

dirk LOUIS
peter MÜLLER



2. Auflage

Android

DER SCHNELLE UND
EINFACHE EINSTIEG IN DIE
PROGRAMMIERUNG UND
ENTWICKLUNGSUMGEBUNG



EXTRA: E-Book inside



Im Internet: Beispiele, Tutorials, JRE
und Android-Bundle



Inklusive: Java-Tutorium für
Ein- und Umsteiger

HANSER

Louis/Müller

Android

Bleiben Sie auf dem Laufenden!



Unser **Computerbuch-Newsletter** informiert Sie monatlich über neue Bücher und Termine. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter



www.hanser-fachbuch.de/newsletter



Hanser Update ist der IT-Blog des Hanser Verlags mit Beiträgen und Praxistipps von unseren Autoren rund um die Themen Online Marketing, Webentwicklung, Programmierung, Softwareentwicklung sowie IT- und Projektmanagement. Lesen Sie mit und abonnieren Sie unsere News unter



www.hanser-fachbuch.de/update



Dirk Louis

Peter Müller

Android

Der schnelle und einfache Einstieg
in die Programmierung
und Entwicklungsumgebung

2. Auflage

HANSER

Die Autoren:

Dirk Louis, Saarbrücken, autoren@carpelibrum.de

Peter Müller, Saarbrücken, leserfragen@gmx.de

Alle in diesem Buch enthaltenen Informationen, Verfahren und Darstellungen wurden nach bestem Wissen zusammengestellt und mit Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Buch enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autoren und Verlag übernehmen infolgedessen keine juristische Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Informationen – oder Teilen davon – entsteht.

Ebenso übernehmen Autoren und Verlag keine Gewähr dafür, dass beschriebene Verfahren usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Buch berechtigt deshalb auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.



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.

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren) – auch nicht für Zwecke der Unterrichtsgestaltung – reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2016 Carl Hanser Verlag München, www.hanser-fachbuch.de

Lektorat: Brigitte Bauer-Schiewek

Copy editing: Sandra Gottmann, Münster-Nienberge

Herstellung: Irene Weilhart

Umschlagdesign: Marc Müller-Bremer, www.rebranding.de, München

Umschlagrealisation: Stephan Rönigk

Gesamtherstellung: Kösel, Krugzell

Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702

Printed in Germany

Print-ISBN: 978-3-446-44598-7

E-Book-ISBN: 978-3-446-45112-4

Inhalt

Vorwort	XV
Teil I: Einführung	1
1 Der Rechner wird vorbereitet	3
1.1 Die nötigen Hilfsmittel	3
1.2 Installation des JDK	4
1.3 Installation von Android Studio	5
1.3.1 Erster Start	7
1.4 Der Android-SDK-Manager	9
1.4.1 Die Android-Versionen	9
1.4.2 APIs/SDKs und anderes nachinstallieren	10
1.4.3 Dokumentation und API-Referenz	12
1.5 Wo Sie weitere Hilfe finden	14
1.6 Nächste Schritte	15
1.7 Frage und Antworten	15
1.8 Übungen	15
2 Auf die Plätze, fertig ... App!	17
2.1 Die Ruhe vor dem Sturm	17
2.2 Das Projekt	18
2.3 Das vorgegebene Codegerüst	26
2.3.1 Die package-Anweisung	28
2.3.2 Die import-Anweisungen	28
2.3.3 Die Klassendefinition	29
2.4 Layout und Ressourcen	31
2.4.1 XML-Layouts	31
2.4.2 Ressourcen	34
2.5 Die App bauen (Build)	37

2.6	Die App im Emulator testen	38
2.6.1	AVD für Emulator anlegen	38
2.6.2	Die App testen	41
2.7	Die App auf Smartphone oder Tablet testen	45
2.8	Nächste Schritte	47
2.9	Fragen und Antworten	47
2.10	Übungen	48
3	Was wann wofür	49
3.1	Was ist zu tun? – Die drei Pfeiler der App-Erstellung	49
3.2	Wer hilft uns? – Bausteine und Klassen	50
3.2.1	Bausteine für den App-Aufbau	50
3.2.2	Klassen zur Adressierung spezieller Aufgaben	54
3.3	Wo wird was gespeichert? – Dateitypen, die Sie kennen sollten	55
3.3.1	Quelldateien	56
3.3.2	Die Datei R.java	56
3.3.3	Assets	57
3.3.4	Die Ressourcendateien	57
3.3.5	Die Manifestdatei <i>AndroidManifest.xml</i>	58
3.3.6	Die APK-Datei	59
3.4	Frage und Antworten	59
3.5	Übung	60
	Teil II: Grundlagen	61
4	Code	63
4.1	Der Editor	63
4.1.1	Syntaxhervorhebung	64
4.1.2	Code Folding (Code-Gliederung)	65
4.1.3	Code Completion (Code-Vervollständigung)	67
4.1.4	Syntaxfehler beheben	69
4.1.5	Informationen über Klassen und Methoden	71
4.1.6	Klammerpaare identifizieren	74
4.1.7	Zeilennummern einblenden	74
4.1.8	Code generieren	75
4.1.9	Refactoring (Code umstrukturieren)	78
4.1.10	Dateiverlauf (Local History)	81
4.2	Neue Klassen anlegen	81
4.3	Fragen und Antworten	84
4.4	Übungen	84

5	Benutzeroberfläche und Layout	85
5.1	Einführung	85
5.2	Der Layout-Designer	86
5.2.1	Die Text-Ansicht (XML-Code)	88
5.2.2	Die Design-Ansicht	92
5.3	Layouts (ViewGroups)	95
5.3.1	Die allgemeinen Layoutparameter	96
5.3.2	ViewGroups	99
5.3.3	Hintergrundfarbe (oder -bild)	109
5.3.4	Der Hierarchy Viewer	112
5.4	UI-Elemente	114
5.5	Richtlinien für das Design von Benutzeroberflächen	118
5.6	Praxisbeispiel: eine Quiz-Oberfläche	121
5.7	Hoch- und Querformat	126
5.8	Das App-Symbol	127
5.9	Views im Code verwenden	128
5.9.1	Layouts laden	128
5.9.2	Zugriff auf UI-Elemente	129
5.10	Fragen und Antworten	131
5.11	Übung	131
6	Ressourcen	133
6.1	Der grundlegende Umgang	133
6.1.1	Ressourcen anlegen	134
6.1.2	Ressourcen verwenden	136
6.1.3	Ressourcen aus dem Projekt entfernen	140
6.2	Welche Arten von Ressourcen gibt es?	140
6.2.1	Größenangaben	140
6.2.2	Farben	141
6.2.3	Strings	142
6.2.4	Strings in mehreren Sprachen (Lokalisierung)	144
6.2.5	Bilder	145
6.2.6	Layouts	146
6.2.7	Menüs	147
6.2.8	Roh- und Multimediadaten	147
6.2.9	Stile	148
6.3	Alternative Ressourcen vorsehen	152
6.3.1	Das Grundprinzip	152
6.3.2	Wie stellt man konfigurationsspezifische Ressourcen bereit?	153
6.4	Fragen und Antworten	155
6.5	Übungen	156

