

La prima applicazione

Benvenuti nel mondo della programmazione Mac! Molti libri di programmazione iniziano con lunghe pagine di storia dei linguaggi di programmazione, introducendo molti concetti astratti prima di mostrare qualcosa di concreto. Anche quando finalmente si arriva a fare qualcosa con il computer, spesso si tratta di scrivere codice per strumenti a riga di comando che visualizzano il testo in una console “senza interfaccia utente”. Questo libro è diverso.

Iniziamo il nostro viaggio insieme creando un’applicazione semplice ma pienamente funzionale che presenta tutte le belle caratteristiche di una tipica applicazione Mac. Il nostro programma si avvia come qualsiasi applicazione Mac, visualizza la propria finestra, mostra la propria barra dei menu, risponde all’input dell’utente e, cosa stupefacente, consente perfino all’utente di stampare utilizzando una stampante collegata al computer. E tutto ciò sarà ottenuto senza scrivere nemmeno una riga di codice.

Nei successivi capitoli utilizzeremo questa semplice applicazione per illustrare come si scrive il codice. Il nostro scopo, perciò, non è solo quello di farvi apprendere un nuovo “linguaggio” di programmazione, ma anche di spiegarvi come si realizza il software *dalla prospettiva del Mac*, utilizzando questo linguaggio all’interno di un’applicazione Mac reale, che creeremo utilizzando gli strumenti per gli sviluppatori di Apple.

In questo capitolo

- 2.1** Introduzione a Xcode
- 2.2** L’evento main
- 2.3** Il framework Cocoa
- 2.4** Risorse dell’applicazione
- 2.5** Riepilogo

2.1 Introduzione a Xcode

Se avete già provato a scrivere codice su altre piattaforme, o se vi siete divertiti a scrivere pagine web, avete potuto scegliere tra una varietà di ambienti di programmazione o strumenti di codifica. Sul Mac generalmente si utilizza *Xcode*, un software fornito gratuitamente da Apple.

Questo software è incluso nei dischi di installazione di Mac OS X come elemento da installare a parte, o in alternativa potete prelevare la versione più recente dal sito Apple Developer Connection. Se non avete ancora installato Xcode, fatelo ora seguendo le istruzioni fornite nell'Appendice B.

Benché il nome sembrerebbe suggerire che Xcode serva solo a scrivere codice, in realtà si tratta di un *ambiente di sviluppo integrato*, o IDE (Integrated Development Environment). Lo utilizzeremo per organizzare i file di codice, avviare strumenti per la realizzazione dell'interfaccia, creare applicazioni dal codice, eseguire applicazioni, correggere eventuali errori e molto altro.

L'ambiente di Xcode

Avviate subito Xcode e create il vostro primo progetto di programmazione. Al primo avvio dovrete vedere una finestra di benvenuto: chiudetela e scegliete *File > New Project*. Apparirà la finestra per la scelta di un template, come quella mostrata nella Figura 2.1. Per realizzare un'applicazione in Xcode si utilizzano molti file diversi. Invece di aggiungere tutti questi file a un progetto del tutto vuoto, Xcode mette a disposizione vari template utilizzabili come punti di partenza per procedere.

Nella parte sinistra della finestra per la scelta del template potete notare molti diversi tipi di progetti Mac OS X. Potreste anche vedere dei tipi di template per progetti iPhone OS (visibili nella Figura 2.1), che però appaiono soltanto se avete installato anche l'iPhone SDK.

Parleremo più avanti dei vari tipi di progetti disponibili, per ora creiamo un'applicazione Mac OS X. Selezionate il tipo *Application* (appena sotto il titolo *Mac OS X*) e vedrete diversi tipi di template elencati nella parte superiore destra della finestra.

Un'applicazione Mac standard può essere di due tipi principali: *Cocoa Application* e *Cocoa Document-based Application*. Per spiegare la differenza utilizziamo degli esempi. Apple Pages e Microsoft Word sono esempi di applicazioni basate su documenti. iTunes e DVD Player, invece, sono applicazioni non basate su documenti, perché non chiedono all'utente di "aprire un file". La differenza non è sempre così chiara, ma questa è la distinzione di base.

Per evitare complicazioni inutili, creeremo un'applicazione Cocoa di base, perciò selezionate il template *Cocoa Application* nella finestra del progetto, disattivate la casella di controllo *Create document-based application* nella parte inferiore della finestra e fate clic sul pulsante *Choose*.

A questo punto vedrete una finestra standard che vi chiederà un nome e una posizione in cui salvare il progetto. Notate che Xcode creerà automaticamente una nuova cartella con lo stesso nome dell'applicazione per contenere tutti i file del progetto, perciò non è necessario che lo facciate voi.

Date al progetto il nome "TextApp" e fate clic sul pulsante *Salva*.

La finestra del progetto

A questo punto dovrete vedere apparire sullo schermo una finestra simile a quella della Figura 2.2. Non fatevi intimorire dal lungo elenco *Groups & Files* sulla sinistra: nella maggior parte dei casi utilizzerete soltanto il primo gruppo dell'elenco *TextApp*.

