Norbert Usadel

Know-how ist blau.

Xcode 4.2 für iOS 5

Mit Xcode 4.2 und Objective-C fürs iPhone programmieren

Einführung in die Software-Entwicklung für iOS 5

> Die neuen Features von Xcode 4.2
> Schnelle Einführung in Objective-C
> Alle Schritte der App-Entwicklung – vom Entwurf bis zum App-Store

14 Workshops zeigen Ihnen, wie Sie Ihre iPhone-App selbst entwickeln.

FRANZIS

Norbert Usadel

Mit Xcode 4.2 und Objective-C fürs iPhone programmieren

Norbert Usadel

Mit Xcode 4.2 und Objective-C fürs iPhone programmieren

Einführung in die Software-Entwicklung für iOS 5

Mit 158 Abbildungen

Bibliografische Information der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;
detaillierte Daten sind im Internet über http://dnb.ddb.de abrufbar.

Lektorat: Anton Schmid
Satz: DTP-Satz A. Kugge, München
art & design: www.ideehoch2.de
Druck: Bercker, 47623 Kevelaer
Printed in Germany

ISBN 978-3-645-60137-5

Inhaltsverzeichnis

1 Über dieses Buch

Dieses Buch handelt von Xcode 4.2, der neuen Entwicklungsumgebung für Mac OS X 10.7 und für iOS 5, das Betriebssystem für iPhones und iPads, welches im Herbst 2011 auf den Markt kommt.

Der Schwerpunkt des Buches liegt auf der App-Entwicklung für das iPhone.

Der Theorieteil ist sehr knapp gehalten. Der Leser erwirbt grundlegende Kenntnisse des Programmierens und der neuen Entwicklungsumgebung durch die zahlreichen Workshops, die dieses Buch anbietet.

Das Buch ist brandaktuell, weil es neben Xcode 4.2 und iOS 5 auch viele neue APIs des Betriebssystems vorstellt. Auf Neuerungen wie Storyboards in Xcode oder die neuen Features des Compilers wird ebenso eingegangen wie auf die neue Umgebung des Interface-Builders.

1.1 Aufbau

Der Aufbau ist einfach. Das Buch beginnt mit einem Theorieteil und endet in einem langen Praxisteil. Der Leser erschließt sich durch das ständige »Machen« den Stoff, den Umgang mit dem neuen Xcode und der objektorientierten Programmierung. Es wurde darauf Wert gelegt, dass der Leser jetzt schon alles über die vielen Neuerungen erfährt, die das neue Betriebssystem iOS 5 und Xcode 4.2 mit sich bringen. So ist der Leser am Puls der Zeit und kann ganz neue Ideen für seine Apps entwickeln.

In den Workshops werden wichtige Themen wie Table-Views, Datenbanken für das iPhone, das Programmieren von Pickern, das Einbinden neuer Frameworks etc. behandelt. Sie stehen als einzelne Projekte da, auf die Sie später aufbauen können.

So erhalten Sie in jedem Workshop Ideen und ein Grundgerüst für Ihre Projekte. Wie gesagt, die Praxis steht im Vordergrund. Der Leser erhält nach dem erfolgreichen Abschluss eines Workshops ein Aha-Erlebnis. Die Code-Beispiele der Workshops können umgestellt und ausgebaut werden, so dass es dem Leser leicht fallen wird, Apps für seinen eigene Bedürfnisse entwickeln zu können.

1.2 Zielgruppe

Das Buch richtet sich sowohl an Einsteiger als auch an Entwickler, die die neue Entwicklungsumgebung von Xcode 4.2 und die Features von iOS 5 kennenlernen wollen.

Einsteiger steigen gleich mit der neuen Entwicklungsumgebung ein und haben am Ende des Buches einen Baukasten für die eigenen Projekte. Sie werden sehen, dass viele Entwickler auch nur mit Wasser kochen. Haben Sie dies erkannt, so ist der Weg zu eigenen Projekten, die in den Store gelangen sollen, frei.

1.3 Über den Autor

Lebt und schreibt im Ruhrgebiet. Studierte Architektur und Kunstgeschichte. Machte eine Ausbildung zum Programmierer und kam auf den Apple. Der Apple ließ ihn nicht mehr los. Er arbeitet seit fast 20 Jahren selbstständig im Apple-Bereich.

Der Autor schrieb in dieser Zeit Artikel über Podcasts, Datenverschlüsselung und GarageBand für die Zeitschrift MacPraxis. Ist Autor des Buches *Inside iPod*.

Der Autor schulte in diesem Abschnitt ca. 2000 Menschen über die Themen SAP, Cobol, Filemaker, Shopsysteme, iPods, iPhones, iTunes, Strukturierte Programmierung, Pagemaker, Photoshop, und App-Programmierung.

Er entwickelt Apps und übernimmt Projektleitungen für die App-Programmierung.

1.4 Website zum Buch

Falls Sie Fragen und Anregungen haben, besuchen Sie meine Website *www.appzitty.de*. Auf diesen Seiten dreht sich alles um die App-Programmierung. Im Bereich *www.appzitty.de/download* liegen die einzelnen Xcode-Projekte aus diesem Buch für Sie bereit. Sie können mich unter der E-Mail-Adresse buch@appzitty.de erreichen. Es würde mich freuen, auf diesem Weg Kontakt mit Ihnen aufnehmen zu dürfen.

Die Downloads zum Buch finden Sie außerdem auch auf *www.buch.cd*.

1.5 Danksagung

Man ahnt es ja gar nicht, aber um ein Buch möglich zu machen, ist auch hinter den Kulissen ein Haufen Leute nötig. Ich danke diesen Leuten. Auch wenn sie hier nicht namentlich genannt sind, sollten sie sich aber gerne angesprochen fühlen, wenn sie diese Zeilen lesen.

5 Objective-C

Ich fange mit Ihnen bei Null an. Sie sollten dieses Buch neben Ihrem Rechner liegen haben und die einzelnen Schritte mit Xcode 4 an Ihrem Rechner nachvollziehen. In diesem Kapitel bekommen Sie die Tipps und Tricks, mit denen Sie die Grundlagen der Programmiersprache erlernen. Gleichzeitig lernen Sie Ihre neue Arbeitsumgebung noch besser kennen. Also, auf geht's: Feuern Sie Ihr Xcode an.

5.1 Eine kleine Erfolgsgeschichte

Es begab sich aber zu der Zeit, als Steve Jobs seine ständig gewachsene Garagenfirma, die mittlerweile an der Börse war und die ihn über Nacht zum Multimillionär gemacht hatte, verlassen musste. Das war 1985. Jobs gründete daraufhin NeXT. Ziel war es, auf einem Unix-basierten Betriebssystem eine grafische Benutzeroberfläche zu schaffen. Um dieses Projekt durchzuführen, wurde Objective-C erfunden. Ziel war es unter anderem, das ganze Projekt objektorientiert durchzuführen. Hierzu wurde ein Interface-Builder geschaffen, mit dem man Objekte auf einer Arbeitsfläche zusammenstellen konnte. Der NIB, der Next-Interface-Builder, wurde so geschaffen. Programmierer kennen das *awake from nib* als Programmierbefehl. Der Befehl kommt aus dieser Zeit. Das ganze Konzept wurde NeXTStep getauft und war seiner Zeit weit voraus.

Natürlich gab es zu dem ganzen Softwarekonzept stylische Hardware. Die NeXT-Station und den NeXT-Cube. Die Hardware musste schwarz sein. In Bild 5.1 sehen Sie eine NeXT-Station.

Bild 5.1: Die NeXT-Station.

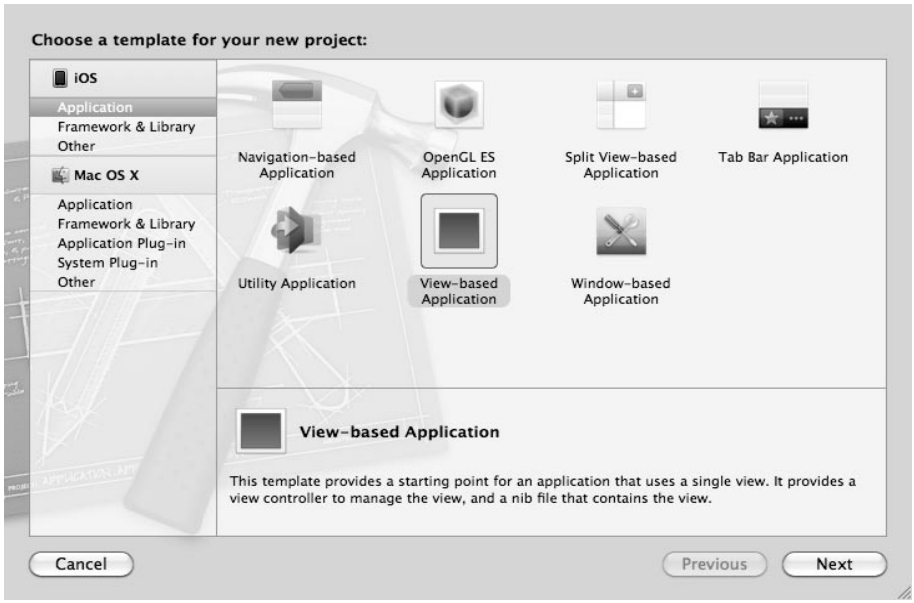
Im Jahre 1997, Apple ging es schlecht, ganz schlecht, wurde Steve Jobs als Retter in den Vorstand geholt und als CEO eingesetzt. Apple kaufte daraufhin NeXT, um das Betriebssystem dann auf Apple-Rechnern zu etablieren. Mac OS X, welches auf NeXTStep basiert, wurde zu einem Meilenstein in der Computergeschichte. Seitdem trat Objective-C als objektorientierte Programmiersprache ihren Siegeszug an. Alles wurde gut. Und wenn Sie nicht gestorben sind, programmieren sie noch heute in Objective-C.

5.2 Ihr Projekt für die Programm-Schnipsel

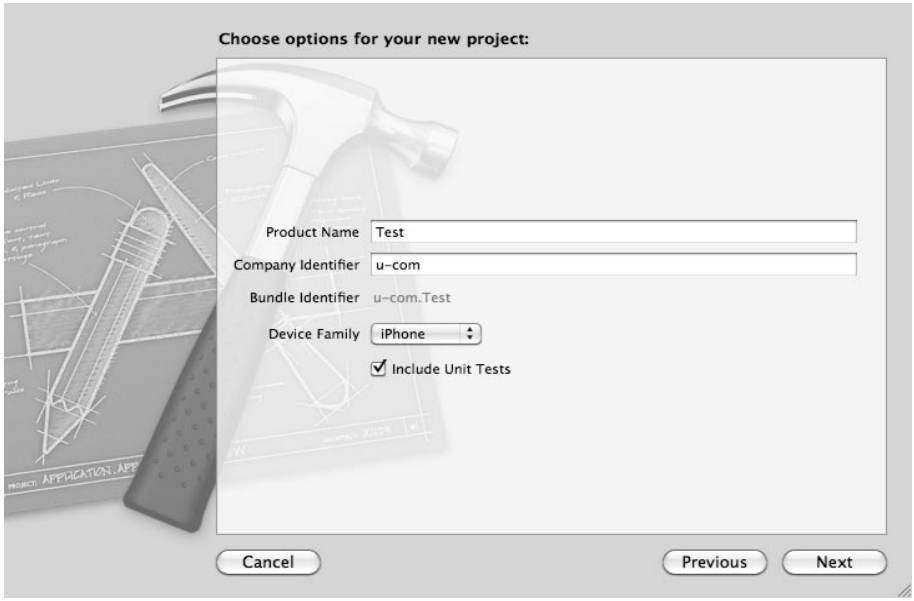
Haben Sie Ihr Xcode angefeuert? Starten Sie es per Doppelklick. In der rechten Seite des Fensters sehen die Projekte, die zuletzt benutzt wurden. In der linken Seite sehen Sie weitere Menüpunkte. Sie kommen so zum Entwicklerportal von Apple, können direkt die Xcode-Hilfe aufrufen oder die Versionskontrolle älterer Projekte starten.

