

2 Installation und ein erstes Beispiel

In diesem Einstiegskapitel lernen Sie zunächst das iPhone Dev Center als zentrale Quelle für Informationen rund um die Entwicklung von Anwendungen für das iPhone kennen. Danach folgt eine Erläuterung der Installation sowie der Deinstallation des iPhone SDK. Zum Abschluss dieses Kapitels wollen wir uns noch ein Beispiel ansehen, das Ihnen ein erstes Gefühl für die iPhone-Anwendungsentwicklung vermitteln soll.

Grundvoraussetzung für die Entwicklung mit dem iPhone SDK

Bevor dieses Kapitel direkt mit der Installation und einem Beispiel loslegt, sollen hier kurz die Grundvoraussetzungen für die Programmierung mit dem iPhone SDK aufgezeigt werden.

Für die Entwicklung mit dem iPhone SDK benötigen Sie einen Apple Mac mit installiertem Mac OS X. Da Sie in diesem Kapitel auch schon Quellcode zu sehen bekommen, sei noch darauf hingewiesen, dass die verwendete Programmiersprache Objective-C ist. Eine Einführung dazu finden Sie im Anhang dieses Buches.

2.1 Das iPhone Dev Center

Alles, was Sie für den Start in die iPhone-Anwendungsentwicklung benötigen, u.a. das iPhone SDK, finden Sie im *iPhone Dev Center*, einem umfangreichen Portal rund um die Entwicklung von iPhone-Anwendungen (<http://developer.apple.com/iphone>). Das iPhone Dev Center ist ein spezieller Bereich innerhalb der *Apple Developer Connection* (<http://developer.apple.com>).

Für den Zugriff auf die im iPhone Dev Center abgelegten Informationen benötigen Sie jedoch einen Zugang bei der *Apple Developer Connection*: Ohne eine Anmeldung erhalten Sie zwar einen Überblick

Das iPhone Dev Center ist ein Unterbereich innerhalb der Apple Developer Connection.

darüber, welche Informationen im iPhone Dev Center vorhanden sind, Sie können aber nicht darauf zugreifen.

Das iPhone Developer
Program

Es ist wichtig zu erwähnen, dass eine Anmeldung zur Apple Developer Connection nicht mit einer Anmeldung zum *iPhone Developer Program* (<http://developer.apple.com/iphone/program/>) zu verwechseln ist.

Apple Developer Connection vs. iPhone Developer Program

Wie erwähnt, ist die Anmeldung zur *Apple Developer Connection* obligatorisch, um das iPhone SDK und auch weiterführende Dokumentation und Beispiele herunterladen zu können. Diese Anmeldung ist kostenlos. Apple möchte wohl einfach etwas Kontrolle über die Informationen haben.

Das *iPhone Developer Program* ist dagegen eine kostenpflichtige Angelegenheit. Eine Mitgliedschaft ist erforderlich, wenn Sie Ihre Anwendung vertreiben wollen, wobei es nicht darauf ankommt, ob dies kostenpflichtig oder umsonst geschehen soll. Eine Installation Ihrer entwickelten Anwendung auf einem iPhone kann nur über den Umweg des iPhone Developer Program erfolgen. Eine ausführliche Beschreibung des iPhone Developer Program finden Sie in Kapitel 6 dieses Buches.

Das iPhone Dev Center können Sie als zentrale Stelle für Informationen rund um die Entwicklung mit dem iPhone SDK verstehen. Die Einstiegsseite enthält drei wesentliche Bereiche:

- Suche (Abb. 2–1)
- Beispiele und mehr (Abb. 2–2)
- Download (Abb. 2–3)

Bereich 1:
Suche

Im oberen Bereich der Seite können Sie über das rechte Eingabefeld eine Volltextsuche innerhalb des kompletten iPhone Dev Center ausführen. Sollten mehrere Versionen des iPhone SDK angeboten werden, können Sie zwischen ihnen am linken Rand der Titelseile umschalten.

Abb. 2–1

Volltextsuche innerhalb
der zur Verfügung
stehenden Dokumente



Wie Sie in Abbildung 2–1 sehen, können Sie über die Schaltfläche *iPhone SDK 3.1* auf diese Version zugreifen. Sollten mehrere Versionen zur Verfügung stehen, befindet sich die neueste höchstwahrscheinlich im Betastatus. Auf solch eine Betaversion erhalten Sie nur Zugriff, wenn Sie beim iPhone Developer Program angemeldet sind.

Bereich 2:
Beispiele und mehr

Unterhalb der Suche finden Sie unter der Beschriftung *Resources for iPhone OS 3.0* Verweise zu wichtigen Hilfen für die Applikationsent-

wicklung mit dem iPhone SDK. Der erste Link mit der Bezeichnung *Downloads* springt lediglich auf der gleichen Seite nach unten, in den eigentlichen Downloadbereich. Über den Link *Getting Started Videos* gelangen Sie zu einer Übersicht mit hilfreichen Videotutorials für den Einstieg in die iPhone-Entwicklung. Für das Betrachten der Videos (inklusive der Übersicht) wird eine installierte iTunes-Version benötigt. Über die Links *Getting Started Documents* und *iPhone Reference Library* gelangen Sie zu Dokumenten rund um die Anwendungsentwicklung für das iPhone OS. Hinter dem Verweis *Coding How-To's* stehen die Antworten auf viele Fragen der täglichen Entwicklung, wie z.B. *How to create a table view?* Ein sehr wichtiger Bereich steht hinter dem Verweis *Sample Code*. Hier erhalten Sie Zugriff auf die zur Verfügung gestellten Beispiele. Diese helfen Ihnen beim Einstieg in die Programmierung.