Alcune delle cartelle di questo gruppo saranno vuote, ma fate clic sul triangolino accanto a *Other Sources* per espanderla e vederne il contenuto: appariranno due file, uno dei quali si chiama *main.m*. Fate clic su di esso e il suo contenuto apparirà nella parte inferiore destra della finestra del progetto, un po' come avviene per il testo di una mail nell'applicazione della posta di Apple. Se fate doppio clic sul file *main.m* nell'elenco *Groups & Files*, lo aprirete in una nuova finestra.

Notate che lungo la parte superiore della finestra del progetto ci sono diversi pulsanti e caselle a discesa: ne parleremo nel momento in cui le useremo.

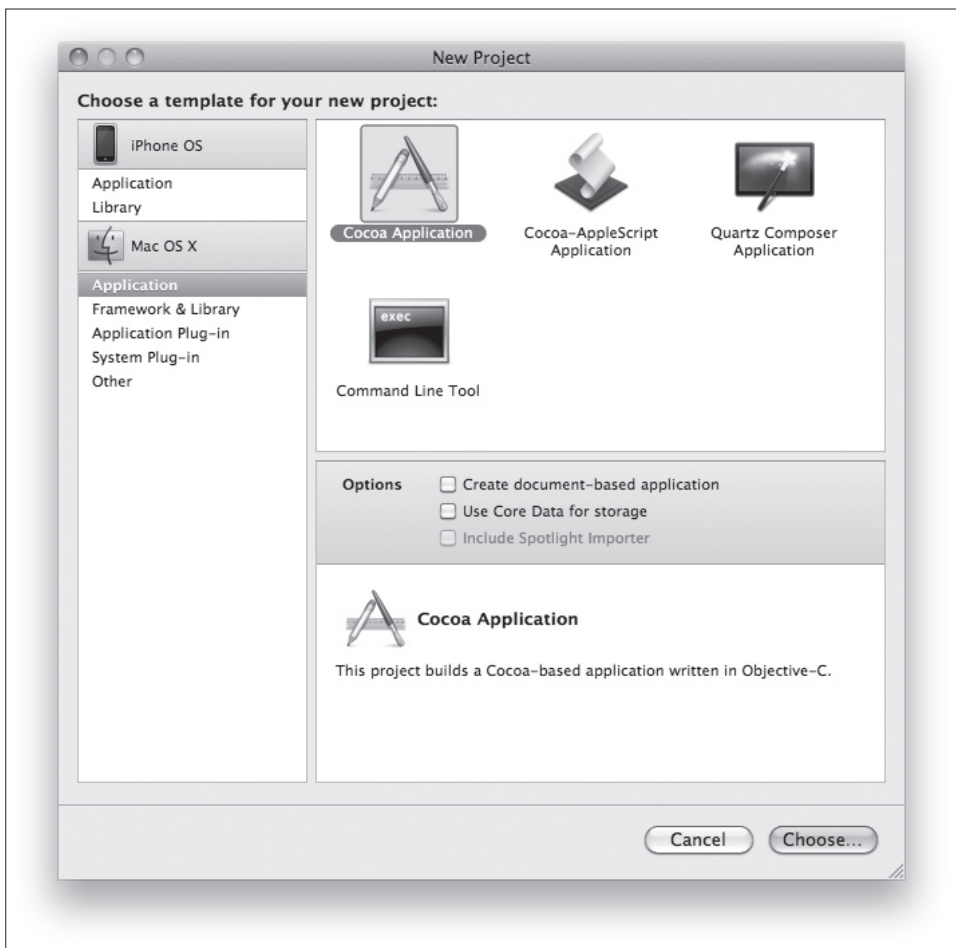


Figura 2.1 La finestra per la scelta di un template di Xcode.