Bitte wählen Sie nun *Create a new Xcode project.*

Sie kommen zum nächsten Fenster. In ihm haben Sie mehrere Vorlagen für verschiedene Projekte, mit denen Sie bestimmen können, für welchen Typ von Programm Ihr Projekt sein soll. Im unteren Beispiel ist es ein Projekt für das iPhone und die Vorlage *View-based Application* wird ausgesucht. Bestätigen Sie mit *Next.*

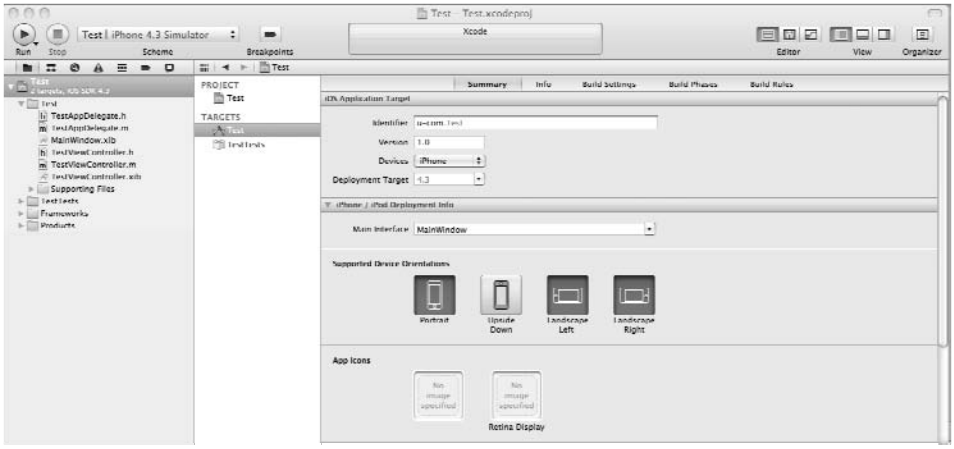


Im nächsten Fenster benennen Sie Ihr Projekt. Sie können dort auch einen Firmennamen eingeben. Nennen Sie Ihr Projekt *Test* und bestätigen Sie mit *Next*.

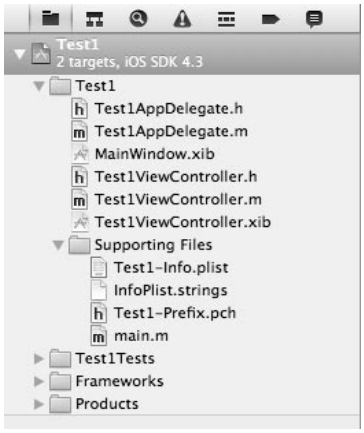


Nun können Sie den Ort auf der Festplatte bestimmen, an dem das Projekt abgespeichert werden soll. Bestätigen Sie mit dem Button *Create*.

Die Arbeitsumgebung von Xcode öffnet sich. Sie sehen in der linken Hälfte den Projektbaum mit der Ordnerstruktur, die Xcode für Sie angelegt hat. Die rechte Hälfte steht Ihnen für die Verwaltung Ihres Projekts zur Verfügung. Hier können Sie Versionsnummern, Icons und die Ausrichtung des Displays einsehen und verändern.



Betrachten Sie nun den Projektbaum und öffnen Sie den Ordner *Supporting Files*. In ihm befindet sich die Datei *main.m*. Diese Datei brauchen Sie, um die Code-Beispiele nachvollziehen zu können. In ihr geben Sie den Quelltext ein.

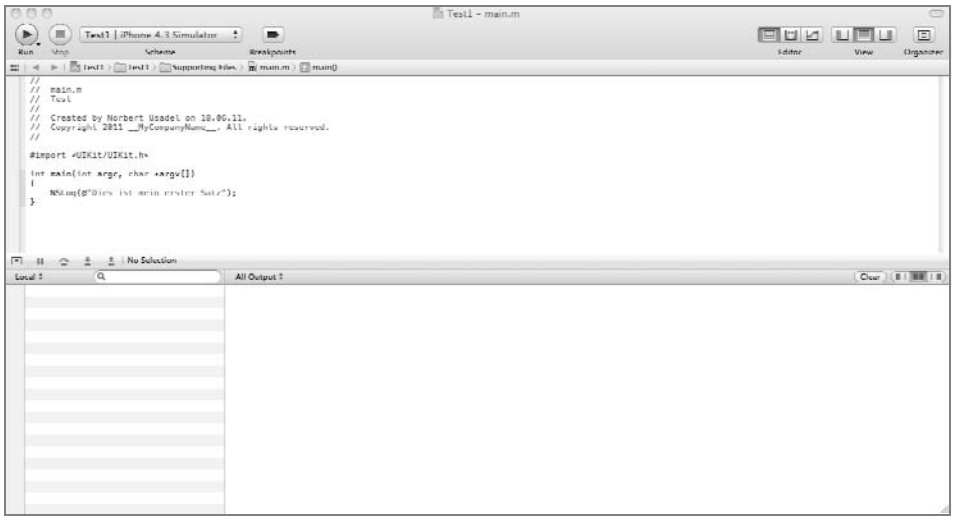


Öffnen Sie die Datei *main.m* mit einem Doppelklick. Sie sehen die Datei mit ein paar Zeilen Code. In dieser Datei setzen Sie die Code-Beispiele ein, die in diesem Kapitel gezeigt werden.

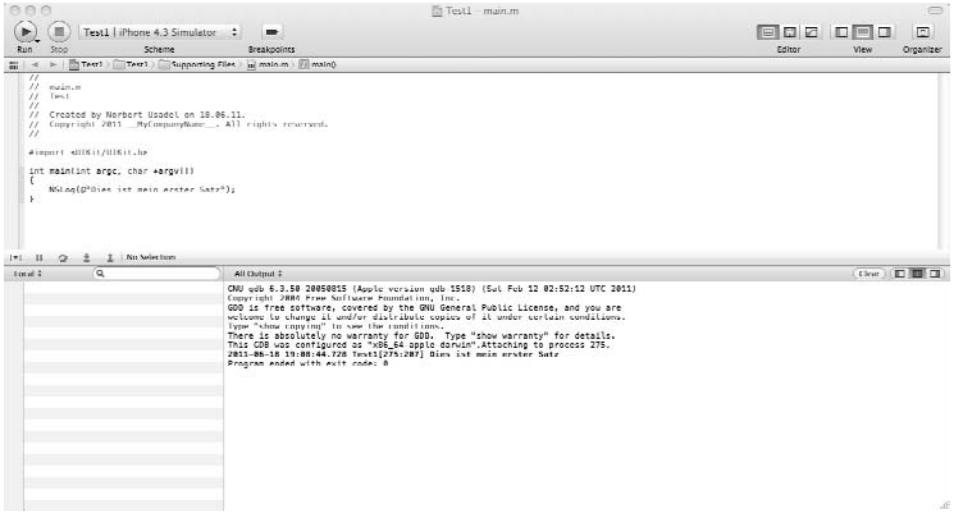
Die nächste Abbildung zeigt Ihnen genau, wie Sie die Beispiele des Quelltextes einsetzen müssen. Diese Zeile Code soll eingesetzt werden:

```
NSLog(@"Dies ist mein erster Satz");
```

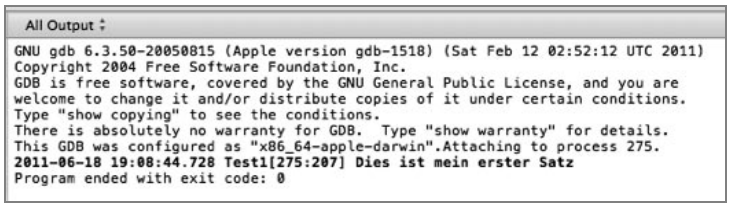
Sie fügen diese Zeile zwischen den beiden geschweiften Klammern ein, wie es Ihnen die nächste Abbildung zeigt. Den Rest, der zwischen den geschweiften Klammern stand, löschen Sie einfach weg.



Jetzt können Sie Ihr Projekt kompilieren. Drücken Sie dazu den *Run*-Button in der oberen Ecke des Fensters, und beachten Sie, was passiert. Ihr Projekt wird kompiliert, und durch den Befehl `NSLog` wird eine Bildschirmausgabe erzeugt, die Sie im unteren Konsolenfenster angezeigt bekommen.



In der nächsten Abbildung sehen sie nochmals einen Ausschnitt der Konsole mit der Ausgabe »Dies ist mein erster Satz«.



Sie haben ein Projekt erstellt, in dem Sie die im Weiteren aufgezeigten Programmier-Schnipsel eingeben und nachvollziehen können.

5.3 Variablen

Im vorherigen Abschnitt haben Sie mittels des `NSLog`-Befehls eine Ausgabe auf das Konsolenfenster erzeugt. Der Befehl wird eigentlich dazu verwendet, Fehlermeldungen anzuzeigen. Er eignet sich aber vorzüglich dazu, Bildschirmausgaben im Konsolenfenster zu erzeugen. Sie können mit ihm alle Programmierbeispiele nachvollziehen, solange Sie keine Programme mit einer GUI haben, die Bildschirmausgaben anzeigen.

Variablen werden in jeder Programmiersprache verwendet. Sie dienen als Platzhalter für Zahlen und Zeichenketten. Sie können vorab mit einem Wert belegt werden. Der Ausdruck y = 2 erzeugt eine Variable mit dem Namen y, die den Wert 2 hat. In Objective-C übersetzt sieht das Ganze dann so aus:

```
y = 2;
```

Das Semikolon beendet die Anweisung. Der Compiler weiß dann, dass es sich um eine Anweisung handelt, und kann der Variablen den Wert 2 zuweisen.

Variablen können aber auch nur zwei Werte enthalten. *Wahr* oder *Falsch* oder eben *Nein* oder *Ja*. In Objective-C werden diese mit dem Namen Bool bezeichnet. Die allgemeingültige Schreibweise zur Deklaration von Variablen ist diese:

```
datentype variablenname = wert;
```

Beispiele für die Deklaration von Variablen sehen in Quelltext übersetzt so aus :

```
int zahl = 2000;
char buchstabe = ' G '
float gleitkommazahl = 130.89
```

Im Folgenden erhalten Sie eine Übersicht über die verschiedenen Variablentypen, deren beanspruchten Speicherplatz und den Wertebereich:

Übersicht der Variablen:

Name	*Wertebereich / Beschreibung*	*Größe*
BOOL	NO oder YES	1 Byte
char	-128 bis 127 / Buchstaben, Sonderzeichen	1 Byte
unsigned char	0 bis 255 / Buchstaben, Sonderzeichen	1 Byte
short	-32768 bis 32767 / Ganzzahlen	2 Byte
unsigned short	0 bis 65538 / Ganzzahlen	2 Byte
int	-2147483648 bis 2147483647 / Ganzzahlen	4 Byte
unsigned int	0 bis 4294967295 / Ganzzahlen	4 Byte
long int	-2147483648 bis 2147483647 / Ganzzahlen	4 Byte
unsigned long	0 bis 4294967295 / Ganzzahlen	4 Byte
long long int	9223372036854775808 bis 9223372036854775807 / Ganzzahlen	8 Byte
unsigned long long	0 bis 18446744073709551615 / Ganzzahlen	8 Byte
float	1.2 E -38 bis 3.4 E +38 / Fließkommazahlen Genauigkeit 6 Stellen	4 Byte
double	2.3 E -308 bis 1.7 E +308 / Fließkommazahlen Genauigkeit 15 Stellen	8 Byte
long double	16 Byte Gleitkommazahl	16 Byte

Sie sehen also, dass es verschiedene Typen von Variablen gibt. Bevor Sie anfangen zu programmieren, müssen Sie sich darüber Gedanken machen, welche Variablen Sie für Ihr Programm benötigen.

