entwickler.press


# iPhone

## Anwendungsentwicklung für Einsteiger

Michael Kain

Michael Kain

# iPhone

## Anwendungsentwicklung für Einsteiger

entwickler.press

Michael Kain
iPhone - Anwendungsentwicklung für Einsteiger
ISBN: 978-3-86802-200-1


Ein Imprint der Software & Support Verlag GmbH

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Ihr Kontakt zum Verlag und Lektorat:
Software & Support Verlag GmbH
entwickler.press
Geleitsstraße 14
60599 Frankfurt am Main
Tel: +49(0) 69 63 00 89 - 0
Fax: +49(0) 69 63 00 89 - 89
lektorat@entwickler-press.de
http://www.entwickler-press.de

# Inhaltsverzeichnis

# E Koautor und Reviewer

*Jeder ist seines Glückes Schmied.*

Appius Claudius Caecus,
römischer Konsul

**Martin Schultz,**
**Koautor und technischer Reviewer**

Martin Schultz ist ein iPhone-Entwickler der ersten Stunde und hat seit dem Start des Apple App Store eine Vielzahl von Applikationen und Spielen veröffentlicht, wie zum Beispiel Bubble Bang, Big Fun Racing oder Chain Timer. Er ist Experte für die Entwicklung von interaktiven 3D-Spielen und Anwendungen mit der Unity-Game-Engine, die auch für das iPhone verfügbar ist. Mit dieser hat Martin unter anderem die Spiele Big Fun Racing und Bubble Bang entwickelt. Ferner verfügt er über viel Erfahrung im Bereich Werbung auf dem iPhone. Im Kapitel „Über den Tellerrand“ gibt er uns einen Ausblick auf die Entwicklung mit Unity sowie die Einbindung von Werbebannern in iPhone-Applikationen auf Basis von AdMob. Darüber hinaus steuerte er das Kapitel über Sounds zu diesem Buch bei.

Mehr über Martin Schultz finden Sie auf seiner Website www.decane.net oder direkt im App Store. Bei weitergehenden Fragen erreichen Sie Martin unter ms@decane.net.

## Guido Keller, technischer Reviewer

Guido Keller ist seit 1998 freiberuflicher IT-Berater und hilft seinen Kunden bei der Umsetzung von Java-Enterprise-Projekten. Sein Schwerpunkt liegt dabei besonders im Bereich Telekommunikation und Mobilfunk. Er hat an der Entwicklung von iPhone-Applikationen mitgearbeitet und ist seit längerem intensiver iPhone-Nutzer.

# V Vorwort

**Lieber Leser,**

zu Beginn dieses Buches will ich ausnahmsweise ehrlich zu Ihnen sein – ausnahmsweise. Ja, das iPhone ist die spannendste Plattform, für die ich je entwickelt habe. Ehrlich. Wobei mir das Adjektiv „spannend" zu zurückhaltend klingt. Der Begriff „Fieber" scheint mir viel angebrachter. Ich habe Entwickler, Familienväter, kennengelernt, welche am nächsten Morgen zur Arbeit mussten, um 22 Uhr ihre Kinder ins Bett brachten und sich bis um 2 Uhr nachts noch mal schnell „daransetzen mussten", weil die Faszination iPhone uns alle in den Bann zieht.

Für keine andere Plattform hat mir das Entwickeln so viel Spaß bereitet wie für das iPhone. Weder Delphi, J2ME, J2SE, J2EE, noch Grails konnten da bisher mithalten. Obwohl ich bis Mitte des Jahres 2008 noch nie einen Mac besessen hatte. Von Objective-C hatte ich zuvor nie gehört. Trotz dieser Vorbedingungen habe ich Ende 2008 meinen Mut zusammengenommen, um dieses Buch zu schreiben. Wenn Sie sich jetzt fragen, ob Sie hier wirklich das Buch eines Experten in den Händen halten, lautet meine ehrliche Antwort: Jein.

Nein, weil ich nicht seit zehn Jahren ausschließlich auf dem Mac und in Xcode entwickle. Ich bin nicht von Mac-Applikationen in Cocoa zu iPhone-Anwendungen auf Basis von Cocoa Touch migriert und leider habe ich auch keinen langen, grauen Bart. Ja, weil ich in den ersten Monaten meiner Zeit als iPhone-Entwickler in fast jede Falle getappt bin, in welche Einsteiger geraten können. Die Vielzahl dieser Stolpersteine ist in meiner Erinnerung noch ganz frisch und vor diesen will ich Sie, lieber Leser, bewahren. Und ja, weil es sich hier um mein zweites Buch handelt, in welchem ich besonderen Wert auf die einfache Verständlichkeit der Inhalte gelegt habe.

## V.1 Aufbau des Buches

Dieses Buch ist in sechs große Kapitel aufgegliedert, welche es einem Einsteiger am schnellsten ermöglichen sollen, sich in der Welt der iPhone-Anwendungsentwicklung zurechtzufinden. Ganz bewusst lehnen sich deshalb die Titel der Kapitel 3, 4 und 5 an die Vorgabe von Apple an. Laut Apple besteht nämlich, von ganz oben betrachtet, das iPhone Developer Program aus drei Stufen: 1. *Develop*, 2. *Test* und 3. *Distribute*. Im ersten Schritt werden Anwendungen auf Basis des iPhone SDK und des iPhone-Simulators entwickelt. Im zweiten Schritt werden diese auf Geräten und mit weiteren Analysewerkzeugen getestet und vervollkommnet, um anschließend im dritten Schritt unter die Leute gebracht zu werden.

