iOS

Prof. Emanuele Papotto

Bibliografia

- Francesco Fanti – Html.it

Introduzione ad iOS

- Cosa faremo?
  - Sviluppare applicazioni su **piattaforma iOS** compatibili con i dispositivi di casa Apple: iPhone, iPad e iPod Touch.

Introduzione ad iOS

- Cos'è?
  - iOS, come anche il diretto concorrente *Android*, è un sistema operativo molto giovane, che vede la luce il 9 Gennaio 2007 durante la presentazione del primo modello di iPhone e che, negli anni successivi, è andato ad equipaggiare tutti gli altri dispositivi *mobile* di Apple come l'iPod Touch e l'iPad.
- Dopo la presentazione dell'**AppStore** (il negozio virtuale nel quale acquistare le applicazioni per il proprio dispositivo) ed il rilascio dell'**SDK ufficiale per la realizzazione di applicazioni**, il sistema operativo ha immediatamente suscitato molto interesse sia verso singoli sviluppatori sia verso le grandi software-house.
- Le applicazioni per iOS hanno subito un incremento esponenziale e una diffusione grandissima, dovuta principalmente alla grande popolarità (e soprattutto alle enormi vendite) dei dispositivi Apple.

Qualche numero…del 2011

- Più di 250 milioni di dispositivi venduti
- 367 apple store in 11 paesi
- l'iPhone rappresenta il 5% del totale dei cellulari nel mondo
- 3 tablet su 4 venduti in USA sono iPad
- 500.000 app presenti nell'AppStore tra cui 140 mila per iPad
- 18 miliardi di App sono state scaricate dalla nascita dell'App Store, ora la frequenza è di circa 1 miliardo al mese
- Apple ha pagato agli sviluppatori 3 miliardi di dollari
- Apple ha venduto 300 milioni di ipod di cui 45 milioni nell'ultimo anno
- l'iPod Touch è il numero uno tra i dispositivi portatili di gioco
- sono presenti più di 20 milioni di brani in iTunes il quale rimane il primo music store del mondo con oltre 16 miliardi di brani scaricati
- Apple con l'iPhone 4s ha avuto un incremento di vendite dal 26% al 43% riducendo il divario con Android che è in vetta.

Perché sviluppare con iOS

- Una grande diffusione: iOS equipaggia più di 250 milioni tra iPhone, iPod e iPad.
- A differenza di Android non c'è frammentazione nei dispositivi: ogni anno viene rilasciato un solo iPhone, un solo iPad ed un solo iPod Touch, quindi sarà molto più agevole produrre applicazioni ottimizzate.
- Un ambiente di sviluppo e una documentazione dell'SDK di ottimo livello.
- Un costo accessibile per l'iscrizione al programma sviluppatori (iOS Developer Program).

Objective-C, Xcode, Interface Builder e Simulatore

- Objective-C
  - Il linguaggio di programmazione che è necessario conoscere per la realizzazione di applicazioni iOS è l'Objective-C che è un linguaggio orientato agli oggetti e di fatto rappresenta un'estensione del linguaggio C. Essendone un'estensione, l'Objective-C, mantiene una compatibilità totale con i costrutti utilizzati nel linguaggio C.
  - E' la base da apprendere per utilizzare le librerie(framework) che Apple mette a disposizione e che consentono lo sviluppo di applicazioni su:
    - OSX
    - iPhone
    - iPodTouch