7	Mit dem Anwender interagieren	157
7.1	Das Grundprinzip	157
7.1.1	Auf ein Ereignis reagieren	158
7.1.2	Welche Ereignisse gibt es?	161
7.1.3	Hintergrund der Ereignisverarbeitung	162
7.2	Vereinfachte Ereignisbehandlung	164
7.2.1	Ereignisbehandlung mit anonymen Listener-Klassen	165
7.2.2	Ereignisbehandlung mit anonymen Listener-Objekten	166
7.2.3	Ereignisbehandlung mithilfe der Activity-Klasse	166
7.3	Eine Behandlungsmethode für mehrere Views	167
7.4	Auf Tipp- und Wischereignisse reagieren	168
7.4.1	Tippereignisse	168
7.4.2	Wischereignisse	170
7.5	Multi-Touch und Gesten erkennen	172
7.5.1	Multi-Touch	172
7.5.2	Gestenerkennung	174
7.6	Frage und Antworten	176
7.7	Übung	176
8	App-Grundlagen und Lebenszyklus	177
8.1	Die Android-Architektur	177
8.2	Der App-Lebenszyklus	179
8.3	Der Activity-Lebenszyklus	181
8.4	Lebenszyklusdemo	183
8.5	Fragen und Antworten	187
8.6	Übung	187
Teil III: Weiterführende Themen		189
9	In Views zeichnen	191
9.1	Das Grundprinzip	191
9.1.1	Die Leinwand	191
9.1.2	Das Atelier	191
9.1.3	Die Zeichenmethoden und -werkzeuge	192
9.1.4	Wie alles zusammenwirkt	192
9.2	Grafikprimitive zeichnen	196
9.3	Bilder laden	200
9.4	In Bilder hineinzeichnen	201
9.5	Bilder bewegen	203
9.6	Verbesserungen	208
9.7	Fragen und Antworten	209
9.8	Übung	209

10	Menüs, Fragmente und Dialoge	211
10.1	Menüs	211
10.1.1	Menüverwirrungen	212
10.1.2	Menüressourcen	213
10.1.3	Menüeinträge in der ActionBar (AppBar)	215
10.1.4	Das Optionen-Menü	216
10.1.5	Das Kontextmenü	217
10.1.6	Popup-Menü	219
10.1.7	Untermenüs	220
10.1.8	Auf die Auswahl eines Menüeintrags reagieren	221
10.2	Fragmente	223
10.2.1	Was ist ein Fragment?	223
10.2.2	Ein Fragment erzeugen	224
10.2.3	Fragment zur Activity hinzufügen	225
10.2.4	Ein Fragmentbeispiel	226
10.2.5	Definition der Fragment-Klassen	229
10.2.6	Die Activity	231
10.3	Dialoge	233
10.3.1	Dialoge erzeugen	234
10.3.2	Dialoge anzeigen	235
10.3.3	Standarddialoge mit AlertDialog	235
10.3.4	Dialoge für Datums- und Zeitauswahl	237
10.3.5	Der Fortschrittsdialog	240
10.3.6	Eigene Dialoge definieren	242
10.4	Benachrichtigungen mit Toasts	244
10.4.1	Toasts im Hintergrund-Thread	244
10.5	Fragen und Antworten	245
10.6	Übungen	246
11	Mehrseitige Apps	247
11.1	Intents	247
11.1.1	Was sind Intents?	248
11.1.2	Explizite und implizite Intents	249
11.1.3	Intent-Filter	249
11.2	Activities starten mit Intents	250
11.2.1	Intent-Objekte erzeugen	251
11.3	Intents empfangen	253
11.4	Ein Demo-Beispiel	253
11.5	Ergebnisse zurücksenden	256
11.6	Fragen und Antworten	257
11.7	Übung	257

12	Daten speichern	259
12.1	Preferences	259
12.2	Dateizugriffe	260
12.2.1	Zugriff auf internen Speicher	261
12.2.2	Externer Speicher (SD-Karte)	264
12.3	Die Reaktions-App	267
12.4	Fragen und Antworten	272
12.5	Übungen	272
13	Quiz-Apps	273
13.1	Aufbau und Benutzeroberfläche	273
13.2	Die Activity (QuizActivity.java)	274
13.3	Die Fragen (Frage.java)	276
13.4	Die Spielsteuerung (SpielLogik.java)	277
13.5	Verbesserungen	279
13.6	Frage und Antwort	280
13.7	Übung	280
14	Multimedia	281
14.1	Audioressourcen	281
14.2	Soundeffekte mit SoundPool	282
14.3	Das Universalgenie: MediaPlayer	284
14.3.1	Audioressourcen abspielen	284
14.3.2	Audiodateien vom Dateisystem abspielen	285
14.3.3	Audiodateien aus dem Internet abspielen	285
14.3.4	Auf das Abspielende reagieren	286
14.3.5	MediaPlayer-Objekte wiederverwenden	287
14.3.6	Ressourcen freigeben	289
14.3.7	Audiodateien wiederholt abspielen	290
14.4	Piepen und andere Töne	290
14.5	Bilddateien anzeigen	292
14.6	Videos abspielen	293
14.7	Fotos und Videos aufnehmen	295
14.8	Fragen und Antworten	298
14.9	Übungen	298
15	Sensoren	299
15.1	Zugriff	299
15.1.1	Was Sie benötigen	300
15.1.2	Welche Sensoren sind verfügbar?	300
15.1.3	Anmeldung beim Sensor	302

15.2	Sensordaten auslesen	303
15.2.1	Beschleunigungswerte ermitteln	305
15.2.2	Lagedaten ermitteln	309
15.3	Fragen und Antworten	312
15.4	Übung	313
16	Einsatz der Datenbank SQLite	315
16.1	Was ist eine relationale Datenbank?	315
16.2	Datenbank anlegen/öffnen	316
16.2.1	onCreate()	317
16.2.2	onUpgrade()	319
16.2.3	close()	319
16.2.4	Datenbanken als Ressourcen mitgeben	319
16.3	Datenzugriffe	320
16.4	Datenbankinhalte mit ListView anzeigen	325
16.5	Fragen und Antworten	329
16.6	Übung	330
17	Geolokation	331
17.1	Zugriff	331
17.1.1	Verfügbarkeit feststellen	331
17.1.2	Daten empfangen	332
17.1.3	Empfänger abmelden	333
17.2	Geokoordinaten	334
17.2.1	Sexagesimale und dezimale Darstellung	334
17.2.2	Das Location-Objekt	334
17.3	Eine GPS-Tracker-App	336
17.4	Fragen und Antworten	340
17.5	Übung	340
18	Brettspiel-Apps (TicTacToe)	341
18.1	Aufbau und Benutzeroberfläche	341
18.2	Die Start-Activity (MainActivity)	343
18.3	Spielfeld und Logik (TicTacToeView)	344
18.3.1	Vorbereitungen	344
18.3.2	Spielfeld zeichnen	345
18.3.3	Spielerzug durchführen	347
18.3.4	Computerzug mit AsyncTask durchführen	348
18.4	Verbesserungen	351
18.5	Frage und Antwort	351
18.6	Übung	352

19	Tipps und Tricks	353
19.1	Das Smartphone vibrieren lassen	353
19.2	UI-Code periodisch ausführen lassen	354
19.3	Bildergalerien mit GridView und BaseAdapter	357
19.3.1	Die Bildressourcen	358
19.3.2	Die Adapter-Klasse	358
19.3.3	Die GridView	361
19.3.4	Angeklickte Bilder als Vollbild anzeigen	362
19.4	Spinner verwenden (Listenfelder)	364
19.4.1	Den Spinner mit Daten füllen	365
19.4.2	Ereignisbehandlung	366
19.5	Mehrsprachige Apps	367
19.6	Schlussbemerkung	370
	Teil IV: Anhänge	371
	Anhang A: Apps veröffentlichen oder weitergeben	373
A.1	Die App vorbereiten	373
A.2	Digitales Signieren	375
A.3	Die App exportieren und signieren	376
A.4	Bei Google Play registrieren	378
A.4.1	Steuerliche Aspekte bei App-Verkauf	379
A.5	App hochladen	380
A.6	Weitergabe an Bekannte	380
	Anhang B: Android Studio	383
B.1	Android-Projekt anlegen	383
B.2	Projekt bauen (Build)	383
B.3	Projekte löschen	385
B.4	Eclipse-ADT-Projekt importieren	385
B.5	Run-Konfigurationen anpassen	385
B.6	Fenster zurücksetzen	386
B.7	Apps exportieren	386
B.8	Kleines Android Studio-Wörterbuch	386
	Anhang C: Emulator, ADM & Debugger	389
C.1	Der Emulator	389
C.1.1	AVD-Dateien	390
C.1.2	Emulator starten	393
C.1.3	Die Emulator-Bedienung	395
C.1.4	Apps installieren und deinstallieren	396