Unter der Beschriftung *Featured Content* finden Sie speziell hervorgehobene Verweise, wie z.B. den Link auf *Cut, Copy and Paste*.

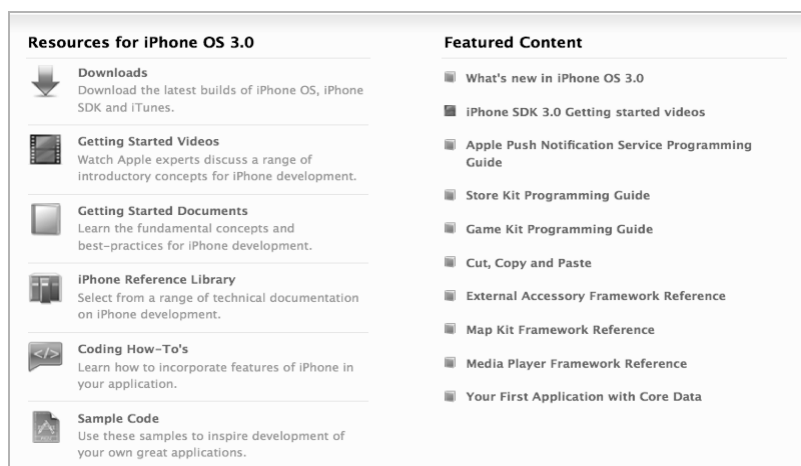


Abb. 2-2
Nachschlagebereich
innerhalb des
iPhone Dev Center


Im Bereich *Downloads* (Abb. 2-3) kann die aktuelle Version des iPhone SDK im Disk-Image-Format (.dmg) heruntergeladen werden. Zusätzlich stehen Ihnen eine Installationsanleitung und die Nutzungsbestimmungen (engl. Agreement) zur Verfügung. Vor dem Einsatz des iPhone SDK empfiehlt sich ein Blick in diese Bestimmungen. Zusätzlich kann es vorkommen, dass weitere wichtige Dokumente im Bereich *Downloads* angeboten werden. In Abbildung 2-3 sehen Sie beispielsweise den Link auf das *iPhone OS 3.0.1 Advisory*.

Bereich 3:
Download

Abb. 2-3

Downloadbereich für das
iPhone SDK

Downloads



iPhone SDK 3.0
With over 1,000 new APIs, iPhone SDK 3.0 provides developers with a range of new possibilities to enhance the functionality of their applications. New APIs also provide support for applications to communicate with hardware accessories attached to iPhone or iPod touch.

Posted: June 17, 2009
Build: 9M2736

Downloads

- iPhone SDK 3.0 (Leopard)
- iPhone SDK 3.0 (Leopard) Read Me
- iPhone SDK 3.0 (Snow Leopard)
- iPhone SDK 3.0 (Snow Leopard) Read Me
- iPhone SDK Agreement
- iPhone OS 3.0.1 Advisory

2.2 Das iPhone SDK installieren

Um das iPhone SDK zu installieren, müssen Sie zunächst die aktuelle Version aus dem Downloadbereich herunterladen. Wie schon erwähnt, steht das Paket als fertiger Installer im dmg-Format zur Verfügung. Dabei werden zwei Varianten angeboten. Ein Installer für das Mac OS X Leopard (Dateiname: `iphone_sdk_3.0_leopard_9m2736_final.dmg`) und ein Installer für Mac OS X Snow Leopard (Dateiname: `iphone_sdk_3.0_snow_leopard_final.dmg`). Der Dateiname kann sich mit zukünftigen Versionen des iPhone SDK ändern. Anzunehmen ist, dass zumindest die Versionsnummer bei zukünftigen Versionen mit hochgezählt wird. Beim Übergang von iPhone OS 2.1 auf 2.2 ist im Dateinamen zusätzlich der Name des Builds (z.B. `9m2736`) hinzugekommen. Das erwähnte Installationspaket für Mac OS X Leopard hat eine stattliche Größe von 2,08 GB. Für Mac OS X Snow Leopard ist die vergleichsweise kleine Größe von 404,1 MB zunächst überraschend, lässt sich jedoch dadurch erklären, dass das Paket ohne die Entwicklungsumgebung Xcode kommt – Xcode 3.2 wird nämlich bereits Bestandteil des Betriebssystems sein. Die weiteren Beschreibungen in diesem Buch beziehen sich auf die Mac-OS-X-Leopard-Variante.

iPhone SDK – passende Version

Für die Beschreibungen in diesem Buch wurde das iPhone SDK 3.0^a verwendet. Der dabei eingesetzte Build ist **9M2736**^b, der am 17.06.2009 erschienen ist. Für die Verwendung ist nach Angaben von Apple im iPhone Dev Center mindestens ein Mac OS X der Version 10.5.7 notwendig. Zusätzlich wird iTunes in der Version 8.2 benötigt. Sollte auf dem Zielsystem noch nicht die richtige Version installiert sein, können Sie dies über die integrierte Softwareaktualisierung von Mac OS X nachholen.

