Hello, Aplicación Compleja Universal

Resumen

Hemos cubierto mucho terreno hasta ahora en estos Primeros tutoriales. Hasta el momento, hemos introducido el conjunto de herramientas, creado e implementado aplicaciones, Outlets y acciones, incluso el patrón MVC. Sin embargo, todo lo que hemos hecho ha sido basado en el iPhone / iPod Touch. En este tutorial final de la serie de introducción, vamos a proceder paso a paso mediante la creación de una aplicación para iPad, a continutación, vamos a seguir a través de la creación de una aplicación universal que funciona igual de bien tanto en el iPhone como en el iPad

Las siguientes imágenes son de la primera aplicación universal vamos a construir:



Requisitos

Este tutorial se basa en los conceptos introducidos en los últimos Tutoriales de introducción. Por lo tanto, es muy recomendable que complete los Tutoriales de Inciaición del 1 al 4 antes de proseguir con este.

Construyendo para el iPad

Desde una perspectiva de aplicación, la programación para iPad es casi el mismo proceso que la programación para iPhone. Todo lo que hemos aprendido hasta ahora en estos tutoriales se aplica a las aplicaciones del iPad. El Patrón MVC todavía se aplica, y aunque en el IPAD hay sólo un par de controles adicionales, el resto de la UIKit también se sigue aplicando.

La principal diferencia entre las aplicaciones del iPhone y las aplicaciones de iPad es en el diseño de la interfaz de usuario, debido a que el IPAD presenta una forma mucho más grande debido a que es casi del tamaño de una hoja de papel común.

Tener este espacio de la pantalla significa la funcionalidad dividida en varias pantallas en el iPhone puede caber en un solo pantalla en el IPAD. Por ejemplo, echemos un vistazo a la configuración de la aplicación en el iPhone:

La misma funcionalidad se reune en un menor número de pantallas en el IPAD:

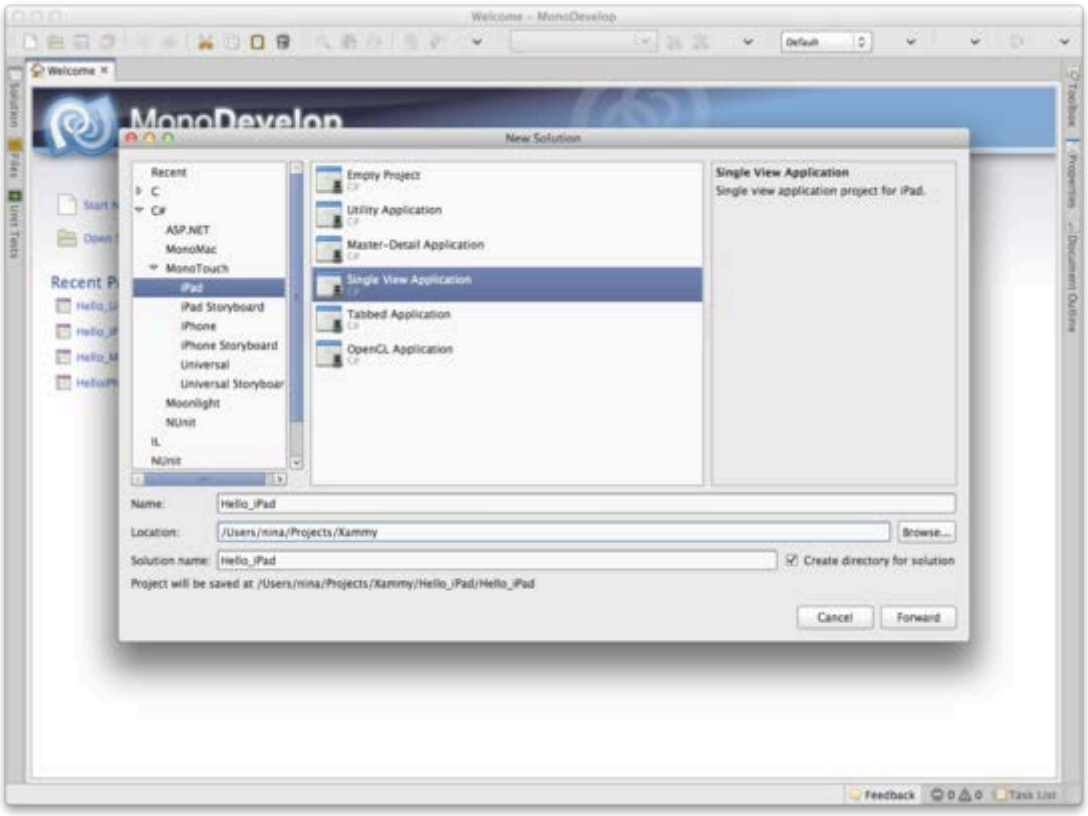
De hecho, con todo este espacio de pantalla, usted puede fácilmente imaginar cómo la configuración de la aplicación podría ser aún más compacta

Esta es la razón por la que aparecen controles adicionales para el IPAD. En la imagen anterior, la configuración de la aplicación utiliza el controlador de la vista dividida(split view), que sólo está disponible en el iPad.

OK, vamos a echar un vistazo a como crear realmente una aplicación para iPad.