Objective C

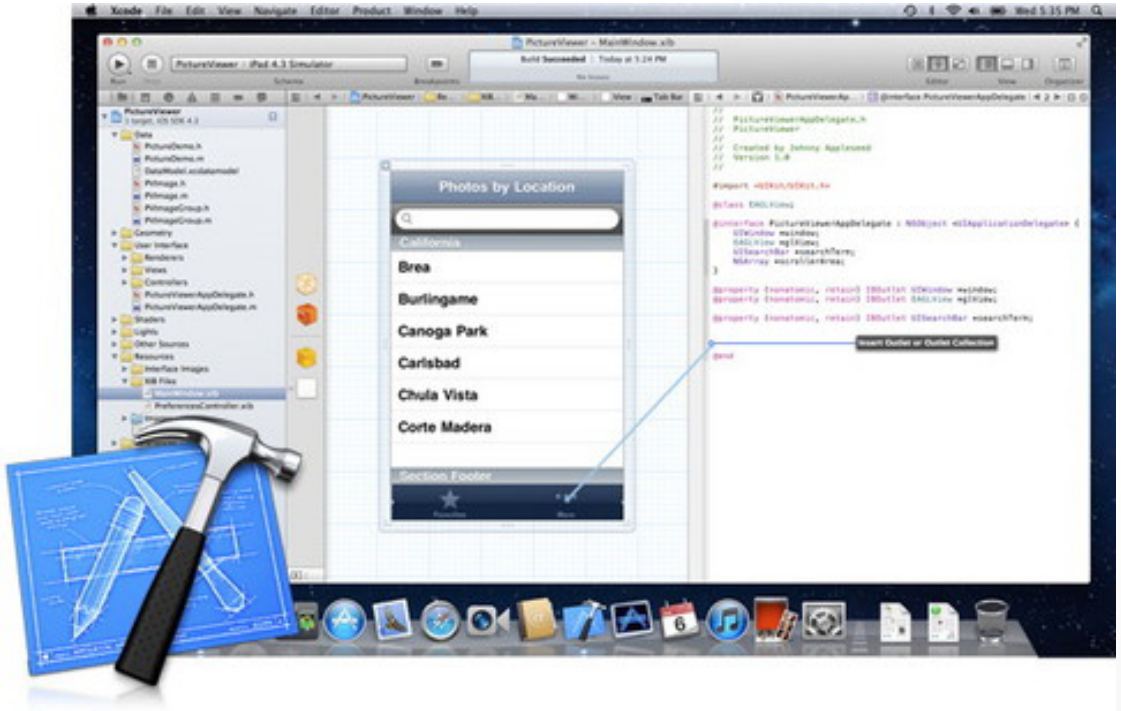
- E' un linguaggio di programmazione ad oggetti (OOP) molto simile a altri linguaggi come Java o C++.
- CLASSI e OGGETTI sono elementi astratti che permettono di rappresentare oggetti reali (o immaginari) in oggetti software.
- Ogni Oggetto appartiene a una Classe e i nostri programmi saranno un insieme di oggetti che dialogano tra loro inviandosi messaggi e ricevendo risposte attraverso metodi e funzioni.

Objective C

- In Objective-C per definire un oggetto, quindi la classe cui esso appartiene, abbiamo bisogno di due file:
  - uno (.h) che definisce l'interfaccia della classe
  - l'altro (.m) che ne definisce l'implementazione
- L'interfaccia descrive le azioni (i metodi e funzioni) della classe e nasconde l'implementazione che definisce il codice vero e proprio, ovvero ciò che le azioni realmente eseguono.

Xcode

- XCode è l'IDE (Integrated development environment) di sviluppo che viene offerto agli sviluppatori per la realizzazione delle proprie applicazioni. L'interfaccia del software è abbastanza chiara e intuitiva e offre notevoli funzionalità (come l'efficientissimo completamento automatico) che permettono di ridurre sensibilmente i tempi di sviluppo.



Interface Builder

- Interface Builder è un tool integrato all'interno di Xcode, che viene usato per la realizzazione delle interfacce grafiche, generando file .xib. L'utilizzo del tool è immediato: è sufficiente trascinare all'interno di un'area (che rappresenta una vista dell'applicazione) gli elementi grafici che si vogliono utilizzare (come bottoni, campi di testo, immagini) per poi passare al loro posizionamento.

Simulatore

- Ultimo strumento che andremo ad utilizzare è il simulatore che ci consente di eseguire l'applicazione realizzata direttamente sul nostro Mac senza necessariamente disporre di un dispositivo fisico come iPhone, iPod Touch o iPad..
- Il simulatore, virtualizza il comportamento del dispositivo abbastanza fedelmente, ma utilizza l'hardware offerto dal nostro Mac e molte funzionalità come il GPS o le Notifiche Push non sono presenti su di esso.