- a) Zum Redaktionsschluss dieses Buchs befand sich das iPhone SDK 3.1 im Betastatus. Dieses Release wird Fehlerbehebungen und kleine Änderungen enthalten. Somit sind alle Beschreibungen in diesem Buch auch für das iPhone SDK 3.1 gültig.
- b) Sollten Sie mit dem iPhone SDK 3 für das iPhone OS 3.0.1 entwickeln wollen, müssen Sie die Anweisungen im Dokument iPhone OS 3.0.1 Advisory befolgen. Dieses Dokument ist verfügbar unter http://developer.apple.com/iphone/download.action?path=/iphone/iphone_sdk_3.0__final/iphone_os_3.0.1_advisory.pdf.

Nach dem Download kann der heruntergeladene Installer mittels Doppelklick gestartet werden. Hierbei wird das dmg-Archiv geöffnet, was aufgrund der Größe etwas Zeit in Anspruch nehmen kann. Das geöffnete Archiv präsentiert sich mit den drei Einträgen *About iPhone SDK*, *iPhone SDK* und *Packages*.

Die PDF-Datei *About iPhone SDK* enthält eine kurze englischsprachige Erläuterung zur heruntergeladenen Version des iPhone SDK.

Der Installer für das iPhone SDK (Name: *iPhone SDK*) steht im mpkg-Format zur Verfügung (ein Meta Package enthält gebündelt alle Dateien für eine Installation).

Im Ordner *Packages* befinden sich zusätzliche Pakete (pkg-Format) für die Installation.

Die endgültige Installation wird per Doppelklick auf das mittige Icon *iPhone SDK* gestartet. Zusätzlich zu Kommandozeilenwerkzeugen wird bei der Installation des iPhone SDK auch noch die Entwicklungsumgebung Xcode in der Version 3.1.3 installiert.

About iPhone SDK

iPhone SDK

Packages

Installation starten

Vorsicht bei einer alten Version von Xcode

Standardmäßig wird seit Mac OS X 10.5 die Entwicklungsumgebung Xcode im Verzeichnis /Developer installiert. Sollte auf Ihrem Rechner bereits eine alte Xcode-Version installiert sein, wird sie durch die Installation von Xcode in der Version 3.1.3 aktualisiert. Besteht der Wunsch zum Parallelbetrieb, müssen Sie für die neue Version ein anderes Verzeichnis (Variante Custom Install) angeben; falls Sie die neue Version aber im Verzeichnis /Developer installieren möchten, müssen Sie die alte vor der Installation in ein anderes Verzeichnis verschieben.

Nach kurzer Zeit erscheint das Fenster des Installers (siehe Abb. 2–4). Der Installer ist im Stile eines typischen Wizards aufgebaut. Das Fenster kann in drei Bereiche eingeteilt werden:

- Bereich 1: Anzeige des aktuellen Schrittes innerhalb der Navigation. Die einzelnen Punkte (z.B. Einführung oder Lizenz) können nicht angeklickt werden, sondern dienen lediglich der Übersicht.
- Bereich 2: Der eigentliche Inhaltsbereich des Installers
- Bereich 3: Der Navigationsbereich mit den Schaltflächen für die Steuerung (Vor oder Zurück) in der Installation

Bei der Verwendung der deutschen Version des Installationsprogramms sollten Sie sich nicht davon verwirren lassen, dass einige Informationen während der Installation trotzdem in englischer Sprache erscheinen (z.B. die Lizenz für das iPhone SDK).

Abb. 2–4

Der Installer des iPhone SDK, aufgeteilt in drei Bereiche



Zu Beginn der Installation werden Ihnen nacheinander die Lizenz zu Xcode und zum iPhone SDK präsentiert. Diesen Lizenzen müssen Sie für eine Installation zustimmen.

Bei dem Schritt *Installationstyp* sollten Sie beachten, dass standardmäßig bereits eine Auswahl vorselektiert ist, die in den meisten Fällen auch passend ist. Das Paket *Developer Tools* kann nicht deaktiviert werden. Es enthält unverzichtbare Elemente für die Entwicklung einer Anwendung mit dem iPhone SDK, wie z.B. Xcode. Zusätzlich sind die Pakete *iPhone SDK*, *System Tools* und *UNIX Development Support* selektiert. Die Tabelle 2–1 enthält eine kurze Beschreibung der sechs Pakete des Installationsdialoges.