Dieser klare, dreistufige Prozess findet sich im Aufbau des Buches wieder. Lediglich davor und danach wurde noch etwas angebaut, um dem Einsteiger entgegenzukommen. Demnach gliedert sich der Inhalt dieses Buches wie folgt:

1. **Einleitung**:

   Zu Beginn dieses Buches klären wir die Voraussetzungen, welche erfüllt sein müssen, um mit der Entwicklung von iPhone-Applikationen zu beginnen. Im Anschluss daran verschaffen wir uns vorab einen kompletten Überblick über die gesamte Plattform der iPhone-Anwendungsentwicklung. Dies erfolgt anhand einer spannenden Analogie, der iPhone-Formel 1.

2. **Einstieg in die Welt des iPhones**:

   Den Einstieg in die Welt des iPhones wagen wir mit der Erstellung einer einfachen Beispielanwendung, anhand derer wir uns in den darauf folgenden Unterkapiteln die kompletten Grundlagen der Entwicklung erarbeiten werden. Wir lernen die drei Bereiche von Objective-C kennen, welche zu den großen Fallstricken für Einsteiger werden können: Klassen und Nachrichten, Variablen und Properties und die manuelle Speicherverwaltung. Im Anschluss daran werfen wir einen Blick auf den Lebenszyklus aller Anwendungen und lernen, wie wir eigene Grafiken, wie z. B. das Application Icon, integrieren.

3. **Entwickeln mit dem iPhone-Simulator**:

   Vieles kann problemlos mit Hilfe des iPhone-Simulators entwickelt und vorab getestet werden. Dementsprechend bunt sind die Themen, welche in diesem Kapitel behandelt werden. Themen wie Properties, Benutzereinstellungen und Internationalisierung sind klare Querschnittsthemen, die jeden etwas angehen. Andererseits kommen wir an den Grundlagen der Oberflächenprogrammierung, wie Autorotation, Text Fields, mehrere Views und Table Views auch nicht vorbei. Fast jede iPhone-Applikation bedient sich dieser Teile des iPhone SDK, die ebenfalls in diesem Kapitel behandelt werden. Die zwei Unterkapitel über Debugging und Sounds runden dieses Kapitel ab.

4. **Testen auf dem Gerät**:

   In diesem Kapitel lernen wir das Development-Provisioning kennen: Was müssen wir befolgen, wenn wir ein Gerät verwenden möchten, um auf diesem unsere Anwendungen zu testen? Zusätzlich lernen wir APIs anhand von einfachen Beispielen kennen, welche stark an Geräte gekoppelt sind, wie z. B. die Kamera, Multi-Touches und die Ortsbestimmung. Zum Abschluss dieses Kapitels sehen wir uns an, wie wir mit Hilfe von Instruments Speicherlöcher in unseren Applikationen finden und beheben können.

5. **Vertrieb**:

   In dem Kapitel über den Vertrieb erfahren wir, was wir hinsichtlich des Distribution-Provisioning beachten müssen. Welche Schritte müssen wir als Entwickler befolgen, wenn wir einen Release-Build erstellen möchten? Außerdem lernen wir, wie wir uns iTunes Connect bedienen, um unsere Applikationen in den App Store einzustellen, und wie wir über unsere Verkäufe auf dem Laufenden bleiben.

6. **Über den Tellerrand**:

   In diesem Kapitel wagen wir gegen Ende dieses Buches einen kurzen Blick über den Tellerrand der klassischen iPhone-Anwendungsentwicklung. Wir lernen das Framework AdMob kennen, mit dessen Hilfe sich über Werbung in iPhone-Applikationen Geld verdienen lässt. Außerdem wird Entwicklern, welche primär an der Entwicklung von Spielen Interesse haben könnten, das 3D-Werkzeug Unity vorgestellt, mit dessen Hilfe sich Anwendungen für das iPhone exportieren lassen.

## V.2 Begriff: iPhone

Für zwei Serien von Geräten können auf Basis des iPhone SDK Applikationen entwickelt werden: das iPhone und den iPod touch. In diesem Buch wird der Begriff iPhone umfassend für alle Geräte verwendet, für die Anwendungen entwickelt werden können. Wenn also vom iPhone die Rede ist, schließt dies normalerweise den iPod touch automatisch mit ein. Dies hat den Hintergrund, dass für einen Großteil der API – aus Sicht des Entwicklers – nicht zwischen den Geräten unterschieden werden muss. An den Stellen, wo es Unterscheide zwischen den Geräten gibt, welche bei der Programmierung beachtet werden müssen, wird explizit darauf hingewiesen, für welches Gerät welche Ausnahme zu implementieren ist.

## V.3 Danksagungen

Zuallererst möchte ich mich mit einer tiefen Verbeugung bei Martin Schultz bedanken, meinem Koautor und technischen Reviewer. Ohne Dich, Martin, würde es dieses Buch nicht geben und ich wäre niemals auf die Idee gekommen, Anwendungen für das iPhone zu entwickeln. Mit Deiner Begeisterung hast Du mich angesteckt. Niemals hast Du Deinen großen Wissensstrom in meine Richtung versiegen lassen und für jede meiner Fragen eine solide Antwort präsentiert. Decane rocks!