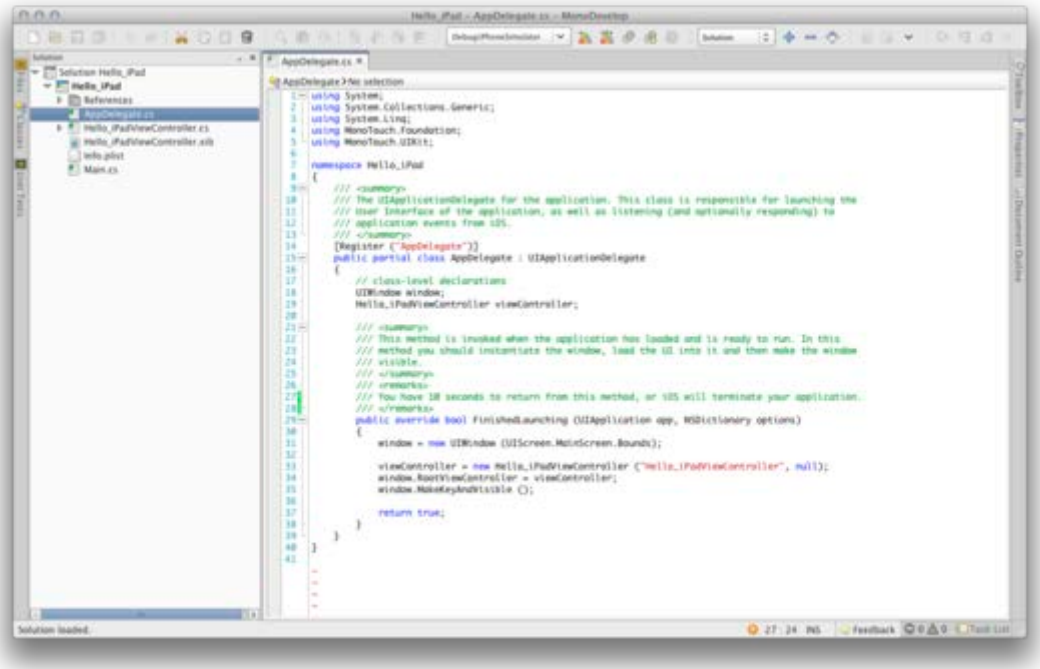
Creando nuestra aplicación para iPad

Abrir MonoDevelop (MD) y crear una nueva solución.En el cuadro de diálogo `New Solution` , seleccionar `C# : MonoTouch : iPad`. Puede comprobar que todas las plantillas para el iPhone también tienen una plantilla de iPad correspondiente:



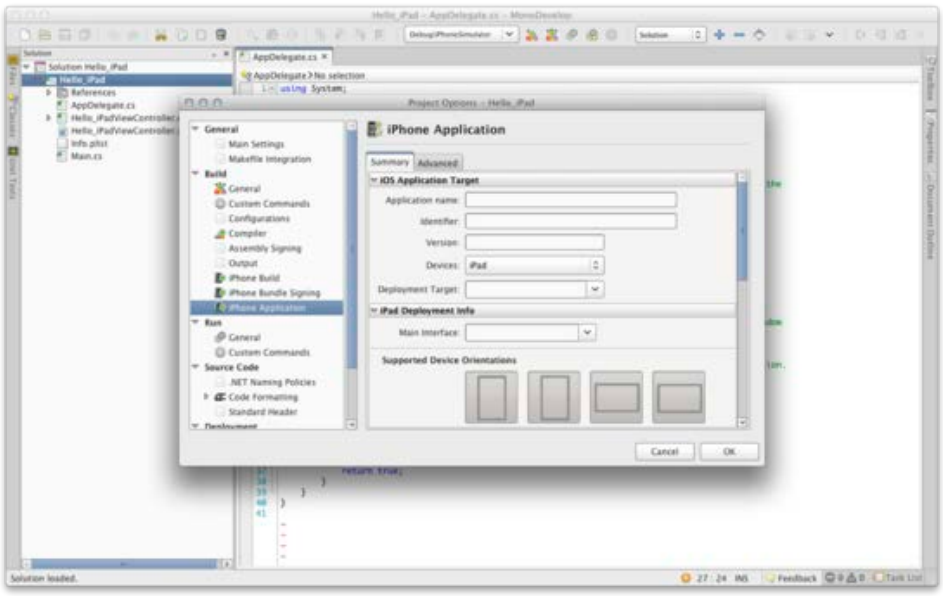
Vamos a hacer una aplicación muy sencilla, así que elige `MonoTouch Single View Application – iPad` y nombrala `Hello_iPad`.

El proyecto resultante es casi idéntico a la aplicación "Hola", la aplicación iPhone que hemos creado en una versión anterior del tutorial de introducción:



Opciones de Proyecto (Project Options)