C.2	Android Device Monitor (ADM)	396
C.3	Der Debugger	401
C.3.1	Debug-Lauf starten	401
C.3.2	Debug-Möglichkeiten	402
C.4	Debugging-Beispiel	404
Anhang D: Das Material zum Buch		409
Anhang E: Lösungen		411
Anhang F: Glossar		427
Index		437

Vorwort

Willkommen in der Android-Welt! Seitdem sich der Touchscreen als Standardoberfläche von Mobilfunktelefonen etabliert hat und vor Kurzem noch völlig unbekannte Features wie GPS-Empfänger und Lagesensor zur Standardausstattung gehören, gibt es kein Halten mehr: Jede Woche erscheinen neue Android-basierte Geräte und die Zahl der verfügbaren Apps im Android Market explodiert geradezu.

Wenn auch Sie dazugehören wollen, wenn Sie nicht bloß Anwender sein möchten, sondern daran interessiert sind, eigene Ideen in Apps umzusetzen – sei es zum Spaß oder auch vielleicht als Einstieg in eine Existenz als selbstständiger Software-Entwickler –, dann kann Ihnen dieses Buch einen guten Einstieg (und ein bisschen mehr) in die Welt der App-Programmierung für Android-Systeme bieten.

Vorkenntnisse und Anforderungen

Wir wollen nichts beschönigen. Die Anforderungen an Android-Programmierer sind hoch. Doch mithilfe dieses Buchs und ein wenig Ausdauer und Mitdenken sollten Sie die größten Hürden meistern können.

Sehen wir uns dazu einmal an, welche Fähigkeiten ein Android-Programmierer besitzen muss und inwieweit Ihnen dieses Buch helfen kann, diese Fähigkeiten zu entwickeln.

- *Gute Kenntnisse der Programmiersprache Java*
Sie erfüllen diesen Punkt nicht? Kein Grund zur Panik, aber lesen Sie unbedingt den nachfolgenden Abschnitt zum „idealen Leser“.
- *Umgang mit der integrierten Entwicklungsumgebung (IDE) Android Studio.*
Alles, was Sie zum Umgang mit Android Studio im Allgemeinen wie auch im Hinblick auf die Erstellung von Android-Apps wissen müssen, lernen Sie in diesem Buch. Zusätzlich finden Sie am Ende des Buchs einen eigenen Anhang zu Android Studio, wo die wichtigsten Aufgaben noch einmal zusammengefasst sind (inklusive eines kleinen Wörterbuchs, das Lesern, die im Englischen nicht so versiert sind, die Eingewöhnung in die durchweg englische Benutzeroberfläche erleichtern soll).
- *Einsatz verschiedener Hilfsprogramme wie HierarchyViewer, Debugger und Emulator.*
Insbesondere der Emulator ist für die Entwicklung von Apps unerlässlich, da Sie mit seiner Hilfe unterschiedlich ausgestattete Android-Geräte simulieren („emulieren“) können, ohne sie tatsächlich als echtes Gerät zu besitzen.

Unnötig zu erwähnen, dass wir Ihnen die wichtigsten Hilfsprogramme in diesem Buch vorstellen und Sie in die Arbeit mit ihnen einführen.

- *Wissen um den Aufbau von Apps und Kenntnis der Android-Klassenbibliothek*

Dies ist das eigentliche Thema dieses Buchs. Wie sieht das Grundgerüst einer Android-App aus, worauf muss ich achten und was für tolle Sachen kann man mit der Android-Klassenbibliothek machen? (Kurzantwort: Nichts ist unmöglich!)

Nach dem erfolgreichen Durcharbeiten dieses Buchs werden Sie sicher noch kein Profi-Android-Entwickler sein. Das können und wollen wir Ihnen gar nicht versprechen, denn der Umfang an Material wäre so groß, dass kein Platz mehr für ausführliche Erläuterungen bliebe.

Sie werden aber eine sehr fundierte Grundlage erhalten, in viele fortgeschrittene Bereiche blicken und alles Notwendige lernen, um tolle Apps erstellen und sich selbstständig weiterbilden zu können.

Der ideale Leser, Java-Kenntnisse und das Java-Tutorium

Da es den idealen Leser im Grunde gar nicht gibt, sollten wir uns lieber fragen, welche Lesergruppen in welchem Umfang von dem vorliegenden Buch profitieren können:

Leser mit guten Java-Kenntnissen, die sicher objektorientiert programmieren können und bereits Erfahrung mit Konzepten wie Überschreibung, Interface-Implementierung, Ereignis-Listener und Threads haben, bilden eine der drei Hauptzielgruppen, für die dieses Buch geschrieben wurde. Sollten Sie zu dieser Gruppe zählen, legen Sie einfach los.

Leser mit grundlegenden Java-Kenntnissen bilden die zweite Hauptzielgruppe und sollten mit diesem Buch ebenfalls gut und schnell vorankommen. Sollten Sie zu dieser Gruppe gehören, achten Sie auf die im Buchtext eingestreuten Hinweise zu den Exkursen des Java-Tutoriums unter <http://files.hanser.de/fachbuch/PDFs.zip>. Mithilfe dieser Exkurse können Sie etwaige Wissenslücken zur Java-Programmierung schließen.

Umsteiger von anderen Programmiersprachen bilden die dritte Hauptzielgruppe. Doch Obacht! Es liegt viel Arbeit vor Ihnen, denn Sie müssen sich parallel auch noch mithilfe des Java-Tutoriums in Java einarbeiten. Sofern Sie allerdings bereits über gute Programmierkenntnisse in einer anderen objektorientierten Sprache (wie z. B. C++ oder C#) verfügen, dürfte dies für Sie keine große Schwierigkeit sein. Sie können das Tutorium vorab oder parallel zu diesem Buch lesen (die ersten Kapitel enthalten zu diesem Zweck Hinweise, wann Sie welche Teile des Tutoriums lesen sollten).

Bleibt die Gruppe der Leser, die über keine oder nur wenig Programmiererfahrung verfügen. Angehörigen dieser Gruppe können wir eigentlich nur empfehlen, sich zuerst einmal in die Java-Programmierung einzuarbeiten (beispielsweise mit unserem Java-Titel). Sie können es aber natürlich auch mit dem Java-Tutorium versuchen. Es geht zwar relativ flott voran, ist aber recht gut verständlich und beinhaltet sogar eine allgemeine Einführung in die grundlegenden Programmierkonzepte.

Aufbau des Buchs

Das Buch ist in drei Teile plus Anhang gegliedert.

- Der erste Teil behandelt die Installation der notwendigen Entwicklerwerkzeuge und die Grundlagen der App-Erstellung.
- Der zweite Teil vertieft die im ersten Teil angesprochenen Grundthemen: Code, Benutzeroberfläche, Arbeiten mit Ressourcen und der App-Lebenszyklus.
- Der dritte Teil behandelt zahlreiche fortgeschrittene Aspekte wie z. B. Grafik, Menüs, Sensoren, Spiele, Datenbanken oder Geolokation. Er unterscheidet sich nicht nur inhaltlich, sondern auch konzeptionell von den beiden vorangehenden Teilen und ist eher im Stile eines Fortgeschrittenenbuchs geschrieben.