Als Nächstes möchte ich in aller Form Guido Keller Danke sagen, dem zweiten technischen Reviewer dieses Buches: Trotz Deiner vollen beruflichen und privaten Auslastung hast Du, Guido, mir in kürzester Zeit für jedes Kapitel ein mehr als lohnendes Feedback gegeben. Deine konstruktiven Vorschläge haben den Inhalt dieses Buches maßgeblich mitbeeinflusst.

Zuletzt möchte ich mich bei Dorothee Kremers, meiner Freundin, bedanken. Wiederum hast Du dieses zweite Buch mit all Deinen Kräften unterstützt! Alles andere sage ich Dir auf Walisch …

## V.4 Widmung

Dieses Buch ist meinen Eltern, Ursula und Ulrich Kain, gewidmet. Ihrer grenzenlosen Unterstützung verdanke ich mehr, als tausend Worte sagen können.

Michael Kain, Hamburg, im Juli 2009

# 1 Einleitung

Einleitend werden in diesem kurzen Oberkapitel zwei Ziele verfolgt. Zum einen werden dem Leser die Voraussetzungen vermittelt, welche notwendig sind, um je nach den persönlichen Zielen mit der Entwicklung bzw. dem Vertrieb von iPhone-Applikationen beginnen zu können. Zum anderen wird anhand einer Analogie mit der Formel 1 die gesamte Plattform der iPhone-Anwendungsentwicklung vorgestellt: Willkommen in der iPhone-Formel 1!

## 1.1 Voraussetzungen

Zwei Arten von Voraussetzungen muss ein Einsteiger mitbringen, bevor er mit der Entwicklung von iPhone-Applikationen beginnen kann: Material und Kopf – oder um es präziser zu formulieren, technische Voraussetzungen in Form von Hard- und Software, sowie Vorwissen in Form von Programmierkenntnissen. Letztere hängen auch damit zusammen, ob dieses Buch für den Leser eine Bereicherung darstellt oder nicht. Aus diesem Grund gehen wir hier auf diese beiden Arten von Voraussetzungen ein.

Beginnen wir mit den technischen Anforderungen. Diese sind in Abbildung 1.1 grafisch dargestellt. Die Basis von allem ist ein Intel-basierter Mac-Rechner, auf dem als Betriebssystem Mac OS X Leopard oder Snow Leopard installiert sein muss. Unter Windows oder anderen Betriebssystemen ist keine iPhone-Programmierung möglich.

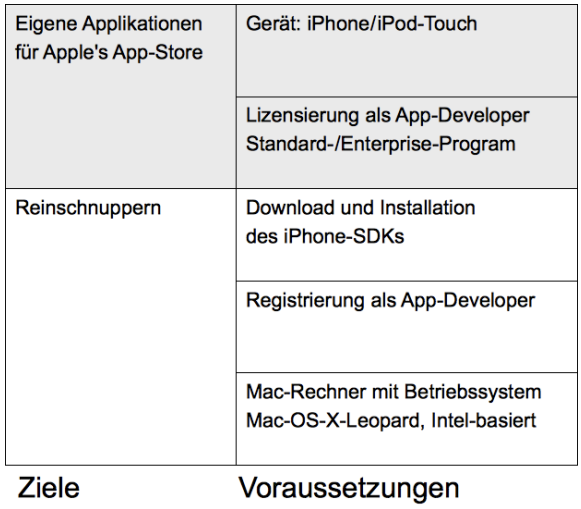


**Abbildung 1.1:** Voraussetzungen der iPhone-Entwicklung

Wer ein wenig mit der iPhone-Plattform experimentieren möchte, kann dies kostenlos tun. Hierfür ist lediglich eine Registrierung als iPhone-Developer im iPhone Dev Center notwendig. Anschließend kann das iPhone-Entwicklerkit (SDK) heruntergeladen und auf dem lokalen Mac installiert werden. Das Entwicklerkit enthält den iPhone-Simulator, um Anwendungen ohne Gerät vorab testen zu können.

Das iPhone Developer Center findet sich unter der URL:
http://developer.apple.com/iphone/

Der nächste Schritt wird erst notwendig, wenn eigene Anwendungen über den Apple App Store vertrieben werden sollen. Durch den Erwerb einer iPhone Developer License öffnet sich dem Entwickler der komplette Zugang zum App Store. Apple bietet diese Lizenzen in zwei Ausführungen an: als Standard Program für 99 US-Dollar oder als Enterprise Program für 299 US-Dollar. Die zweite Variante dürfte lediglich für größere Firmen interessant sein, die eine Inhouse-Distribution ihrer eigenen Anwendungen anstreben.

Erst eine iPhone Developer License ermöglicht es, unsere Anwendungen auf dem persönlichen Gerät zu installieren und zu testen. In Frage kommt hierfür jedes iPhone bzw. jeder iPod touch. Der iPhone-Simulator alleine reicht nicht aus. Er unterstützt beispielsweise keine Kamerafunktionalität und keine vollwertigen Multi-Touch-Gesten. Hinzu kommt, dass die physikalischen Restriktionen der Plattform, wie etwa Speichervolumen und Performanz, erst auf einem Gerät offensichtlich werden. Zum Wohle der Anwender ist ein Gerät also unumgänglich.

Bei der Erstellung dieses Buches liegt das iPhone SDK in der Version 3.0 vor. Um dieses auf seinem lokalen Mac installieren zu können, benötigt man entweder Mac OS X Leopard ab Version 10.5.7 oder Mac OS X Snow Leopard ab Version 10.6.0. Ältere Versionen von Leopard und Snow Leopard können über den Apple-Software-Update auf diese Versionen aktualisiert werden.
Damit das Entwicklerkit nahtlos mit dem eigenen Gerät zusammenarbeitet, muss dieses ebenfalls auf das iPhone OS der Version 3.0 aktualisiert werden. iTunes 8.2 und eine vorhandene Internetverbindung lösen diese Aufgabe: es installiert das iPhone OS 3.0 auf dem jeweiligen iPhone oder iPod touch, damit das Gerät und SDK voll kompatibel sind.