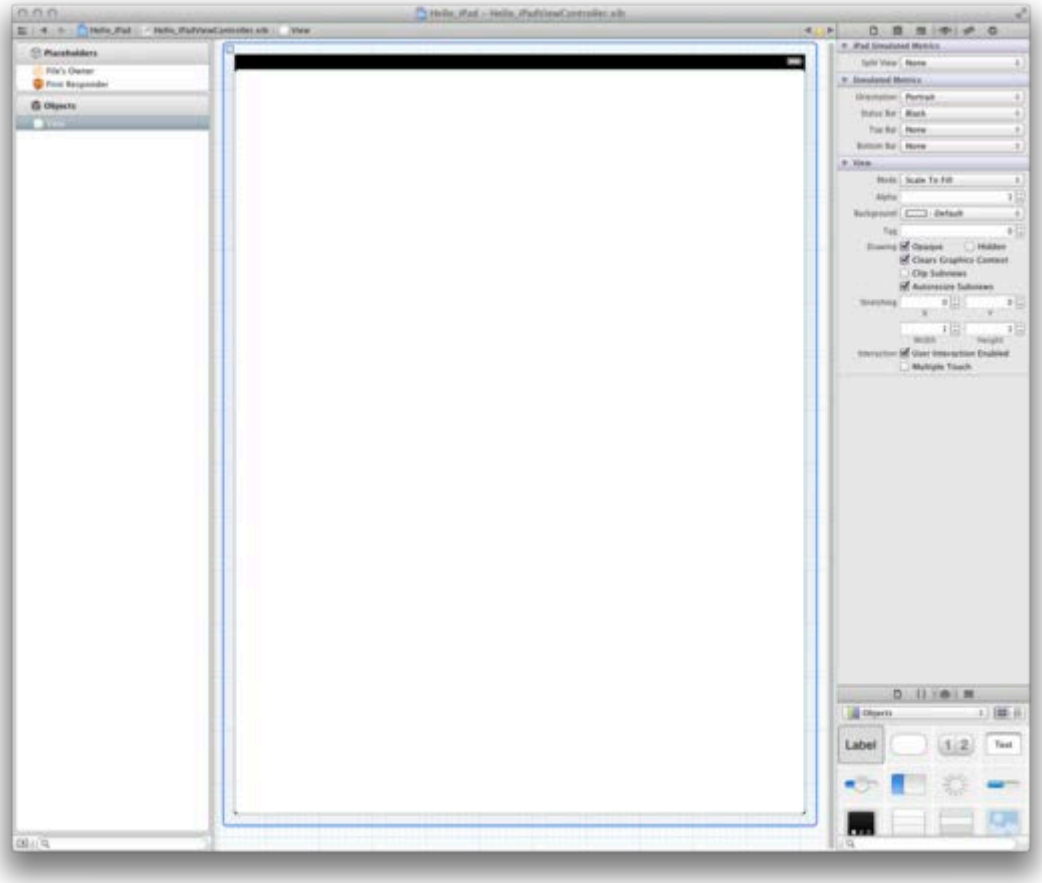
De hecho, la única diferencia real está en nuestro archivo .plist. Si hace clic derecho en el proyecto, y selecciona `Options`,vamos a ver que la configuración en `Devices` se establece en iPad:



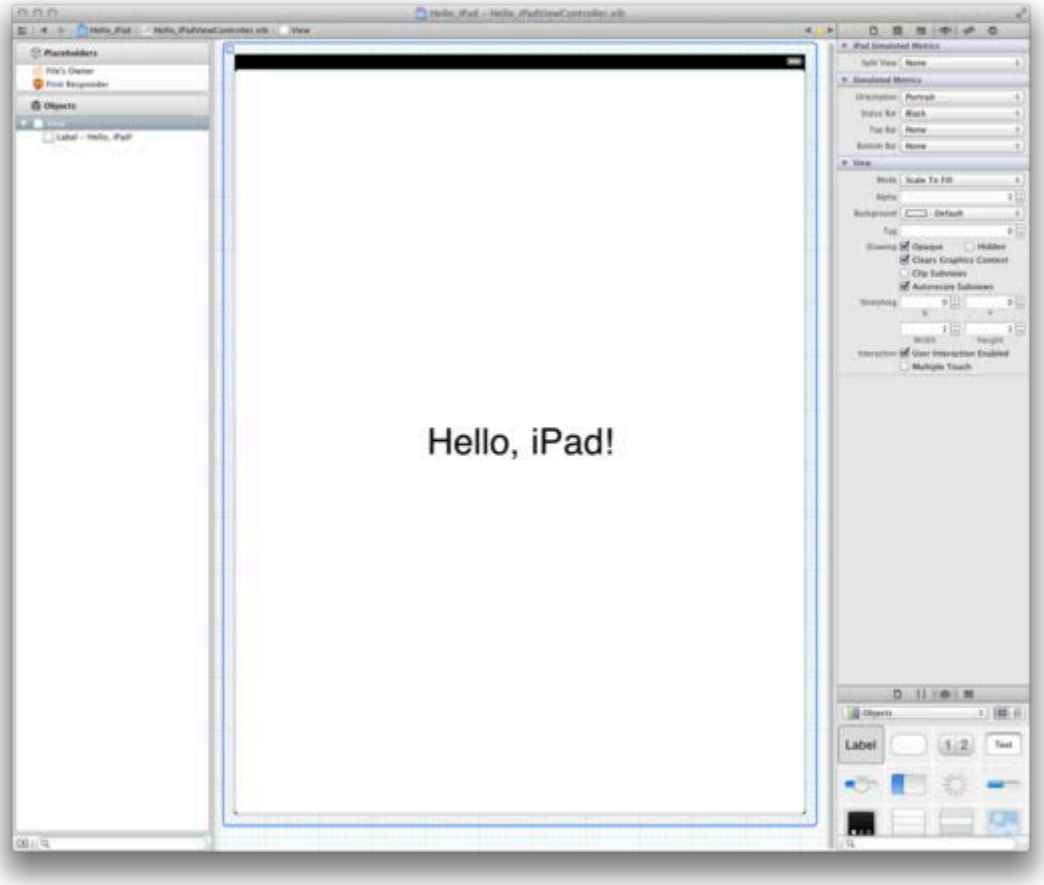
Al cambiar los ajustes de dispositivos para iPad, el diálogo también ofrece opciones específicas para el dispositivo.

El Constructor de Interfaz y las Vistas iPad

Editando los archivos .xib del iPad en el Consturctor de Interfaz (IB), es muy parecido a editar los archivos .xib del iPhone, excepto que el diseñador es mucho más grande.
Haga doble clic en `Hello_iPadViewController.xib`y se abrirá en el IB:



Vamos a agregar una etiqueta a la vista, y a guardar:



No necesitamos nada de Outlets o acciones, pero debe agregar esto es el mismo proceso que para el iPhone, se abre el Assistente del Editor, y arrastra desde el control al archivo .h .