Abgerundet wird das Buch mit Anhängen zur Veröffentlichung von Apps, zur Entwicklungsumgebung Android Studio sowie zu weiteren Werkzeugen wie Emulator, ADM und Debugger, einem Glossar und einem ausführlichen Index.

Software, Beispiel-Apps und sonstiges Material zum Buch

Die Android-Entwicklungsumgebung, die Beispielsammlung und die Tutorials stehen für Sie zum Download bereit. Die Download-Links finden Sie in Anhang D: Das Material zum Buch. Bitte beachten Sie, dass es für die Android-Entwicklungsumgebung und den Java-SDK mehrere Download-Links gibt. Wählen Sie einfach den Link, der zu Ihrem Betriebssystem passt.

Die Website zum Buch

Wir haben dieses Buch mit großer Sorgfalt erstellt. Falls Sie auf Probleme oder Fehler stoßen, sollten Sie nicht zögern, uns eine E-Mail unter Angabe von Buchtitel und Auflage zu senden. Schauen Sie auch einmal auf unserer Buch-Website

www.carpelibrum.de

nach. Neben zusätzlichem Material, den Lösungsprojekten zu den Übungen, Aktualisierungen und Errata finden Sie dort auch weitere Bücher zum Thema Programmieren in Java, C++, C# u. a.

Viel Spaß in der Android-Welt wünschen Ihnen

Dirk Louis (autoren@carpelibrum.de)

Peter Müller (leserfragen@gmx.de)



Teil I: Einführung

In diesem Teil geht es darum, dass Sie sich mit dem Aufbau von Android-Programmen und den Entwicklerwerkzeugen vertraut machen, die wir für die Erstellung von Android-Programmen (fortan kurz „Apps“ genannt) benötigen. Doch zunächst müssen wir diese natürlich erst einmal installieren.

1

Der Rechner wird vorbereitet

Bevor Sie mit der App-Programmierung beginnen können, müssen Sie sicherstellen, dass Sie das nötige Arbeitsgerät zur Verfügung haben. Die gute Nachricht ist: Alles, was Sie zum Schreiben eigener Apps benötigen, gibt es kostenlos beim Material zum Buch (siehe Anhang D) oder verteilt im Internet. Die weniger gute Nachricht ist: Sie müssen die nötigen Entwicklungswerkzeuge erst einmal installieren, konfigurieren und – siehe nächstes Kapitel – testen. Aber wie sagt schon ein auf Seneca zurückgehendes Sprichwort:

per aspera ad astra

(„Durch Mühsal zu den Sternen“ oder wie es in Anlehnung an Hesiod heißt: „Vor den Erfolg haben die Götter den Schweiß gestellt.“)

■ 1.1 Die nötigen Hilfsmittel

Um Apps schreiben zu können, benötigen Sie:

■ **Android Studio**

Android Studio ist eine integrierte Entwicklungsumgebung (kurz IDE), die viele spezialisierte Werkzeuge und Hilfsprogramme zur App-Entwicklung in einer gemeinsamen Oberfläche zusammenfasst. Dazu gehören beispielsweise ein Editor zum Aufsetzen der Programmquelltexte, ein Compiler zum Übersetzen der Quelltexte in Programmcode und ein Debugger zur schrittweisen Ausführung einer App zwecks Fehleranalyse, um nur die wichtigsten Helfer zu nennen. Android Studio erleichtert Ihnen als Entwickler die Arbeit und erlaubt zudem eine intensivere Zusammenarbeit der eingesetzten Hilfsprogramme (wie z. B. die Zuordnung von Compiler-Fehlermeldungen zu Quelltextzeilen oder die Anzeige der aktuellen Ausführungsposition beim Debuggen im Editor).

■ **Android SDK**

Das Android SDK¹ enthält alle wichtigen Tools, Klassenbibliotheken und Dokumentationen, die für die Erstellung von Android-Apps benötigt werden.

¹ SDK steht für „Software Development Kit“, grob übersetzt also ein „Bausatz zur Entwicklung von Software“.

Das Android SDK wird normalerweise zusammen mit dem Android Studio installiert. Wir werden es allerdings im Nachhinein noch etwas anpassen.

■ **Java Development Kit (JDK)**

Zum Ausführen von Android Studio und zum Kompilieren von Android-Apps wird ein aktuelles Java SDK (kurz JDK) benötigt. Es umfasst neben diversen Werkzeugen auch die Java-Laufzeitumgebung, die sogenannte JRE (Java Runtime Environment).

Die JRE wird nur für Android Studio und seine Hilfsprogramme benötigt. Die erzeugten Android-Apps brauchen natürlich auch eine Laufzeitumgebung, allerdings ist dies nicht die klassische JRE, sondern eine spezielle Variante, die vom Android-Gerät bereitgestellt wird (je nach Version heißt diese Laufzeitumgebung Dalvik oder ART).

- ein Android-Smartphone zum Testen (optional)
- einen nicht zu alten Rechner mit mindestens 5 Gbyte freiem Festplattenspeicher und mindestens 4 Gbyte Hauptspeicher und einem geeigneten Betriebssystem (unseres Wissens nach eignet sich jedes nicht zu alte Windows-, Mac OS- oder Linux-System). Realistisch betrachtet muss man hier anfügen, dass dies eine Minimalkonfiguration ist, die funktioniert, aber stellenweise viel Geduld erfordert. Für flottes Entwickeln brauchen Sie einen Rechner mit 8 bis 12 Gbyte und einer SSD.
- Spaß am Programmieren, Ausdauer und auch ein bisschen Mut

■ 1.2 Installation des JDK

Wie bereits erwähnt, benötigen wir zur App-Entwicklung unbedingt ein JDK in einer aktuellen Version (falls Sie bereits ein JDK installiert haben, können Sie es möglicherweise verwenden; sicherer ist aber eine Neuinstallation).

1. In Anhang D finden Sie den Download-Link der zu Ihrem System passenden JDK-Setup-Datei.

Windows- und Linux-Anwender müssen zudem die zu ihrem System passende 32-Bit- oder 64-Bit-Setup-Datei wählen.

2. Nach dem Download doppelklicken Sie auf die Setup-Datei, um das Setup-Programm zu starten, und folgen Sie den Anweisungen des Setup-Programms.



32 ODER 64 BIT

Sie sind unsicher, ob Sie über einen 32-Bit- oder 64-Bit-Rechner verfügen? Wenn es sich um einen Windows-Rechner handelt, rufen Sie die Systemsteuerung auf, schalten Sie die **Anzeige** ggf. auf **Kleine Symbole** und klicken Sie auf **System**. Auf der erscheinenden Seite werden Ihnen Betriebssystem- und Prozessortyp angezeigt.



Auf der Oracle-Website

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

können Sie das jeweils neueste Java SDK herunterladen.

1.3 Installation von Android Studio

In Anhang D finden Sie den Link zu der zu Ihrem System passenden Installationsdatei. Nach dem Download starten Sie das Programm und folgen Sie den weiteren Anweisungen (meistens einfach auf **Next** klicken). Achten Sie allerdings darauf, als Installationsort für Studio und SDK nicht die Vorgaben zu übernehmen, sondern einen neuen Ordner anzulegen (beispielsweise `c:\Android`). Die Erfahrung hat gezeigt, dass Sie damit vielen kleinen Problemen aus dem Weg gehen. Für den Rest des Buches gehen wir davon aus, dass Sie Android Studio und Android SDK nach `c:\Android` installiert haben.