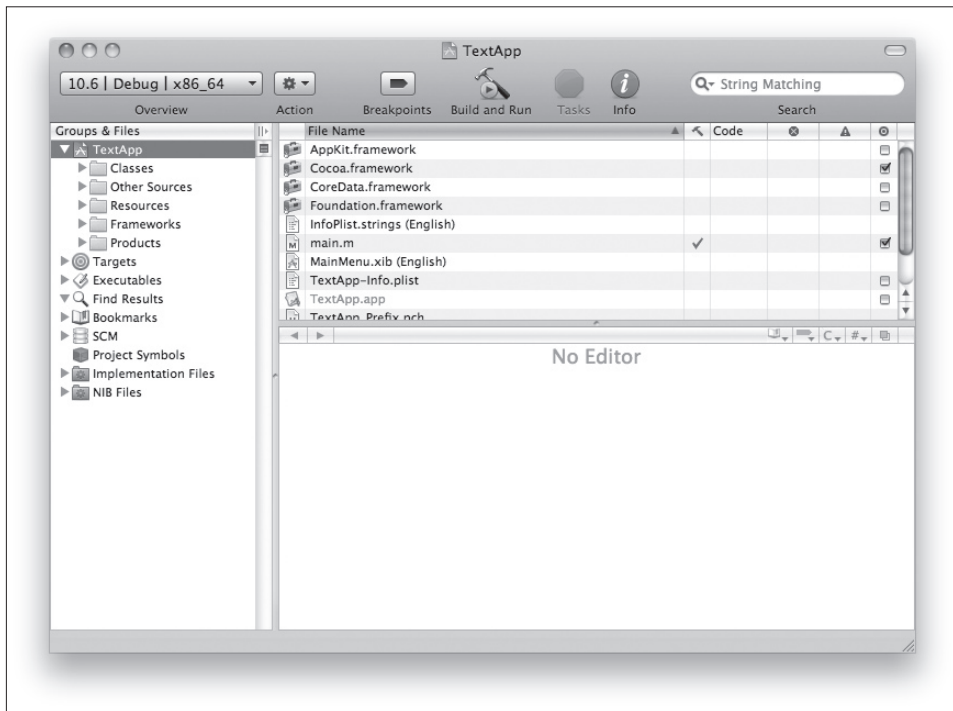


Figura 2.2 La finestra del progetto TextApp.

2.2 L'evento main

La maggior parte dei libri introduttivi alla programmazione che parlano di varianti del linguaggio C presenta prevalentemente codice scritto all'interno del file `main`. Quando si scrivono applicazioni standard Mac, tuttavia, in realtà accade raramente di modificare questo file. All'inizio di questo capitolo abbiamo detto che non avremmo *scritto* nemmeno una riga di codice, perciò manteniamo la promessa, ma è comunque utile dare uno sguardo al codice.

Più avanti tratteremo i dettagli relativi al layout e alla sintassi del codice, ma per ora limitiamoci a una rapida panoramica su ciò che si vede in questo particolare file. Quando fate doppio clic sul file `main.m`, vedrete aprirsi una finestra contenente il codice seguente:

YourFirstApp/TextApp/main.m

```
//
// main.m
// TextApp
//
// Created by Tim Isted on 08/09/2009.
// Copyright 2009 MyCompanyName . All rights reserved.
//
```

```
#import <Cocoa/Cocoa.h>
```

```
int main(int argc, char *argv[])
{
    return NSApplicationMain(argc, (const char **) argv);
}
```

Se non avete ancora modificato le impostazioni predefinite di Xcode, il codice di questo file dovrebbe essere visualizzato in vari colori che hanno lo scopo di aiutarvi nella codifica, perché consentono di individuare porzioni di codice “a colpo d’occhio”.

Notate che le prime righe appaiono in verde. Ognuna di queste righe inizia con due barre slash, come la seguente:

```
// TextApp
```

Queste righe sono *commenti* e vengono completamente ignorate durante l’esecuzione del codice. In questo caso sono utilizzate per fornire informazioni sul file, quali il nome del file, il nome del progetto, l’autore e i dati di copyright. Questi dettagli in particolare sono stati inseriti automaticamente nel momento in cui abbiamo utilizzato il template del progetto Cocoa Application.

Come vedrete nel prosieguo del libro, i commenti possono essere utilizzati in vari modi. Uno dei loro impieghi principali è quello di *documentare* il codice. Per esempio, potreste avere bisogno di svolgere un calcolo geometrico complesso per determinare come disegnare una forma a stella regolare all’interno di un insieme di coordinate di disegno. Il codice corrispondente potrebbe essere ben chiaro a voi mentre lo scrivete, ma sei mesi più tardi potrebbe risultare assolutamente impossibile comprenderlo, senza qualche commento inserito qua e là per spiegare che cosa accade.

Un’altra possibilità è quella di *contrassegnare come commenti* particolari righe di codice. Supponiamo che il vostro codice per disegnare una stella non funzioni bene come vorreste. Potreste decidere di disegnare un semplice rettangolo al posto della stella, per assicurarvi che i calcoli delle coordinate siano corretti. Invece di eliminare tutte le righe di codice per il disegno della stella, potreste limitarvi a contrassegnarle come commenti, in modo che più tardi possiate reinserirle di nuovo, una riga alla volta, senza doverle digitare nuovamente per intero.

Dopo le parti dei commenti in verde, c’è una riga marrone e rossa che inizia con `#import`. Non pensateci troppo, per ora, e concentratevi invece sulle ultime quattro righe del file:

```
int main(int argc, char *argv[])
{
    return NSApplicationMain(argc, (const char **) argv);
}
```

Che ci crediate o no, queste quattro righe contengono tutta la vostra applicazione Mac OS X, dall’inizio alla fine. Per semplificare un po’ il processo, quando un utente fa doppio clic sull’applicazione nel Finder, il sistema operativo cerca all’interno del codice questa porzione principale (`main`) e quindi esegue il codice racchiuso tra le parentesi graffe.

Come abbiamo già detto, non vi capiterà spesso di dover modificare il file `main.m`. Tenendo conto di ciò, vediamo che cosa accade quando si esegue l’applicazione.