Alle Anwendungen, welche seit der Veröffentlichung des iPhone SDK 3.0 in den App Store eingestellt werden sollen, müssen auf Basis des iPhone SDK 3.0 gebaut und kompiliert werden.

Welches Vorwissen ein Einsteiger in die Entwicklung für das iPhone mitbringen muss, kann nur recht schwammig formuliert werden. Aus unserer Sicht macht die Begeisterung bei dem Einstieg in ein neues Themenfeld sowieso die halbe Miete aus. Wie wir alle wissen: Wer begeistert ist, lernt am schnellsten.

- Allgemein: Wer solide Grundkenntnisse in einer objektorientierten Programmiersprache hat und den Umgang mit Entwicklungsumgebungen gewöhnt ist, sollte keine Probleme mit dem Einstieg haben. Ein Beispiel hierfür wären gute Kenntnisse und etwas Erfahrung in der Softwareentwicklung mit Java und Eclipse.
- Spezieller: Von Vorteil sind Kenntnisse im Umgang mit dem Mac bzw. dem iPhone, obwohl diese aus Sicht des Autors auch während der Entwicklung erlernt und erfragt werden können bzw. sich von selbst einstellen. Die idealen Voraussetzungen sind natürlich Kenntnisse von Objective-C und in der Entwicklung mit Cocoa, dem Framework für die Entwicklung von Anwendungen für den Mac.
- Dieses Buch ist so aufgebaut, dass es nur solide Kenntnisse in einer objektorientierten Programmiersprache voraussetzt. Wer Schleifen, Bedingungen, grundlegende Datentypen und die Konzepte der Objektorientierung aus anderen Programmiersprachen kennt, sollte mit diesem Buch problemlos zurechtkommen. Aus diesem Grund werden in dem Einstiegskapitel dieses Buches die Eckpfeiler von Objective-C erläutert, welche Einsteigern am meisten Verständnisprobleme bereiten können. Zu diesen gehören der Aufbau von Klassen, die Verwendung von Variablen und Properties und die Speicherverwaltung ohne Garbage-Collection. Wer mehr über Objective-C im Speziellen wissen möchte, sei auf weitere Literatur verwiesen. Dieses Buch erhebt den Anspruch, alle wesentlichen Grundlagen für einen kompletten Einstieg abzudecken, ohne den Leser, welcher Kenntnisse in der Entwicklung von Software haben dürfte, mit Schleifen-Konstrukten und Ähnlichem in den Schlaf zu wiegen.

## 1.2 Die iPhone-Formel 1

Willkommen in der iPhone-Formel 1! In diesem Unterkapitel wollen wir uns einen Überblick über die gesamte iPhone-Plattform verschaffen. Hierzu bedienen wir uns einiger Analogien mit der Formel 1. Vergleicht man das Schreiben einer Anwendung für das iPhone mit der Durchführung eines Rennens in der Formel 1, finden sich einige erstaunliche Parallelen zwischen beiden Welten. Wir gehen im Folgenden anhand dieser auf die Säulen der iPhone-Plattform und deren Werkzeuge ein.

Ähnlich zur Formel 1 liegt die iPhone-Entwicklung gerade voll im Trend. Neben der reinen Entwicklungstätigkeit räumt Apple dabei dem einzelnen Entwickler eine weitere Perspektive ein: er kann zum Einzelunternehmer werden. Diese Perspektive jedoch beinhaltet bereits ein Szenario, welches für den Leser dieses Buches zugrunde liegen könnte, nämlich einen soliden Programmierer, welcher neu auf der iPhone-Plattform ist und lieber heute als morgen seine Applikationen über den App Store vertreiben möchte.

Jeder Entwickler kann seine Produkte weltweit über den App Store verkaufen und den Preis hierfür selbst festlegen. Apple behält pro Applikationsverkauf 30 % des Verkaufspreises ein, der Rest wird an den jeweiligen Entwickler ausbezahlt. Wie man im App Store seine Applikation präsentiert und diese bewirbt, ist dem Entwickler selbst überlassen. Um die ganzen Abrechnungsangelegenheiten kümmert sich Apple, und dies weltweit über eine Plattform.

### 1.2.1 Das Qualifying – Objective-C und Cocoa Touch

Vor jedem Rennen versuchen sich die Fahrer bestmöglich zu platzieren, um im Rennen eine gute Ausgangsposition zu haben. Bei der iPhone-Programmierung findet das Qualifying in Objective-C und Cocoa Touch statt, da iPhone-Anwendungen in Objective-C realisiert werden und Cocoa Touch die zentrale API des iPhone OS darstellt.

Bei Objective-C handelt es sich generell um eine objektorientierte Erweiterung von C, deren Syntax und Konzeption stark an Smalltalk angelehnt ist. Entwickelt wurde diese bereits in den 80er Jahren von Brad Cox.

