



**2.**

Auflage



# iOS-Apps Christian Bleske programmieren mit **Swift**

Der leichte Einstieg in die Entwicklung für iPhone,  
iPad und Co. – inkl. Apple Watch und Apple TV

**dpunkt.verlag**



**Christian Bleske** ist Autor, Trainer und Entwickler. Sein Arbeitsschwerpunkt ist die Entwicklung von Client/Server- und mobilen Anwendungen. In vielen namhaften Entwicklerzeitschriften erscheinen seine Fachaufsätze. Er lebt in Witten im Ruhrgebiet.

Papier  
plus<sup>+</sup>  
PDF.

Zu diesem Buch – sowie zu vielen weiteren dpunkt.büchern – können Sie auch das entsprechende E-Book im PDF-Format herunterladen. Werden Sie dazu einfach Mitglied bei dpunkt.plus<sup>+</sup>:

[www.dpunkt.de/plus](http://www.dpunkt.de/plus)

**Christian Bleske**

# **iOS-Apps programmieren mit Swift**

**Der leichte Einstieg in die Entwicklung für iPhone,  
iPad und Co. – inkl. Apple Watch und Apple TV**

2., aktualisierte und erweiterte Auflage

 **dpunkt.verlag**

Christian Bleske  
cb.2000@hotmail.de  
christianbleske.wordpress.com

Lektorat: René Schönfeldt, Sandra Bollenbacher  
Copy-Editing: Petra Kienle, Fürstfeldbruck  
Satz: Ill-satz, Husby  
Herstellung: Susanne Bröckelmann  
Umschlaggestaltung: Helmut Kraus, www.exclam.de  
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

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.

ISBN:  
Print 978-3-86490-438-7  
PDF 978-3-96088-074-5  
ePub 978-3-96088-075-2  
mobi 978-3-96088-076-9

2., aktualisierte und erweiterte Auflage  
Copyright © 2017 dpunkt.verlag GmbH  
Wiebinger Weg 17  
69123 Heidelberg

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.

Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.

Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.

5 4 3 2 1 0

---

# Vorwort

*Liebe Leserinnen und Leser,*

mobile Geräte wie Smartphones, Tablets und Smartwatches erobern den Erdball, gleichzeitig stagniert der Absatz von PCs oder ist sogar rückläufig. Durch den massenhaften Absatz mobiler Geräte wird auch der Hunger nach Anwendungen, die darauf ausgeführt werden können, immer größer – ein spannender und manchmal sogar lukrativer Markt für Entwickler.

Im Jahr 2011 schrieb ich mein erstes Buch für Entwickler, die sich diesen Markt über Windows Phone erschließen wollten (»Windows Phone 7 Apps entwickeln«). Im Jahr darauf erschien das zweite Buch zur Entwicklung von Apps, dieses Mal für Android (»Java für Android«). Nun, mit dem vorliegenden Buch zur iOS-Programmierung, schließt sich der Kreis.

## **iOS und Swift**

iOS ist das universelle Betriebssystem von Apple, das sowohl die unterschiedlichen iPhone- und iPad-Modelle sowie iPod Touch, CarPlay, Apple TV (tvOS) und Apple Watch (Basis von watchOS ist iOS) antreibt.

Neben den unterschiedlichen Geräten gibt es seit nunmehr zwei Jahren bei Apple auch eine passende moderne Programmiersprache, die die Entwicklung für die iOS-Plattform erleichtern soll: Swift. Swift ist der Nachfolger von Objective-C, das mit seinen fast 30 Jahren langsam, aber sicher abgelöst werden soll. Viele Entwickler setzen ihre neuen Projekte deshalb nur noch mit Swift um.

In diesem Jahr ist Swift in der Version 3.0 erschienen und es enthält viele Änderungen. Dies ist die zweite Auflage des Buches, neben der Aktualisierung auf Swift 3.0 finden Sie in dieser Auflage auch neue Themen. So ist ein Kapitel zum Thema Apple TV 4 (tvOS) hinzugekommen. In diesem erfahren Sie, wie Apps für diese Plattform geschrieben werden. Außerdem gibt es im Buch jetzt ein Kapitel zum Thema parallele Verarbeitung. Auch die Programmierung von Apps mit Unterstützung für Handoff, Spotlight und 3D Touch wird demonstriert. Zuletzt wird nun auch das Thema Drucken aus Apps heraus behandelt.

## Leser

Konzentrieren werde ich mich auch in Bezug auf die Leserschaft dieses Buches. Es spricht Leser an, die bisher noch keine iOS-Anwendungen entwickelt haben und auch mit Swift nicht vertraut sind. Auch mit Apples Entwicklungsumgebung Xcode müssen Sie sich nicht auskennen, denn die lernen Sie hier ausführlich kennen. Sie sollten aber grundlegende Kenntnisse in einer beliebigen anderen Programmiersprache haben und über einen Mac-Computer als Entwicklungsrechner verfügen.

Damit ausgestattet sollten Sie kaum Probleme haben, meinen Ausführungen zu folgen und die Beispiele nachzuvollziehen. Nach der Lektüre werden Sie dann genügend über das Betriebssystem iOS, die Entwicklungsumgebung Xcode und die Programmiersprache Swift gelernt haben, um eigene Apps damit zu programmieren.

## Beispielcode

Wie in meinen Büchern üblich, so finden Sie auch hier wieder viele Beispiele. Sie sind allesamt mit Swift 3 und Xcode 8 entwickelt worden, und ihren Code können Sie online von der Buch-Webseite herunterladen unter

*[www.dpunkt.de/swift2](http://www.dpunkt.de/swift2)*

Als Alternative hierzu ist der Code ebenso über meinen GitHub-Account <https://github.com/christianbleske> verfügbar. Dort können die Beispiele auch einzeln heruntergeladen werden. Wenn neue Swift-Versionen erscheinen, dann werde ich an dieser Stelle aktualisierte Versionen der Beispiele zum Download bereitstellen.

Den Rückmeldungen der Leserinnen und Leser zu meinen bisherigen Büchern habe ich entnommen, dass sich viele etwas komplexere Beispiele und eine »richtige« App wünschen. Diesem Wunsch habe ich versucht in diesem Buch gerecht zu werden. So werden die meisten Themen anhand einer kompletten App, etwa einer App zur Wettervorhersage, erläutert.

Darüber hinaus zieht sich eine besondere Beispiel-App durchs ganze Buch: eine Passwortverwaltung, die in verschiedenen Kapiteln immer wieder aufgegriffen und themenbezogen weiterentwickelt wird.

Ein weiteres besonderes Beispiel ist der Schnelleinstieg, in dem ich Ihnen in kompakter Form die wichtigsten Dinge zeige, die die iOS-Entwicklung mit Swift ausmachen. Das komplette Einstiegsbeispiel finden Sie zusätzlich in Form eines kleinen Videos auf der Buch-Webseite.