Registrazione al developer program e download di Xcode

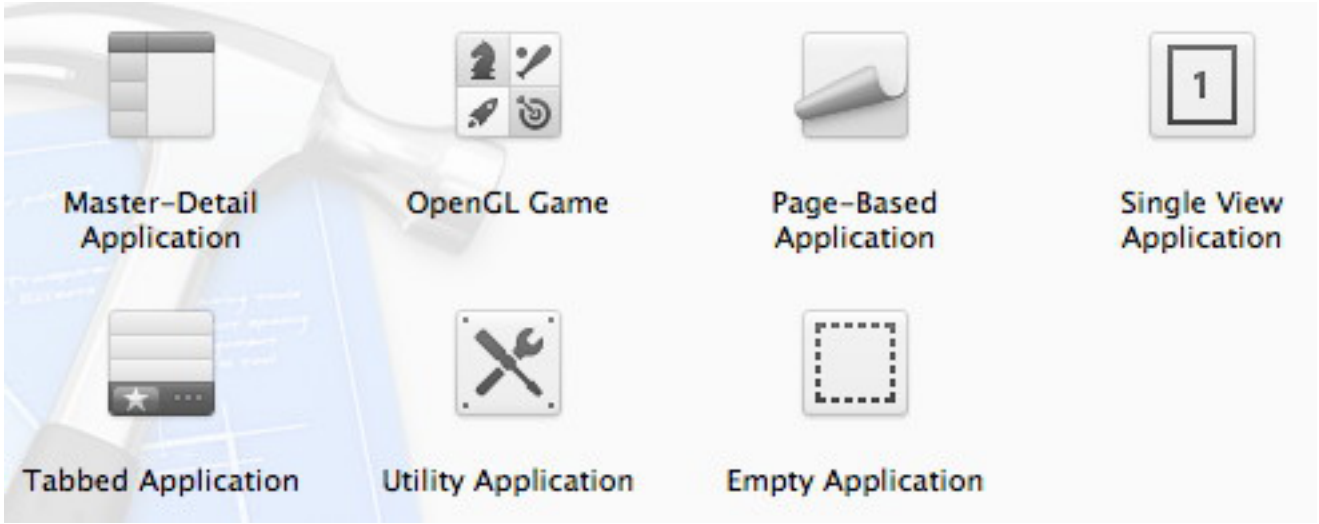
- Prima di poter iniziare a sviluppare applicazioni per iOS, dobbiamo eseguire due operazioni preliminari: iscriverci al developer program e scaricare Xcode.
- Il riferimento principale per ogni sviluppatore è la sezione dedicata ai developer dalla quale è possibile accedere alle varie sezioni dedicate agli sviluppatori iOS, Mac e Safari.
- L'iscrizione al developer program ha un costo di 79 Euro e una validità di un anno. La registrazione al developer program è necessaria per poter svolgere due operazioni: installare l'applicazione sul proprio dispositivo fisico e pubblicare le proprie applicazioni sull'AppStore.

Xcode: descrizione dei template offerti

- Per prima cosa apriamo Xcode e scegliamo l'opzione Create a new Xcode project.

Xcode

- Ci troviamo di fronte ad una serie di template offerti dai quali potremo partire per sviluppare la nostra applicazione.