Name des Paketes	Beschreibung
Developer Tools Essentials 5,2 GB	Installiert Xcode, Interface Builder, Instruments, Dashcode, Quartz Composer, GCC 4.0.1, GCC 4.2.1, LLVM-GCC 4.2.1, GDB und weitere Entwicklungswerkzeuge. Zusätzlich wird das SDK für Mac OS X 10.4 und Mac OS X 10.5 installiert. Für dieses Paket kann das Verzeichnis der Installation über »Ort« gesetzt werden. Standardmäßig ist das Basisverzeichnis auf /Developer gesetzt.
iPhone SDK 23,9 MB	Entwicklungswerkzeuge, Header-Dateien, Bibliotheken und Dokumentation für die Anwendungsentwicklung. Die Installation erfolgt innerhalb des /Platforms-Verzeichnisses im Entwicklungsverzeichnis, standardmäßig ist dies /Developer/Platforms. Das iPhone SDK benötigt einen Intel-basierten Mac mit mindestens Version 10.5.7 (oder höher) von Mac OS X.
System Tools 95,4 MB	Beinhaltet generelle Werkzeuge wie Shark zur Messung der Performance. Shark ist Bestandteil der CHUD-Werkzeugsammlung (Computer Hardware Understanding Development) rund um das Thema Performance. Zusätzlich werden DTrace-Komponenten installiert. Wichtig ist, dass nur eine Version (und zwar immer die neueste) der System Tools installiert sein kann. Diese wird auch immer im /Developer-Verzeichnis abgelegt.
UNIX Development Support 596 MB	Enthält zusätzliche Werkzeuge zur Entwicklung auf der Kommandozeile. Installiert einen zusätzlichen GCC-Compiler und Kommandozeilenwerkzeuge in Verbindung mit dem reinen Xcode-Entwicklungspaket. Daneben werden Header-Dateien, Bibliotheken und andere Dateien für die Softwareentwicklung mit Mac OS X installiert. Dieses Paket wird mit Shellskripten ausgeliefert, die auf Dateien in vordefinierten Verzeichnissen zugreifen; darum ist eine Änderung des Installationsortes nicht möglich.
Mac OS X 10.3.9 Support 75,3 MB	Unterstützung für die Anwendungsentwicklung für Mac OS X 10.3.9. Enthält die Apple-Version 3.3 von GCC und das SDK für Mac OS X. GCC 3.3 kann nur im vorgegebenen Verzeichnis installiert werden.
Web Objects 370 MB	Entwicklungswerkzeuge, Beispiele und Dokumentation zu Web Objects. Der Installationsort kann nicht geändert werden und ist /Developer.

Tab. 2–1

Die Distribution des iPhone SDK enthält mehr als nur das reine SDK. Die Bestandteile können bei der Installation ausgewählt werden.

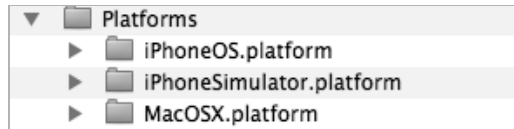
Für die Installation der vorselektierten Pakete werden insgesamt ungefähr 5,9 GB Speicherplatz benötigt. Eine Änderung des Installationsverzeichnis ist möglich, sollte vorher aber gründlich überdacht werden. Für die ersten Schritte ist es sicherlich sinnvoll, die Verzeichnisse beizubehalten. Sollten Sie später doch noch ein anderes Verzeichnis vorziehen, können Sie das iPhone SDK immer noch deinstallieren und in einem neuen Verzeichnis installieren.

Nach der Auswahl der entsprechenden Pakete wird die letztendliche Installation der einzelnen Pakete über *Fortfahren* gestartet. Das Ende der Installation wird mit der Meldung *Installation erfolgreich* bestätigt.

Nach der erfolgreichen Installation des iPhone SDK sollte unterhalb des gewählten Basisverzeichnisses mindestens die in der nachfolgenden Abbildung dargestellten drei Verzeichnisse vorhanden sein. Ohne Änderung ist das Basisverzeichnis das Verzeichnis `/Developer`.

Abb. 2-5

Die Verzeichnisse nach der
Installation

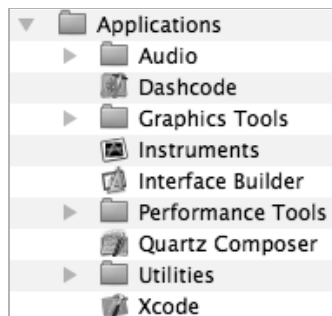


Wie am Namen schon zu erkennen, befinden sich im Verzeichnis `iPhoneOS.platform` Dateien rund um das sogenannte iPhone OS, das Betriebssystem des iPhones. Im Verzeichnis `iPhoneSimulator.platform` sind die Dateien zum iPhone Simulator enthalten.

Zusätzlich zum iPhone SDK werden, wie schon erwähnt, die Entwicklungsumgebung Xcode und zusätzliche Tools installiert. Diese befinden sich unterhalb des Verzeichnisses `/Developer/Applications`.

Abb. 2-6

Werkzeuge des
iPhone SDK



2.3 Das iPhone SDK deinstallieren

In der kurzen Startdokumentation¹ für das iPhone SDK wird beschrieben, dass bei der Installation einer neuen Version des iPhone SDK die letzte Version überschrieben wird. Falls Sie auf Nummer sicher gehen möchten, besteht auch die Möglichkeit, ein bereits installiertes iPhone SDK vorher zu entfernen. Für eine Deinstallation stehen vier unterschiedliche Varianten zur Verfügung, die inklusive einer kurzen Erläuterung in der nachfolgenden Tabelle 2–2 beschrieben werden. Die Erläuterungen gehen davon aus, dass das iPhone SDK im vorgegebenen Standardverzeichnis `/Developer` installiert wurde. Falls das Verzeichnis geändert wurde, müssen Sie den Basispfad `/Developer` in den Befehlen anpassen.