Auf der Website

<http://developer.android.com/sdk/index.html>

können Sie die jeweils neueste Version des Android Studio herunterladen. Tun Sie dies aber möglichst erst nach Durcharbeiten dieses Buches! Android Studio wird fortlaufend überarbeitet, sodass sich beim Herunterladen der aktuellen Version Abweichungen zu den Abbildungen und Beschreibungen im Buch ergeben können.

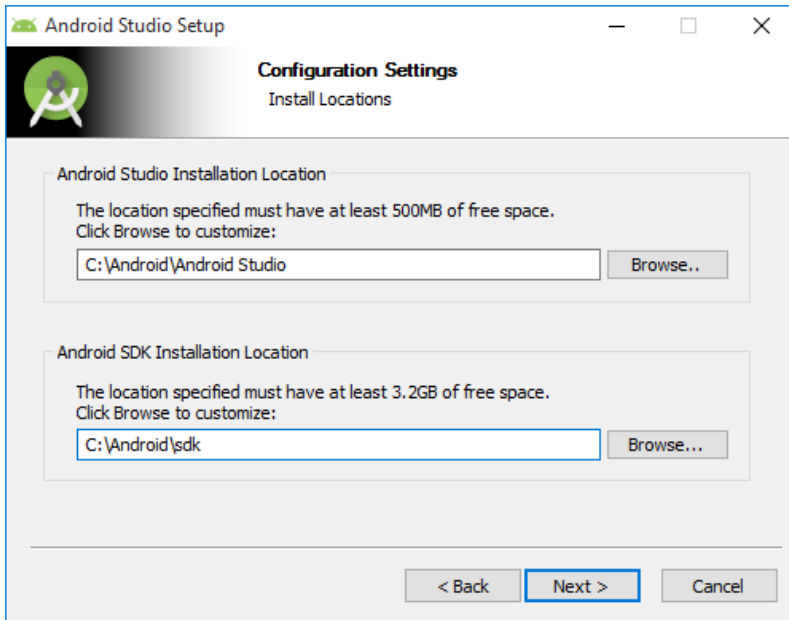


Bild 1.1 Installationsort festlegen

Nach erfolgter Installation bietet Ihnen das Setup-Programm die Option an, Android Studio für Sie zu starten. Halten Sie hier bitte einen kurzen Moment inne.

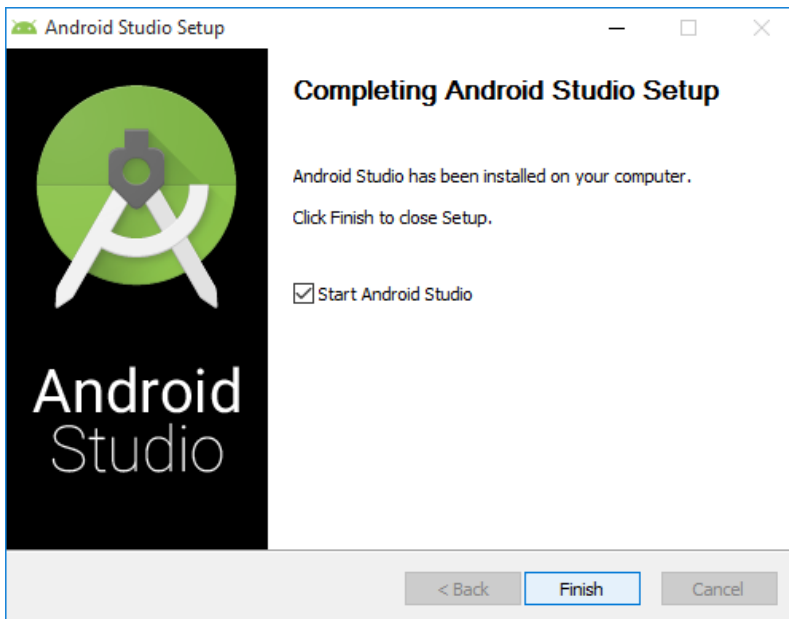


Bild 1.2 Die Installation ist fast beendet

Falls Sie ein 64-Bit System haben (und daher die 64-Bit-Version von Android Studio verwenden) und in der glücklichen Lage sind, über mindestens 8 Gbyte Hauptspeicher zu verfügen, dann sollten Sie an dieser Stelle eine Einstellung von Android Studio ändern. Öffnen Sie hierzu die Datei `C:\Android\Android Studio\bin\studio64.exe.vmoptions` mit einem Texteditor. Editieren Sie die Zeile mit dem Eintrag `-Xmx` und setzen Sie den für Android Studio verfügbaren Speicher hoch (speichern nicht vergessen!), z.B.:

```
-Xmx2500m
```

Stellen Sie als Nächstes sicher, dass Sie eine aktive Internetverbindung haben.

Kehren Sie nun zum obigen Abschluss-Dialog des Setup-Programms zurück. Lassen Sie das Häkchen für den Start von Android Studio aktiviert und beenden Sie mit **Finish** die Installation. Android Studio wird danach automatisch zum ersten Mal gestartet.



Android Studio basiert auf der IDE IntelliJ IDEA von JetBrains und ist unter professionellen Java-Programmierern sehr beliebt.

Falls Sie noch nie mit einer IDE gearbeitet haben, wird Sie die Fülle an Möglichkeiten einer IDE am Anfang „erschlagen“. Aber keine Sorge. Wir werden daher gerade in den ersten Kapiteln dieses Buches nebenbei auch des Öfteren auf den Umgang mit Android Studio selbst eingehen. Der Aufwand lohnt sich allemal, denn die Alternative – Android-Programmierung mit bloßem Texteditor und dem reinen Android SDK – birgt noch weitaus mehr Tücken.

Die wichtigsten Arbeitsschritte mit Android Studio haben wir überdies für Sie noch einmal im Anhang zusammengefasst.

1.3.1 Erster Start

1. Android Studio wird automatisch nach Abschluss der Installation gestartet. Unter Windows ist es auch über das Startmenü mit **Programme/Android Studio** bzw. **Alle Apps/Android Studio** aufrufbar.



ACHTUNG

Falls eine Fehlermeldung erscheint, dass keine JRE- oder JDK-Installation gefunden werden konnte, dann setzen Sie die Umgebungsvariable `JAVA_HOME` auf das Verzeichnis, wo Ihre Java-Installation liegt (unter Windows setzen Sie Umgebungsvariablen über das Startmenü: **System/Systemeigenschaften/Erweitert/Umwgebungsvariablen**).

2. Beim ersten Start von Android Studio erscheint möglicherweise eine Meldung Ihrer Firewall, weil ein Zugriff auf das Internet versucht wird. Erlauben Sie dies, da weitere Komponenten heruntergeladen bzw. auf den letzten Stand gebracht werden.

Falls Sie eine Firewall-Software verwenden, die nicht von selbst beim Benutzer nachfragt, müssen Sie Android Studio manuell bei der Firewall eintragen und Internet-Zugriff erlauben.

3. Üblicherweise fragt Android Studio auch noch diverse Einstellungen für die erste Einrichtung ab. Klicken Sie sich dann einfach durch die Dialoge und übernehmen Sie die Voreinstellungen. Wenn Android Studio mit dem Download beginnt, machen Sie sich in Ruhe einen Kaffee oder Tee und warten Sie, bis Android Studio fertig ist.

Eventuell wird eine Fehlermeldung angezeigt, dass eine SDK-Komponente nicht installiert wurde. Klicken Sie in diesem Fall einfach auf **Retry**, dann verschwindet das Problem in der Regel.