**Das Blog zum Buch**

Begleitend zum Buch gibt es außerdem ein Blog, in dem ich Ihnen zusätzliche Informationen sowie ggf. Fehlerkorrekturen zum Buch bereitstelle und auf Änderungen in Xcode und der iOS-API hinweise. Sie finden es unter

*<http://christianbleske.wordpress.com>*

Über das Blog oder per E-Mail unter *cb.2000@hotmail* dürfen Sie mich auch gerne persönlich ansprechen, wenn bei der Lektüre zu diesem Buch Fragen auftreten sollten.

*Christian Bleske*, im September 2016



---

# Inhaltsübersicht

1	<b>Einleitung</b>	1
2	<b>Schnellstart mit Swift</b>	41
3	<b>Einstieg in Swift</b>	57
4	<b>Objektorientierte Programmierung mit Swift</b>	99
5	<b>Grundlagen der App-Entwicklung</b>	153
6	<b>Fehlersuche und Problembehandlung</b>	227
7	<b>Tabellen und Controller</b>	245
8	<b>Core Data</b>	273
9	<b>Internet und Netzwerke</b>	291
10	<b>Sensoren</b>	317
11	<b>iOS Maps</b>	339
12	<b>Lokalisierung</b>	347
13	<b>Universal Apps</b>	363
14	<b>App Extensions</b>	373
15	<b>WatchKit</b>	381
16	<b>Apps für Apple TV 4</b>	393
17	<b>NSOperation und Dispatch Queue</b>	403
18	<b>Handoff-API</b>	411
19	<b>Das Core Spotlight Framework</b>	417
20	<b>3D Touch verwenden</b>	427
21	<b>Drucken unter iOS</b>	437
22	<b>Apps verteilen</b>	445
	<b>Nachwort</b>	459
	<b>Index</b>	461



---

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	iPhone, iPad & Co. . . . .	2
1.2	Für wen ist dieses Buch gedacht? . . . . .	2
1.3	Benötigte Hard- und Software . . . . .	3
1.3.1	Welcher Mac genügt? . . . . .	3
1.3.2	Welche macOS-Version? . . . . .	3
1.4	Xcode und das iOS-SDK . . . . .	4
1.4.1	Download und Installation von Xcode . . . . .	5
1.4.2	Xcode im Detail . . . . .	6
1.4.3	Projektübersicht . . . . .	9
1.4.4	Quellcodefenster . . . . .	9
1.4.5	Interface Builder . . . . .	11
1.4.6	Object Library & Co. . . . .	11
1.4.7	Inspektoren . . . . .	13
1.5	Vorlagen in Xcode . . . . .	15
1.5.1	Schnellstart: Hallo Playground . . . . .	15
1.5.2	Die Projektvorlagen von Xcode für iOS-Apps . . . . .	16
1.5.3	Schnellstart: Die Master-Detail-Application-Vorlage . . . . .	16
1.5.4	Schnellstart: Die Page-Based-Application-Vorlage . . . . .	19
1.5.5	Schnellstart: Die Single-View-Application-Vorlage . . . . .	20
1.5.6	Schnellstart: Die Tabbed-Application-Vorlage . . . . .	21
1.5.7	Bestandteile eines Projekts . . . . .	22
1.6	Apps ausführen . . . . .	26
1.6.1	App im Simulator . . . . .	27
1.6.2	Die App auf dem iOS-Gerät . . . . .	29
1.6.3	Der iOS-Simulator im Detail . . . . .	29
1.7	Mehrere Xcode-Versionen parallel verwenden . . . . .	31
1.7.1	Ältere Versionen von Xcode finden . . . . .	31
1.7.2	Weitere Xcode-Versionen installieren . . . . .	33
1.8	Das Apple Developer Program – Anlaufstelle für Entwickler . . . . .	33
1.8.1	Von Zertifikaten, Profilen und Identitäten . . . . .	35
1.8.2	App auf ein angeschlossenes iOS-Gerät übertragen . . . . .	38

<b>2</b>	<b>Schnellstart mit Swift</b>	<b>41</b>
2.1	Das Projekt anlegen	42
2.2	Die Oberfläche mit dem Interface Builder bauen	46
2.3	Controls und Ereignisse	50
2.4	Mit Swift zum Ergebnis	53
<b>3</b>	<b>Einstieg in Swift</b>	<b>57</b>
3.1	Variablen, Konstanten und Zuweisungen	57
3.1.1	Zuweisungsoperator	57
3.1.2	Konstanten	58
3.2	Datentypen	59
3.2.1	Strings	59
3.2.2	Ganze Zahlen und Fließkommazahlen	62
3.2.3	Wahrheitswerte	64
3.2.4	Aufzählungen (Enumerationen)	65
3.2.5	Felder (Arrays)	67
3.2.6	Dictionarys	69
3.2.7	NS-Datentypen	71
3.2.8	NSNumber	72
3.2.9	NSString	73
3.2.10	NSDate	73
3.2.11	NSRange	74
3.2.12	TimeInterval	74
3.2.13	NSArray/NSMutableArray	75
3.2.14	NSDictionary und NSMutableDictionary	75
3.3	Operatoren	75
3.3.1	Boolesche Operatoren	75
3.4	Arithmetische Operatoren	76
3.5	Kontrollstrukturen	77
3.5.1	Die Fallunterscheidung (if)	77
3.5.2	Mehrfachauswahl	79
3.6	Schleifen	80
3.6.1	Kopfgesteuerte Schleifen (while-Schleife)	81
3.6.2	Fußgesteuerte Schleifen (repeat...while-Schleife)	81
3.6.3	Zählschleifen (for-Schleife)	81
3.7	Funktionen	84
3.7.1	Funktionen – Parameter übergeben	85
3.7.2	Funktionen – Werte zurückgeben	86
3.7.3	Funktionen – Mehrere Werte zurückgeben (Tupel)	87
3.7.4	Funktionen – Externe Parameternamen	87

3.7.5	Funktionen – Parameter vorbelegen	88
3.7.6	Funktionen – Weitere Parametertypen	89
3.7.7	Verschachtelte Funktionen	91
3.7.8	Closures	91
3.7.9	Optionals	93
3.8	Strukturen Teil 1 (Structs)	95
<b>4</b>	<b>Objektorientierte Programmierung mit Swift</b>	<b>99</b>
4.1	Grundlagen der OOP	99
4.1.1	Strukturen aus objektorientierter Sicht (Strukturen Teil 2)	99
4.1.2	Was sind Objekte?	100
4.1.3	Was sind Klassen?	101
4.1.4	Public, Private und wer noch?	105
4.2	Eigenschaften (Properties)	110
4.3	Methoden	114
4.3.1	Instanzmethode (Instance Methods)	114
4.3.2	Klassenmethode (Type Methods)	117
4.3.3	Der Initialisierer (Konstruktor)	118
4.3.4	Der Deinitialisierer (Dekonstruktor)	121
4.4	Vererbung	121
4.4.1	Erben (Ableitung) von Klassen	122
4.4.2	Zugriff auf Elemente der vererbenden Klasse	123
4.4.3	Initializer delegation	123
4.4.4	Überschreiben von Methoden	125
4.4.5	Überschreiben von Eigenschaften	126
4.4.6	Das Überschreiben von Elementen verhindern	127
4.5	Schnittstellen	128
4.5.1	Was sind Protokolle?	128
4.5.2	Ableitung von Protokollen	130
4.6	Erweiterung von Typen (Klassen, Strukturen und Enumerationen)	132
4.6.1	Subscripts	132
4.6.2	Verschachtelte Typen (Nested Types)	133
4.6.3	Erweiterungen (Extensions)	134
4.6.4	Optional Chaining	137
4.7	Typumwandlung (is & as) und (Any & AnyObject)	139
4.8	Generics	144
4.9	Speicherverwaltung (Automatic Reference Counting)	145