In Ihren Vorüberlegungen sollte die Schreibweise von Variablen eine Rolle spielen:

Gute Variablen-Namen:

zahl8a

zah8l

za_hl

Nicht erlaubte Variablen-Namen:

zahl 3 (enthält ein Leerzeichen)

3zahl (beginnt mit einer Ziffer)

Nicht empfohlene Namen:

Zahl3 (beginnt mit einem Großbuchstaben)

Schreibweise:

In der Schreibweise von Variablen hat sich die sogenannte Kamelschreibweise durchgesetzt. Sie dient dazu, alles immer schön übersichtlich zu gestalten. Sie dürfen keine reservierten Wörter in den Variablen verwenden. Das sind die Wörter, die in Objective-C eine besondere Bedeutung haben.

Sie sollten bei der Vergabe Ihrer Variablen in der Mitte einen Großbuchstaben verwenden.

Zum Beispiel wäre schrottFeld ein schöner Variablen-Name. Diese Kamelschreibweise oder Binnenmajuskel-Verwendung in Variabeln dient dazu, Programmierfehler in der Syntax zu reduzieren.

Reservierte Wörter:

Die folgende Übersicht enthält die reservierten Wörter. Diese dürfen Sie nicht für Ihre Variablen benutzen. Sie sind der Programmiersprache Objective-C vorbehalten. Die Variablen dürfen nicht mit »NS« anfangen.

Die reservierten Wörter der Programmiersprache Objective-C:

asm	const	FALSE	int	SEL	typedef
auto	continue	float	long	short	union
bool	default	for	new	signed	unsigned
BOOL	do	goto	nil	sizeof	void
break	double	id	Nil	static	volatile
case	else	if	NO	struct	wchar_t
char	enum	IMP	register	switch	while
Class	extern	inline	return	TRUE	YES

5.3.1 Berechnungen mit Variablen

Natürlich können Sie, wie früher im Mathematikunterricht, mit Variablen Berechnungen anstellen. Dazu bedienen Sie sich folgender Vergleichsoperatoren:

```
+, // Addition
-, // Subtraktion
*, // Multiplikation
/. // Division

// dies ist übrigens ein Kommentar
```

Eine typische Berechnung sieht in Objective-C dann so aus:

```
int a = 20;
int b = 30;
int c;
c = a * b; NSLog (@" c=%i",c);
c = a / b; NSLog (@" c=%i",c);
```

Die Variable c nimmt den Wert Ihrer Berechnungen auf. Die Variablen a und b wurden vorher mit den Werten 20 und 30 belegt. Die Variable c hat also nach der Berechnung den Wert 600. Im zweiten Fall hat die Variable den Wert von 0,66.

Sie können sich die Berechnung in Xcode jederzeit in der main.m-Datei anzeigen lassen. Geben Sie, wie oben beschrieben, die Code-Zeilen ein. Die %i-Zeichen dienen zur Formatierung Ihrer Ausgabe.

5.4 Verzweigungen

Um ein Programm zu steuern, bedarf es Schleifen und Verzweigungen. Im Prinzip werden Werte verglichen, um über den Vergleich eine Auswahl zu treffen. Dazu benutzt man Vergleichsoperatoren.

Die Vergleichsoperatoren

Operator	*Beispiel*	*Beschreibung*
==	a == b	wahr, wenn a gleich b ist
!=	a != b	wahr, wenn a ungleich b ist
‹	a ‹ b	wahr, wenn a kleiner b ist
›	a › b	wahr, wenn a größer b ist
‹=	a ‹= b	wahr, wenn a kleiner oder gleich b ist
›=	a ›= b	wahr, wenn a größer oder gleich b ist

5.4.1 Die if-Verzweigung

```
if (condition) {
   statements // Block 1
}
else {
   Statements // Block 2-
}
```

Die Anweisungen in Block 1 werden nur ausgeführt, wenn die Bedingung »condition« wahr ist. Ist sie das nicht, wird in Block 2 weitergemacht.

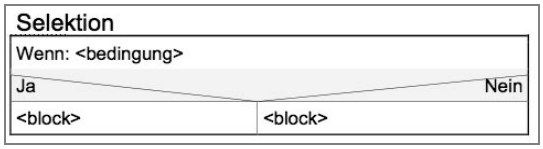


Bild 5.2: Die Selektion oder auch `if`-Abfrage.

So sieht der Programmcode für ein solches Konstrukt aus:

```
int main(int argc, char *argv[]){
   int a = 2;
   int b = 1;
   if (a==b){
      NSLog (@"a ist gleich b");
   }
else if (a<b){
      NSLog (@"a ist kleiner als b");
   }
else {
      NSLog (@"b ist gleich a");
   }
```

Die `if`-Verzweigung ist eine bedingte Anweisung, die prüft, ob etwas wahr ist. Ist die Bedingung wahr, wird nur ein bestimmter Programmteil ausgeführt.

5.4.2 Das case-Konstrukt

Man könnte die zuvor genannten `if`-Abfragen ineinander verschachteln. Da dies aber unübersichtlich wird, wird das `case`-Konstrukt benutzt.

```
int main(int argc, char *argv[]){
   int tagderwoche = 3;
   switch (tagderwoche)
   {
      case 1 : NSLog(@"Montag");
         break;
      case 2 : NSLog(@"Dienstag");
         break;
      case 3 : NSLog(@"Mittwoch");
         break;
      case 4 : NSLog(@"Donnerstag");
         break;
      case 5 : NSLog(@"Freitag");
         break;
      case 6 : NSLog(@"Samstag");
         break;
```

```
      case 7 : NSLog(@"Sonntag");
         break;
}
```

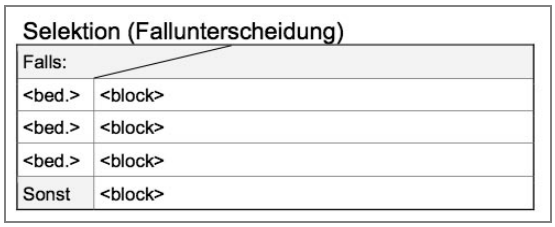


Bild 5.3: Das `case`-Konstrukt.

5.5 Schleifen

Ein Schleifendurchlauf wird so lange wiederholt, bis die Bedingung erfüllt ist. In Objective-C gibt es drei verschiedene Schleifenarten:

die `for`-Schleife

die `while`-Schleife

die `do-while`-Schleife

5.5.1 Die for-Schleife

Die `for`-Schleife nennt man auch Zähl-Schleife. Sie hat folgende Syntax:

```
for (Initialisierung; Bedingung; Endanweisung)
   Anweisung;
```

```
i = 1
for (i=1; i<=1000; i++)
   NSLog(@"$d", i)
```

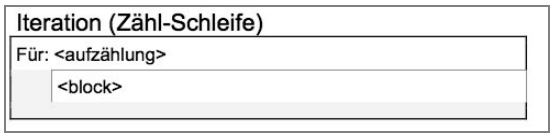


Bild 5.4: Die Zähl-Schleife.

Die Variable, die Sie als Zähler in der Schleife verwenden, muss eine Integerzahl, also eine Ganzzahl, sein.

5.7 Klassen

Klassen dienen dazu, das Rad nicht immer neu erfinden zu müssen. Programmierer erstellen Programmcode, den sie immer wieder aufrufen können, um bestimmte Routinen nicht neu programmieren zu müssen. In Objective-C gibt es eine ganze Reihe dieser Klassen.

Die Implementierung einer Klasse erfolgt in den .h- und .m-Dateien Ihres Projekts. Die allgemeingültige Syntax einer Klasse sieht so aus:

```
@interface ClassName : Superclass {
     //instance variable declarations
 }
 // method declarations
 @end
```

Die Klassendeklaration schreibt man zwischen `@interface` und `@end`. Nach dem Begriff interface folgt der Klassenname; nach dem Doppelpunkt die Basisklasse. Die Deklaration der Instanzvariablen erfolgt in dem Anweisungsblock.

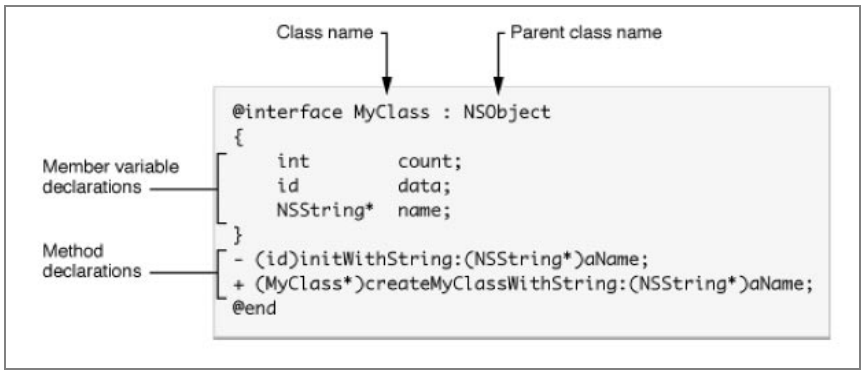


Bild 5.9: Die Schreibweise einer Klasse.

5.7.1 Implementation einer Klasse

Die Anweisung fängt mit `@implementation` an und hört mit `@end` auf.

Die allgemeine Schreibweise für die Implementierung einer Klasse:

```
@implementation ClassName
// method definitions
@end
```

Implementiert man eine Klasse, so erhalten die einzelnen Methoden der Klasse ihre Funktion. Sie werden über die Instanzvariablen angesteuert. Die einzelnen Methoden

der Klasse werden in den Rumpf des Gebildes und in Blöcken mit geschweiften Klammern gesetzt. Soll bei einer Methode ein Wert zurückgeliefert werden, schreibt man die Anweisung `return` in die Methode.

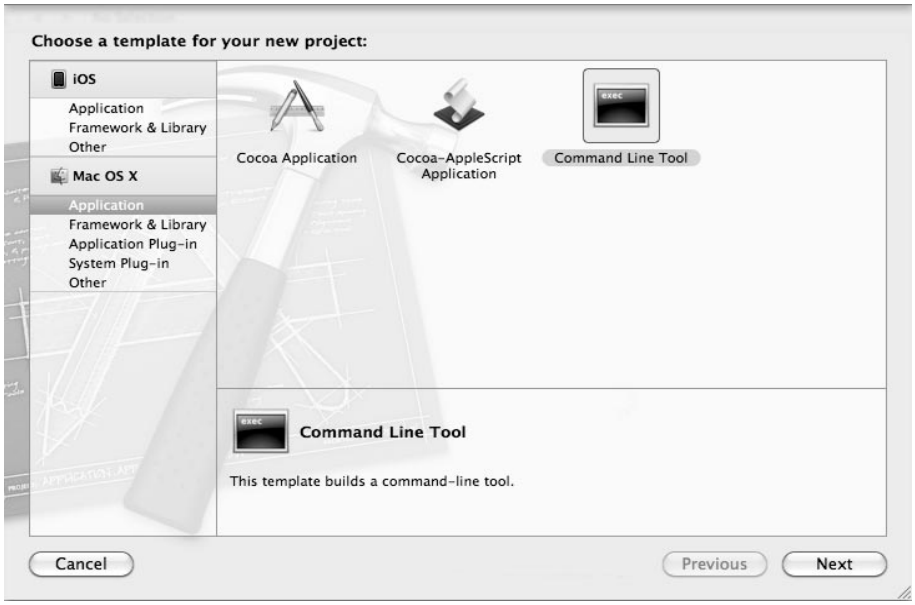
Beispiel für die Implementation einer Klasse:

```
@implementation MyClass
- (id)initWithString:(NSString *)aName
{
    self = [super init];
    if (self) {

        name = [aName copy];

    }

    return self;

}
+ (MyClass *)createMyClassWithString: (NSString *)aName

{

    return [[[self alloc] initWithString:aName] autorelease];

}

@end
```