Beschreibung	Abzusetzender Befehl ^a
Komplette Deinstallation ^b des iPhone SDK und der dazugehörigen Komponenten, d.h. inklusive der Entwicklungsumgebung Xcode.	<code>sudo /Developer/Library/ uninstall-devtools --mode=c=all</code>
Deinstallation des reinen iPhone SDK. Mit dieser Variante wird die Entwicklungsumgebung Xcode beibehalten.	<code>sudo /Developer/Library/ uninstall-devtools --mode= systemsupport</code>
Deinstallation des Unix Development Support. Die anderen Bestandteile, wie z.B. Xcode, werden beibehalten.	<code>sudo /Developer/Library/ uninstall-devtools --mode=unixdev</code>
Deinstallation von Xcode. Neben dem angegebenen Befehl kann für eine Deinstallation auch einfach der Xcode-Ordner auf den Papierkorb gezogen werden.	<code>sudo /Developer/Library/ uninstall-devtools --mode=xcodedir</code>

Tab. 2–2

*Varianten zur
Deinstallation des
iPhone SDK*

- Die angegebenen Befehle sind über das Terminal auszuführen.
- Eine Deinstallation ist selbstverständlich nur mit Administratorrechten möglich.
- Ohne Angabe des Parameters `mode` wird immer `--mode=all` herangezogen.

Die Deinstallation des kompletten Paketes kann einige Minuten in Anspruch nehmen. Das Terminalfenster sollten Sie während dieser Deinstallation nicht schließen, um den Prozess auch sauber beenden zu können.

1. Das iPhone SDK Readme ist online verfügbar unter der URL http://developer.apple.com/iphone/download.action?path=/iphone/iphone_sdk_3.0_final/iphone_os_3.0_sdk_readme.pdf.

2.4 Ein erstes Beispiel

Als Einstiegsbeispiel habe ich nicht die übliche Hello-World-Anwendung gewählt, sondern eine einfache Anwendung zur Verwaltung der eigenen alltäglich anstehenden Aufgaben. Der Name des kleinen Helferleins soll *i.do* sein. Als Basis verwenden wir dafür die Anwendung *SQLiteBooks*. Dieses Beispiel soll einen ersten Einblick geben, wie die Entwicklung mit dem iPhone SDK aussieht. Die grundlegenden Erläuterungen folgen in den späteren Kapiteln.

Bevor ich Ihnen die wichtigsten Schritte bei der Entwicklung der Anwendung *i.do* beschreibe, soll kurz aufgezeigt werden, welche Funktionen die Anwendung beinhalten soll:

- Eingabe und Änderung einer Aufgabe (eine Aufgabe soll einen Titel und ein Fälligkeitsdatum besitzen)
- Anzeige der einzelnen Aufgaben in einer Liste
- Löschen einer Aufgabe

Der Name des verwendeten Beispiels *SQLiteBooks* (in der Version 1.8) rührt übrigens daher, dass die Anwendung die interne SQL-Datenbank SQLite (<http://www.sqlite.org/>) verwendet. Die Maske in Abbildung 2-7 zeigt die Eingabe einer einzelnen Aufgabe.

Abb. 2-7

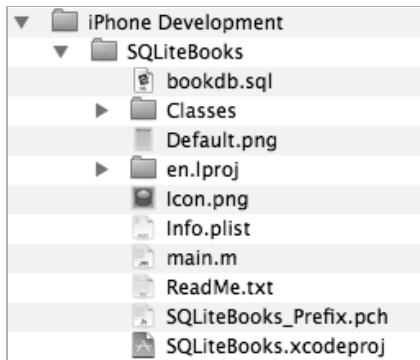
Details einer Aufgabe
in *i.do*



Dieses erste Beispiel geht nicht näher auf die Details der Entwicklung einer Anwendung für das iPhone ein, sondern zeigt lediglich die Schritte, die notwendig sind, um eine erste lauffähige Version der Anwendung zu erhalten.

Wie bereits zu Anfang erwähnt, wird die Anwendung nicht komplett neu entwickelt, sondern auf Basis einer Beispielanwendung, die ursprünglich aus dem iPhone Dev Center stammt, aber seit der Einführung des iPhone OS 3 dort nicht mehr angeboten wird. Daher laden Sie das Beispiel von der Buchwebseite (<http://www.dpunkt.de/iphone>, Link »SQLiteBooks«) herunter oder unter <http://www.dpunkt.de/leseproben/3261/SQLiteBooks.dmg>.

Dann extrahieren Sie dieses Archiv in ein beliebiges Verzeichnis. Nach dem Entpacken präsentiert sich das Beispiel wie in der Abbildung 2–8 (dieser Beschreibung liegt die Datei SQLiteBooks.zip in der Version 1.8 vom 7.7.2008 zugrunde).



Schritt 1:

Der Download

Abb. 2–8

Das entpackte Beispiel

SQLite Book List

Für die Bearbeitung der Anwendung öffnen Sie sie über einen Doppelklick auf die Datei SQLiteBooks.xcodeproj. Nach diesem Doppelklick öffnet sich nach kurzer Zeit die Entwicklungsumgebung Xcode mit dem Projekt SQLiteBooks (siehe Abb. 2–9).