<b>5</b>	<b>Grundlagen der App-Entwicklung</b>	<b>153</b>
5.1	Storyboard und Interface Builder	153
5.1.1	App im Storyboard	153
5.2	Bausteine einer iOS-App	159
5.2.1	View, ViewController und wer noch?	161
5.2.2	View-Ereignisse und View-Lebenszyklus	169
5.2.3	Outlets und Actions	171
5.2.4	MVC – Model View Controller	173
5.2.5	Controls	175
5.2.6	UIAlertController	186
5.3	Delegate	192
5.4	Gestenverarbeitung (Touch Events)	196
5.5	AutoLayout	204
5.6	Workshop – Passwortverwaltung – Teil 1	213
5.6.1	Planung der App	214
5.6.2	Umsetzung des Projekts – Teil 1	217
<b>6</b>	<b>Fehlersuche und Problembehandlung</b>	<b>227</b>
6.1	Breakpoints im Quellcode setzen	227
6.2	Inspizieren von Variablen	229
6.3	View Debugging	231
6.4	Fehlerbehandlung mit »try catch«, (NS)Error & Co.	234
6.5	Fehlerbehandlung bei knappem Speicher	240
6.6	Voraussetzungen prüfen, Fehler vermeiden	242
<b>7</b>	<b>Tabellen und Controller</b>	<b>245</b>
7.1	Schnellstart: App mit Master-Detail-Application-Vorlage	245
7.1.1	Datenquelle für Master-Detail hinzufügen	247
7.1.2	Das Projekt testen	251
7.2	UITableView und Controller	252
7.3	UITableViewCell	254
7.4	CustomCell	256
7.5	NavigationBar anpassen	261
7.6	Workshop – Passwortverwaltung – Teil 2	263
<b>8</b>	<b>Core Data</b>	<b>273</b>
8.1	Was ist Core Data?	273
8.2	Ein neues Projekt mit Core Data	273

---

8.3	Ein Modell für die Speicherung	275
8.4	Core-Data-Klassen verwenden	279
8.5	Workshop – Passwortverwaltung – Teil 3	281
<b>9</b>	<b>Internet und Netzwerke</b>	<b>291</b>
9.1	Das UIWebView	291
9.2	Wetter mit JSON	297
9.2.1	Mit der App zum (aktuellen) Wetter	302
9.3	Dateiaustausch mit AirDrop	307
9.4	Daten in der iCloud	311
<b>10</b>	<b>Sensoren</b>	<b>317</b>
10.1	Kamera	317
10.2	Mikrofon	320
10.3	Beschleunigungssensor	325
10.4	Positionsdienste	329
10.5	Workshop – Passwortverwaltung – Teil 4	336
<b>11</b>	<b>iOS Maps</b>	<b>339</b>
11.1	Das MapKit-Framework	339
11.2	Das MapView-Control	341
11.3	Markierung mit Annotation	342
11.4	Markierung auf der Karte anzeigen	343
11.5	Icon ändern	344
<b>12</b>	<b>Lokalisierung</b>	<b>347</b>
12.1	Statische Texte in der GUI	347
12.2	Eine Sprache hinzufügen	349
12.3	Lokalisierung von Bildern	352
12.4	Dynamische Texte	354
12.5	Lokalisierung des App-Namens	357
12.6	Workshop – Passwortverwaltung – Teil 5	357
<b>13</b>	<b>Universal Apps</b>	<b>363</b>
13.1	Universal Apps	363
13.2	Ein Projekt als Universal App anlegen	363
13.3	Bilder in der App	365

---

13.4	Portrait oder Landscape? .....	368
13.5	iPad oder iPhone? .....	369
13.6	Icons und Launch Images .....	370
13.7	Workshop – Passwortverwaltung – Teil 6 .....	371
<b>14</b>	<b>App Extensions</b> .....	<b>373</b>
14.1	Widgets und iOS .....	373
14.2	Ein Projekt für Widgets .....	374
14.3	Today Extension anlegen .....	375
14.4	Code und GUI des Widgets entwerfen .....	377
14.5	Test des Widgets .....	378
<b>15</b>	<b>WatchKit</b> .....	<b>381</b>
15.1	Apple und die Watch .....	381
15.2	Zweiteilung .....	382
15.3	Das Projekt im Projekt .....	383
15.4	Das WatchKit-Projekt .....	385
15.5	GUI für die Watch-App .....	386
15.6	Der Code im anderen Abschnitt .....	388
15.7	Wie kommt die App auf die Watch? .....	390
<b>16</b>	<b>Apps für Apple TV 4</b> .....	<b>393</b>
16.1	Apps im Fernsehen .....	393
16.2	Projektvorlagen für Apple-TV-Apps .....	394
16.3	Auflösungen berücksichtigen .....	395
16.4	Fokus und Steuerung der App .....	395
16.5	Der Simulator und Apple Remote .....	396
16.6	Würfel im Fernsehen .....	397
16.7	Die Programmlogik der Würfel-App .....	398
16.8	Icons hinzufügen .....	400
16.9	Installation der App .....	402
<b>17</b>	<b>NSOperation und Dispatch Queue</b> .....	<b>403</b>
17.1	Operationen und Queues .....	403
17.2	Grand Central Dispatch und Queues im Überblick .....	403
17.2.1	Serielle Queues .....	404
17.2.2	Konkurrierende Queues .....	404