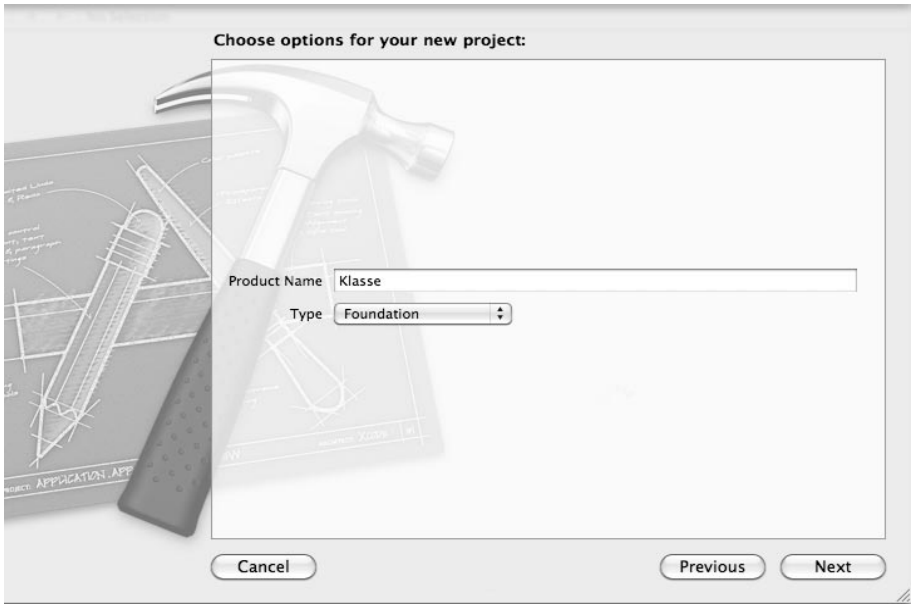
5.7.2 Eine Klasse in Xcode erstellen

Wenn Sie eine eigene Klasse in Xcode erstellen wollen, gehen Sie wie folgt vor:

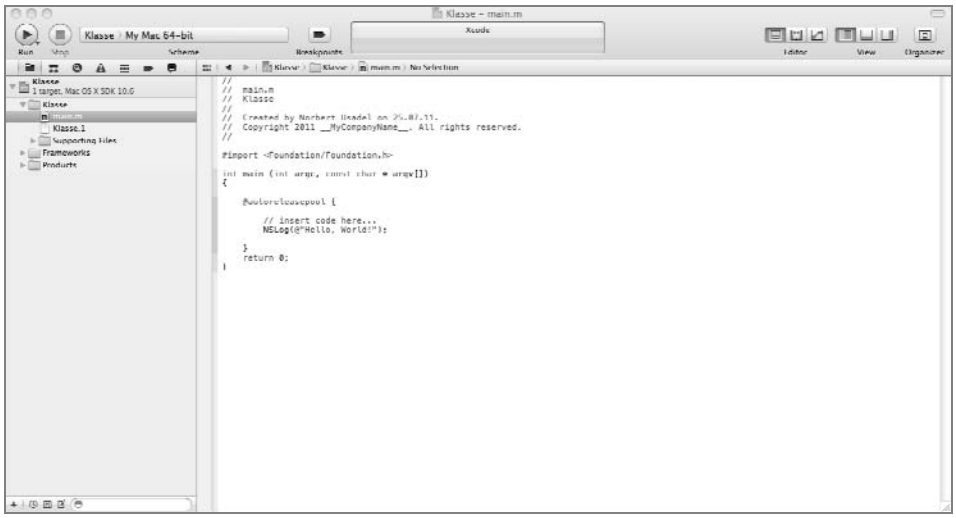
Starten Sie Xcode, und wählen Sie unter dem Menüpunkt Mac OS X *Application* aus. Im Template-Bereich wählen Sie als Vorlage *Command Line Tool* aus. Quittieren Sie dann mit *Next*.



Benennen Sie danach Ihr Projekt *Klasse,* und wählen Sie den Type *Foundation* aus.



Nun können Sie in der Datei *main.m* Ihre Klasse eingeben. Die Ausgabe Ihres Programms können Sie dann im Konsolenfenster überprüfen.



5.7.3 Klassen finden

Wenn Sie in Xcode die Hilfe aufrufen und oben in der Suchfunktion der Filter-Bar *iOS Library* eingeben, erhalten Sie eine Übersicht über alle Frameworks, in denen sich die Klassen wiederfinden.

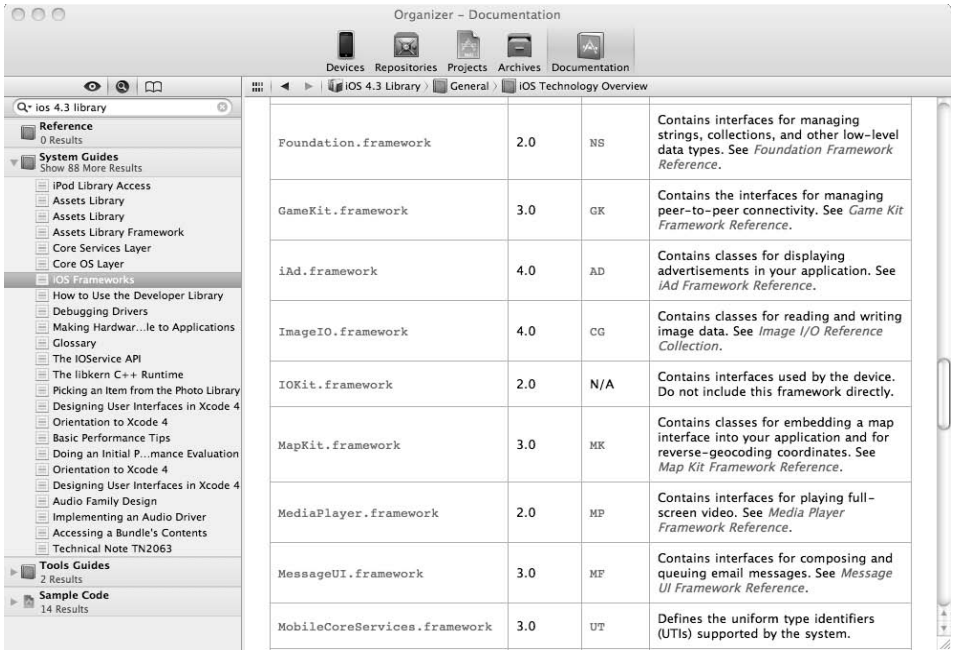


Bild 5.10: Die Übersicht in der iOS Library über Klassen und Frameworks.

Die Klassen werden also in Frameworks zusammengefasst und können jederzeit in Ihr Programm eingebaut werden. In den Workshops dieses Buches wird immer wieder auf diese Frameworks eingegangen.

Sollten Sie ein Programm für das iPhone entwickeln wollen, können Sie sich in einer Beispiel-App alle UI-Klassen anzeigen lassen. Hierzu können Sie wieder die Hilfe von Xcode benutzen und sich den Quelltext der App UICatalog herunterladen und diese dann im iPhone-Simulator betrachten. Sie bekommen so die Übersicht über die Klassen und haben gleichzeitig den Quellcode vor sich.

Bild 5.11: Die App UICatalog aus der iOS Library zeigt Ihnen Klassen und das UI-Framework.

Eine Klasse beschreibt Objekte, die immer wieder abrufbar sind und sich in den Quelltext einbauen lassen. Eine Klasse ist also ein Bauplan für Objekte.

5.7.4 Die Vererbung von Klassen

Die Vererbung von Klassen dient dazu, nicht immer von vorne zu beginnen, wenn Sie in einer Programmiersprache ein Problem lösen wollen. Sie können in einem Programm auf dem aufbauen, was schon vorher programmiert worden ist. Wollen Sie nur einen Teilaspekt lösen, brauchen Sie diesen dann nur Ihrem Code hinzuzufügen.

Die Klasse, die in Objective-C alles bestimmt, ist die Klasse NSObject. Sie hat zahlreiche Unterklassen. Wenn Sie selbst eine Klasse bestimmen, wird diese in fast allen Fällen eine Unterklasse oder eine Subklasse von der Klasse NSObject sein.

Sie finden einen Überblick in der iOS Library, wenn Sie in der Suchabfrage NSObject eingeben.

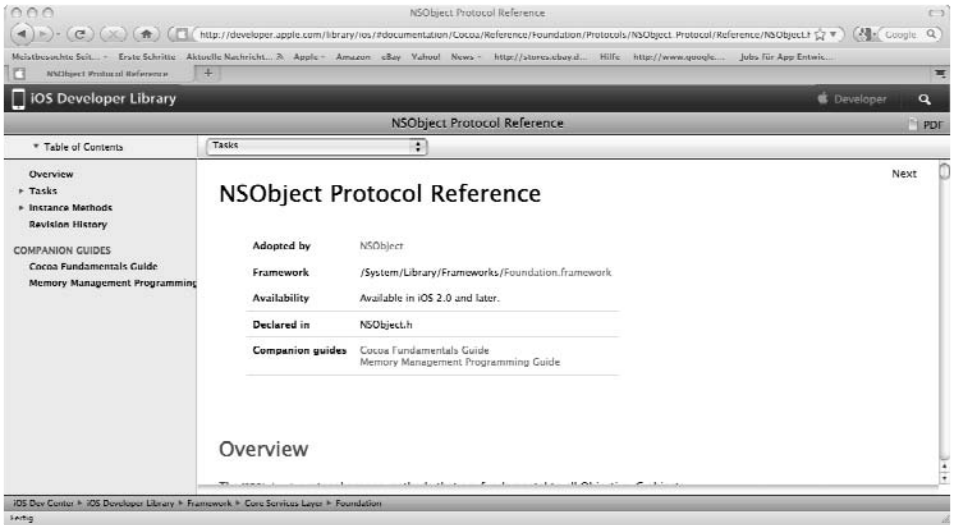


Bild 5.12: NSObject; die Mutter aller Klassen.

Die allgemeine Schreibweise einer Klassendeklaration mit einer Vererbungsbeziehung wird so abgebildet:

```
@interface ClassName : Superclass {
    //instance variable declarations
 }

// method declarations
 @end
```

In der Regel erben alle Klassen von der Basisklasse NSObject.

5.8 Methoden

Eine Klasse erstellt mittels Methoden ein Verfahren, das ein Objekt ausführen kann. Das kann zum Beispiel ein Wert sein, den ein Schalter zurückliefert. Dieser Wert kann dann im Quellcode weiterverarbeitet werden. Die Klasse legt für die Objekte, die zu ihr gehören, Speicherplatz im Hauptspeicher des Geräts an, sodass Informationen gespeichert werden können. Wird ein Schalt-Button angelegt, so wird der Button als Objekt erstellt. Die einzelnen Funktionen werden jedoch nicht im Speicher abgelegt, sondern nach Bedarf im Quelltext aufgerufen.

5.8.1 Wo sind die Methoden?

Apple stellt für Sie ja etliche Methoden bereit. Wie finden Sie diese? Der Weg führt über die Suche im Interface-Builder.

In Bild 5.13 sehen Sie einen Ausschnitt eines Projekts. Dort wurde ein NSTextField verwendet. In der rechten Hälfte des Fensters wird Ihnen dazu Hilfe angeboten. Dieses Projekt gibt es hier im Buch als Workshop.