Aus dem Blickwinkel eines Objective-C-fremden Entwicklers, beispielsweise eines Java-Entwicklers, sticht ein Merkmal von Objective-C besonders hervor: das Konzept der Nachrichten, welches wir in Kapitel 2.2 im Detail kennenlernen werden. In Objective-C werden auf Instanzen keine Methoden aufgerufen, sondern es werden Nachrichten zwischen Objekten ausgetauscht. Ein Sender schickt eine Nachricht an einen Empfänger. Das heißt, dass der Empfänger entscheidet, wie er mit einer empfangenen Nachricht umgeht. In einem kurzen Syntax-Beispiel sieht dies wie folgt aus: `[object message]`. Die eckigen Klammern kapseln hierbei den Nachrichtenversand (in anderen Sprachen den Methodenaufruf). Nochmals verdeutlicht wird dies anhand eines etwas konkreteren Beispiels:

```
Car *car = [[Car alloc] init];
NSString *carType = [car carType];
```

**Listing 1.1: Nachrichten-Syntax in Objective-C**

Der Klasse `Car` werden die Nachrichten `alloc` und `init` geschickt, die dazu führen, dass eine Instanz dieser Klasse initialisiert wird. Anschließend wird der Default-Wagentyp dieser Instanz abgefragt und einer Variablen vom Typ `NSString` zugewiesen.

Zu den weiteren Sprachmerkmalen von Objective-C gehören unter anderem das späte Binden und eine strenge Typisierung, sowie die Konzepte der Protokolle, Kategorien und Klassendefinitionen. Darüber hinaus finden sich die aus der Java-Welt bekannten Konzepte der Reflection und der Klassenobjekte ebenfalls in Objective-C wieder. Das Wichtigste hiervon ist Bestandteil dieses Buches.

Seit Mac OS X 10.5 liegt Objective-C in der Version 2.0 vor. Es ist damit für die iPhone-Entwicklung maßgebend. Einzig die in Objective-C 2.0 eingeführte Garbage-Collection steht auf dem iPhone nicht zur Verfügung, was sich durch dessen limitierte Hardwareressourcen erklären lässt. Für alle anderen Erweiterungen aus 2.0 gilt dies nicht. Sie können problemlos auf dem iPhone eingesetzt werden. Die Rede ist hierbei von solchen Merkmalen wie der Properties- oder der Dot-Notation, sowie der schnellen Enumeration (siehe hierzu Kapitel 2.3).

Cocoa Touch ist die höchste Abstraktionsschicht auf dem iPhone OS. Dieses Framework erlaubt Zugriff auf die wichtigsten Systemkomponenten, wie z. B. die Kamera und das Fotoalbum, sowie den Accelerometer oder das Adressbuch. Es stellt alle bekannten iPhone-typischen Oberflächenkomponenten zur Verfügung und kapselt den Zugriff auf andere Ressourcen.

### 1.2.2 Das Rennen – die eigene Anwendung

Bei einem Formel-1-Rennen verfolgt jeder Fahrer seine Strategie. An diese hält sich auch das gesamte Team des Fahrers. Jeder Fahrer hat eine Idee im Kopf, wie er das Rennen mit seinem bestmöglichen Ergebnis bestreiten will. Die Motivationen hierfür mögen unterschiedlich sein. Der eine verfolgt ein möglichst hohes Preisgeld – den Verkauf möglichst vieler Applikationen –, der andere hat einfach so viel Spaß am Rennen, dass er seine Applikationen kostenlos in den App Store stellen möchte.

Alle beginnen jedoch zuerst mit einer Idee, die es anschließend umzusetzen gilt. Zu beachten sind hierbei, was die Konzeption einer iPhone-Anwendung anbelangt, vor allem die Restriktionen des mobilen Geräts, wie z. B. dessen reduzierte Eingabemöglichkeiten, verringerte Bildschirmgröße und die durchschnittlich sehr kurzen Nutzungsdauern von mobilen Applikationen.

Ein Motto von Apple hierfür lautet: *Chop it off!* Reduziere alle Anforderungen auf den kleinsten gemeinsamen Nenner – das, was ein mobiler Nutzer wirklich braucht und nutzen kann. Habe ein gutes Gefühl dabei, wenn Du wieder eine Anforderung streichst und die Applikation dadurch vereinfachst.

Ein empfehlenswertes Werkzeug für die Konzeption einer iPhone-Applikation ist ein Pinboard. Auf dieses klebt man die Skizzen der unterschiedlichsten Screens der gedachten Anwendung und versucht über aufgeheftete Fäden das Benutzerverhalten zu simulieren. Dieses Vorgehen ermöglicht es bereits frühzeitig, die Stärken und Schwächen der eigenen Idee zu simulieren, bevor eine Zeile Code geschrieben wurde. Für die Implementierung selbst stellt Apple alle Bordmittel zur Verfügung.

### 1.2.3 Der Rennwagen – Xcode

Das wichtigste und am meisten beachtete Werkzeug eines jeden Rennfahrers ist sein Rennwagen. Um manche Fahrzeuge wird ein regelrechter Kult betrieben. Sie bestehen aus einer Vielzahl unterschiedlichster Komponenten. Das Gleiche gilt für Xcode, dem Herzstück der iPhone-Entwicklung. Es vereint alle lokalen Komponenten, die für die iPhone-Anwendungsentwicklung notwendig sind, unter einem Dach und ist die zentrale Komponente des iPhone SDK.

Apple definiert Xcode als eine Suite von Entwicklungswerkzeugen, um Anwendungen für den Mac oder das iPhone zu erstellen. Es besteht aus verschiedenen Unteranwendungen, Kommandozeilen-Werkzeugen, Frameworks und Bibliotheken. Im Folgenden sollen die wichtigsten Komponenten kurz beleuchtet werden.