---

17.3	Queues in der praktischen Anwendung .....	405
17.4	Serielle Abarbeitung .....	407
17.5	NSOperation Queues .....	408
<b>18</b>	<b>Handoff-API</b>	<b>411</b>
18.1	Einrichtung von Handoff .....	411
18.2	Die Klasse NSUserActivity .....	412
18.3	Das Projekt einrichten .....	413
18.4	Activity anlegen .....	414
18.5	Update und Ereignisse .....	415
<b>19</b>	<b>Das Core Spotlight Framework</b>	<b>417</b>
19.1	Indizierung mit Spotlight .....	417
19.2	Neue Struktur für ein Zitat .....	419
19.3	Den Text analysieren .....	419
19.4	NSUserActivity verwenden .....	421
19.5	Activity-Instanz prüfen .....	423
19.6	Ein Activity wiederherstellen .....	424
<b>20</b>	<b>3D Touch verwenden</b>	<b>427</b>
20.1	Was ist 3D Touch? .....	427
20.2	3D Touch verwenden .....	428
20.3	Waage mit 3D Touch .....	429
20.4	Shortcuts definieren .....	431
20.5	Shortcut im Code .....	433
<b>21</b>	<b>Drucken unter iOS</b>	<b>437</b>
21.1	Was ist Airprint? .....	437
21.2	Der Printer Simulator .....	438
21.3	Drucken aus einem TextView-Control .....	439
21.4	Generierung eines PDF-Dokuments .....	441
	21.4.1 Drucklayout mit HTML .....	442
	21.4.2 PDF im Code erzeugen .....	443
<b>22</b>	<b>Apps verteilen</b>	<b>445</b>
22.1	Der Distributionsprozess .....	445
22.2	iTunes Connect verwenden .....	447
	22.2.1 Meine Apps (App registrieren) .....	448

---

22.3	Apps verteilen . . . . .	453
22.3.1	App Store . . . . .	453
22.3.2	Ad Hoc Deployment . . . . .	456
22.3.3	Enterprise Deployment . . . . .	457
22.3.4	Bereitstellung einer App zur Verteilung . . . . .	457
	<b>Nachwort</b>	<b>459</b>
	<b>Index</b>	<b>461</b>

# 1 Einleitung

Viele Normalbenutzer waren wohl relativ enttäuscht, als auf der *World Wide Developers Conference* (WWDC) 2014 keine Uhr, kein Telefon bzw. kein neuer Rechner vorgestellt wurde – die Entwicklergemeinde freute sich dafür umso mehr. Denn im Juni 2014 wurde auf der WWDC eine neue Programmiersprache nebst passender Integration in Apples Entwicklungswerkzeug Xcode präsentiert. *Swift* heißt die neue Sprache, was man unter anderem als »Mauersegler« übersetzen kann, und der gilt als ganz schneller Vogel. Aber warum eine neue Programmiersprache? Mit Objective-C hatte man doch eigentlich die Mutter aller Sprachen im Programm – oder etwa doch nicht?

Um diese Frage zu beantworten, sehen wir uns folgenden Objective-C-Code-Auszug an: Hier wird einer String-Variablen eine URL zugewiesen und anschließend in einem Browser-Control aufgerufen:

```
NSString *urlString=@"http://";
urlString = [urlString stringByAppendingString:Id];
NSURL *url = [NSURL URLWithString:[NSString
    stringWithFormat:@"%@" ,urlString]];
NSURLRequest *requestObj = [NSURLRequest
    requestWithURL:url];
[uiWebView loadRequest:requestObj];
```

Selbst wenn man ein Fan von Objective-C ist, wird man zugeben müssen, dass die Syntax dieser Sprache im Vergleich zu anderen (Assembler einmal ausgenommen) gewöhnungsbedürftig ist. Gerade Ein- und Umsteigern bereitet die komplexe Syntax von Objective-C oft Probleme. Apple bzw. die Geräte des Unternehmens zeichnen sich dadurch aus, dass sie besonders einfach zu bedienen sind. Von der Haus- und Hofsprache Objective-C kann man das leider nicht sagen.

Es musste also etwas Neues her, um sicherzustellen, dass der Strom von neuen Apps für die iOS-Plattform und die damit verbundenen Geräte auch zukünftig nicht abreißt. So kam man bei Apple vor einigen Jahren auf die Idee, Objective-C noch eine zweite Sprache zur Seite zu stellen – eine moderne Sprache mit einem zeitgemäßen Funktionsumfang (z.B. Generics), die leicht zu erlernen sein soll.

Man kann durchaus festhalten, dass es Apple mit Swift gelungen ist, genau die »Problemzonen« von Objective-C anzugehen und trotzdem kompatibel zum Rest von Apples SDK zu bleiben. Denn neben den bereits genannten Vorzügen von Swift gibt es auch eine Brücke in die »alte« (Objective-C-)Welt. Natürlich ist auch das passende API (Cocoa Touch) in Swift-Apps nutzbar. So muss nicht alles neu entwickelt werden.

## 1.1 iPhone, iPad & Co.

Eine Programmiersprache allein – und sei sie noch so gut – bringt nichts, wenn die zugrunde liegende Plattform bzw. das Betriebssystem keinen Erfolg hat. Das kann man von iOS nicht sagen, denn es ist auf vielen Geräten im Einsatz.

Daher wird in diesem Buch bewusst nicht allein das iPhone angesprochen, denn iOS treibt darüber hinaus natürlich auch das iPad und den iPod Touch, Apple TV und die Apple Watch an. Seinen Ursprung hat iOS in Apples Mac-Betriebssystem macOS (OS X). Im Prinzip handelt es sich um eine abgespeckte Variante dieses Betriebssystems, die für die Bedienung mit der Hand und für die Ausführung der Software auf schwächerer Hardware optimiert wurde.

An dieser Stelle sollen Sie aber nicht mit der Historie rund um Apples Hard- und Software gequält werden. Die ganze Story ist sicherlich bekannt.

### Hinweis

Seit der WWDC 2016 heißt das Betriebssystem für Macs (wieder) macOS 10.12 anstatt OS X.

## 1.2 Für wen ist dieses Buch gedacht?

Dieses Buch ist für Einsteiger mit Programmierkenntnissen gedacht. Fachbegriffe aus dem Bereich der Softwareentwicklung, wie z.B. IDE (Entwicklungsumgebung), Variable oder Schleife, sollten Sie also kennen. Ideal wäre es, wenn Sie bereits mit einer anderen Programmiersprache (z.B. C, C# oder Java) gearbeitet haben. Es wird also vorausgesetzt, dass Sie sich mit dem Thema Programmierung bereits beschäftigt haben. Leser ohne diese Kenntnisse werden es schwer haben, mit dem Buch zurechtzukommen.

Außerdem wird davon ausgegangen, dass Sie die Bedienung von macOS beherrschen, also beispielsweise wissen, wie man Dateien kopiert oder Anwendungen aus dem App Store installiert.

Die Entwicklungsumgebung Xcode müssen Sie nicht kennen.

## 1.3 Benötigte Hard- und Software

Eine wichtige Frage ist, welche Hard- und Software man benötigt, um Apps für iOS zu entwickeln – vor allem, wenn man bisher auf einer anderen Plattform, z. B. Windows, gearbeitet hat.