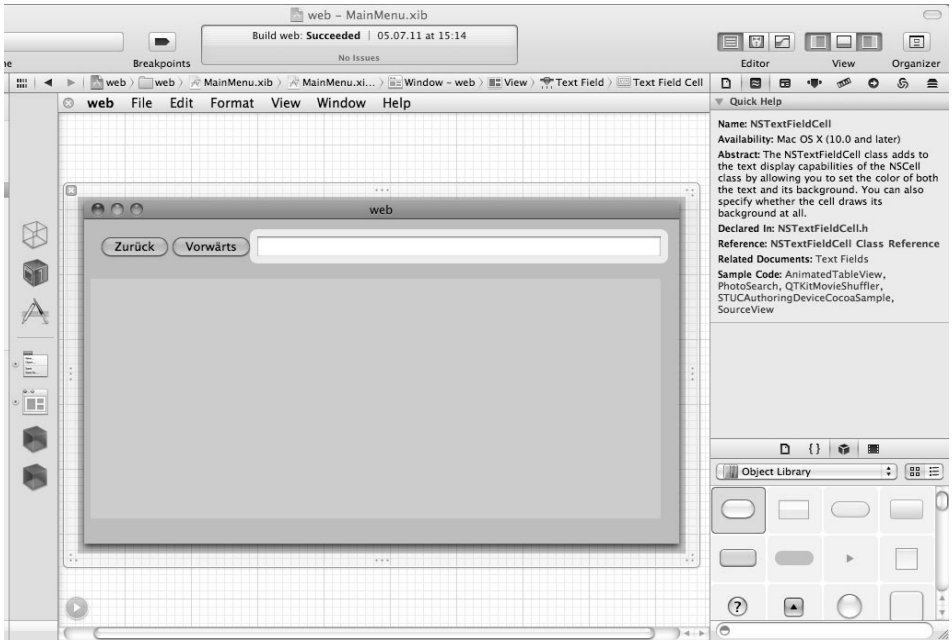


Bild 5.13: Ausschnitt aus einem Projekt in diesem Buch. Gesucht werden Informationen über das NSTextField.

NSTextField ist ein Objekt in einer Klasse, die in einer Methode beschrieben ist. Wenn Sie darüber Informationen brauchen, finden Sie diese in dem rechten Fenster des Interface-Builders. Über einen Link erhalten Sie die komplette Übersicht über die Klassen-Reference, in der sich die Methode verbirgt.

Dort erhalten Sie alle Informationen, die Sie benötigen.

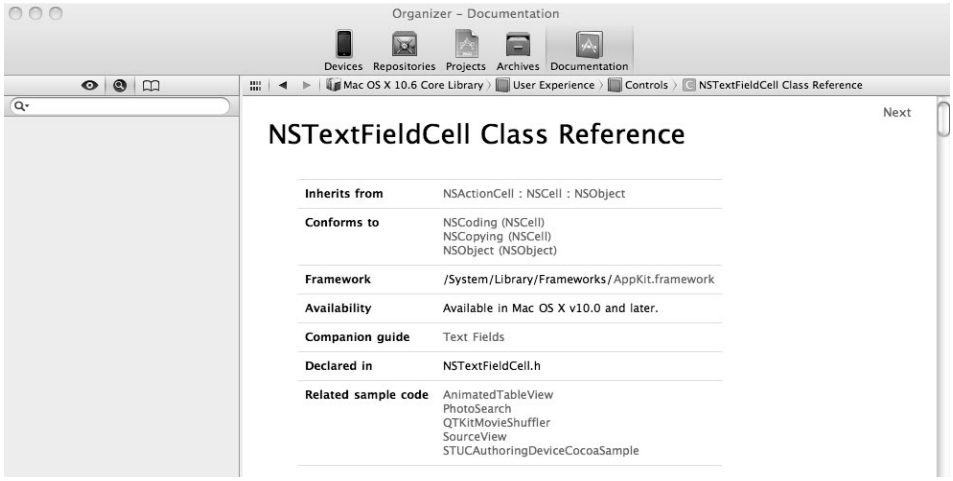


Bild 5.14: Das Suchergebnis des NSTextFields.

Sie können sich zu der Methode aber auch Beispielcode in einem Projekt anzeigen lassen. Weiter unten erhalten Sie ein Beispiel, wie Ihre angefragte Klasse in einem Projekt mit dem Code dazu eingebunden ist.

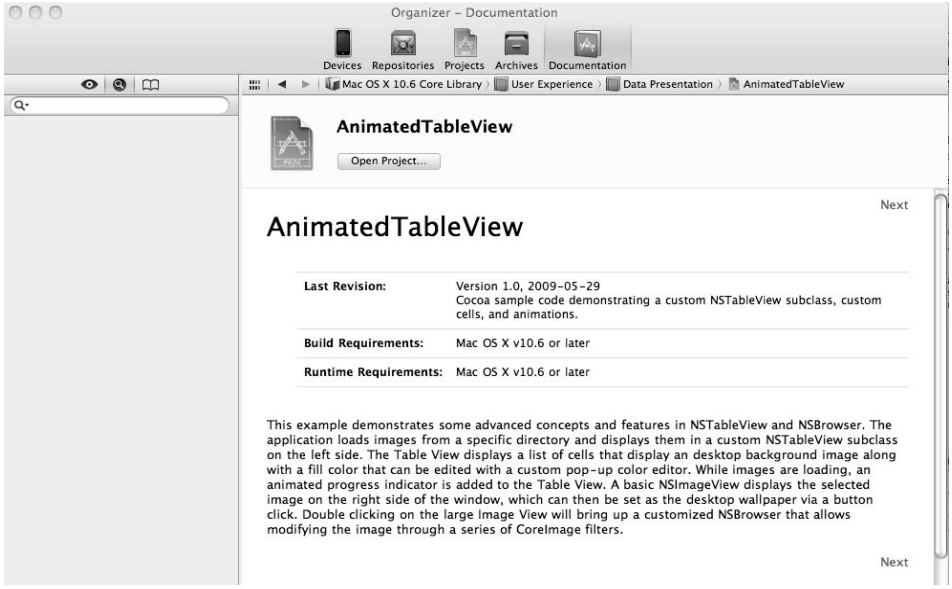


Bild 5.15: Ein Beispiel mit einem NSTextField in der iOS Library.

5.9 Die Frameworks

Apple hat alle Klassen und Methoden in Frameworks zusammengefasst. Das erspart Ihnen eine Menge Tipparbeit. So brauchen Sie keine Knöpfe oder Schaltflächen zu programmieren. Sie bekommen den Code aus den einzelnen Frameworks zur Verfügung gestellt. Immer, wenn Apple eine neue Version eines Betriebssystems ankündigt, kommen zu den Frameworks neue APIs hinzu. APIs sind Programmierschnittstellen, mit denen sich im Nu neue Features für Programme erstellen lassen.

Sie bekommen unter dem Link *http://developer.apple.com* immer eine Übersicht über die neuesten Schnittstellen. Unter dem Link *http://developer.apple.com/technologies/ios5/* sehen Sie Infos über das brandneue iOS-5-Betriebssystem für iPhone und iPod Touch (vgl. Bild 5.16). Dieses wirbt mit mehr als 1500 neuen APIs.

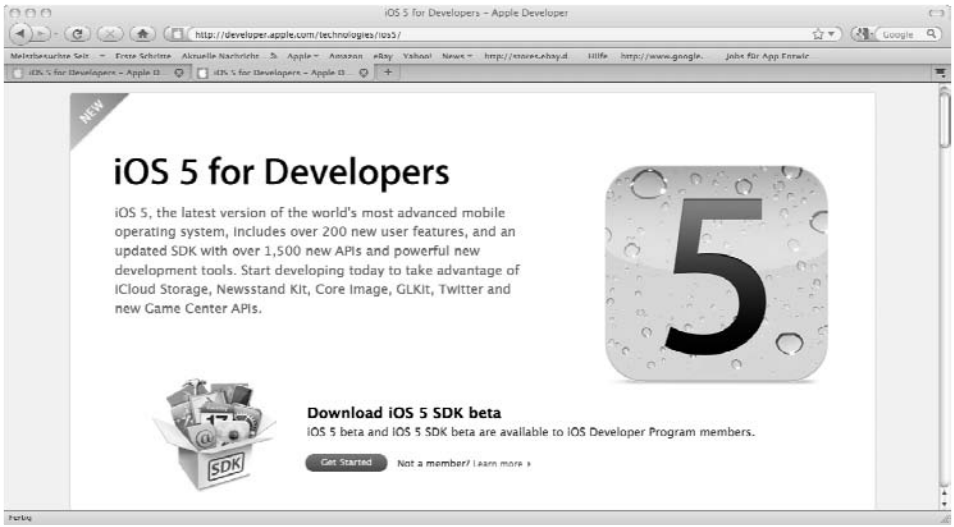


Bild 5.16: Neue Frameworks in iOS 5 für neue Projekte.

Die Ankündigung eines neuen Betriebssystems zieht auch immer ein neues SDK oder Entwicklertool nach sich. In diesem Fall kam nach der Ankündigung des iOS 5 Xcode 4.2 auf den Markt. Das Betriebssystem iOS 5 wurde bereits in Kapitel 4 vorgestellt.

Die Frameworks finden Sie auch gut dokumentiert in den einzelnen Libraries.

Die Library für iOS 5 finden Sie unter diesem Link: *http://developer.apple.com/library/ios/navigation/*

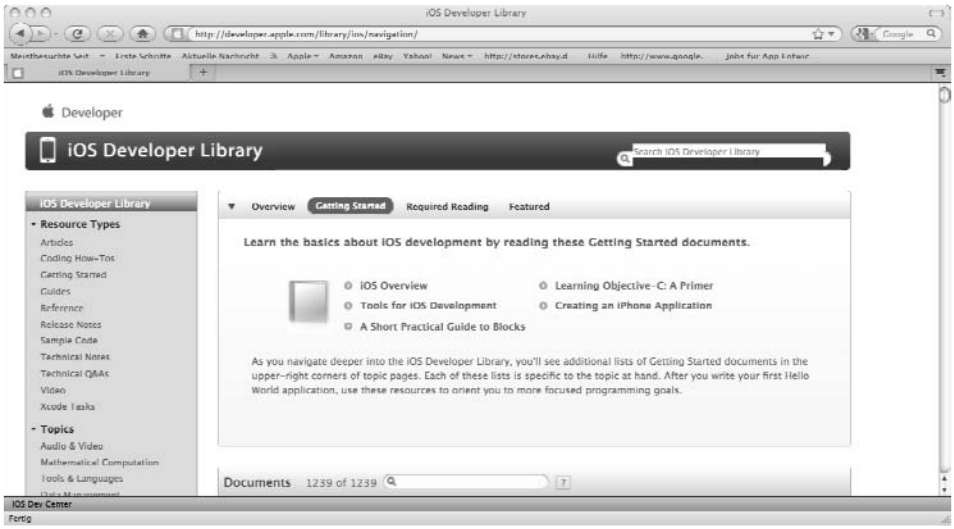


Bild 5.17: Die iOS Developer Library.

Die Library für Mac OS X ist hier zu finden: *http://developer.apple.com/library/mac/navigation/*.

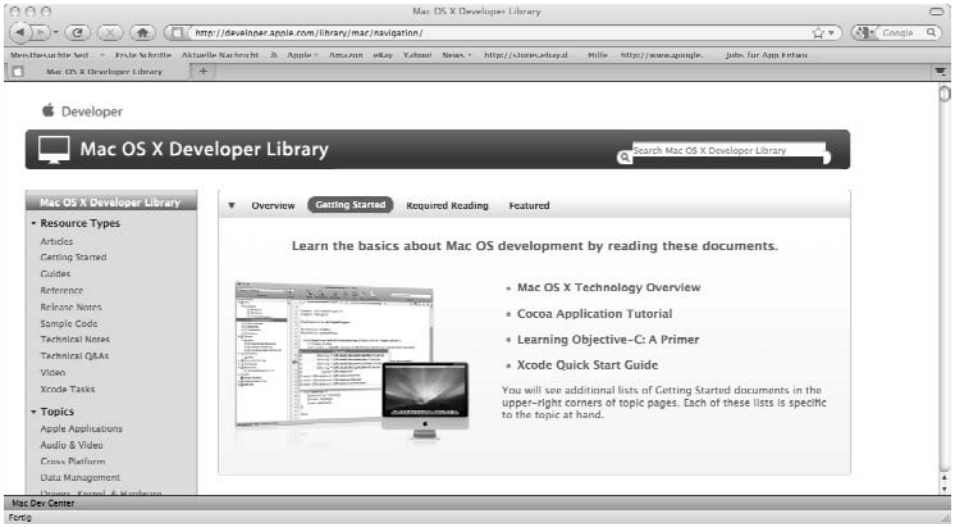


Bild 5.18: Die Mac OS X Developer Library.