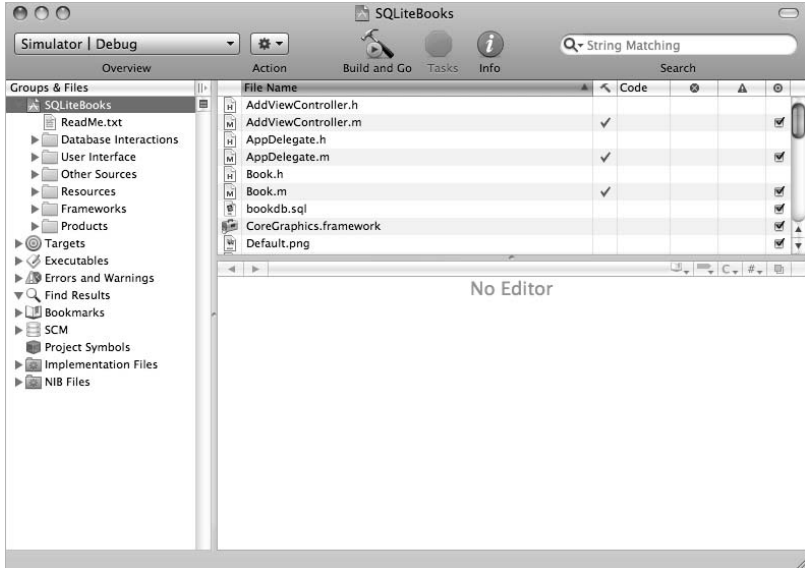
Schritt 2:

Entwicklungsumgebung

öffnen

Abb. 2-9

Xcode mit dem geöffneten
Projekt SQLiteBooks

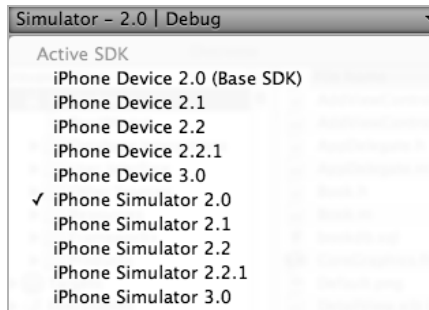


Schritt 3:
Die Version des iPhone OS
festlegen

Mit dem iPhone SDK 3 besteht die Möglichkeit, eine Anwendung für eine ältere Version des iPhone OS zu entwickeln. Also sollten Sie sich zunächst für eine Version entscheiden. Dies tun Sie über das linke obere Auswahlfeld in Xcode (siehe Abb. 2-10).

Abb. 2-10

Auswahl der Zielversion,
für die die Anwendung
erzeugt werden soll



Für unser Beispiel wählen Sie die Version iPhone Simulator 3.0 aus.

Schritt 4:
Das Beispiel mit den
Änderungen starten

Um die Anwendung ein erstes Mal zu starten, müssen Sie das Beispiel übersetzen. Dies erfolgt mittels Klick auf die Schaltfläche *Build and Go* (siehe Abb. 2-11).

**Abb. 2-11**

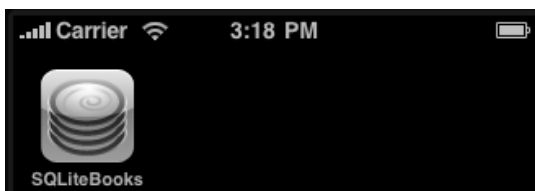
Das Beispiel übersetzen
und starten

Nach kurzer Zeit ist der Quellcode übersetzt und die Anwendung wird im mitgelieferten iPhone Simulator gestartet. Danach präsentiert sich sofort die Anwendung SQLiteBooks (Abb. 2-12). Den iPhone Simulator können Sie per Maus bedienen. Der Klick auf die Pfeile in der Liste zeigt den entsprechenden Inhalt an. Über das »+«-Zeichen in der Navigationsleiste wird ein neuer Eintrag hinzugefügt.

**Abb. 2-12**

SQLiteBooks gestartet

Bei der Betätigung des schwarzen iPhone-Knopfes in der Mitte des unteren Randes kommt der Nutzer auf das Hauptmenü zurück. Im Hauptmenü ist die Beispielanwendung SQLiteBooks mit einem eigenen Icon vertreten (siehe Abb. 2-13).

**Abb. 2-13**

SQLiteBooks im
Hauptmenü

Schritt 5: Für das Schließen der Anwendung muss der iPhone Simulator über den Menüpunkt *iPhone-Simulator beenden* geschlossen werden, alternativ funktioniert auch die Tastenkombination *cmd+Q*. Nach dem Beenden können Sie nun Änderungen an der Anwendung vornehmen.

Schritt 6: Für die Eingabe einer neuen Aufgabe in unserer Beispielanwendung *i.do* werden wir die vorhandene Maske zur Eingabe eines Buches verwenden (siehe Abb. 2-14). Ziel ist es, die Maske an die zu Anfang dieses Abschnitts gezeigte Darstellung anzupassen (siehe Abb. 2-7).

Abb. 2-14
Eingabe der Daten zu
einem Buch



Zunächst gilt es herauszufinden, an welcher Stelle die Maske definiert ist. Dies ist die Datei `DetailViewController.m` im Verzeichnis `classes`. In Xcode finden Sie diese Datei im logischen Ordner *User Interfaces* innerhalb von *Groups & Files*. Zur Darstellung wird eine Tabelle mit drei Bereichen verwendet. Zuerst soll die Überschrift der Maske geändert werden. Die Methode, die hierfür modifiziert werden muss, ist `initWithNibName`.

```

- (id)initWithNibName:(NSString *)nibNameOrNil
bundle:(NSBundle *)nibBundleOrNil {
    if (self = [super initWithNibName:nibNameOrNil
bundle:nibBundleOrNil]) {
        self.title = @"Info";
        dateFormatter = [[NSDateFormatter alloc] init];
        [dateFormatter setDateStyle:NSDateFormatterMediumStyle];
        [dateFormatter setTimeStyle:NSDateFormatterNoStyle];
    }
    return self;
}