Chiudete il file `main.m` in modo da tornare alla finestra del progetto `TextApp`. Fate clic sul pulsante *Build & Run* nella barra degli strumenti nella parte inferiore di questa finestra

Tasti di scelta rapida

La maggior parte dei comandi di menu presentati nel libro può essere eseguita anche premendo le combinazioni di tasti corrispondenti, indicate con i nomi utilizzati di consueto per descrivere i tasti Mac.

Riportiamo di seguito un elenco di corrispondenze tra i simboli presenti sulla maggior parte delle tastiere Mac e i nomi dei tasti corrispondenti:

^	Control o Ctrl
⇧	Maiusc
⌥	Opzione, Opt o Alt
⌘	Comando o Mela
⌘	Delete or Del
↵	Invio (Return)
⌘	Invio

e pazientate mentre Xcode *compila* il progetto utilizzando i vari file del template e poi esegue l'applicazione risultante.

Se tutto va secondo i piani, l'applicazione verrà avviata. Dovrebbe apparire una finestra vuota, che potete spostare e ridimensionare. Notate che la barra dei menu è cambiata: mostra “TextApp” come nome dell'applicazione in alto a sinistra. Osservate gli elementi di ciascun menu. Troverete un menu *File* standard, con diversi comandi quali *New* e *Open visualizzati in grigio* per indicare che non sono disponibili. Il menu *Edit* contiene le azioni standard della *pasteboard* (l'area di memoria temporanea) quali *Copia* e *Incolla*. Il menu *Window* contiene comandi che agiscono sulla finestra vuota visibile sullo schermo; potete ridurre tale finestra a un'icona sul Dock o ingrandirla a tutto schermo.

Uno dei più importanti principi per la realizzazione di software destinato alla piattaforma Mac è che le applicazioni devono seguire un insieme standard di linee guida per l'interfaccia stabilite da Apple. Una di queste linee guida afferma che certe voci di menu devono apparire in tutte le applicazioni e devono essere raggruppate in modo specifico. Per esempio, dovrete trovare sempre i comandi *Taglia*, *Copia* e *Incolla* nel menu *Modifica*, e sempre in tale ordine. Se la vostra applicazione segue queste linee guida, risulterà molto più facile da utilizzare, perché avrà un comportamento corrispondente alle attese degli utenti.

Ora uscite dall'applicazione TextApp nel modo che preferite. Potete selezionare il comando *Quit TextApp* dal menu *TextApp* oppure premere il tasto di scelta rapida corrispondente, Mela+Q. Potete anche fare clic con il pulsante destro del mouse (o Ctrl+Clic) sull'icona *TextApp* che è apparsa nel Dock nella parte inferiore dello schermo e selezionare *Esci*. Sono tutti modi del tutto leciti per uscire dall'applicazione, e li troverete già belli e pronti.

2.3 Il framework Cocoa

Ricordate che nel paragrafo precedente abbiamo osservato il contenuto del file `main.m` e abbiamo visto una riga di codice che apparentemente eseguiva l'applicazione dall'inizio alla fine? Sembra piuttosto strano che questa singola riga possa provvedere a tutte le funzionalità che abbiamo visto in azione.

Un modo per scrivere applicazioni per computer potrebbe essere quello di scrivere codice che, letteralmente, disegnasse ogni singolo pixel sullo schermo per rappresentare l'interfaccia utente. Se decideste di scrivere `TextApp` in questo modo, dovrete disegnare una barra lungo il lato superiore della finestra per i menu, poi visualizzare il testo di ogni menu, prima di disegnare il contorno della finestra e il suo contenuto. Senza considerare la necessità di visualizzare ciò che sta nello sfondo dello schermo dell'utente, che sia la scrivania o altre applicazioni, e tralasciando il fatto che dovremmo scrivere codice per far cambiare tutti i pixel in corrispondenza delle interazioni dell'utente con l'applicazione. Ricordate che abbiamo parlato di una serie di linee guida per l'interfaccia delle applicazioni? Una *finestra* ha un aspetto ben definito, per esempio, e i menu si comporteranno tutti in un certo modo; se ogni programmatore dovesse scrivere codice simile per ottenere lo stesso comportamento di base, si arriverebbe a una grande quantità di funzionalità duplicate tra le applicazioni. Immaginate che cosa accadrebbe se le linee guida cambiassero leggermente: si dovrebbe modificare ogni applicazione per adeguarla al nuovo standard. Tutti questi problemi si risolvono utilizzando un *framework*, o infrastruttura. Un framework fornisce un'ampia quantità di codice già scritto, che possiamo utilizzare per evitare di "reinventare la ruota". Quando scriviamo software Mac, utilizziamo il framework *Cocoa*, fornito da Apple.

Ricorderete che, quando abbiamo creato il progetto `TextApp`, abbiamo scelto il template `CocoaApplication`. Creando un'applicazione con Cocoa, non dobbiamo preoccuparci delle funzionalità di base esibite dalla maggior parte delle applicazioni Mac, e possiamo concentrarsi sulla scrittura del codice per le funzionalità peculiari della nostra applicazione. Aprite ancora `main.m` per osservare quella riga importante racchiusa tra parentesi graffe:

```
return NSApplicationMain(argc, (const char **) argv);
```

Per ora, ignorate il fatto che la sintassi possa incutere timore. Tutto ciò che fa questa riga è creare un'applicazione Cocoa e assegnarle il controllo.