Der Kern des Ganzen ist die Xcode-Anwendung selbst. Mit ihrer Hilfe werden iPhone-Projekte angelegt und bearbeitet. Sie bietet einen Code-Editor mit Syntax-Highlighting und Code-Completion, sowie einen Debugger und eine Laufzeit-Konsole. Die Dokumentation und die API-Referenzen sind integriert. Dies gilt auch für die Anbindung an Werkzeuge zur Teamentwicklung wie Subversion oder CVS. Die Abbildung 1.2 zeigt uns das Fenster der Xcode-Anwendung, in welchem unser erstes gemeinsames iPhone-Projekt geöffnet ist, welches wir in Kapitel 2.1 neu anlegen und kennenlernen werden.

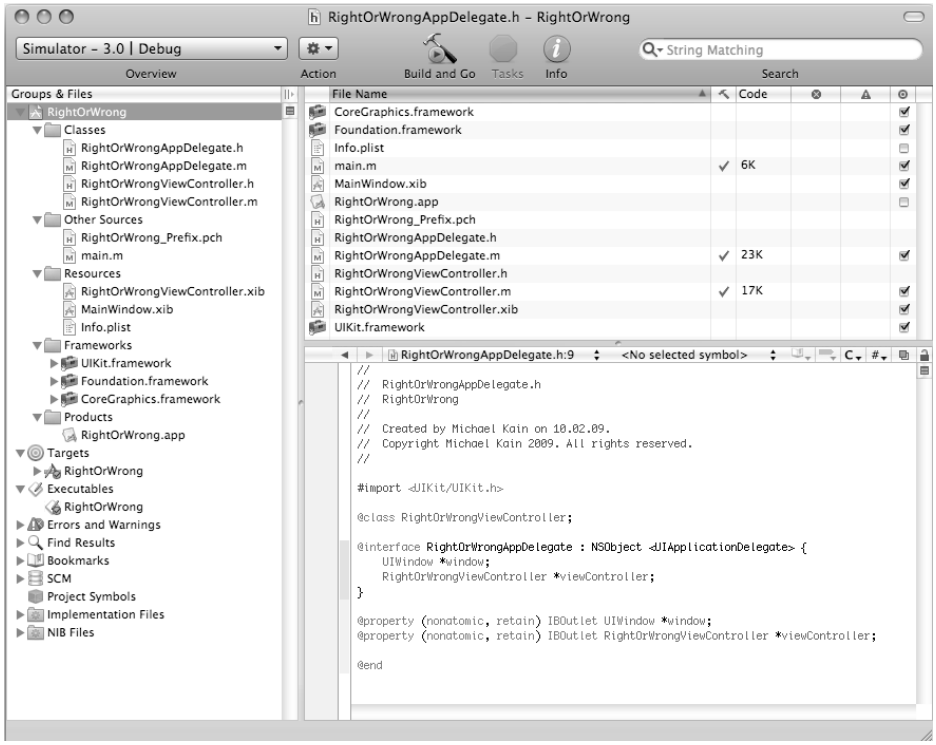


**Abbildung 1.2:** Projektfenster in Xcode

Die kommende Beschreibung von Xcode ist rein informell und als schnelle theoretische Einführung gedacht – ohne Rechner. Was wir für die Praxis brauchen, lernen wir alles ab Kapitel 2.

Ganz oben im Fenster befindet sich die Toolbar. Diese besteht aus sechs Komponenten, wovon wir vier nur beim Namen nennen und die zwei einführen, welche für einen Einsteiger am wichtigsten sind:

1. Pop-up-Menü *Overview*: Mit Hilfe dieses Pop-up-Menüs wird konfiguriert, auf Basis welches SDKs unsere iPhone-Applikationen kompiliert und gestartet werden sollen, ob im iPhone-Simulator oder auf einem Gerät, ob auf Basis der Version 2.2.1 oder 3.0. Es werden das *Active SDK* und die *Active Configuration* festgelegt.
2. Pop-up-Menü *Action*
3. Button *Build and Go*: Durch Drücken dieses Buttons wird das Projekt kompiliert und ausgeführt. Wer sich das Hello World von Apple herunterlädt, muss einzig auf diesen Button drücken, um die Anwendung im iPhone-Simulator zur Ausführung zu bringen.
4. Button *Tasks*
5. Button *Info*
6. Textfeld *Search*

Unterhalb der Toolbar befinden sich drei weitere Fenster, eines auf der linken und zwei auf der rechten Seite. Diese Anordnung gilt als das standardmäßige Layout eines Projektfensters. Nachdem wir in diesen drei Fenstern innerhalb dieses Buches ständig zugange sein werden, wollen wir diese ebenfalls kurz einführen:

1. Bereich *Groups & Files*: Dieses Fenster stellt alle Inhalte dar, welche zu einem Projekt gehören: Dateien, Verzeichnisse, Targets, Executables und andere Projektbestandteile bzw. -informationen. Üblicherweise sind diese in Gruppen organisiert. Der hier dargestellte Inhalt hat keinen Bezug zum Dateisystem, sondern folgt einer eigenen Ordnung.
2. *Detail View*: Wie der Name suggeriert, werden in diesem Fenster alle Details zu den Elementen aus den *Groups & Files* angezeigt, beispielsweise der Inhalt eines Ordners oder speziellere Informationen zu einem Element.
3. *Editor-Pane*: In diesem Fenster werden Inhalte editiert, allen voran der Quellcode. Hier werden wir die meiste Zeit bei der Entwicklung unserer iPhone-Applikationen verbringen.