```

Listing 2-1

Die nicht geänderte
Methode
initWithNibName

Die Änderung besteht darin, der Property title den Wert ToDo Detail anstatt Info zuzuweisen.

```

- (id)initWithNibName:(NSString *)nibNameOrNil
bundle:(NSBundle *)nibBundleOrNil {
    if (self = [super initWithNibName:nibNameOrNil
bundle:nibBundleOrNil]) {
        self.title = @"ToDo Detail";
        dateFormatter = [[NSDateFormatter alloc] init];
        [dateFormatter setDateStyle:NSDateFormatterMediumStyle];
        [dateFormatter setTimeStyle:NSDateFormatterNoStyle];
    }
    return self;
}

```

Listing 2-2

Die Methode
initWithNibName mit
dem neuen Titel

Für unsere Anwendung benötigen wir nur zwei Bereiche: einen für den Namen der Aufgabe und einen für das Datum, bis zu dem die Aufgabe erledigt sein soll. Somit muss ein Bereich ausgeblendet werden. Die Methode, die geändert werden muss, ist numberOfSectionsInTableView.

```

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tv {
    // 3 sections, one for each property
    return 3;
}

```

Listing 2-3

Die ursprüngliche
Methode
numberOfSections
inTableView

Die erste Änderung besteht darin, den Rückgabewert auf den Wert 2 zu ändern. Dadurch werden in der Darstellung nur noch zwei Elemente erwartet.

```

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tv {
    // 2 sections, one for each property
    return 2;
}

```

Listing 2-4

Die geänderte Methode

Nun ändern wir noch die übrigen Beschriftungen der Maske. Hierzu müssen Sie die Methode tableView in DetailViewController.m ausfindig machen.

Listing 2-5

Die unveränderte
Methode tableView

```
-(NSString *)tableView:(UITableView *)tv
titleForHeaderInSection:(NSInteger)section {
    switch (section) {
        case 0: return @"Title";
        case 1: return @"Copyright";
        case 2: return @"Author";
    }
    return nil;
}
```

Für die Anpassung müssen Sie lediglich die zwei Beschriftungen ändern. Zusätzlich können Sie die dritte Beschriftung löschen, denn die Eingabemaske hat nur zwei Bereiche. Die Bezeichnungen dieser zwei Bereiche (❶ und ❷) sind nun an die neue Anwendung angepasst.

Listing 2-6

Die geänderte Methode
tableView

```
-(NSString *)tableView:(UITableView *)tv
titleForHeaderInSection:(NSInteger)section {
    // Return the displayed title for the specified section.
    switch (section) {
        case 0: return @"ToDo"; ❶
        case 1: return @"Erledigung bis"; ❷
    }
    return nil;
}
```

Die Detailanzeige einer Aufgabe ist somit abgeschlossen. Nun geht es noch darum, kleine Änderungen für das Hinzufügen einer Aufgabe durchzuführen. Die notwendige Datei hierfür ist `AddViewController.m`. Die erste Änderung betrifft wieder die Überschrift der Maske. Die Änderung müssen Sie in der Methode `viewDidLoad` vornehmen.

Listing 2-7

Die unveränderte
Methode viewDidLoad

```
-(void)viewDidLoad {
    self.title = @"New Book";
    self.navigationItem.leftBarButtonItem = [[[UIBarButtonItem
alloc] initWithBarButtonSystemItem:UIBarButtonSystemItemCancel
target:self action:@selector(cancel:)] autorelease];
    self.navigationItem.rightBarButtonItem = [[[UIBarButtonItem
alloc] initWithBarButtonSystemItem:UIBarButtonSystemItemSave
target:self action:@selector(save:)] autorelease];
}
```

Auch hier müssen Sie nun einfach wieder die Property `title` auf den neuen Wert `New ToDo` ändern.

```

- (void)viewDidLoad {
    self.title = @"New ToDo";
    self.navigationItem.leftBarButtonItem = [[[UIBarButtonItem
alloc] initWithBarButtonSystemItem:UIBarButtonSystemItemCancel
target:self action:@selector(cancel:)] autorelease];
    self.navigationItem.rightBarButtonItem = [[[UIBarButtonItem
alloc] initWithBarButtonSystemItem:UIBarButtonSystemItemSave
target:self action:@selector(save:)] autorelease];
}

```

Listing 2-8

Die Methode
viewDidLoad mit der
geänderten Überschrift

Um eine neue Aufgabe hinzuzufügen, muss sie über die Schaltfläche *Save* abgespeichert werden. Diese Schaltfläche wird in der Methode `viewWillAppear` abhängig vom Inhalt aktiviert bzw. deaktiviert. Da wir nun nur noch zwei (statt drei) Werte eingeben können, müssen wir die Methode anpassen.

```

(void)viewWillAppear:(BOOL)animated {
    [super viewWillAppear:animated];
    // Conditionally enable the "Save" button
    self.navigationItem.rightBarButtonItem.enabled =
(self.book.title && self.book.title.length > 0 &&
self.book.copyright && self.book.author &&
self.book.author.length > 0);
}