2.4 Risorse dell'applicazione

È facile dire che stiamo assegnando il controllo a un'applicazione Cocoa, ma non abbiamo ancora scoperto da dove vengono la barra dei menu e le finestre.

Se in precedenza avete fatto doppio clic sul file `main.m` per aprirlo in una finestra separata, chiudete tale finestra per tornare alla finestra del progetto di Xcode per `TextApp`. Nella parte sinistra della finestra, il gruppo *Other Sources* dovrebbe apparire ancora espanso. Sotto di esso vedrete un altro gruppo denominato *Resources*; fate clic sul triangolino accanto a esso per espanderlo, e vedrete altri tre file. Fate clic una volta sola sul file `TextApp-Info.plist` (se utilizzate una versione precedente di Xcode, questo file potrebbe chiamarsi `Info.plist` anziché `TextApp-Info.plist`) e lo vedrete apparire nella parte inferiore destra della finestra del progetto, come illustrato nella Figura 2.3.

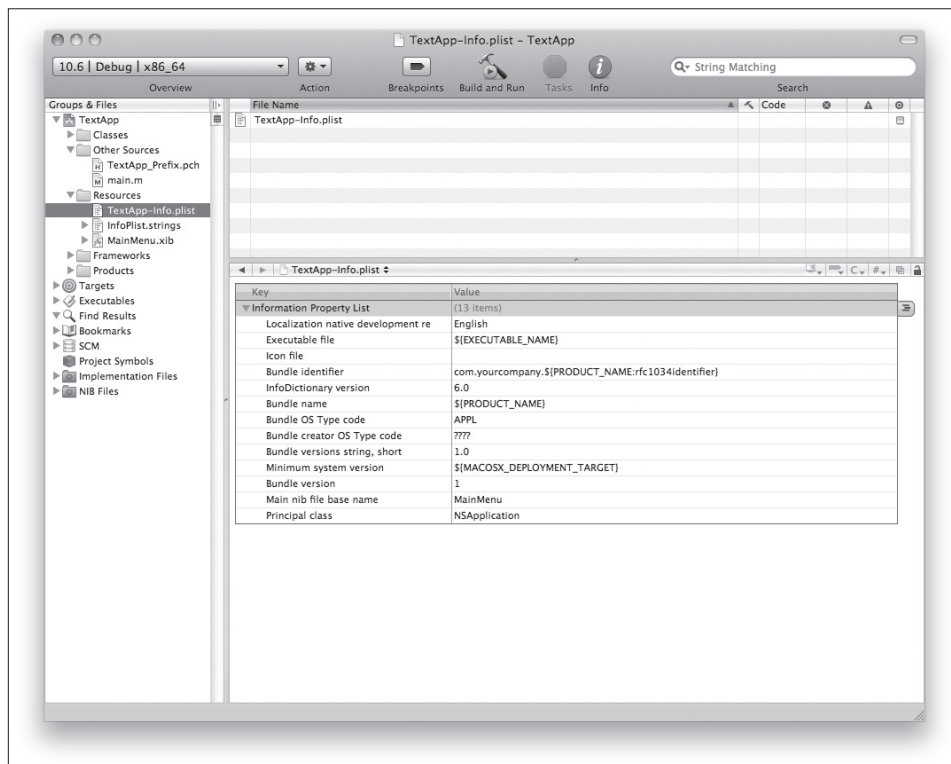


Figura 2.3 Il file TextApp-Info.plist.

Questo file contiene molte informazioni, ma la riga su cui dobbiamo concentrarci è quella che recita *Main nib file base name*. Vedrete che la colonna *Value* per questa riga contiene *MainMenu*.

Il file MainMenu.xib

Quando create un'applicazione utilizzando Cocoa, il framework cerca il valore che abbiamo appena citato all'interno del file *ApplicationName-Info.plist* e utilizza il file con quel nome per creare l'interfaccia di base dell'applicazione. Guardate ancora nel gruppo *Resources* sulla sinistra della finestra del progetto: vedrete che la terza risorsa elencata si chiama *Main-Menu.xib*. Fate doppio clic su questo file per aprirlo.

Xcode avvia un'altra applicazione per sviluppatori, denominata *Interface Builder*, per la modifica di questo file. Quando l'applicazione si apre, presenta sullo schermo varie finestre: la principale è simile a quella illustrata nella Figura 2.4.

Questa finestra *MainMenu.xib* contiene una varietà di elementi; i due da considerare con attenzione ora sono *Main Menu* e *Window (TextApp)*.