Wer generell mehr über Xcode wissen möchte, soweit dies über den Rahmen dieses Buchs hinausgeht und deshalb dafür zwar nicht erforderlich ist, aber dennoch von Interesse sein kann, sei auf zwei Apple-Guides verwiesen. Diese stehen im iPhone Dev Center zum Download bereit: der Guide *Xcode Overview* und der *Xcode Workspace Guide*. Beide befinden sich unter *ADC Home* > *Reference Library* > *Guides* > *Tools* > *Xcode*.

Durch Klicken auf *Build and Go* wird aus Xcode heraus ein iPhone-Projekt gestartet. Mit der Einstellung *Active SDK – iPhone Simulator 3.0* im Pop-up-Menü *Overview* öffnet sich der iPhone-Simulator, um die eigenen Anwendungen vorab auf dem lokalen Rechner zu testen. Abbildung 1.3 stellt diesen mit geöffnetem Home-Screen dar.


**Abbildung 1.3:** Home-Screen des iPhone-Simulators

Ebenfalls in die Xcode-Anwendung integriert ist der Interface Builder: das Werkzeug von Apple für die grafische Gestaltung von Benutzeroberflächen. Die einzelnen Dialoge der eigenen Anwendungen können per Drag-and-Drop zusammengestellt und konfiguriert werden. Der Interface Builder stellt alle Elemente hierfür in seiner Bibliothek bereit. Er enthält fast alle auf dem iPhone bekannten GUI-Elemente.

Als Letztes verbleiben noch der Organizer und die zwei Analysewerkzeuge Instruments und Shark. Ersterer verwaltet alle Geräte, Entwicklerprofile und Anwendungen, die auf dem eigenen iPhone bzw. iPod touch installiert sind. Das Analysewerkzeug Instruments unterstützt unter anderem bei der Beobachtung des Speicherverbrauchs und bei der Auffindung von Speicherlöchern. Shark dient zur Optimierung hinsichtlich der Performanz der eigenen Anwendung.

### 1.2.4 Der Boxenstopp – Dev Center und Program Portal

Neben dem Rennwagen ist für das Gelingen eines Formel-1-Rennens eine zweite Komponente von zentraler Bedeutung: der Boxenstopp. Jeder Fahrer begibt sich während eines Rennens mindestens einmal in die Box, um nachzutanken oder Reparaturen an seinem Wagen vornehmen zu lassen. Der Fahrer selbst hat, während er in der Boxengasse steht, einen relativ geringen Handlungsspielraum, da er sich auf die Leistung seines Mechanikerteams verlassen muss.

Das Szenario eines Boxenstopps ist vergleichbar mit den externen, im Web verfügbaren Werkzeugen, die bei der iPhone-Programmierung eines große Rolle spielen: dem iPhone Dev Center und dem Program Portal. Diese befinden sich nicht auf dem lokalen Rechner, sind im Vergleich zur tatsächlichen Konzeptions- und Entwicklungszeit relativ wenig im Einsatz, aber dennoch zwingend erforderlich, um Anwendungen in den App Store einzustellen.

Beim iPhone Dev Center kann man sich kostenlos registrieren. Es stellt die komplette Dokumentation in Form von Guides, Referenzen, Videos und Code-Beispielen zur Verfügung. Auch das iPhone-Entwicklerkit lässt sich von dort herunterladen (siehe Voraussetzungen).

Nach dem Erwerb einer iPhone Developer License öffnet sich innerhalb des iPhone Dev Center (Abbildung 1.4) eine Schaltfläche, die auf das Program Portal verlinkt. Innerhalb des Program Portal erledigt der Entwickler all die technischen Aufgaben, die mit dem Testen von Applikationen auf den eigenen Geräten oder mit der Distribution von Applikationen zu tun haben. Der Fokus liegt hierbei klar auf dem technischen Sicherheitsaspekt. Die eigenen Geräte müssen registriert werden, für jede eigene Anwendung muss eine App-ID vergeben werden und für jede Art der Installation einer Anwendung muss ein eigenes Profil beantragt werden: eines für den Test und eines für die App Store-Distribution.


**Abbildung 1.4:** Startseite des iPhone Dev Center

### 1.2.5 Die Vermarktungsrechte – iTunes Connect und iTunes

Mit der Vergabe ihrer verschiedenen Vermarktungsrechte verdient die Formel 1 ihr Geld. Bei der iPhone-Anwendungsentwicklung erfüllen diesen vertrieblichen Aspekt zwei weitere, webbasierte Werkzeuge: iTunes Connect und iTunes.

Mit Hilfe von iTunes Connect verwaltet jeder Entwickler seine Anwendungen im iTunes-App Store: er editiert deren Screenshots und Werbetexte, legt sie Apple zur Überprüfung und Freigabe vor, beobachtet deren Downloads und Verkäufe und sieht seine monatlichen Abrechnungen ein.

Es empfiehlt sich, frühzeitig die Aktivierung des Paid Applications Agreement in Angriff zu nehmen. Dieses muss zwingend aktiviert sein, wenn Anwendungen über den App Store verkauft werden sollen. Möglich ist dies in iTunes Connect, unter *Contracts, Tax & Banking Information*. Dieser Prozess hat sich als relativ zeitaufwendig erwiesen. Wer ausschließlich kostenlose Anwendungen über den App Store vertreiben möchte, kann sich diese Mühen sparen. Mehr zum Paid Applications Agreement lesen Sie in Kapitel 5.2.2.