Arrancando una aplicación iPad

Ahora vamos a generar y ejecutar nuestra aplicación para iPad. Si nos fijamos en `Project menu > iPhone Simulator Target`, se establece como `Default`, pero ya hemos establecido que el `Device` sea un iPad en el archivo .plist, el simulador de iOS se abrirá y simulará el iPad, que se ve casi igual que el simulador de iPhone, excepto por el tamaño:

¡Eso es! Hemos construido nuestra primera aplicación iPad MonoTouch!

Iconos iPad

Los iconos de la aplicación iPad se especifican en el archivo Info.plist, justo esta en el mismo lugar que los iconos de la aplicación iPhone. La unica diferencia el tamaño
necessario para cada uno. La siguiente tabla lista los iconos y tamaños necessarios
para la aplicación iPad:

Uso del Icono	Tamaño para iPad (en pixels)
Icono de Aplicación	72x72
Foco (Spotlight) y Ajustes (Settings)	Foco: 50x50 Ajustes: 29x29
Icono App Store (solo necesario si se va a publicar en la App Store)	512x512

Para agregar estos iconos, siga el mismo procedimiento que se indicó en la anterior aplicación iPhone "Hola": botón derecho en el proyecto y seleccionar Add : Add Files.

Para configurar estos iconos, haga doble-click en el archivo `Info.plist` en el Navegador de Proyecto, o doble-click en el proyecto i elija iPhone Application en el cuadro de dialogo de Opciones.

iPad Pantallas de carga por Defecto

Especificando un pantalla de carga en el iPad es conceptualmente lo mismo que con
aplicaciones para el iPhone, sólo hay que mover las imágenes correctamente con nombre y tamaño a la
raíz del proyecto, y el IOS automáticamente las mostrará mientras se pone en marcha. Sin embargo, las cosas son un poco más complicadas en las aplicaciones de iPad porque en lugar de dos imágenes (como las hay para el iPhone), puede opcionalmente especificar un máximo de cuatro, una para cada orientación.Además, a diferencia del iPhone, la barra de estado se puede ver durante el lanzamiento de las aplicaciones iPad, lo que significa que usted debe tener en cuenta que 20 píxeles en la parte superior de la pantalla ya están ocupados. Eso también depende de la orientación, debido a que la barra de estado está siempre en el

la parte superior. Por lo tanto, si el iPad está en modo de vertical, la imagen de carga tiene que ser 768x1004 píxeles, y si es en apaisado, necesita ser de 1024x748 píxeles.La siguiente tabla muestra los tipos de imágenes de carga del iPad, así como su tamaño (ancho x alto) y las orientaciones. Las imágenes se enumeran en orden de precedencia, las imágenes más abajo en la lista tienen prioridad sobre imágenes más altas en la lista:

Nombre de Archivo	Tamaño (en pixels)	Uso
Default~ipad.png	768x1004	Esta es la imagen básica de carga que se usa si otra imagen específica no esta disponible. Si no se proveen imagenes específicas esto se debería tener presente.
Default-Portrait~ipad.png	768x1004	Esta imagen se utiliza para la zona posterior derecha (botón de inicio en la parte inferior) en orientación vertical y toma prioridad sobre Default~ipad.png. Por lo tanto, si usted proporciona esta imagen, Default~ipad.png no se utiliza. Además, si Default-PortraitUpsideDown~ipad.png no se suministra, esta imagen tambien se utiliza.
Default-Landscape~ipad.png	1024x748	Esta es la imagen de carga genérica apaisada. Esta imagen tiene prioridad sobre Default~ipad.png. Por lo tanto, si usted proporciona esta imagen, Default~ipad.png no se utiliza. Además, si Default-LandscapeLeft~ipad.png o Default-LandscapeRight~ipad.png no es suministrada, tambien se usa esta imagen.
Default-PortraitUpsideDown~ipad.png	768x1004	Esta es la imagen de carga en vertical boca abajo(Botón de inicio en la parte superior).
Default-LandscapeLeft~ipad.png	1024x748	Esta es la imagen de carga izquierda en apaisado (botón inicio a la izq.).
Default-LandscapeRight~ipad.png	1024x748	Esta es la imagen de carga derecha en apaisado(botón inicio a la derecha).

A menos que quieras tener imágenes diferentes para cada orientación, sólo es necesario suministrar Default~ipad.png si su aplicación sólo es compatible con el modo retrato. Si su aplicación sólo es compatible con la orientación horizontal, sólo

necesita la imagen Default-Landscape~ipad.png. Si usted soporta tanto orientaciones horizontal como vertical, entonces debe proporcionar por lo menos estas dos imágenes. Esto reduce significativamente la cantidad de imágenes que debe crear,suministrar una para cada orientación suele ser excesivo.

Para obtener más información acerca de la carga de imágenes, consulte la documentación de Apple en Guia de Iconos personalizados y directrices de creación de imagen(Custom Icon and Image Creation Guidelines).

Bien, ahora que hemos cubierto las aplicaciones del iPad, echemos un vistazo a cómo crear aplicaciones que se dirigen tanto al iPhone como al IPAD.