Fate doppio clic sull'icona *Main Menu* per aprire l'editor di menu (potrebbe anche essere già aperto sullo schermo). Questo strumento (visibile nella Figura 2.5) contiene la barra dei menu visualizzata quando si esegue l'applicazione *TextApp*. Se fate clic sul titolo di

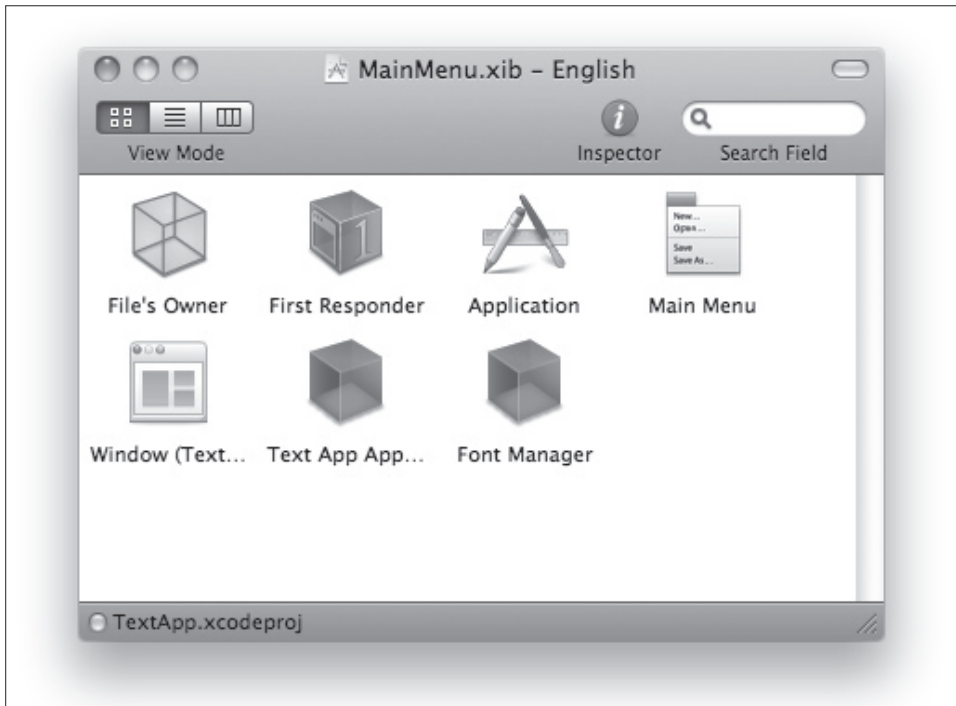


Figura 2.4 Il file MainMenu.xib aperto in Interface Builder.

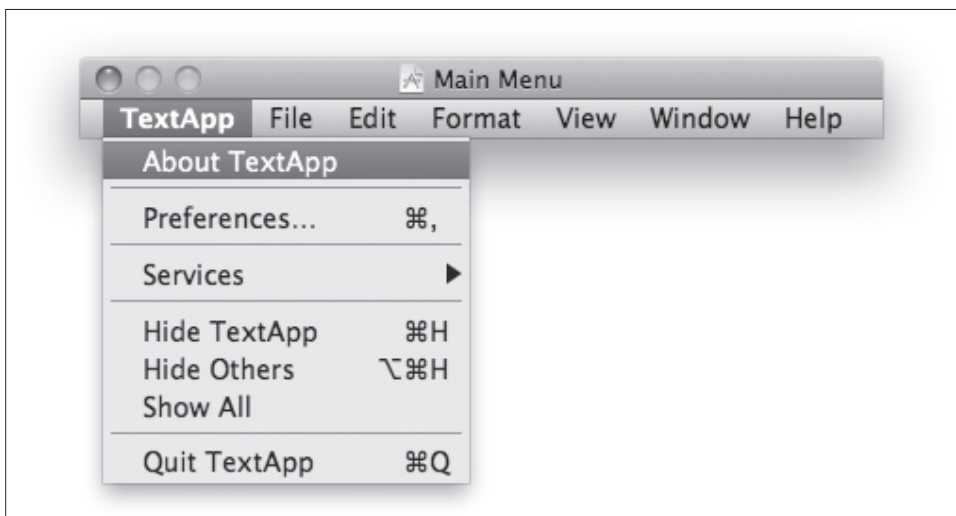


Figura 2.5 L'editor di menu in Interface Builder.

un menu, lo strumento apre il menu corrispondente in modo che possiate apportare le modifiche desiderate ai suoi elementi.

Fate clic sul menu *TextApp* per visualizzarlo, e poi fate doppio clic sul primo elemento, *About TextApp*. Il titolo dell'elemento diventa modificabile e potete sostituirlo con quello che preferite. Cambiate il nome in *Informazioni sulla mia splendida applicazione TextApp*. In alcune versioni precedenti di Xcode, il template del progetto non assegna correttamente il nome di alcuni degli elementi dei menu. L'elemento *About* potrebbe apparire come *About NewApplication*, e il comando *Quit* potrebbe apparire come *Quit NewApplication*. In tal caso, potete rinominare gli elementi nel modo appena descritto. Anche il menu dell'applicazione potrebbe apparire come *NewApplication*, ma al momento dell'esecuzione effettiva diventerà, come per magia, *TextApp*.