Xcode: i template

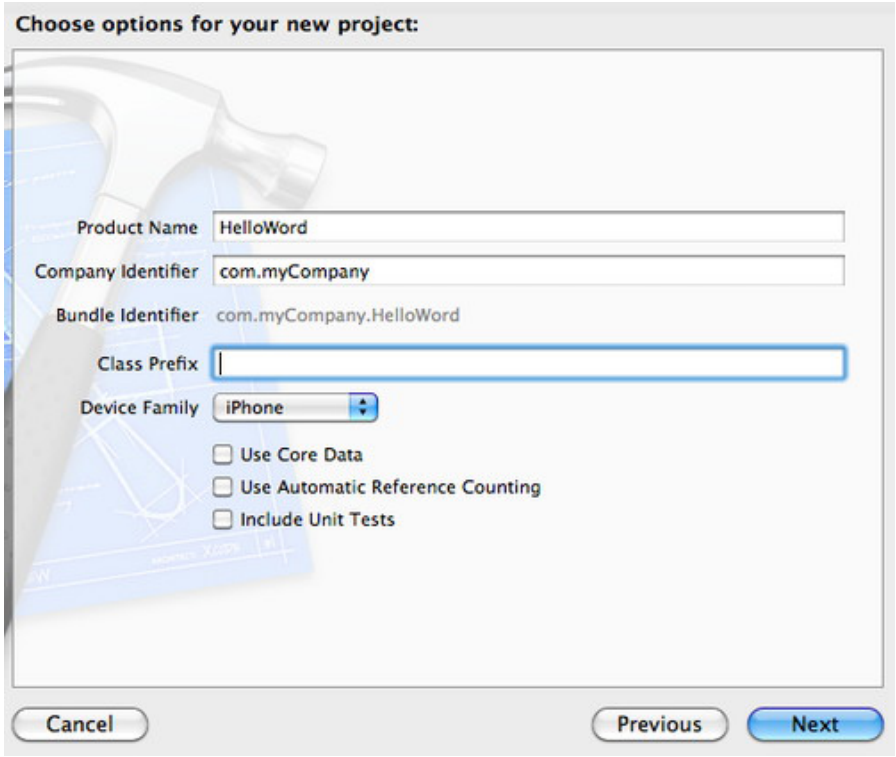
- **Master-Detail Application:** questo template è utilizzabile solo su iPad e offre la classica interfaccia suddivisa in due aree; a sinistra un *View Controller* (Master) utilizzato normalmente per la ricerca/consultazioni di dati ed un secondo *View Controller* (Detail) nella parte destra utilizzato per la visualizzazione dettagliata del contenuto scelto nel Master.
- **Single View Application:** questo template può essere usato nel caso in cui la nostra applicazione faccia uso di una singola *view*. Avremo infatti (oltre all'AppDelegate e alla Window, che spiegheremo tra poco) un *View Controller* per la gestione della *view* e il suo file *.xib* per la gestione dell'interfaccia grafica.

Xcode: i template

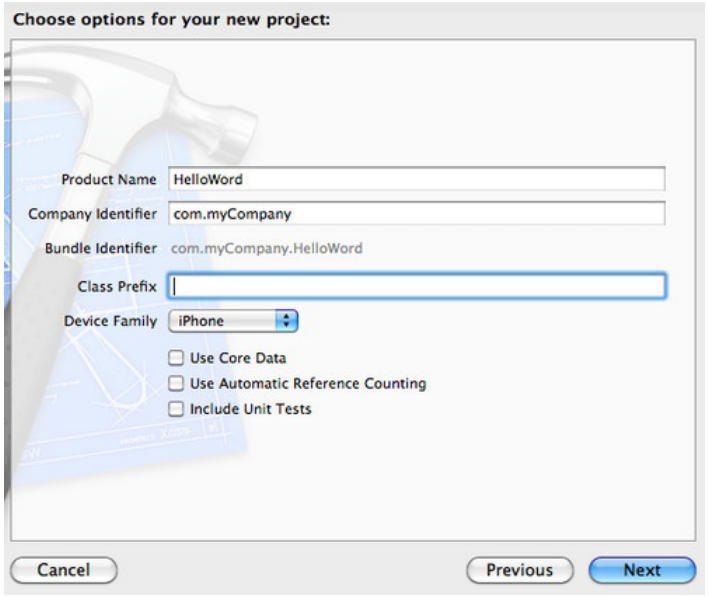
- **Tabbed Application:** questo template offre un punto di partenza per la realizzazione di un'applicazione basata sull'oggetto *UITabBarController* (la barra di selezione collocata nella parte bassa dello schermo in moltissime applicazioni iOS).
- **Empty Application:** questo template è il più semplice di tutti in quanto crea solamente l'AppDelegate della nostra applicazione e la Window. Anche se in questo primo momento può risultare di scarso interesse, questo è in realtà il template migliore da cui partire perché ci lascia completa libertà sulla realizzazione dell'applicazione. Inoltre, ci slega completamente dai file *.xib* e dunque dall'utilizzo di Interface Builder.