Aplicaciones Universales

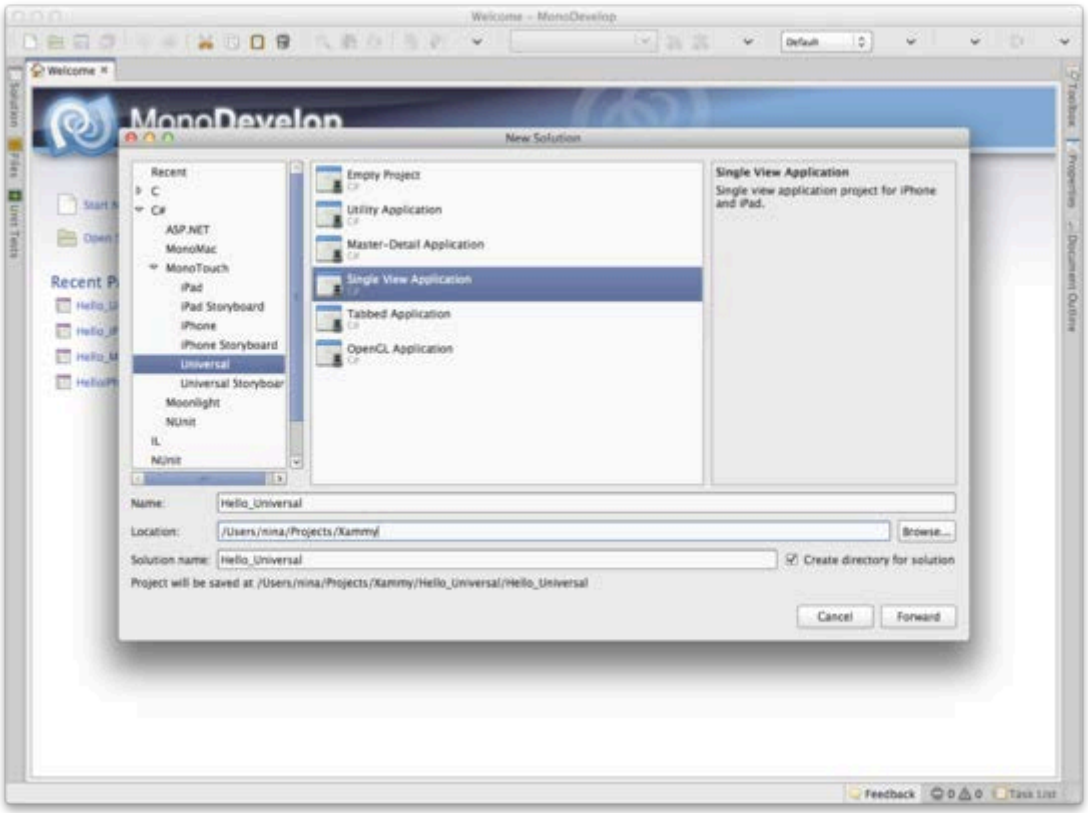
Las aplicaciones iOS Universales son las que se cargan en el interfaz de usuario apropiado, en función del dispositivo en que se ejecuten. Por ejemplo, podría tener una interfaz de usuario definido para el iPhone, y otro interfaz de usuario definido para el iPad. Una aplicación universal cargará automáticamente la correcta Interfaz de usuario.

Esto se realiza generalmente cuando el método `FinishedLaunching` del delegado de la aplicación detecta el dispositivo en que se está ejecutando, y luego lanza la interfaz de usuario adecuada.

Afortunadamente, MonoDevelop viene con una plantilla de proyecto universal que contiene el código básico para lograr esto. Echemos un vistazo.

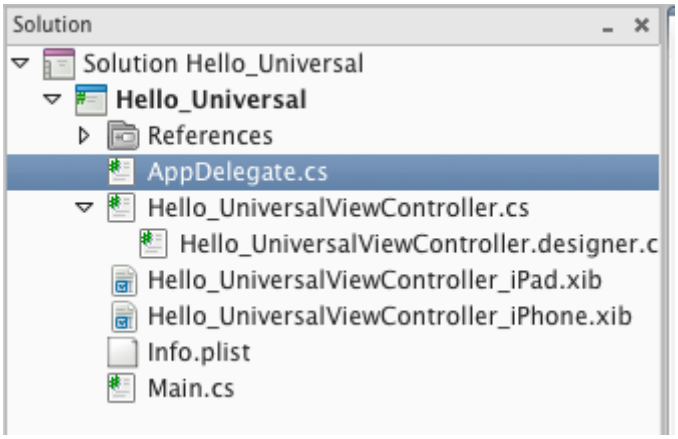
Plantilla de Proyecto Universal

Vamos a crear una nueva solución, esta vez vamos a elegir C# : MonoTouch : Universal : MonoTouch Single View Application – Universal:



Vamos a nombrar el proyecto `Hello_Universal`.

Si echamos un vistazo a la estructura del proyecto creado a partir de esta plantilla, se dará cuenta de algo un poco diferente. Hay una vista controlador, pero dos archivos **.xib** asociados:



Detección de Dispositivo desde código

Para entender cómo funciona esto, echemos un vistazo al método `FinishedLaunching` en la clase `AppDelegate`:

```
>public override bool FinishedLaunching (UIApplication app,
NSDictionary options)

{

// crear una nueva instancia de la ventana  basada en el tamaño de la
pantalla

window = new UIWindow (UIScreen.MainScreen.Bounds);

if (UIDevice.CurrentDevice.UserInterfaceIdiom ==
UIUserInterfaceIdiom.Phone) {

    viewController = new Hola_ViewControllerUniversal

    ("Hola_ViewControllerUniversal_iPhone", null);

} else {

    viewController = new Hola_ViewControllerUniversal

    ("Hola_ViewControllerUniversal_iPad", null);

}

window.RootViewController = viewController;

window.MakeKeyAndVisible ();

return true;

}
```

La magia comienza aquí, con esta línea:

```
>(UIDevice.CurrentDevice.UserInterfaceIdiom ==
UIUserInterfaceIdiom.Phone) { …
```

El objeto `UIDevice.CurrentDevice` expone una propiedad llamada `UserInterfaceIdiom` que devuelve el enumerado `UIUserInterfaceIdiom` que nos permite saber el dispositivo actual. Hay dos posibles valores de esta

enumeración: `Pad` y `Phone`, correspondiendo a iPad y iPhone/iPod Touch, respectivamente.