Beide Bibliotheken sind gefüllt mit guten Dokumentationen, Beispiel-Code und Anleitungen zu den Entwicklerprogrammen. Sie sind mit der Xcode-Hilfe verzahnt. Wenn Sie in Xcode die Hilfe aufrufen, wird auf beide Bibliotheken zurückgegriffen.

In den einzelnen Workshops in diesem Buch erfahren Sie mehr darüber, wie Objective-C in der Praxis eingesetzt wird.

5.10 Das Schichtenmodell

Nun haben Sie schon einiges über Objekte, Klassen, Methoden und Frameworks gelernt. Die Frameworks lassen sich in einem Schichtenmodell darstellen. Alles zusammen ergibt dann das Betriebssystem iOS oder Mac OS X. Die beiden Systeme unterscheiden sich nur unwesentlich voneinander. Das iOS hat ein anderes User-Interface, da es für mobile Endgeräte entwickelt worden ist. Das Weltbild der beiden Betriebssysteme wird im unten stehenden Vier-Schichten-Modell abgebildet.

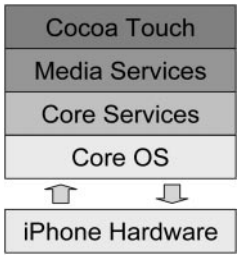


Bild 5.19: Die vier Schichten des iOS-Betriebssystems.

Die einzelnen Frameworks finden Sie in dem Vier-Schichten-Modell wieder:

Cocoa Touch (NSFoundation, UIKit)

Der erste Strang in diesem Framework ist dafür zuständig, alles zu regeln, was im Hintergrund einer App abläuft. Das können zum Beispiel die Datenspeicherung und das Dateimanagement sein. Diese Funktionen werden unter dem Namen NSFoundation-Kit zusammengefasst.

Der andere Strang an Klassen ist das AppKit. Das AppKit fasst alles zusammen, was mit dem Benutzer-Interface zu tun hat.

Das AppKit nennt man auch UIKit, es wurde für das iPhone/iPad neu gebündelt, da die Eingabe der Daten über einen Touchscreen stattfindet. Es können so, speziell für das iPhone, folgende Aspekte berücksichtigt werden:

- Multi-Touch-Verhalten
- Screen-Rotation
- View Controller
- Events und Gesten

In der Cocoa-Schicht können Sie aber auch noch andere Frameworks nutzen:

- Apple Push Notification Service
- AdresssBook UI Framework
- In-App E-Mail
- Map Kit Framework
- Peer-to-Peer-Support

Media (Quartz, Core Animation, OpenGL, Audio, ...)

Die Media-Schicht oder auch der Media Layer verwaltet Grafik-, Audio- und Videotechnologien. Die Bestandteile sind:

- Quartz, eine Bibliothek für vektorisierbares Zeichnen
- Core Animation, die zur Umsetzung visueller Effekte dient
- OpenGL ES, dient zur Herstellung schneller 2D- und 3D-Grafiken
- AV Foundation: Dieses Framework steht für das Aufnehmen und Abspielen von Audio-Inhalten
- Core Audio: dient ebenfalls zum Bearbeiten von Audio-Daten. Bietet aber mehr Möglichkeiten als die AV Foundation
- OpenAL: dient als Audio-Bibliothek für Spiele
- Video: dient zum Abspielen von Videos in verschiedenen Formaten

Core Services (Core Location, Adress Book, SQLite, CFNetwork, ...)

Diese Schicht dient zum Verwalten und Speichern von Daten. Es können hier Datenbanken angelegt werden. Es ist aber auch möglich, mit dem Core Location Framework Positionsbestimmungen per WLAN, Mobilfunk oder GPS durchzuführen und umzusetzen.

Core OS (I/O Threads, Sockets, Bonjour, Key Chain, ...)

In der untersten Schicht des iOS-Betriebssystems können Hardwarekomponenten direkt angesprochen werden, es lassen sich Netzwerkzugriffe programmieren. Hier können Sie direkt auf den Kernel des Unix-Betriebssystems zugreifen.

6 Testen auf dem iPhone und der Weg in den Store

In diesem Kapitel lernen Sie, wie man eine App auf dem iPhone installiert und welche Vorarbeiten zu treffen sind, um Ihre App in den App-Store zu bringen. Voraussetzung dafür ist, dass Sie am Entwicklerprogramm teilnehmen.

6.1 Entwickler werden

Um am Entwicklerprogramm teilzunehmen, melden Sie sich unter dem Link *http://developer.apple.com/programs/ios/* an.

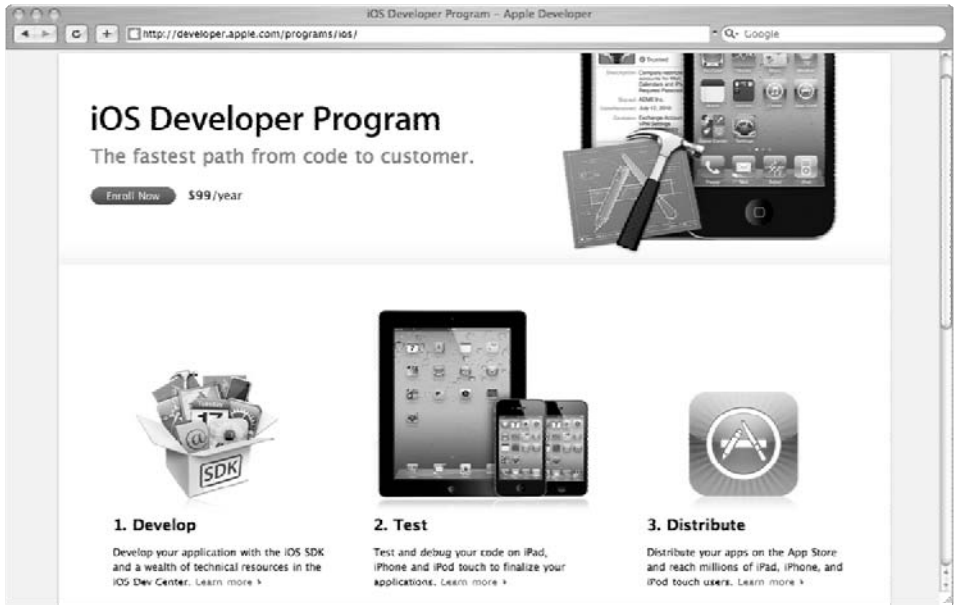


Bild 6.1: Das iOS-Entwicklerprogramm.

Es kostet Sie 99 Dollar im Jahr. Dafür erhalten Sie folgende Leistungen:

- Zugriff auf sämtliche Ressourcen wie z. B. das aktuelle SDK, Dokumentationen, Beta-Versionen und Programmierbeispiele

- das Erstellen von Entwicklertests und Adhoc-Profilen
- technischen Support
- Zugriff auf alle Entwicklerforen
- die Möglichkeit zur Veröffentlichung Ihrer App im App-Store

Nach der Registrierung werden Sie aufgefordert, einen Handelsregisterauszug oder Gewerbeschein Ihrer Firma an Apple zu faxen. Bis Sie freigeschaltet werden, können zehn bis zwanzig Tage vergehen.

Haben Sie den Entwicklerstatus erlangt, können Sie sich unter der Seite *http://developer.apple.com/devcenter/ios/index.action* mit Ihrer AppleID und Ihrem Passwort einloggen.

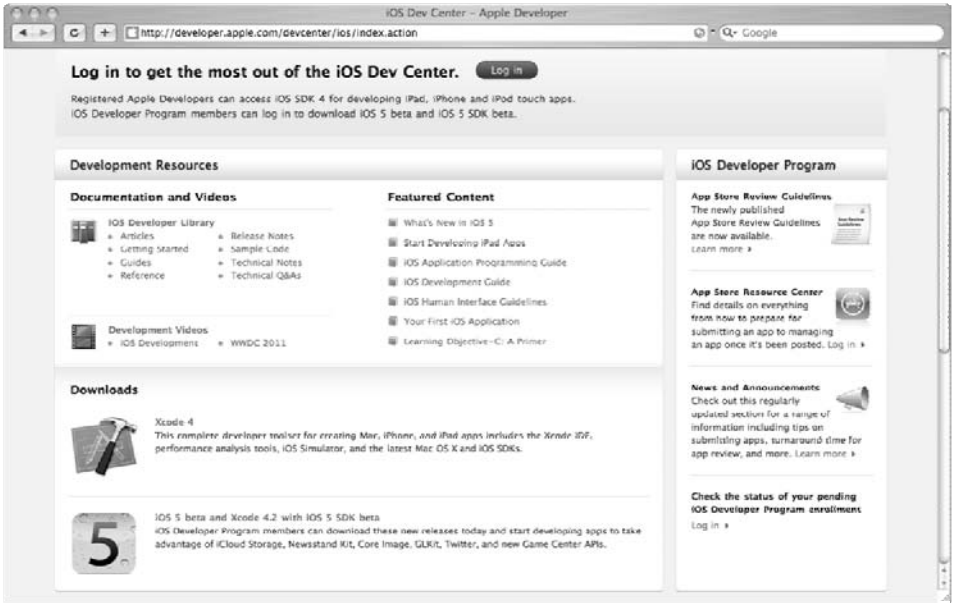


Bild 6.2: Das iOS-Devcenter.

6.2 Zertifikate anlegen

Wenn Sie Zugang zum Dev-Center haben, legen Sie ein iOS Development Certificate an. Sie benötigen es, um Ihre App in den Store zu bringen und um eine App auf einem iPhone zu installieren.

Sie fertigen ein Certificate Signing Request (CSR) an, indem Sie das Programm *Schlüsselbund* benutzen.

Öffnen Sie auf Ihrem Mac das Programm unter dem Pfad *Programme* → *Dienstprogramme*. Im Menüpunkt *Schlüsselbund* → *Voreinstellungen* wählen Sie *Einstellungen* an und stellen das OSCP- und das CRL-Protokoll auf *Aus*.

Bild 6.3: Der Menüpunkt Einstellungen im Programm Schlüsselbund.

Nun wählen Sie aus dem Menü *Schlüsselbund* den Zertifikatsassistenten an und folgen den einzelnen Schritten. Füllen Sie die Zertifikatsinformationen aus und quittieren Sie mit *Fortfahren*.

Bild 6.4: Hier personalisieren Sie Ihr Zertifikat.

Dann stellen Sie im letzten Schritt die Schlüssellänge auf *2048 Bit* und den Algorithmus auf *RSA* ein. Je länger Ihr Schlüssel ist, desto sicherer ist er. Apple wendet hier ein asymmetrisches Verschlüsselungsverfahren an, das sehr sicher ist. Es besteht aus einem privaten und einem öffentlichen Schlüssel. Der private Schlüssel ist geheim, der zweite Schlüssel wird dem Zertifikat hinzugefügt.

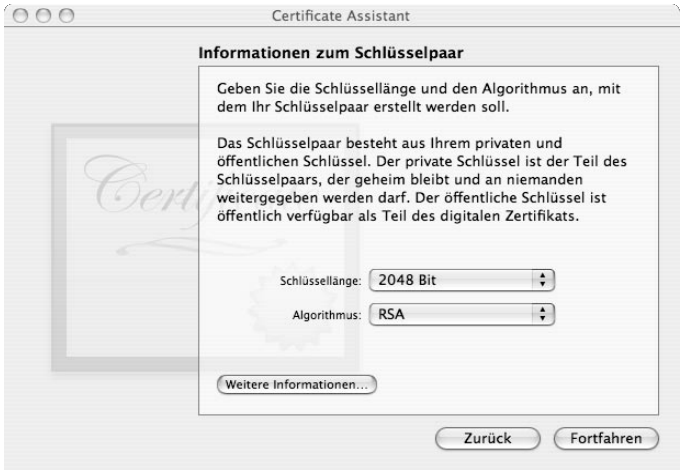


Bild 6.5: Einstellung der Schlüssellänge.