4. Klicken Sie zum Schluss auf **Finish**. Der Startbildschirm von Android Studio erscheint.

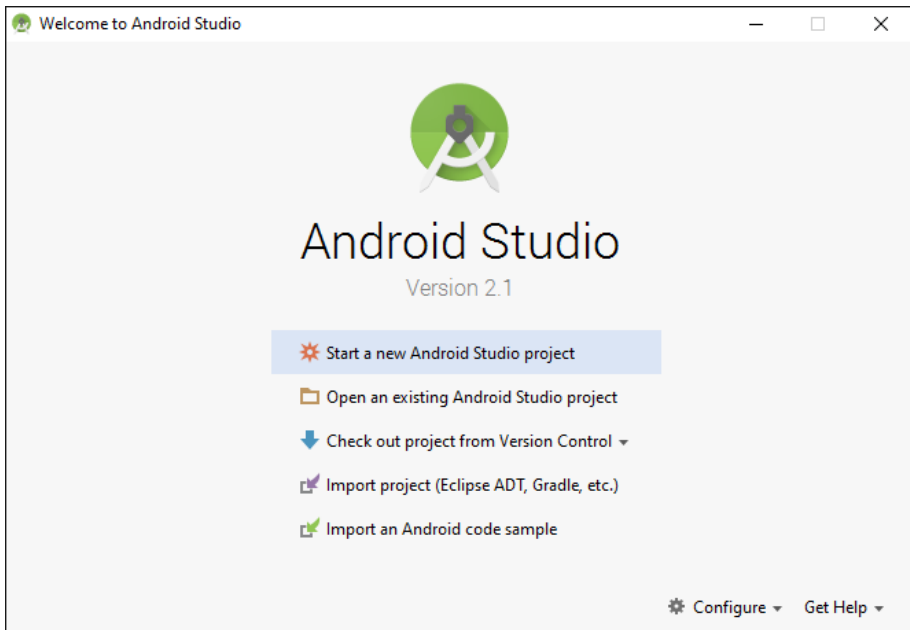


Bild 1.3 Android Studio wurde das erste Mal gestartet



Englischkenntnisse sind für die Arbeit mit dem englischsprachigen Android Studio hilfreich. Eine Umstellung der IDE auf Deutsch ist nicht möglich. Für Leser, die mit der englischen Fachterminologie noch nicht so vertraut sind, haben wir daher im Anhang ein kleines Wörterbuch mit wichtigen Begriffen zusammengestellt.



Da Android Studio ständig aktualisiert wird, werden Sie von Anfang an Meldungen erhalten, dass es Updates für diverse Android Studio-Komponenten gibt. Solange Sie sich noch mithilfe dieses Buches in Android Studio einarbeiten, sollten Sie Android Studio möglichst nicht aktualisieren, da es sonst unnötige Abweichungen zwischen den Beschreibungen im Buch und der Software geben kann.

■ 1.4 Der Android-SDK-Manager

Eigentlich könnten wir an diesem Punkt die Installation bereits als abgeschlossen ansehen und mit der Android-Programmierung beginnen. Doch es gibt da noch einen kleinen Aspekt, dem wir unsere Aufmerksamkeit schenken sollten.

1.4.1 Die Android-Versionen

Android gibt es in verschiedenen Versionen, manchmal auch Plattformen genannt. Und zu jeder Version gibt es eine zugehörige API², die benötigt wird, um Android-Apps für die betreffende Android-Version zu schreiben, beispielweise gehört zu Android 6.0 (Codename Marshmallow) die API-Version 23. Wir verwenden im Buch Android 6.0, weil dies zu dem Zeitpunkt unserer Überarbeitung die aktuelle Version war. Wenn Sie das Buch in Händen halten, ist vermutlich schon 7.0 mit der API 24 oder 25 aktuell. Dies soll für Sie aber keine Rolle spielen. Verwenden Sie einfach zum Einstieg in die Android-Programmierung wie beschrieben Version 6.0 und API 23. Wenn Sie den Einstieg geschafft haben und dann in Ihren Apps auch Neuerungen von Android 7.0 nutzen möchten, können Sie jederzeit upgraden.

Android Studio (siehe Anhang D) installiert die jeweils aktuelle API (im Buch API 23). Mit dieser API können Sie nicht nur alle Neuerungen nutzen, sondern auch Apps schreiben, die zu älteren APIs, und damit auch älteren Android-Versionen, abwärtskompatibel sind.

Letzter Punkt ist extrem wichtig, denn nur die wenigsten Android-Anwender (sprich Kunden für Ihre Apps) aktualisieren bei jeder neu erscheinenden Android-Version ihr Smartphone.

Version	Codename	API	Distribution
2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	3.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.9%
4.1.x	Jelly Bean	16	10.0%
4.2.x		17	13.0%
4.3		18	3.9%
4.4	KitKat	19	36.6%
5.0	Lollipop	21	16.3%
5.1		22	13.2%
6.0	Marshmallow	23	0.5%

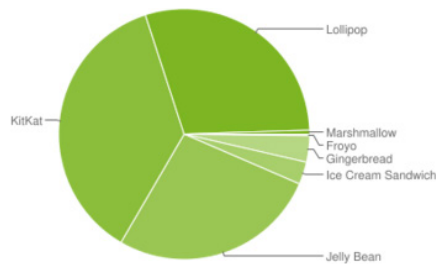


Bild 1.4 Aktuelle Verteilung der Android-Versionen im Januar 2016 (Screenshot der Webseite <http://developer.android.com/resources/dashboard/platform-versions.html>)

² API ist das Akronym für „Application Programming Interface“ und die Schnittstelle zu den Bibliotheksklassen, die Sie für die App-Programmierung benutzen.

Wie Sie sehen können, konnten im Januar 2016 nur 0,5% der Android-Anwender Apps auf ihren Smartphones ausführen, die auf Android 6.0 (API 23) basieren, während Apps, die mit der API 8 (Android 2.2 Froyo) auskommen, auf fast 100% aller Android-Smartphones ausführbar sind (nämlich alle, die 2.2 und höher installiert haben).

Wenn Sie nun eigene Apps schreiben, können Sie beispielsweise festlegen, dass Sie zur Erstellung der App zwar die API 23 nutzen, die App aber abwärtskompatibel bis API 8 sein soll. Allerdings lauert hierbei eine kleine Falle.

Sie können zwar angeben, dass Ihre App zu einer älteren API abwärtskompatibel sein soll, aber die Android-Entwicklungsumgebung bewahrt Sie leider nicht davor, in einer solchen App Elemente zu verwenden, die dieser Abwärtskompatibilität widersprechen.

Sie sollten daher die Abwärtskompatibilität stets noch einmal explizit testen, bevor Sie eine App in die weite Android-Welt entlassen. Keine Angst, dies bedeutet nicht, dass Sie einen Satz von Smartphones mit den verschiedenen Android-Versionen anschaffen müssen. Zur Android-Entwicklungsumgebung gehört ein Emulator, mit dem Sie die verschiedensten Smartphones simulieren können. Um jedoch ein Smartphone für eine bestimmte Android-Version simulieren zu können, muss die zugehörige API (inklusive eines eigenen SDK) auf Ihrem Rechner installiert sein.

Und genau aus diesem Grund empfehlen wir Ihnen, schnell noch die API 16 zu installieren, um ggf. die Abwärtskompatibilität zu Smartphones ab Android 4.1 testen zu können. Damit haben Sie den größten Teil der aktiv genutzten Android-Versionen abgedeckt.



VERSIONEN UND APIS

Wie von jeder Software kommen auch von dem Android-Betriebssystem ständig neue und erweiterte Versionen (Plattformen) heraus. Zu jeder dieser Plattformen gibt es eine eigene API.