Una vez que hemos determinado en qué dispositivo se está ejecutando la aplicación, a continuación, crea una instancia de nuestro controlador de vista, y pasa el nombre de archivo **.xib** apropiado para crear una instancia:

```
>if (UIDevice.CurrentDevice.UserInterfaceIdiom ==
UIUserInterfaceIdiom.Phone) {

       viewController = new Hello_UniversalViewController

             ("Hello_UniversalViewController_iPhone", null);

} else {

       viewController = new Hello_UniversalViewController

             ("Hello_UniversalViewController_iPad", null);

}
```

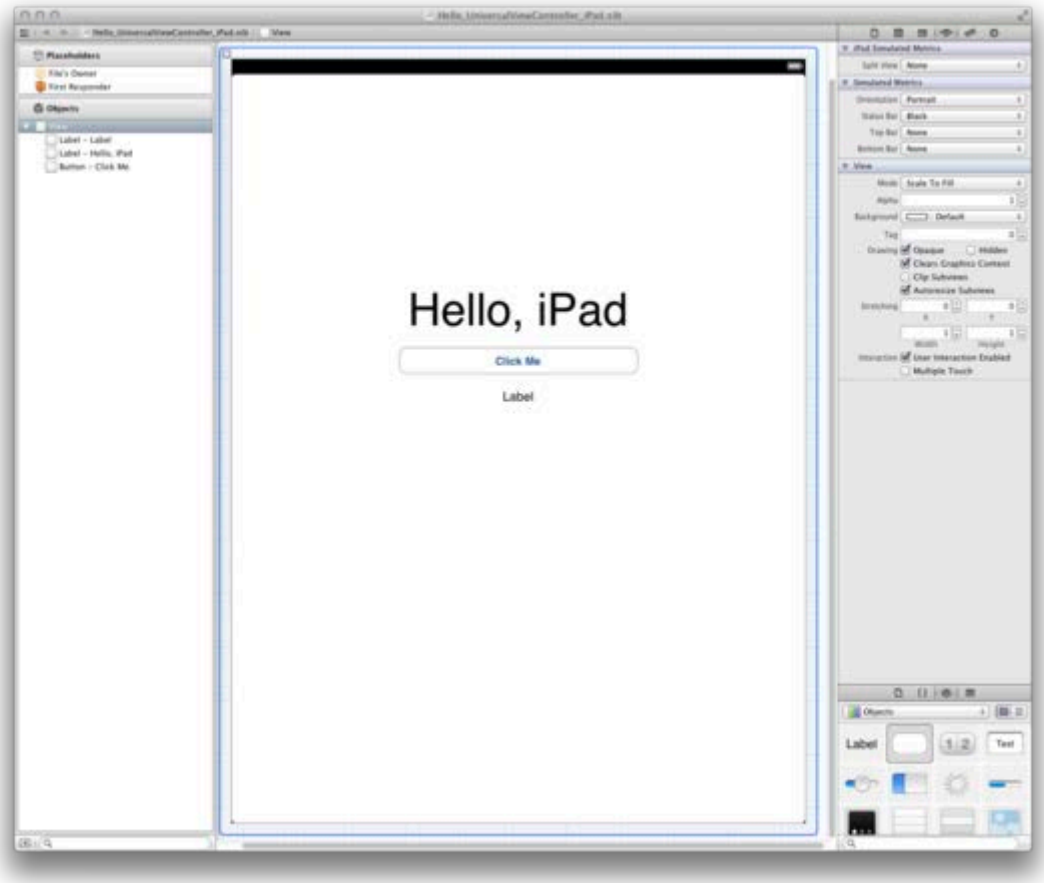
Esto funciona, porque si nos fijamos en el constructor del controlador, podemos ver lo siguiente:

```
>public Hello_UniversalViewController (string nibName, NSBundle
bundle) : base (nibName, bundle)
```

La clase base cargará el archivo **.xib** basado en el nombre (rellamada desde el *tutorial nº 3 del Inicio* donde **Nib** es otro nombre para los archivos **.xib**). En este caso, estamos pasando ya sea `Hello_UniversalViewController_iPhone` , o `Hello_UniversalViewController_iPad` , dependiendo del dispositivo.<