Folgen Sie dann den weiteren Schritten, die sich selbst erklären, und legen Sie Ihr Zertifikat an.

Nun übertragen Sie Ihr Zertifikat in den Developer-Bereich von Apple. Loggen Sie sich dazu im Entwicklerbereich ein, und wählen Sie dort *iOS Provisioning Portal* an.

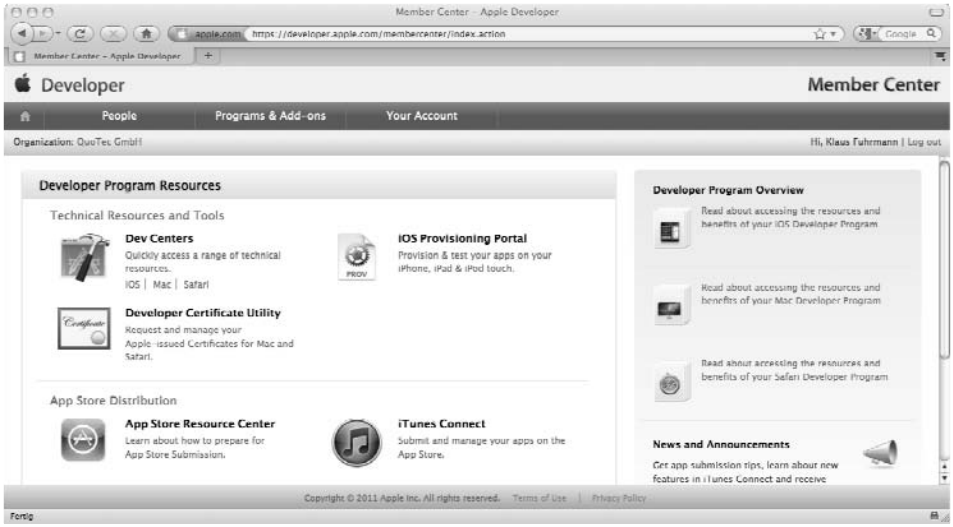


Bild 6.6: Der Developer-Bereich von Apple.

Im linken Bereich des Portals wählen Sie den Menüpunkt *Certificates* an. Danach klicken Sie auf den Button *Request Certificates*. Folgen Sie den Schritten, und bestätigen Sie zum Schluss mit dem *Submit*-Button.

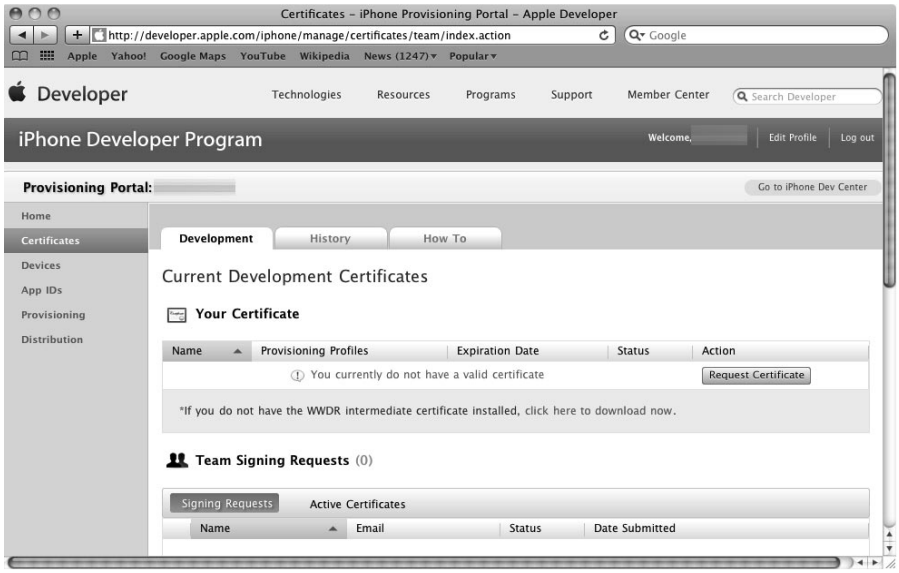


Bild 6.7: Anfordern des Zertifikats mit *Request Certificate*.

Wenn Sie diese ganze Prozedur abgeschlossen haben, wird Ihr Zertifikat wie in Bild 6.8 angezeigt. Ihr Upload war erfolgreich, und das Zertifikat ist aktiv.

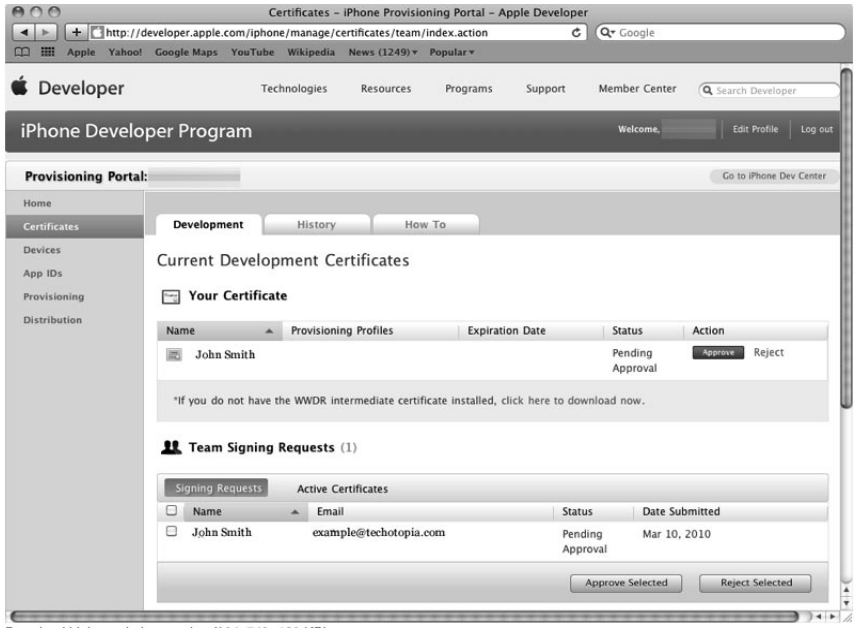


Bild 6.8: Übertragung des Zertifikats in das Provisioning-Portal.

Im letzten Schritt bringen Sie Ihre Zertifikate zurück auf Ihren Rechner, mit dem Sie Ihre Projekte entwickeln und auf dem Ihr Xcode installiert ist. Dazu klicken Sie den *Download*-Button auf der Website des Provisioning-Portals auf der Certificates-Page an. Wenn der Download erfolgreich war, wird er Ihnen in Ihrem Schlüsselbund angezeigt. Ein Doppelklick auf das Programm Schlüsselbund genügt, um Ihnen das Resultat anzuzeigen.

Bild 6.9: Das Zertifikat ist in Ihrem Programm Schlüsselbund angekommen.

6.3 iPhone zum Testen benutzen

Nachdem Sie das Entwicklerzertifikat installiert haben, können Sie Apps direkt auf Geräten testen. Sie können im Jahr bis zu 100 Geräte für Testzwecke oder den Adhoc-Vertrieb verwenden. Um ein Gerät nutzen zu können, brauchen Sie die Unique-Device-Identifier-Nummer (UDID) Ihres Geräts. Um diese herauszufinden, schließen Sie das Gerät mit dem Dock-Connector an den Mac an. Starten Sie Xcode, und wählen Sie aus dem Organizer den Menüpunkt Devices an. Dort wird Ihnen im Fenster eine Übersicht über das angeschlossene Gerät angezeigt. In der Aufstellung bekommen Sie unter dem Punkt Identifier die UDID Ihres Geräts angezeigt. Klicken Sie dann auf *Use for Development.*

Bild 6.10: Die Übersicht über Ihr angeschlossenes Gerät.

Sie werden mit dem Provisioning-Portal verbunden. Dort wählen Sie unter dem Menüpunkt *Devices* den Reiter *Manage* an und tragen die UDID und einen Namen für das Gerät ein.

Bild 6.11: Geben Sie die Device ID Ihres Testgeräts an.

6.4 App ID erstellen

Um eine App ID zu erstellen, klicken Sie im Provisioning-Portal auf den Menüpunkt *App ID*. Sie brauchen diese ID, wenn Sie Ihre App in den App-Store bringen wollen.

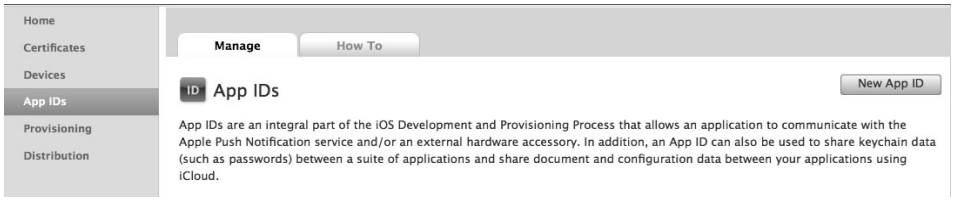


Bild 6.12: Erstellung der App ID über den Button *New App ID*.

Klicken Sie danach auf den Button *New App ID*. Folgendes Fenster baut sich auf:

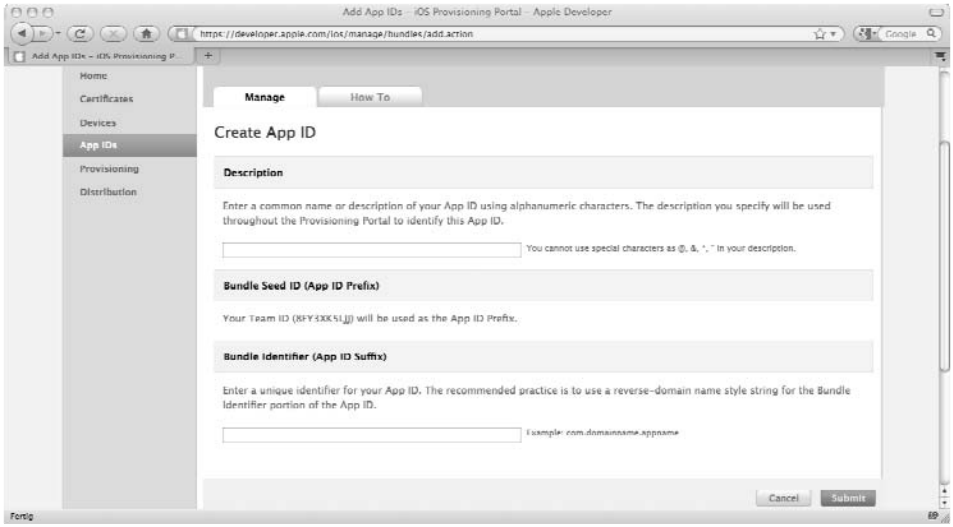


Bild 6.13: Anlegen einer neuen App ID.

Hier tragen Sie einen Bundle Identifier für Ihre App ein, wie zum Beispiel *com.domainname.appname*. Nachdem Sie nach diesem Schema einen Namen vergeben haben, klicken Sie auf den *Submit*-Button.

6.5 iOS Development-Provisioning-Profile anlegen

Rufen Sie in Xcode den Organizer auf und wählen Sie den Menüpunkt *Devices* an. Ein weiteres Fenster erscheint. Wählen Sie, wie in der Abbildung unten, den Menüpunkt Provisioning-Portal an. Ein Login-Menü klappt auf und will Sie mit dem Portal verbinden. Geben Sie Ihren Usernamen und das Passwort Ihrer Apple-ID ein.