Salvate il file *MainMenu.xib* in Interface Builder e tornate a Xcode. Fate clic sul pulsante *Build & Run* nella barra degli strumenti per avviare l'applicazione. Ora, all'avvio di *TextApp*, vedrete che il comando di menu "About" appare con il suo nuovo nome impostato in Interface Builder. Scegliete il comando *Quit TextApp* per uscire dall'applicazione.

Aggiungere elementi all'interfaccia di base

Cocoa non si limita a fornire elementi di base dell'interfaccia utente come finestre e menu, mette a disposizione anche una serie di altri *controlli* con cui si possono aggiungere funzionalità all'applicazione.

Nei prossimi capitoli avremo bisogno di un luogo in cui visualizzare delle informazioni di testo. Ora, per illustrare quante funzionalità possono offrire i controlli "integrati", utilizziamo un controllo per consentire all'utente di digitare del testo nella finestra. Più avanti utilizzeremo altri controlli.

Tornate a Interface Builder (c'è un tasto di scelta rapida molto utile per passare da un'applicazione all'altra in Mac OS X: tenete premuto il tasto Mela e premete il tasto Tab; apparirà una finestrella in cui potrete scegliere tra tutte le applicazioni aperte. Quando rilasciate il tasto Mela, l'applicazione selezionata sarà portata in primo piano) e assicuratevi che il file *MainMenu.xib* sia ancora aperto. Selezionate il menu *Tools* e aprite la palette *Library*. Questa palette, mostrata nella Figura 2.6, contiene i controlli che potete utilizzare come tali nei vostri progetti o anche estendere con funzionalità aggiuntive.

Nella parte inferiore della palette *Library* c'è una casella di ricerca; digitate **text view** e vedrete che nella palette rimarrà un solo elemento, quello che stiamo per utilizzare.

Dobbiamo aggiungere la nostra nuova *text view* (o "vista di testo") alla finestra di *TextApp*. Per assicurarci che questa finestra sia visibile, facciamo doppio clic sull'icona *Window (TextApp)* nella finestra di *MainMenu.xib*: si aprirà come finestra vuota sullo schermo.

Trascinate un oggetto *Text View* dalla palette *Library* nella finestra vuota. Quando passate con l'oggetto sopra la finestra, noterete diverse guide blu che vi aiuteranno a posizionarlo. Allineate l'angolo superiore sinistro con le linee blu a poca distanza dall'angolo superiore interno della finestra. Utilizzando i quadratini di posizionamento che appaiono attorno all'oggetto, ingrandite quest'ultimo finché il suo angolo inferiore destro si allinea con le linee blu che appaiono a poca distanza dall'angolo inferiore destro della finestra. Dovreste ottenere qualcosa di simile a quanto si vede nella Figura 2.7.

È il momento di un test della nuova *text view*. Salvate il file di Interface Builder, tornate a Xcode e fate clic su *Build & Run*. All'avvio dell'applicazione vedrete che la finestra principale presenta una *text view* al suo interno, in attesa di input. Notate che potete