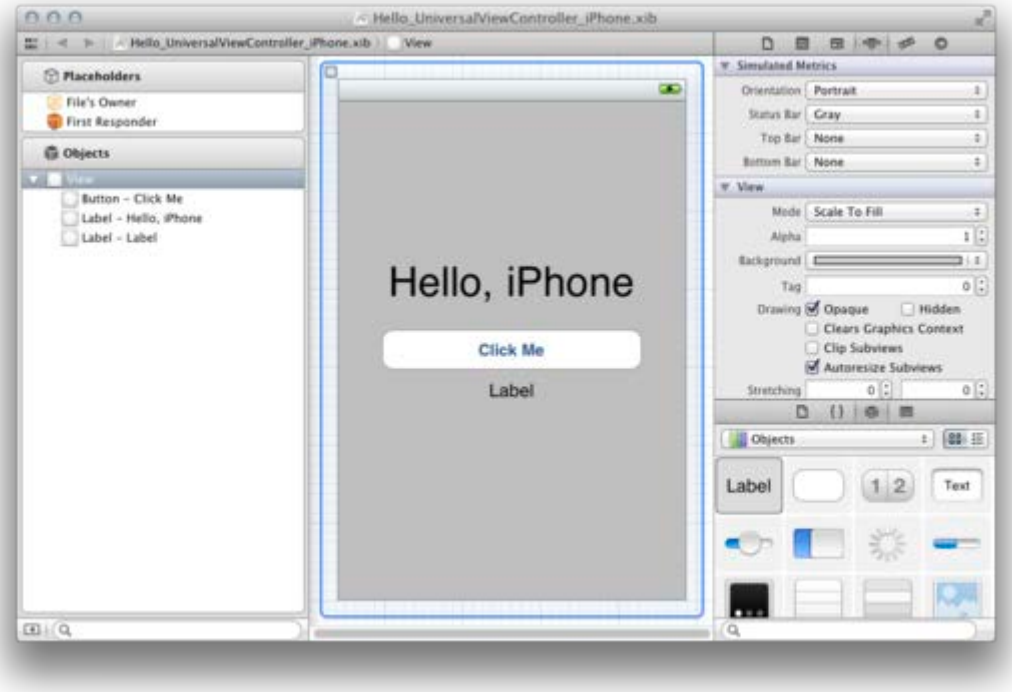
Creando el Interfaz de Usuario

Hasta ahora, todo va bien, pero vamos a poner esto en acción. En primer lugar, vamos a abrir el archivo `Hello_UniversalViewController_iPad.xib` en Xcode y añadir un par de etiquetas y un botón:



Vamos a actualizar la etiqueta de fondo con un poco de texto cuando se hace click sobre el botón,así que asegúrese de que es bastante amplio, para que el texto no quede truncado.

A continuación, vamos a hacer lo mismo
con `Hello_UniversalViewController_iPhone.xib` :



Una vez más, asegúrese de que la etiqueta de la parte inferior sea bastante amplia para que el texto no se vea truncado.

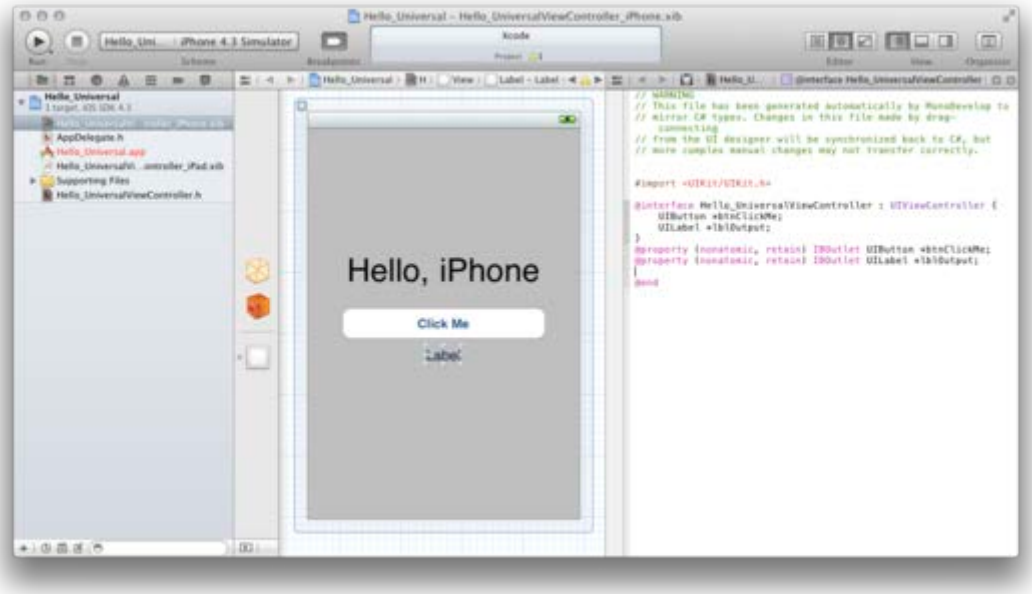
Compartir salidas y acciones a través de múltiples vistas

Así que ahora tenemos dos vistas que comparten el mismo controlador, pero ¿cómo
creamos salidas y acciones que funcionen a través de las dos vistas? Bueno, resulta que esto es muy similar a cómo se comparten las acciones.Primero vamos a crear nuestras salidas. Abre la vista iPhone y el correspondiente
archivo .h como controlador compartido en Xcode y arrastre el Control para
crear las siguientes salidas:

- btnClickMe
- lblOutput

The `btnClickMe` outlet should be wired to the button, and the `lblOutput` should be wired to the lower label. When we're done we should have something like this:

La salida `btnClickMe` debe ser conectada al botón, y `lblOutput` se debe conectar a la etiqueta inferior. Cuando hayamos terminado deberíamos tener algo como esto:



Nuestro archivo .h debe tener el siguiente código de salida:

```
>@property (nonatomic, retain) IBOutlet UIButton *btnClickMe;



@property (nonatomic, retain) IBOutlet UILabel *lblOutput;
```

A continuación, vamos a abrir la vista iPad. Esta vez, vamos a arrastrar el control a las salidas existentes que hemos creado cuando la vista iPhone estaba abierta:

Ahora nuestras salidas se comparten entre los dos Archivos **xib**. Vamos a salvar nuestro
trabajo y volver a MonoDevelop. Ahora podemos usar la misma salida de la misma manera, tanto sea si la aplicación se ejecuta en un iPhone o en un iPad. Vamos a modificar nuestro controlador `Hello_UniversalViewController` de la clase de viewDidLoad
para ver esto en acción:

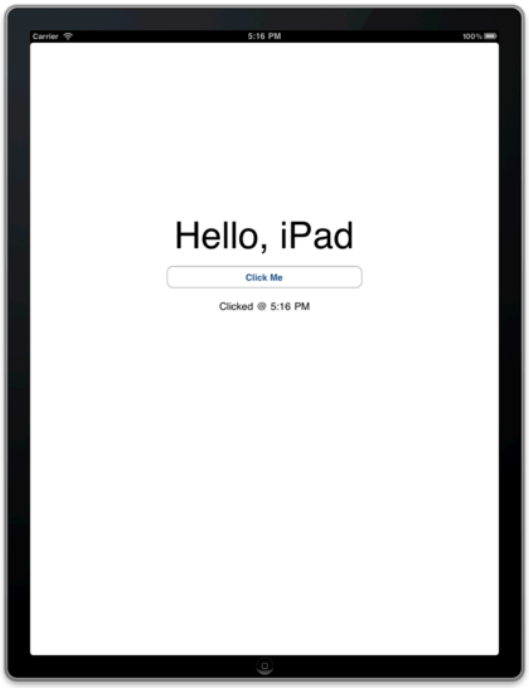
```
>public override void ViewDidLoad ()

{

    base.ViewDidLoad ();

    this.btnClickMe.TouchUpInside += (sender, e) => {

        this.lblOutput.Text = "Clicked @ " +

        DateTime.Now.ToShortTimeString();

        };



}
```