Wenn Sie im Provisioning-Portal sind, gehen Sie wie folgt vor:

- Klicken Sie auf den *New Profile*-Button.
- Vergeben Sie dann einen Namen für Ihr Profil.
- Wählen Sie eine App ID aus dem Menü aus.
- In einer Checkbox können Sie Namen anderer Entwickler angeben, die das Provisioning-Profile nutzen können.
- Wählen Sie das Gerät aus, mit dem Sie Ihr Projekt testen wollen.
- Danach klicken Sie auf den Submit-Button.

Bild 6.14: Das Login in das Provisioning-Portal.

Ihr Profil wird dann übermittelt und erscheint in Ihrem Xcode-Fenster.

Bild 6.15: Das Provisioning-Profile.

Der nächste Schritt zeigt die Verknüpfung zwischen Bundle-Identifier und Ihrer App. Dazu klicken Sie im Projektbaum von Xcode auf die blaue Projektdatei. In der zweiten Spalte wählen Sie *Targets* und dann den Reiter *Info* an. Im Fenster erscheint eine Übersicht. Dort können Sie den Bundle-Identifier, den Sie vorher angelegt haben, Ihrer App zuordnen.

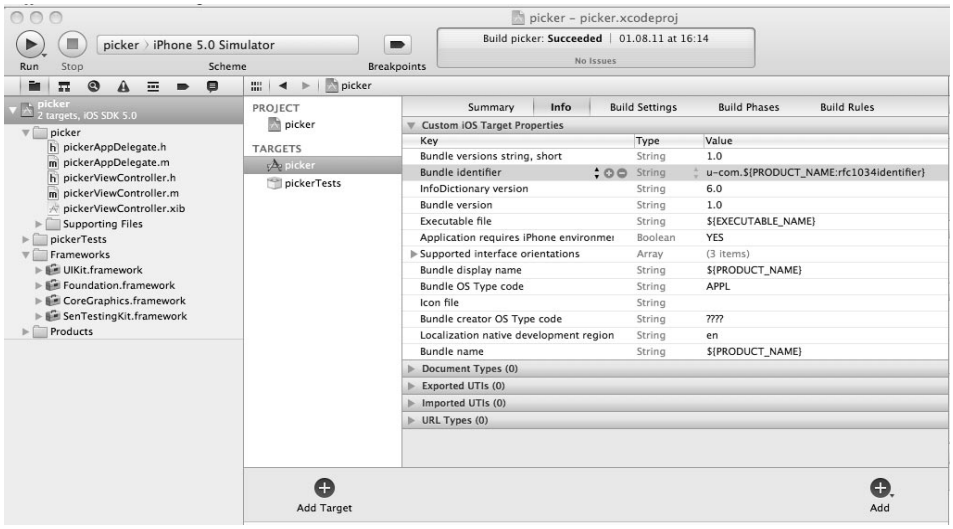


Bild 6.16: Der Bundle-Identifier wird eingetragen.

Der Weg Ihrer App in den Store oder auf ein iPhone, auf dem sie getestet wird, ist gleich. Bild 6.17 zeigt Ihnen noch einmal die Schritte.

Bild 6.17: Der Weg in den App-Store.

6.6 iTunes Connect

Nachdem Sie mit allen Zertifikaten versorgt sind, bringen Sie Ihre App durch iTunes Connect in den App-Store. In der folgenden Abbildung sehen Sie das Startportal von iTunes Connect unter dem Link *http://itunesconnect.apple.com*. Um eine App dort hochzuladen, klicken Sie auf den Menüpunkt *Manage your Application*. Folgen Sie den weiteren Schritten, um Ihre App in den App-Store zu bringen.

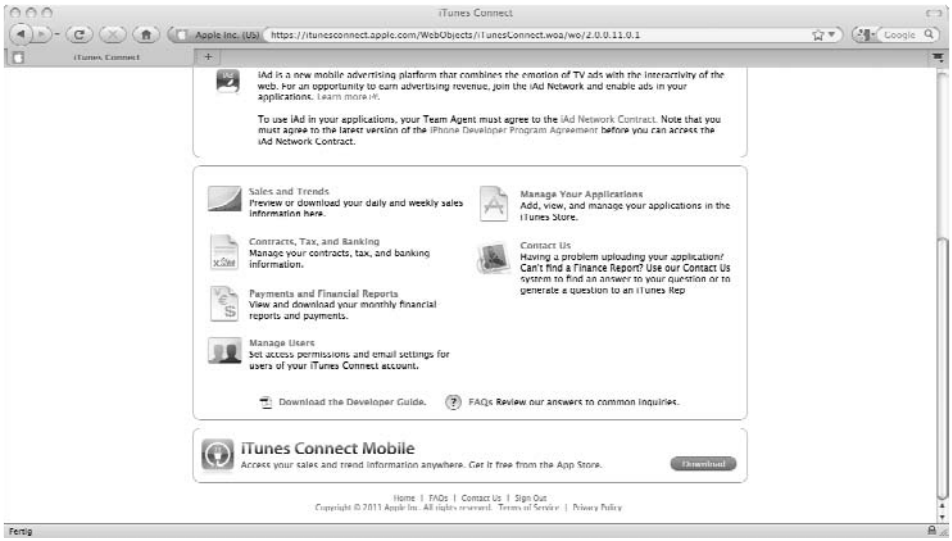


Bild 6.18: Der Startbildschirm von iTunes Connect.

Weitere Funktionen, die iTunes Connect für Sie bereithält:

Verkaufsberichte

Tägliche oder wöchentliche Verkaufsberichte.

Verträge

Verwaltung Ihrer Verträge, Steuer- und Bankdaten.

Werbung

Hier haben Sie den Einstieg in das iAd-Netzwerk, um in Ihrer App Werbung zu schalten.

Finanzen

Ansicht und Download Ihrer monatlichen Finanzberichte und Zahlungen.

App-Verwaltung

Hinzufügen, Anzeigen und Verwalten Ihrer App im App-Store.

Benutzerverwaltung

Verwaltung von Zugriffsberechtigungen und E-Mail-Einstellungen für Benutzer des iTunes-Connect-Kontos.

7 Die Workshops, Besonderheiten in Objective-C und ein Webbrowser

In diesem Kapitel überschreiten Sie eine Landmarke. Zuvor haben Sie den Theorieteil des Buches absolviert. Nun kommt der praxisorientierte Teil des Buches. Die anschließenden Workshops zeigen Ihnen mit ein paar Tricks, wie einfach es ist, eine App zu erstellen. Sie erlernen das Grundgerüst zu speziellen Themen. Wenn Sie den einen oder anderen Workshop zusammenfügen, haben Sie schon eine komplette App erstellt. Peppen Sie diese noch mit grafischen Elementen, die der Interface-Builder Ihnen bietet, auf, haben Sie (fast) schon eine App, die Sie in den Store bringen könnten. Wie einfach es sein kann, mit Xcode ein Programm zu erstellen, sehen Sie in dem anschließenden Workshop.

In Objective-C gibt es einige Besonderheiten, die vor dem Start in den Praxisteil geklärt werden sollten. Sie werden ständig mit diesen Datentypen und den Getter- und Setter-Methoden in den Workshops konfrontiert werden. Vergleichen Sie hierzu auch »iPhone-Apps entwickeln« von Dr. Dirk Koller, Abschnitt 3.5 ff., ebenfalls im Franzis-Verlag erschienen.

7.1 Datentypen und Properties

Es gibt bei den Datentypen in Objective-C Besonderheiten, die hier im Überblick zusammengefasst sind.

id

Mit `id` wird ein Pointer bezeichnet, der auf ein beliebiges Objekt zeigen kann. Er wird erst zur Laufzeit benutzt, weil erst dann klar werden kann, auf welches Objekt er zeigen soll.

Bool

Variablen vom Typ `Bool` können den Wert YES oder NO annehmen.

SEL

Der Datentyp `SEL` nimmt Methodennamen auf, die während der Laufzeit aufgerufen werden.

Nil

Mit `Nil` wird eine Variable auf Null gesetzt.

Properties

Instanzvariablen, die in der Header-Datei deklariert werden. Sie stehen für die `get`- und `set`-Methode. Um sich viel Schreibarbeit zu ersparen, wenn man Methoden im Quelltext anwenden will, wurden die Hilfsmittel `@property` und `@synthesize` geschaffen. Sie vereinfachen die sogenannten Getter- und Setter-Methoden.

Alles dreht sich um diese sogenannten Zugriffsmethoden, die die Eigenschaften der abgefragten Objekte verändern.

Getter-Methoden sind Abfragemethoden. Sie fragen die Instanzvariablen eines Objekts nach einem Wert ab.

Setter-Methoden sind Zugriffsmethoden, die die Eigenschaft eines Objekts verändern. Vor der Änderung wird ein Wert auf seine Gültigkeit abgefragt, um die Änderung des Objekts zu veranlassen.

@property

Durch diese Anweisung werden Accessoren und Mutatoren deklariert.

```
@property (nonatomic, retain) NSString *name;
```

ersetzt diese Anweisung:

```
-(void)setName: (NSString*) aName;
-(NSString*) name;
```

@synthesize

Durch diese Anweisung werden Methoden implementiert.

```
@synthesize name
```

erzeugt folgende Implementierung:

```
-(void)setName: (NSString*) aName;
{
   name = aName;
}
-(NSString *) name
{
   return name;
}
```

Stichwortverzeichnis

Norbert Usadel

Mit Xcode 4.2 und Objective-C fürs iPhone programmieren

Einführung in die Software-Entwicklung für iOS 5

Apple hat seine Entwicklungsumgebung Xcode modernisiert. Mit Xcode 4.2 wird es noch leichter, Apps zu entwickeln. Dieses Buch führt Sie in 14 praxisnahen Workshops an die Programmierung mit Xcode und Objective-C für das neue iOS 5 heran. Erstellen Sie Ihre eigene iPhone-App – dieses Buch zeigt Ihnen wie!

Einführung in Xcode 4.2 und die Entwicklertools

Mit Xcode stellt Apple eine sehr umfangreiche und übersichtliche Entwicklungsumgebung bereit. Hier lernen Sie den Umgang mit den Entwicklertools – wie Instruments, Dashcode oder dem iPhone-Simulator. Und auch die neuen Features von Xcode 4.2 – z.B. das Arbeiten mit Storyboards – werden hier vorgestellt und sofort angewandt.

Objective-C – kurz und verständlich

Die Basis der Entwicklung für Apple-Geräte – egal ob Mac, iPad, iPhone oder iPod – ist die Sprache Objective-C. Hier erhalten Sie eine Einführung, die Ihnen schnell das Wichtigste beibringt, um selbst in die Programmierung einzusteigen.

Die Workshops

- Webbrowser
- Temperatur-Umrechner
- Währungs-Umrechner
- Date-Picker
- Geodaten auf dem iPhone
- Arbeiten mit Storyboards
- Tab-Bar
- Table-View
- SQLite-Datenbank
- Core-Data-Datenbank
- Kartendarstellung mit Google Maps
- Werbung mit iAd
- Audiodaten
- Kunststoff-Daten-App

Aus dem Inhalt:

Über den Autor:

Norbert Usadel arbeitet seit 20 Jahren selbstständig im Apple-Bereich. In dieser Zeit schulte er ca. 2.000 Menschen über die Themen SAP, Cobol, Filemaker, Shopsysteme, iPods, iPhones, iTunes, Strukturierte Programmierung, Pagemaker, Photoshop und App-Programmierung. Außerdem schreibt er Artikel über Podcasts, Datenverschlüsselung und GarageBand und ist Autor des Buches Inside iPod. Er entwickelt Apps und übernimmt Projektleitungen für die App-Programmierung.

Alle Beispieldateien zum Download auf **www.buch.cd**

9 783645 601375

30,– EUR [D]

ISBN 978-3-645-60137-5