### 1.3.1 Welcher Mac genügt?

Haben Sie bereits einen Mac oder suchen Sie noch ein passendes Gerät? Generell gilt: Jeder Mac, auf dem Yosemite oder El Capitan läuft, kann zur Entwicklung von iOS-Apps verwendet werden. Die neueste Version von macOS 10.12 (Sierra) kann auf den folgenden Macs installiert werden:

- iMac (Late 2009 oder neuer)
- MacBook (2009 oder neuer)
- MacBook Air (Late 2010 oder neuer)
- MacBook Pro (Mid 2010 oder neuer)
- Mac mini (Mid 2010 oder neuer)
- Mac Pro (Mid 2010 oder neuer)

Ferner werden mindestens 2 Gigabyte Arbeitsspeicher (besser 4) und 8 Gigabyte Festplattenspeicher benötigt. Sehen Sie diese Werte als Minimalwerte an – mehr ist immer besser!

Ein Tipp, wenn Sie noch keinen Mac haben und einen kaufen möchten: Schauen Sie sich einmal einen Mini an. Warum? Wenn Sie bereits einen anderen Computer (PC) zu Hause haben, dann können Sie vorhandene Hardware (z. B. Monitor, USB-Tastatur, USB-Maus) weiterverwenden. Sofern Sie nicht auf gebrauchte Hardware setzen, ist das die günstigste Variante (ab ca. 600 Euro), um in die Mac-Welt einzusteigen.

### 1.3.2 Welche macOS-Version?

Yosemite, El Capitan oder Sierra – diese Versionen können Sie zur Entwicklung von Apps mit Swift verwenden. Ältere Versionen von macOS, z. B. Mountain Lion, können *nicht* zur Entwicklung verwendet werden. Welche Version ist besser? Sierra unterstützt zurzeit die größte Anzahl an APIs. Das liegt daran, dass Sierra die aktuellste macOS-Version ist. Man sollte aber generell darauf achten, dass es mit der neusten Version von macOS nicht zu Stabilitätsproblemen kommt. Konservative Zeitgenossen (der Autor zählt auch dazu) warten deshalb immer auch darauf, dass die aktuellste macOS-Version mindestens ein oder zwei Updates erhalten hat, bevor ein Umstieg erfolgt.

**Hinweis**

Wenn Sie aus der Windows-Welt kommen, dann sind Sie es (bisher) in der Regel nicht gewohnt, ständig (also mindestens einmal im Jahr) auf eine neue Betriebssystem-Version zu aktualisieren. Unter macOS ist es mittlerweile quasi Standard, dass einmal pro Jahr ein Major-Update, also eine neue macOS-Version, erscheint. Normalerweise funktioniert das Update auf die neueste Version auch problemlos. Trotzdem sollte man warten, bis mindestens die Version XX.1 oder besser XX.2 erreicht ist.

**Benötigen Sie ein iOS-Gerät?**

Nicht zwingend, aber es erleichtert die Entwicklung von Apps doch sehr. Außerdem kann man nur auf einem »echten« Gerät bestimmte Funktionen (z.B. den Beschleunigungssensor) auch aus dem Code heraus ansprechen. Ob Sie sich für ein iPhone, ein iPad oder gar einen iPod touch entscheiden, das bleibt Ihrem Geschmack überlassen. Möchten Sie in Ihrer App auch auf Telefoniedienste zugreifen, so muss das Testgerät natürlich ein iPhone sein. Achten Sie auch auf darauf, wie lange es noch Updates für das jeweilige Gerät gibt. Nichts ist ärgerlicher, als viel Geld für ein Gerät auszugeben, um dann festzustellen, dass das Gerät vom Hersteller nicht mehr mit neuen Versionen des Betriebssystems versorgt wird.

**1.4 Xcode und das iOS-SDK**

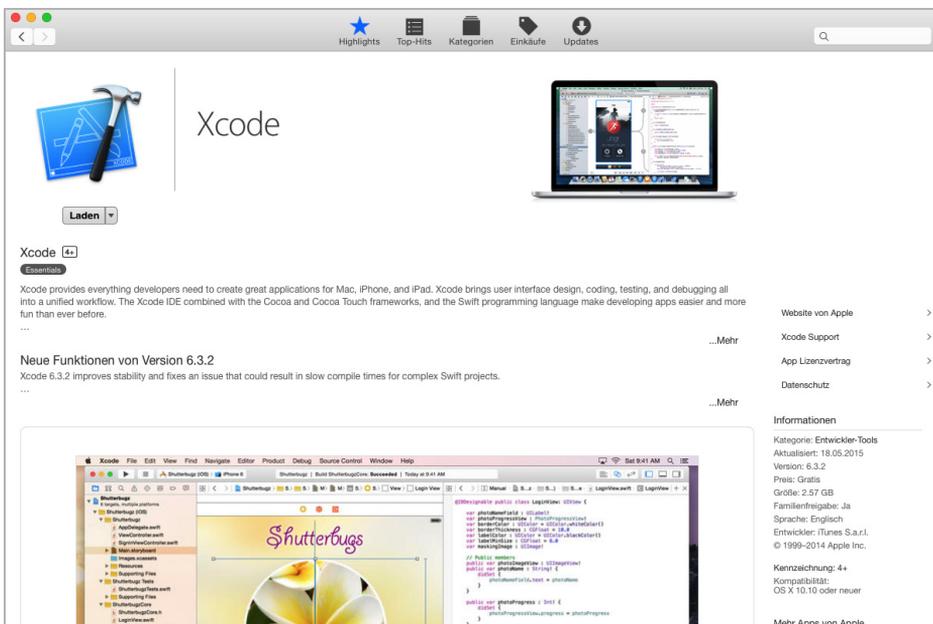
Apple hat eine Entwicklungsumgebung, die für die Entwicklung von Anwendungen sowohl für iOS als auch für macOS verwendet wird. Diese IDE trägt den Namen Xcode. Es gibt sie seit vielen Jahren (genauer gesagt seit 2003), und mittlerweile ist Xcode relativ umfangreich. Xcode unterstützt nicht nur eine Programmiersprache, denn neben Swift können Sie damit auch Programme in Objective-C, C++ und C schreiben. Im Gegensatz zu den bekannten Java-IDEs (Eclipse/Net Beans) gibt es Xcode nur für den Mac bzw. macOS.

Xcode besteht aus mehreren Komponenten. Zum einen ist da natürlich der Quellcodeeditor, in dem Sie den Code schreiben. Wenn eine grafische Oberfläche für eine Anwendung erstellt werden muss, dann kann man dafür den grafischen Designer von Xcode verwenden. Er trägt den Namen *Interface Builder*. Mit dem Interface Builder lassen sich Oberflächen von Anwendungen via Drag & Drop zusammenstellen. Das bedeutet: Ähnlich wie in Microsofts Visual Studio gibt es einen Bereich mit vorgefertigten Controls, die auf einem Formular (View) abgelegt werden können.

### 1.4.1 Download und Installation von Xcode

Neben Xcode benötigen Sie für die Entwicklung von Apps für iOS noch das entsprechende SDK. Es handelt sich dabei aber nicht um einen separaten Download neben Xcode, sondern Sie laden einfach die neueste Xcode-Version aus dem App Store herunter. Diese Version enthält dann alle benötigten Komponenten.