Ahora, cuando se ejecute y haga clic en el botón, deberíamos ver algo como esto:

Tenga en cuenta que el simulador de iPhone se lanza por defecto. Podemos forzar que la aplicación se ejecute en el simulador de iPad mediante la selección `iPad Simulator 4.3` del menú **Project > iPhone Simulator Target**

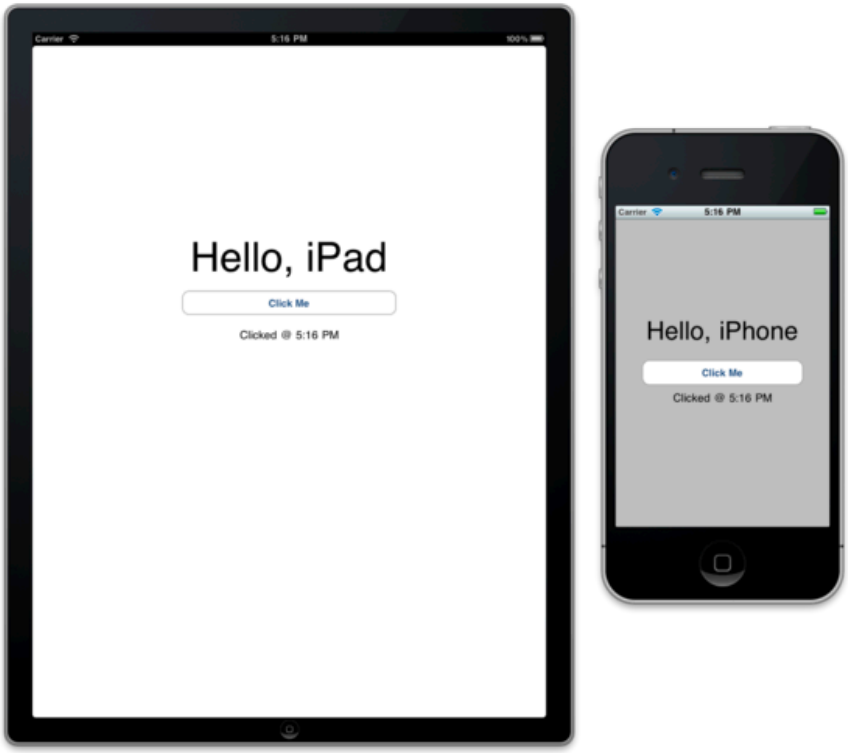
.

Ahora cuando se ejecute la aplicación se pondrá en marcha en el simulador iPad, y cuando se haga clic en el botón, el código se ejecutará tal y como lo hizo cuando se utilizó el simulador de iPhone:



Aplicaciones Universales Complejas

El enfoque que hemos tomado hasta ahora, donde un solo controlador gestiona dos
diferentes vistas basadas en el dispositivo, funciona bien para las pantallas simples de las
con las mismas salidas, acciones e interacciones entre los usuarios en la pantalla del iPhone y la pantalla del iPad, pero esto no es lo habitual en el mundo real. Por lo general, las aplicaciones de iPad y aplicaciones para el iPhone
difieren sustancialmente en términos de interfaz de usuario y interacción con el usuario (denominado Usuario eXperience [UX] cuando se habla de ellos en conjunto). Por ejemplo, consideremos las siguientes dos interfaces diferentes para la misma aplicación en diferentes dispositivos:



En este caso, nos gustaría tener dos controladores diferentes, porque las Salidas serían diferentes. Esto es obviamente una interfaz de usuario artificial, pero que puede imaginar que en algunas aplicaciones, la interfaz de usuario es radicalmente diferente entre los dispositivos. Para cargar un controlador diferente en función del dispositivo, lo primero que debemos hacer es declarar
un `UIViewController` genérico en nuestra clase `AppDelegate`, en vez de uno específico (por ejemplo `homeScreen_iPad` o`homeScreen_iPhone`):

```
> public partial class AppDelegate : UIApplicationDelegate

 {

 // nivel de declaraciones de clase

    UIWindow window;

    UIViewController homeScreen;

    …

 }
```

Luego, en el método `FinishedLaunching` , cuando se determina que dispositivo está en uso, cargamos el controlador apropiado y, a continuación establece que controlador es nuestro root:

```
if (UIDevice.CurrentDevice.UserInterfaceIdiom ==
UIUserInterfaceIdiom.Phone)

{

     homeScreen = new Screens.HomeScreen_iPhone();

}

else

{

    homeScreen = new Screens.HomeScreen_iPad();

}

window.RootViewController = homeScreen;
```

Ahora, cuando ejecute la aplicación, se mostrará el controlador adecuado para el dispositivo.

Sumario

Enhorabuena, este fue el tutorial final de la serie de Introducción!
En este tutorial hemos aprendido cómo crear una aplicación solo para iPad,
así como dos enfoques diferentes para la creación de aplicaciones universales.
Si usted ha seguido estos tutoriales de principio a fin, está usted en camino
de convertirse en un competente desarrollador de MonoTouch iOS!