1.4.2 APIs/SDKs und anderes nachinstallieren

1. Zum Installieren weiterer SDK-Komponenten müssen Sie den sogenannten SDK-Manager öffnen. Klicken Sie hierzu auf der Startseite von Android Studio im Quick-Start-Bereich auf die Schaltfläche **Configure** (siehe Bild 1.3), dann auf die Schaltfläche **SDK Manager**.
2. Nun können Sie im Fenster des SDK-Managers unter dem Reiter **SDK Platforms** eine Übersicht sehen, welche API-Versionen bereits installiert sind bzw. welche installiert werden können. Wählen Sie nun aus, welche API nachinstalliert werden soll: API 16 (für Android 4.1).

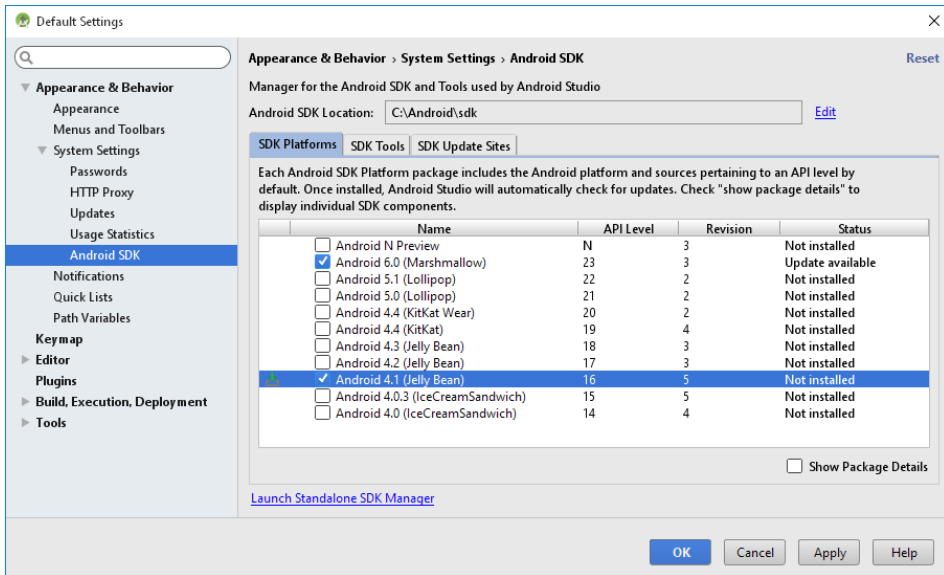


Bild 1.5 Auswahl der zu installierenden Android SDKs

3. Wechseln Sie jetzt zum Reiter **SDK Tools**. Wählen Sie dort die Einträge **Google USB Driver** und – falls noch nicht installiert – **Intel x86 Emulator Accelerator**.

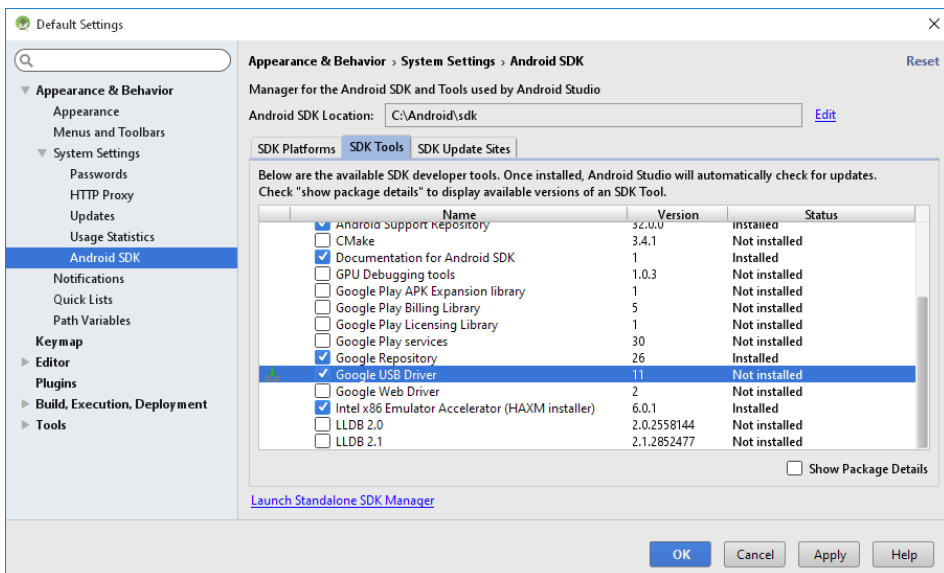


Bild 1.6 Auswahl weiterer Tools

4. Lassen Sie die ausgewählten Komponenten installieren, indem Sie auf **OK** klicken. Unter Umständen erscheint ein Dialog, in dem Sie die Lizenzvereinbarungen annehmen müssen.

Das Herunterladen und Installieren der Komponenten wird vermutlich etwas länger dauern. Haben Sie also ein wenig Geduld. Anhand des Fortschrittsbalkens im **SDK Manager**-Fenster können Sie das Fortschreiten der Installation verfolgen.

5. Beenden Sie die Installation durch Klick auf **Finish**.

1.4.3 Dokumentation und API-Referenz

Lassen Sie uns nun kurz durch das Installationsverzeichnis des Android SDK streifen, um zu sehen, ob alle wichtigen Komponenten heruntergeladen wurden.

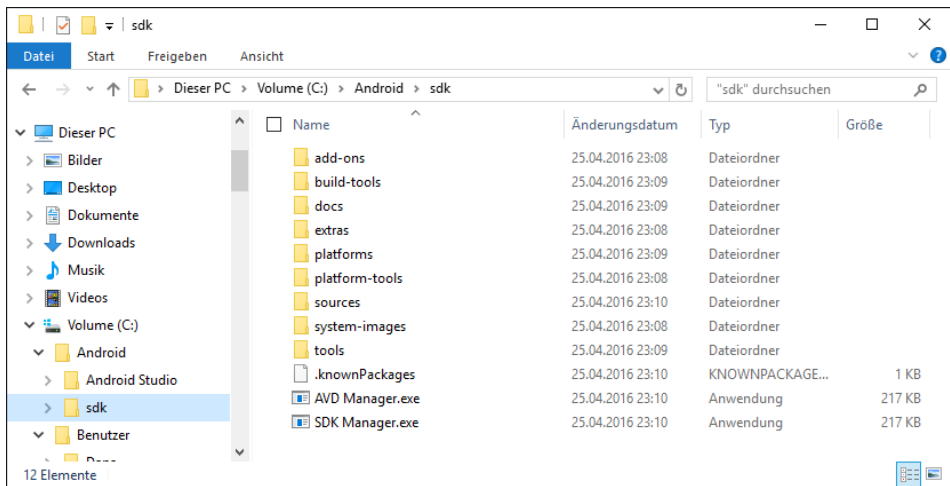


Bild 1.7 Das Verzeichnis des Android SDK

Tabelle 1.1 Wichtige Verzeichnisse des Android SDK

Unterverzeichnis	Inhalt
<i>tools</i>	Hier sind die verschiedenen Hilfsprogramme des SDK zusammengefasst; darunter z. B. auch der Emulator, der ein Android-Smartphone simuliert und mit dessen Hilfe Sie Ihre Apps direkt auf dem PC testen können.
<i>platforms</i>	Für jede Plattform (Android-Version), die Sie mithilfe des SDK-Managers heruntergeladen haben, finden Sie hier ein eigenes Unterverzeichnis mit der Programmierbibliothek (<i>android.jar</i>), individuellen Oberflächen für den Emulator (<i>skins</i>) und diversen anderen plattformspezifischen Dateien.
<i>platform-tools</i>	Enthält plattformspezifische Hilfsprogramme
<i>docs</i>	Hilfe und Dokumentation

Besondere Aufmerksamkeit verdient das Verzeichnis *docs*. Hier finden Sie die vollständige Dokumentation zu Android, inklusive Programmierhandbüchern und API-Referenz, quasi ein lokales Abbild der Android-Website. Ausgangspunkt ist die Datei *index.html*.