Creazione del primo progetto e interfaccia di Xcode

- Dal menu dei template selezioniamo il template **Empty application** e clicchiamo su *next*. Ci apparirà una nuova schermata nella quale ci verranno richieste alcune informazioni per la creazione del nuovo progetto:



Primo progetto



- Gli unici campi che momentaneamente ci interessano sono:
- **Product Name**: rappresenta il nome che vogliamo dare al progetto.
- **Company Identifier**: è un campo che risulta necessario all'atto della pubblicazione dell'applicazione su *AppStore*. Per il momento possiamo inserire, come *Company identifier*, il testo mostrato nello screenshot.
- **Device Family**: indica il dispositivo sul quale l'applicazione verrà eseguita.
- Detto questo, clicchiamo su *next*, decidiamo dove collocare il nostro progetto e clicchiamo su *create*.

Schermata principale di Xcode

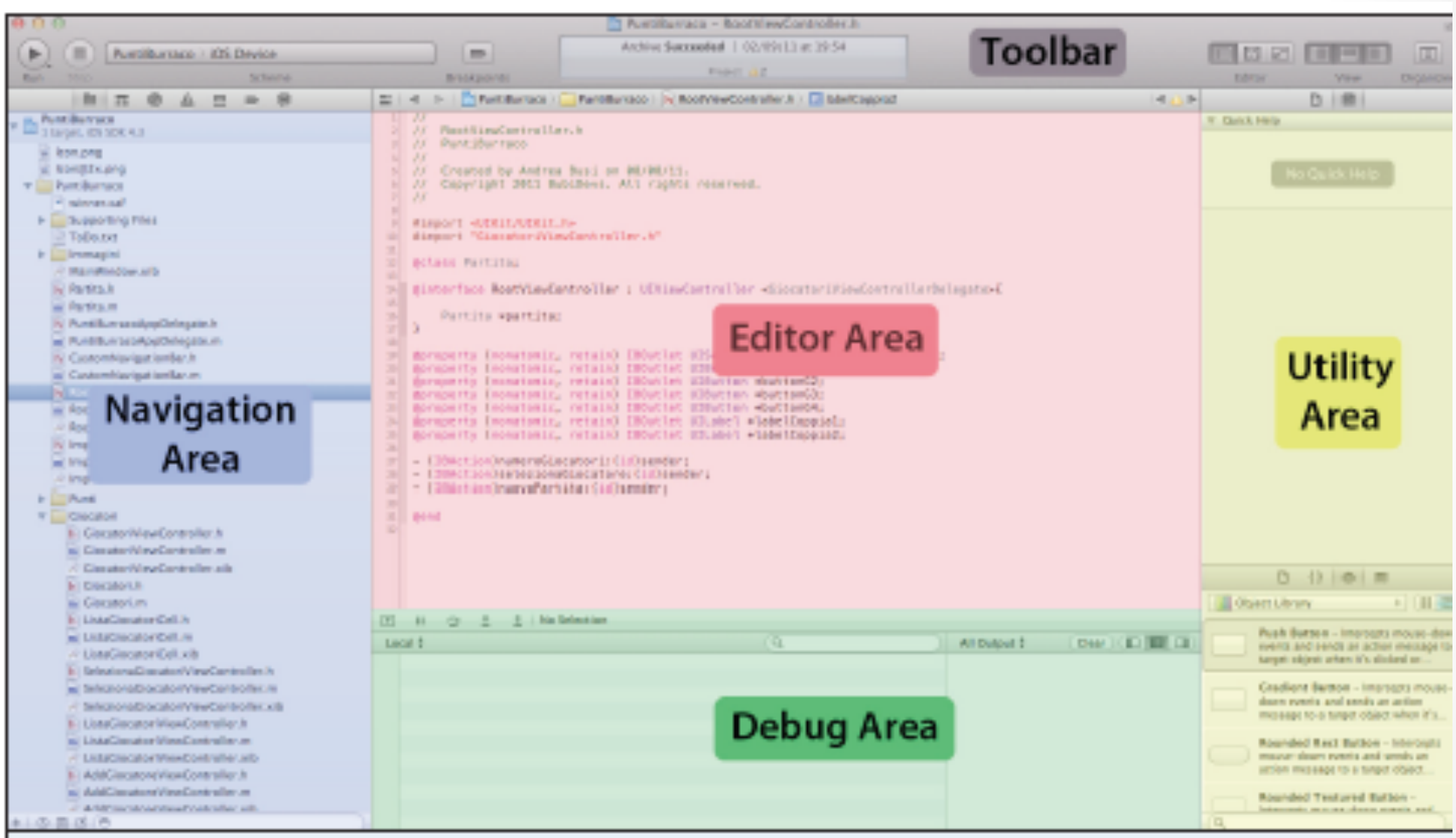
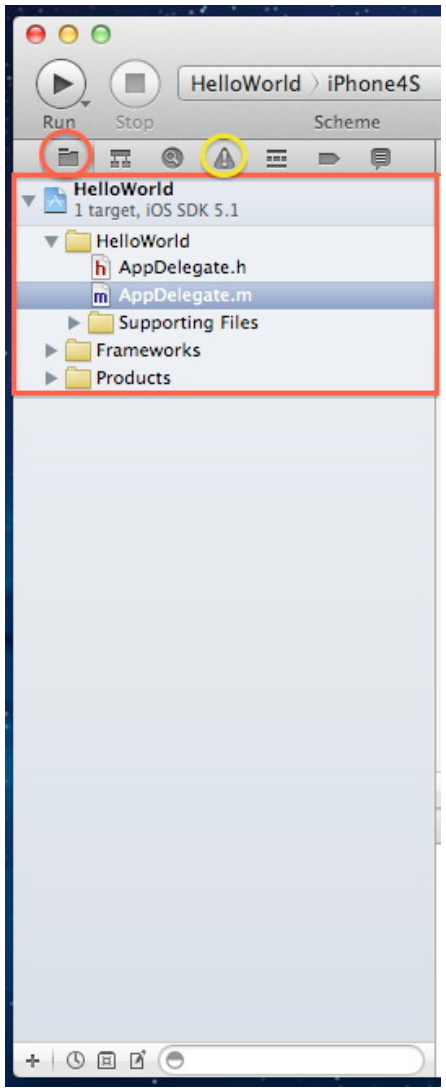