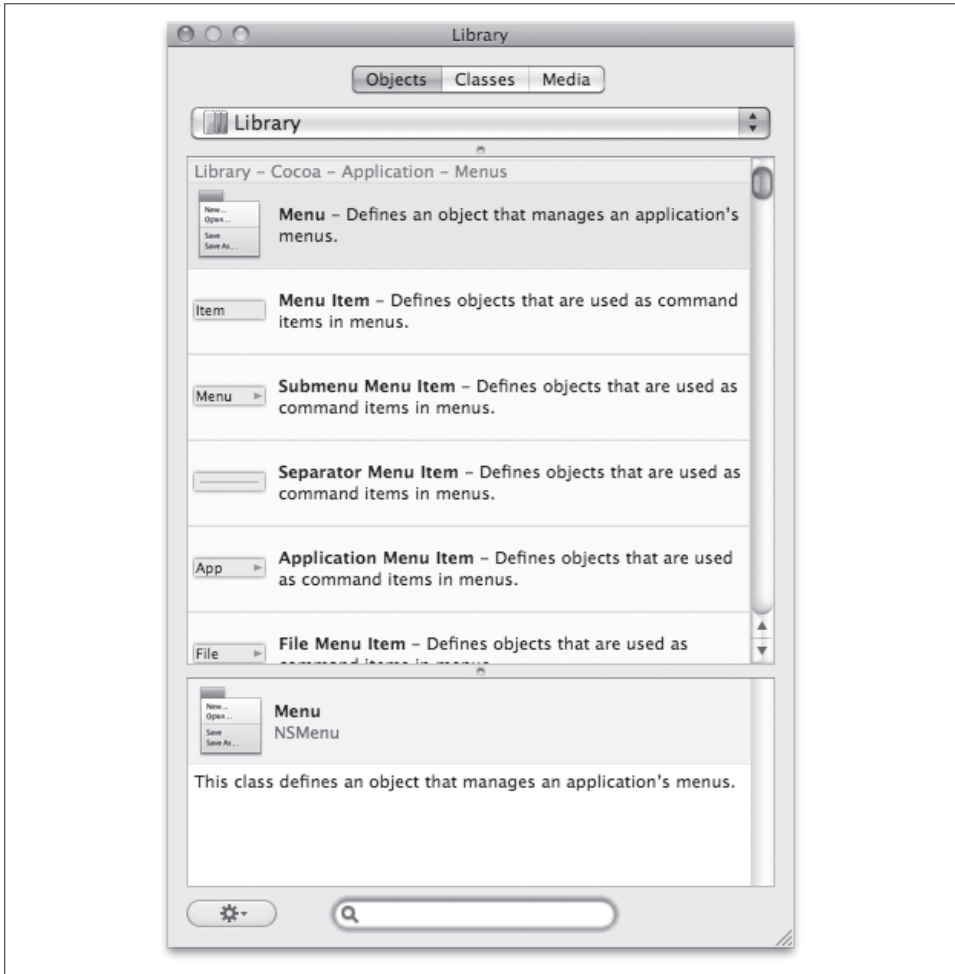


Figura 2.6 La palette Library in Interface Builder.

digitare qualsiasi cosa, selezionare caratteri con il mouse, trascinare elementi, copiare e incollare da altre applicazioni, e persino formattare il testo utilizzando la palette *Fonts* disponibile nel menu *Format* di *TextApp*. Stupefacente, vero? E non abbiamo ancora scritto una riga di codice.

2.5 Riepilogo

Abbiamo fatto un rapido viaggio nel fantastico mondo in cui si realizzano applicazioni Mac OS X utilizzando il framework Cocoa di Apple. Senza scrivere una sola riga di codice, abbiamo creato un'applicazione funzionale con alcune notevoli caratteristiche per la modifica di testi, semplicemente utilizzando le risorse disponibili in un'applicazione Cocoa.

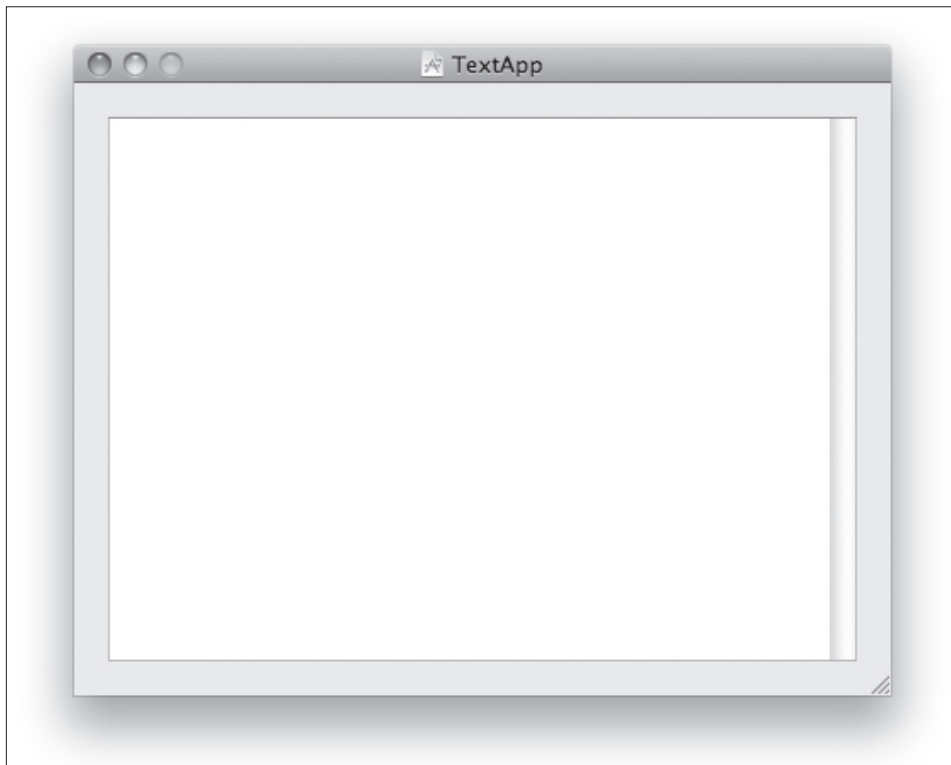


Figura 2.7 La text view all'interno della finestra.

Fin qui va tutto bene, ma per creare applicazioni che siano utili e funzionali nel mondo reale (notate per esempio che la nostra applicazione TextApp non prevede comandi di annullamento o di salvataggio dei file) dovrete imparare a scrivere del codice. Nei prossimi capitoli apporteremo delle modifiche a TextApp, utilizzandola per visualizzare vari tipi di output e per collaudare diverse caratteristiche dello sviluppo di software Mac.

Il capitolo seguente introduce alcuni principi di base della programmazione, e vi farà iniziare realmente a scrivere codice. Intanto fate tutte le prove che volete con i vari oggetti, palette e funzionalità forniti da Interface Builder, ma assicuratevi di mantenere una copia “pulita” di TextApp che vi servirà per il prossimo capitolo.

Prelevare il codice

Potete prelevare i progetti di Xcode e il codice utilizzati in questo libro collegandovi al sito web dell'editore americano, nella sezione dedicata a questo libro:

<http://www.pragprog.com/titles/tibmac>