Es gibt mehrere Wege, um Xcode auf Ihren Mac herunterzuladen. Der einfachste Weg soll Ihnen an dieser Stelle vorgestellt werden. Er führt über den App Store von macOS zum Ziel. Öffnen Sie den App Store, und geben Sie im Suchfeld einfach »Xcode« ein. Nach Aktivierung der Suche sollte gleich der erste Treffer in den Suchergebnissen Xcode sein.



**Abb. 1-1** Download und Installation von Xcode über den App Store

Wenn Sie den Xcode-Link auswählen, kommen Sie auf die Xcode-Homepage im App Store. Zum Download und zur Installation müssen Sie einfach den *Laden*-Button einmal anklicken und anschließend die grüne Schaltfläche mit der Beschriftung *App installieren* betätigen. Der Rest läuft vollautomatisch ab. Das heißt, nach dem Download wird Xcode auch gleich automatisch installiert. Informationen zur Größe des Downloads und zur aktuellen Versionsnummer finden Sie ebenfalls auf der Xcode-Homepage. Nach dem Download und der Installation können Sie Xcode über das Launchpad starten. Dort wurde ein entsprechendes Icon angelegt. Alternativ ist natürlich der Start aus dem Verzeichnis *Programme* möglich. Alle notwendigen Dateien von Xcode befinden sich im Xcode-Package. Das bedeutet: Wenn Sie Xcode wieder loswerden möchten, müssen Sie

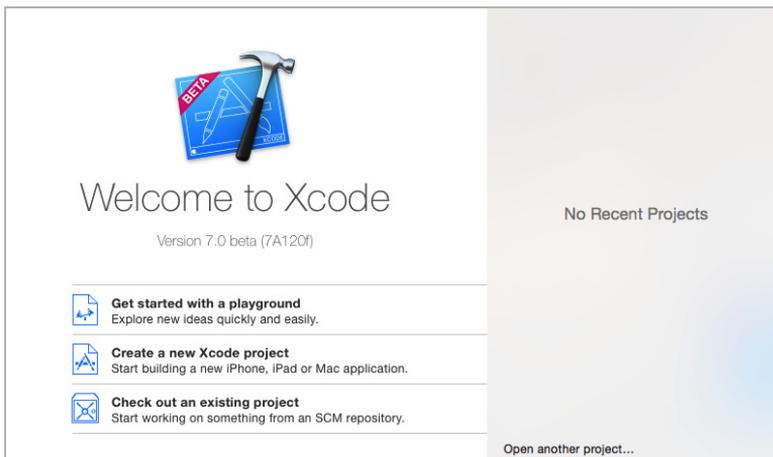
nur das entsprechende Paket aus dem *Programme*-Verzeichnis auf den Papierkorb ziehen.

### Hinweis

In Abschnitt 1.7 wird Ihnen noch eine alternative Möglichkeit zur Installation von Xcode vorgestellt.

## 1.4.2 Xcode im Detail

Nachdem Start von Xcode gibt es zwei Möglichkeiten, wie sich Xcode meldet: Entweder zeigt Xcode den Willkommens-Dialog an (siehe Abb. 1–2) oder es öffnet automatisch das bzw. die zuletzt geöffneten Projekte. Direkt nach der Installation wird der Willkommens-Dialog angezeigt.

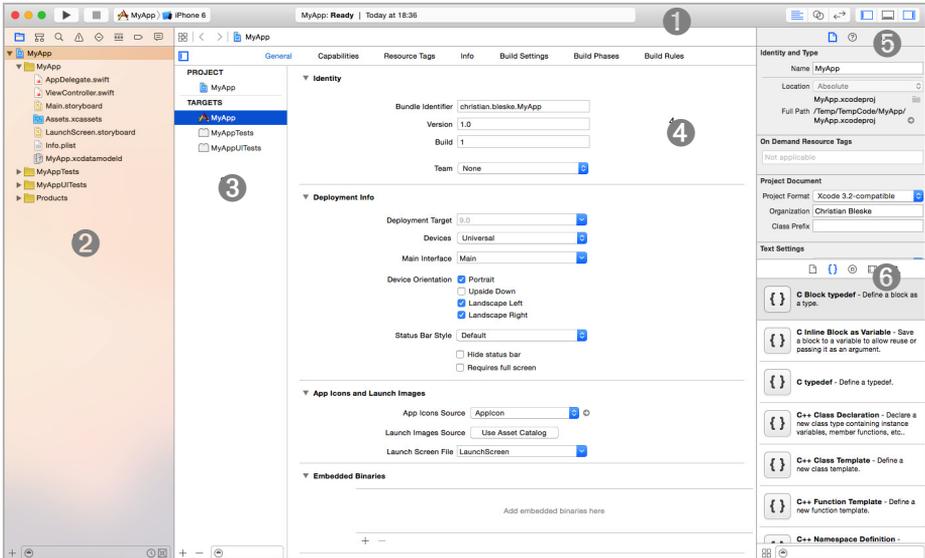


**Abb. 1–2** Der Willkommens-Dialog von Xcode

Der Willkommens-Dialog von Xcode bietet unterschiedliche Optionen. Im linken Bereich werden unter anderem folgende Punkte angeboten:

- Neues Playground-Projekt
- Neues Xcode-Projekt (App)
- Ein Projekt aus einem angeschlossenen Versionskontrollsystem auschecken

Wenn man bereits ein paar Projekte mit Xcode erstellt bzw. bearbeitet hat, dann werden zusätzlich im rechten Bereich die bisher geöffneten Projekte angezeigt. Wird ein neues Projekt angelegt, so zeigt Xcode das Projekt anschließend in der Übersicht an. In dieser Ansicht sind nun mehrere Bereiche von Xcode gut sichtbar (siehe Abb. 1–3).



**Abb. 1-3** Die Bereiche von Xcode mit einem geöffneten Projekt

Im oberen Bereich von Xcode (1 in Abb. 1–3) befindet sich die Toolbar von Xcode. In der Toolbar wird mittig das gerade geöffnete Projekt angezeigt. Direkt unterhalb des Projektnamens befindet sich ein Textfeld, dem Sie den aktuellen Status des Projekts entnehmen können. Wenn Sie beispielsweise ein Projekt starten, so zeigt hier ein Fortschrittsbalken (blau), wie lange es noch dauert, bis das Projekt übersetzt und gestartet wird.

Im linken Bereich der Toolbar befindet sich die *Build and Run*-Schaltfläche, mit der das Projekt (im Simulator oder auf einem angeschlossenen iOS-Gerät) gestartet wird. Direkt danach kommt die Schaltfläche zum Beenden eines laufenden Projekts. Dann folgen der Projektname und das Gerät, auf dem die App gestartet wird. Auch wenn es nicht offensichtlich ist: Es handelt sich hierbei um ein Auswahlfeld. Aufgelistet finden Sie hier die möglichen (simulierten) Geräte sowie ein eventuell angeschlossenes iOS-Gerät.