Fig. B: *Sezioni che compongono Xcode*

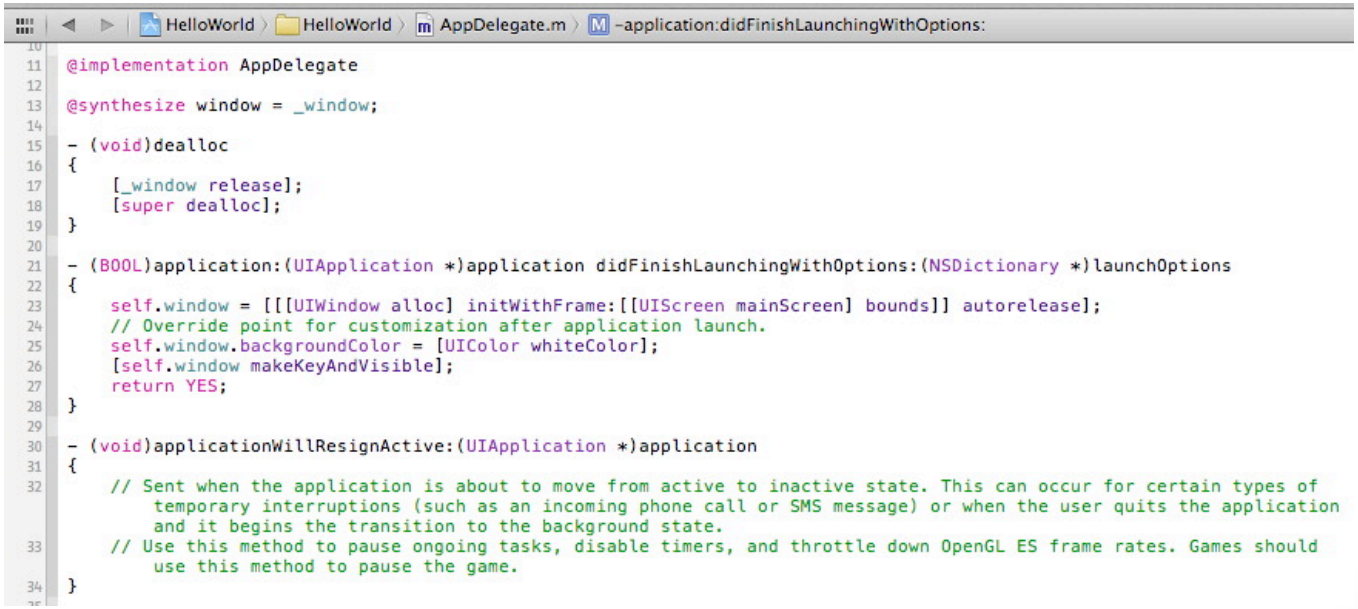
Xcode

- Come prima cosa vediamo come accedere ai file del nostro progetto per poter iniziare ad modificare il codice sorgente.
- Nel menu di sinistra, cliccando sulla prima icona a forma di cartella ci verrà mostrata la gerarchia dei file del nostro progetto. I file sorgenti veri e proprio (ovvero quelli dove risiede il codice Objective-C dell'applicazione) sono all'interno della cartella avente lo stesso nome del progetto (in questo caso "HelloWorld").



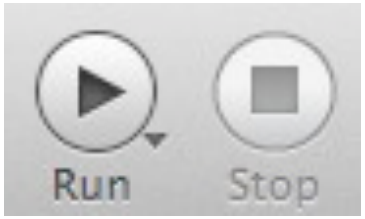
Xcode

- Per modificare un file dovremo semplicemente cliccare sul nome dello stesso. L'unico altro pulsante di nostro interesse è quello cerchiato in giallo nella figura precedente: questo ci permetterà di accedere al menu dove Xcode riporterà *warning* ed *errori*.
- A questo punto nella parte di destra ci verrà mostrato il contenuto del file e potremo modificarlo.



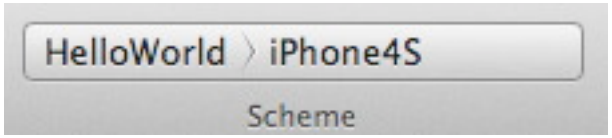
Xcode

- Altra parte molto importante dell'interfaccia, sono i pulsanti per l'esecuzione e lo stop dell'applicazione mostrati in figura. Tramite il tasto con l'icona di play potremo compilare ed eseguire l'applicazione, mentre con il tasto di stop la fermeremo. Nell'interfaccia di Xcode manca però il tasto per compilare (il tasto play compila ed esegue); esistono tuttavia delle scorciatoie da tastiera sia per compilare (Build) lanciare (Run) e fermare (Stop) che sono molto comode e veloci da utilizzare:
- **Build**: shift+cmd+B
- **Run**: cmd+R
- **Stop**: *shift+cmd+invio*.
- (tastiera mac)



Xcode

- Alla destra dei pulsanti *Run* e *Stop* troviamo un'altro pulsante molto utile che ci permetterà di decidere quale schema utilizzare nella fase di compilazione e su quale dispositivo installare l'applicazione. Facendo clic sul nome del progetto (*HelloWorld*) modificheremo gli schemi, mentre facendo clic sul nome del dispositivo, potremo scegliere se installare sul dispositivo fisico o sul simulatore.
- Normalmente, la situazione è la seguente: il dispositivo fisico e i due simulatori (iPhone e iPad).



Xcode

- In basso troviamo la console dove potremo stampare tutto ciò che vogliamo durante l'esecuzione dell'applicazione. L'utilizzo della console, unito ad un corretto uso dei *breakpoint*, è la base indispensabile per poter trovare e correggere gli errori!

Xcode

- Possiamo mostrare e nascondere la console tramite il pulsante cerchiato di rosso. Quello verde serve invece per mostrare il menu di sinistra contenente la gerarchia dei file, mentre quello azzurro ci permetterà di accedere al menu di Interface Builder.

- L'ultimo elemento di nostro interesse è il pulsante per abilitare e disabilitare i breakpoint. Una volta inseriti i breakpoint (che come in ogni IDE si inseriscono cliccando sul numero della linea di codice) potremo decidere se abilitarli o meno tramite questo pulsante.