```

Listing 2-9

Die unveränderte
Methode viewWillAppear

Für die Änderungen müssen Sie nun die Prüfungen auf Inhalte von `self.book.author` entfernen.

```

- (void)viewWillAppear:(BOOL)animated {
    [super viewWillAppear:animated];
    // Conditionally enable the "Save" button
    self.navigationItem.rightBarButtonItem.enabled =
(self.book.title && self.book.title.length > 0 &&
self.book.copyright);
}

```

Listing 2-10

Die angepasste Methode
viewWillAppear

Die letzte Änderung an der Anwendung soll nun die Überschrift am Hauptfenster sein. Hierfür müssen Sie die Methode `viewDidLoad` in der Datei `MasterViewController.m` erweitern.

```

- (void)viewDidLoad {
    self.navigationItem.rightBarButtonItem = self.editButtonItem;
}

```

Listing 2-11

Die unveränderte
Methode viewDidLoad

Der Unterschied zu den zwei Überschriften zuvor ist, dass diese Überschrift nicht im Quelltext gepflegt wird, sondern in der Oberflächendefinitionsdatei (mehr dazu in Kapitel 4). Die Werte aus der Oberflächendefinitionsdatei können aber überschrieben werden. Hierfür

müssen Sie lediglich wieder die Ihnen schon bekannte Property `title` auf den Wert `ToDo`s setzen.

Listing 2-12

Die angepasste Methode
`viewDidLoad`

```
- (void)viewDidLoad {
    self.navigationItem.rightBarButtonItem = self.editButtonItem;
    self.title=@"ToDo"s;
}
```

Es wären sicherlich noch einige Änderungen notwendig, um aus `i.do` eine ausgewachsene Anwendung zu machen. Aber für eine erste eigene Anwendung soll dies genügen. Die Änderungen können Sie wieder über *Build and Go* in Xcode testen.

2.5 Beispiele aus dem iPhone Dev Center nutzen

Für einen schnellen Einstieg in die Programmierung fürs iPhone bieten sich die Beispiele aus dem iPhone Dev Center an. Diese erreichen Sie über den Link <http://developer.apple.com/iphone/library/navigation/SampleCode.html>. Auf der Übersichtsseite sehen Sie alle Beispiele mit ihrer jeweiligen thematischen Einordnung. Bei der Auswahl eines Beispiels (z.B. das klassische »HelloWorld«) erscheint das selektierte Beispiel.

Beschreibungsseite eines
Beispiels

Die Einzelseiten der Beispiele sind alle nach dem gleichen Muster aufgebaut (siehe Abb. 2-15). Nach der Überschrift mit dem Namen des Beispiels enthält die nächste Zeile die aktuelle Versionsnummer. Darauf folgt das Datum, an dem die Anwendung in das iPhone Dev Center hochgeladen wurde (Beschriftung: *Posted*). In der Zeile *Build Requirements* folgt eine Aufzählung der Anforderungen an das Entwicklungssystem, um das zur Verfügung gestellte Beispiel selbst zu erzeugen.

Die letzte Informationszeile (Beschriftung: *Runtime Requirements*) enthält die notwendigen Anforderungen an das System, auf dem die Beispielanwendung ablaufen soll. Eine interessante Funktion steht über die Auswahlliste bei *View Source Code* zur Verfügung: Hier können Sie die einzelnen Quelldateien des Beispiels direkt im Browser betrachten. Zuletzt wird ein Link für den Download des Beispiels in einem Zip-Archiv angeboten. Unterhalb dieses Links befinden sich eine kurze Beschreibung und eine Versionshistorie des Beispiels.

HelloWorld

Version: 1.7
 Posted: 2008-07-07
 Build Requirements: Mac OS X 10.5.3, Xcode 3.1, iPhone OS 2.0
 Runtime Requirements: Mac OS X 10.5.3, iPhone OS 2.0
 View Source Code:
 Download Sample Code ("HelloWorld_iPhone.zip", 346.4K)

Description
 HelloWorld demonstrates how to use a keyboard to enter text into a text field and how to display the text in a label.

Document Revision History

Date	Notes
2008-07-07	First public release.

Abb. 2-15

Aufbau der Seiten
 der Beispiele im
 iPhone Dev Center

Alle Beispiele können Sie wie das bereits verwendete SQLiteBooks nach dem Extrahieren direkt über die enthaltene .xcodeproj-Datei in Xcode starten und übersetzen.

2.6 Zusammenfassung

In diesem Kapitel habe ich Ihnen zunächst das iPhone Dev Center als das zentrale Informationszentrum rund um die iPhone-Entwicklung vorgestellt. Danach haben wir einen kurzen Blick auf die Installation sowie die Deinstallation des iPhone SDK geworfen. Anschließend haben wir auf Basis des im iPhone Dev Center zur Verfügung gestellten Beispiels SQLiteBooks eine erste eigene Anwendung entwickelt. Hier haben wir die Buchverwaltung zu einer einfachen Aufgabenliste mit dem Namen i.do umgewandelt. In die Tiefen des iPhone SDK ist dieses erste Beispiel noch nicht vorgedrungen, aber trotzdem sollten Sie nun schon ein erstes Gefühl für die Anwendungsprogrammierung mit dem iPhone SDK entwickelt haben. Außerdem wissen Sie nun, wo Sie innerhalb des iPhone Dev Center den Quellcode von weiteren Anwendungen finden und wie Sie ihn schnell testen können.