Im rechten Bereich der Toolbar befinden sich sechs Schaltflächen. Die erste (von links nach rechts) aktiviert den *Standard Editor*. Darin werden die Projektdetails angezeigt, z. B. die Versionsnummer oder Informationen zum Deployment (Zielversion von iOS oder Zielgerät).

Unterhalb der Toolbar befindet sich im linken Bereich (2 in Abb. 1–3) der *Project Navigator*. Diesem können Sie die aktuelle Struktur des Projekts entnehmen. Es handelt sich hierbei allerdings nicht um eine reine Ansicht auf Dateiebene, sondern um eine logische Ansicht des Projekts. Änderungen, die an dieser Stelle vorgenommen werden können, müssen sich aber nicht auf die physikalische Struktur des Projekts auswirken. Abhängig vom im *Project Navigator* ausgewählten Element des Projekts ändert sich auch die Ansicht in Xcode. Wählt



**Abb. 1–4** Auswahl des iOS-Geräts zur Ausführung des Projekts

man an dieser Stelle beispielsweise eine Codedatei aus (ihr Name endet auf *.swift*), dann wird zentral das Codefenster von Xcode geöffnet.

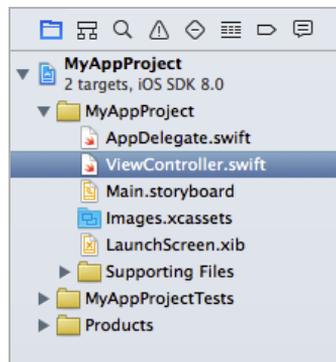
Von links folgt dann die *Project and Target*-Übersicht (siehe ③ in Abb. 1–3). Darin können Sie zwischen den Einstellungen für das Projekt selbst (*Project*) und der Konfiguration zur Erstellung der Anwendung (*Target*) umschalten. Im Target-Bereich befindet sich auch die Konfiguration der Anwendungstests.

In der Mitte (④ in Abb. 1–3) sehen Sie das zentrale Fenster, in dem beispielsweise die Projektkonfiguration, der Quellcodeeditor oder auch der Interface Builder angezeigt werden.

Am rechten Rand von Xcode (⑤ und ⑥ in Abb. 1–3) befinden sich dann noch zwei Bereiche mit unterschiedlichen Funktionen. Im Bereich von ⑤ werden die sogenannten Inspektoren angezeigt. Je nach Auswahl im Project Navigator oder Interface Builder werden in einem Inspektor unterschiedliche Informationen angezeigt bzw. können dort bearbeitet werden. Der letzte Bereich (⑥ in Abb. 1–3) enthält ebenfalls unterschiedliche Werkzeuge. Hier befindet sich unter anderem die *Object Library* (enthält die Controls), die *File Template Library* (Vorlagen für Dateien) oder auch die *Code Snippet Library* (Codevorlagen).

### 1.4.3 Projektübersicht

Im letzten Abschnitt wurde unter anderem der *Project Navigator* bereits kurz vorgestellt. Dieser erlaubt die Navigation im Projekt. Im *Project Navigator* ausgewählte Elemente haben Einfluss auf das, was zentral in Xcode angezeigt wird. Wählt man in diesem Bereich z.B. eine Quellcodedatei aus, so wird automatisch der Codeeditor von Xcode geöffnet. Aber auch der Interface Builder lässt sich hier aktivieren. Hierzu müssen Sie im *Project Navigator* nur eine passende Datei auswählen, die den Interface Builder zur Darstellung verwendet. Sie erkennen diese Dateien an der Endung *\*.storyboard* und *\*.xib*.



**Abb. 1-5** Der *Project Navigator* von Xcode

Neben dem *Project Navigator* gibt es aber auch noch andere Funktionen, die in diesem Teil der IDE angezeigt werden. In der Leiste direkt oberhalb des *Project Navigator* gibt es dafür noch weitere Schaltflächen. Das erste Symbol (Ordner) aktiviert den *Project Navigator*. Der zweite Button aktiviert den *Symbol Navigator*, in dem die Klassen eines Projekts angezeigt werden. Es folgt der *Find Navigator*, über den sich im Projekt suchen lässt. Das nächste Icon aktiviert den *Issue Navigator*, der Informationen zu Problemen im Projekt enthält. Danach folgen der *Test Navigator* (Übersicht der Tests im Projekt), der *Debug Navigator*, der *Breakpoint Navigator* und der *Report Navigator*.

### 1.4.4 Quellcodefenster

Wählt man im *Project Navigator* eine Quellcodedatei (diese Dateien enden auf *\*.swift*) aus, so wird automatisch der Quellcodeeditor geöffnet.

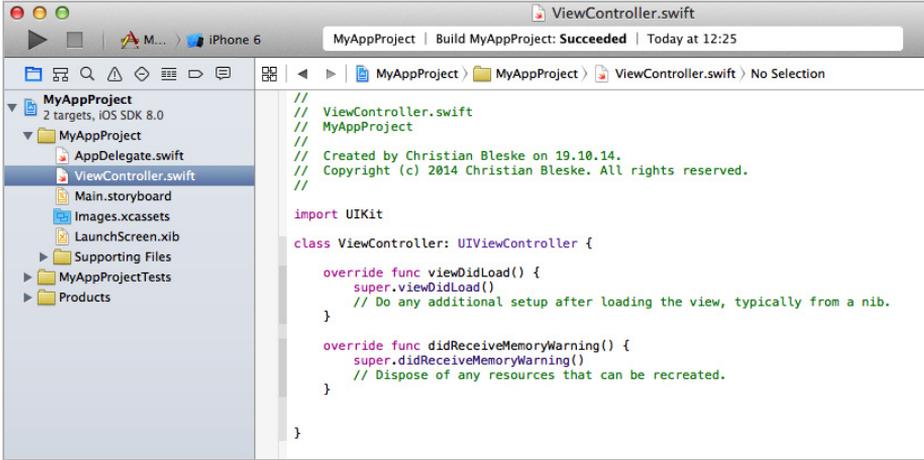


Abb. 1-6 Der Codeeditor von Xcode

Im Editor stehen alle bekannten Funktionen (Bearbeiten, Kopieren, Einfügen) zur Bearbeitung von Texten zur Verfügung. Die gerade geöffnete Quellcodedatei kann nicht nur dem *Project Navigator* entnommen werden, sie wird auch direkt oberhalb des Editors angezeigt. Eine nützliche Funktion ergibt sich in Verbindung mit der *Code Snippet Library*.