Über iTunes selbst und dessen App Store werden schließlich weltweit die Anwendungen direkt von Apple an die Endkunden verkauft. Mit Hilfe von iTunes kann der Entwickler die Darstellung seines Endproduktes überprüfen und weltweite Kundenrezensionen aus unterschiedlichsten Teilen der Welt abrufen.

# 2 Einstieg in die Welt des iPhones

Unseren Einstieg in die Welt des iPhones beginnen wir mit einem Beispiel: unserer ersten gemeinsamen iPhone-Applikation. Diese geht ein wenig über das Niveau eines Hello World hinaus, aus gutem Grund. In den auf dieses Beispiel folgenden Kapiteln werden wir uns auf diese erste Anwendung beziehen, um uns verschiedene Aspekte der Anwendungsentwicklung zu erarbeiten. Der Schwerpunkt liegt dabei auf den Grundlagen der Programmiersprache für das iPhone, Objective-C. Den Anfang macht ein Kapitel über den Aufbau von Klassen und die Definition von Nachrichten in Objective-C. Daran schließen sich zwei weitere Kapitel mit ähnlichem Themenschwerpunkt an. Im ersten Kapitel werden Variablen und Properties beleuchtet, wohingegen im zweiten Kapitel der Schwerpunkt auf der manuellen Speicherverwaltung liegt. Dieses Kapitel sollte uns als Einsteigern besonders am Herzen liegen, zumal es auf dem iPhone trotz der Unterstützung von Objective-C 2.0 keine Garbage-Collection gibt. Den Abschluss dieses Einstiegskapitels bilden zwei für sich thematisch recht alleinstehende Unterkapitel: eines über den Lebenszyklus von iPhone-Applikationen und eines über die Verwendung von eigenen Grafiken.

## 2.1 Meine erste Applikation: Richtig oder Falsch?

Sie sind Ihrem Traum nun ganz nah. In Kürze werden Sie Ihre erste Anwendung für das iPhone selbst implementieren und auf Ihrem eigenen Mac zum Laufen bringen, mit Hilfe von Xcode und dem iPhone-Simulator. Sie werden überrascht sein, wie wenig Code hierfür notwendig ist und welches fortgeschrittene Ergebnis Sie bereits am Ende dieses Kapitels vorliegen haben.

Üblicherweise beginnen IT-Fachbücher an dieser Stelle mit einer Hello-World-Anwendung. Wir brechen mit dieser Tradition – aus guten Gründen. Zum einen stellt Apple bereits selbst eine Hello-World-Applikation als Einstiegsbeispiel zur Verfügung. Zum anderen ist unsere erste Anwendung umfangreicher und anspruchsvoller als jedes lahme Hello World und bringt somit hoffentlich mehr Spaß. Außerdem lassen sich komplexere Zusammenhänge der iPhone-Anwendungsentwicklung einfacher anhand eines zentralen Beispiels veranschaulichen. Es genügt also dieses eine Beispiel, um alle für einen ersten Eindruck relevanten Zusammenhänge zusammenzufassen und zu erklären. Aus diesem Grund werden sich viele der nachfolgenden Unterkapitel auch auf dieses Beispiel beziehen.

Wir werden gemeinsam eine Anwendung mit dem Namen RightOrWrong realisieren. Abbildung 2.1 stellt einen Screenshot dieser Anwendung dar. Sie besteht aus zwei Buttons und einer farbigen Fläche. Diese ist entweder – wie zu Beginn – grün oder aber rot gefärbt. Wenn zuletzt der Right-Button gedrückt wurde, zeigt die Fläche immer grün an, analog dazu rot, wenn vorher der Wrong-Button gedrückt wurde. Es wäre also theoretisch möglich, diese Anwendung dazu zu nutzen, um durch das Hochhalten seines iPhones Zustimmung oder Ablehnung zu etwas zu signalisieren. Bis wir diese aber auf unserem iPhone installieren, dauert es noch etwas. Konzentrieren wir uns jetzt auf die Umsetzung unserer ersten gemeinsamen App mit Hilfe von Xcode und dem Interface Builder.


**Abbildung 2.1:** Screenshot RightOrWrong

Dieses Kapitel wird am besten ohne ein verbundenes iPhone / iPod touch ausgeführt, da dies den Einsteiger mehr irritieren kann als es ihm nützt und in diesem Kapitel nicht erforderlich ist.

Um mit Xcode ein neues Projekt anzulegen, müssen wir Xcode vorab starten. Dies geschieht durch Doppelklick auf das Xcode-Symbol im Verzeichnis */Developer/Applications*. Im Anschluss daran öffnet sich der Xcode-Willkommensbildschirm. Dieser ist in Abbildung 2.2 zu sehen.

Dieses Fenster kann bedenkenlos geschlossen werden. Wie auf Mac OS X üblich, bleibt die Xcode-Anwendung daraufhin trotzdem aktiv. Davon kann man sich mit einem Blick in die oberste Leiste des Bildschirms überzeugen. Xcode ist immer noch die ausgewählte und geöffnete Applikation. Ein neues Projekt wird in Xcode über das Menü FILE | NEW PROJECT... angelegt. Anschließend öffnet sich der New Project Assistant. Dieser ist in Abbildung 2.3 dargestellt. Unter iPHONE OS | APPLICATION stehen sechs verschiedene Projektvorlagen zur Auswahl.