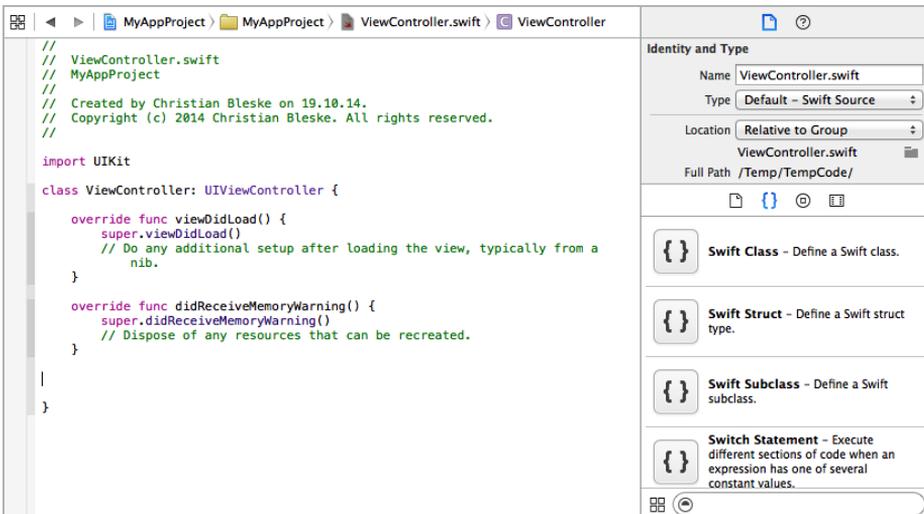


Abb. 1-7 Code Snippet Library und Codeeditor

Aus der *Code Snippet Library* können Codeblöcke via Drag & Drop im Editor abgelegt werden. An der abgelegten Stelle wird dann der jeweilige Codeblock (z.B. eine Klasse) automatisch eingefügt.

### 1.4.5 Interface Builder

Der Interface Builder ist das Werkzeug in Xcode, um Oberflächen für Anwendungen zu entwickeln. Ähnlich wie in anderen Entwicklungsumgebungen, so wird auch in Xcode eine Anwendung Formular für Formular entwickelt. Controls – beispielsweise Textfelder oder Schaltflächen – werden aus der *Object Library* (siehe auch Abschnitt 1.4.6) via Drag & Drop in das gerade im Interface Builder geöffnete View eingefügt.

In Abbildung 1–8 ist links neben dem *Interface Builder* das sogenannte *Document Outline* zu sehen. Nachdem das Label-Control in Abbildung 1–8 in das View eingefügt wurde, wird neben der Ansicht im Interface Builder auch das *Document Outline* aktualisiert. Darin werden die im View eingefügten Elemente innerhalb einer hierarchischen Ansicht angezeigt. Das *Document Outline* kann so auch zur Selektion von Elementen in einem View verwendet werden, wenn die direkte Auswahl im Interface Builder vielleicht nicht möglich ist – beispielsweise weil ein anderes Control das auszuwählende verdeckt. Um Eigenschaften eines Views oder eines Controls zu bearbeiten, werden die Inspektoren von Xcode verwendet.

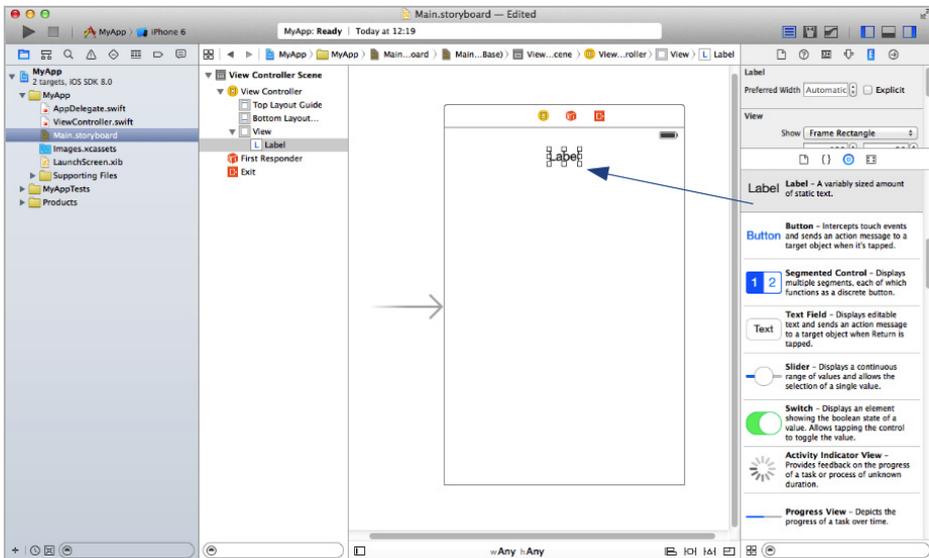
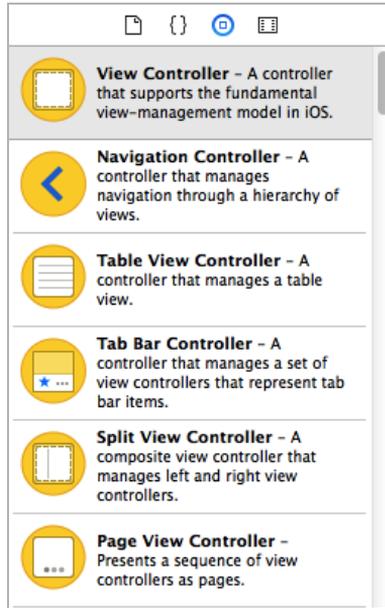


Abb. 1–8 Der Interface Builder von Xcode

### 1.4.6 Object Library & Co.

Im letzten Abschnitt wurde bereits eine Funktion angesprochen, die im rechten Bereich der IDE angesiedelt ist. Die *Object Library* enthält unter anderem Elemente (Controls), die innerhalb eines Views eingefügt werden können. Das ist aber längst nicht alles.



**Abb. 1-9** Die Object Library von Xcode

In Abbildung 1-9 ist gut zu sehen, dass beispielsweise auch ein View Controller innerhalb der *Object Library* vorhanden ist. Wenn man eine App erstellt, die über mehrere Views verfügt, dann können in einer *\*.storyboard*-Datei zusätzliche Views eingefügt werden. Es ist also nicht erforderlich, für jedes neue View eine eigene Datei anzulegen.

Ein weiteres Fenster in diesem Bereich ist die bereits vorgestellte *Code Snippet Library*, die kurze Codeschnipsel zum Einfügen in den Xcode-Texteditor enthält. Neben diesen beiden Bibliotheken gibt es auch noch die *File Template Library*. In ihr sind Vorlagen für Dateien enthalten, die in ein Projekt eingefügt werden können. Auch das geschieht via Drag & Drop. Hierbei ist aber nicht der Interface Builder das Ziel der Drop-Aktion, sondern der *Project Navigator*. Um beispielsweise eine neue Swift-Code-Datei anzulegen, müssen Sie nur die entsprechende Vorlage (*Swift-File*) aus der *File Template Library* in das aktuelle Projekt einfügen, das gerade im *Project Navigator* angezeigt wird.