Die Android-Hilfe ist komplett auf Englisch.

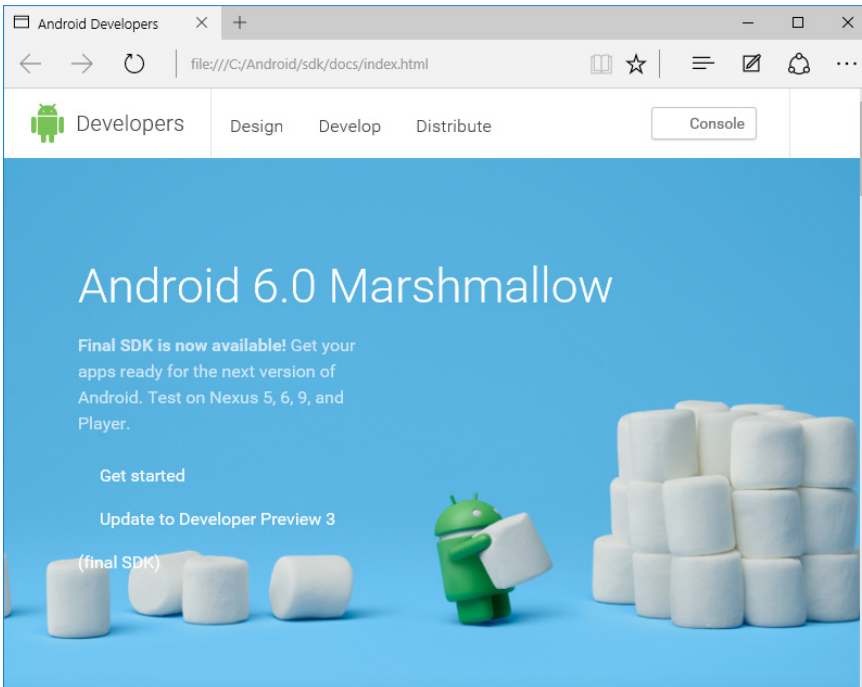


Bild 1.8 Die Startseite der lokal installierten Android-Hilfe

Tabelle 1.2 Wichtige Links der lokalen Android-Hilfe

Datei	Inhalt
<i>file:///C:/Android/sdk/docs/sdk/index.html</i>	Hier können Sie das neueste SDK herunterladen, sich über die Systemvoraussetzungen informieren oder Hilfe zur Installation finden.
<i>file:///C:/Android/sdk/docs/training/index.html</i>	Hier finden Sie viele hilfreiche Tutorien.
<i>file:///C:/Android/sdk/docs/guide/index.html</i>	Hier finden Sie Hinweise, Artikel und Hintergrundinformationen zu praktisch allen Aspekten der Android-Programmierung.
<i>file:///C:/Android/sdk/docs/reference/packages.html</i>	Dies ist die Referenz der Elemente aus der Android-Bibliothek. Die Referenz ist analog der Java-API-Dokumentation aufgebaut. Zuerst wählen Sie links oben ein Paket (<i>Package</i>) aus, dann links unten eine der im Paket enthaltenen Klassen (<i>Classes</i>) oder sonstigen Elemente. Danach wird im rechten Bereich die Dokumentation zu dem ausgewählten Element angezeigt.

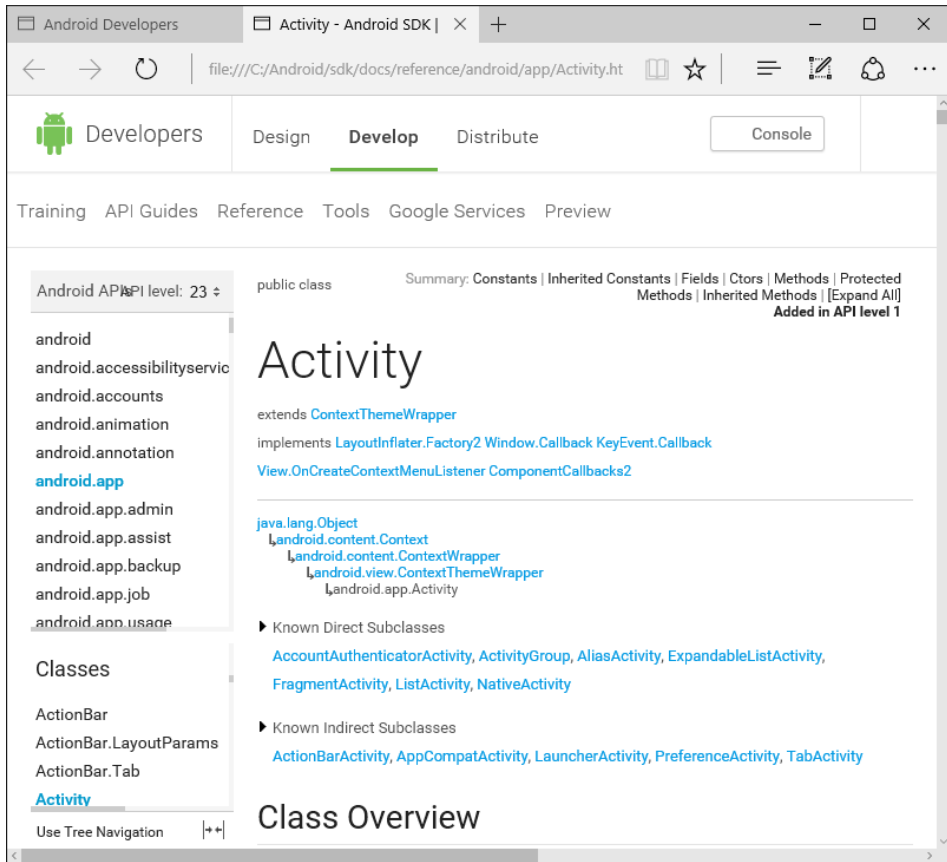


Bild 1.9 Referenzdokumentation zur Android-Klasse Activity

■ 1.5 Wo Sie weitere Hilfe finden

Weitere Informationen finden Sie auf der Support-Site zu diesem Buch

www.carpelibrum.de

und auf den Websites von Oracle

www.oracle.com

und Android

developer.android.com.

Sollten die Hinweise im Buch und auf der Website nicht ausreichen, haben Sie keine Scheu, sich per E-Mail an uns zu wenden (autoren@carpelibrum.de oder leserfragen@gmx.de).

■ 1.6 Nächste Schritte

Der Rechner ist vorbereitet, die nötigen Hilfsmittel sind zusammengetragen, die Programmierumgebung ist eingerichtet. Nun wollen wir sehen, wie wir mithilfe dieser Programmierumgebung Apps erstellen, mit welchen Aufgaben wir bei der App-Programmierung konfrontiert werden und wie Apps grundsätzlich aufgebaut sind.

Für Programmieranfänger und Umsteiger von anderen Programmiersprachen ist jetzt ein guter Zeitpunkt, um mit dem Java-Schnellkurs unter <http://files.hanser.de/fachbuch/PDFs.zip> zu beginnen. Programmieranfänger sollten die Kapitel 1 bis 3 des Java-Tutoriums lesen. Umsteiger von anderen objektorientierten Sprachen können Kapitel 3 überspringen (sofern sie mit Begriffen wie Klassen, Instanziierung, Konstruktor, Methoden und Vererbung vertraut sind).

Wenn Sie möchten, können Sie auch bereits Kapitel 4 des Java-Tutoriums durcharbeiten, in dessen Zuge das Grundgerüst einer Java-Anwendung analysiert und das Konzept der Java-Bibliotheken und -Pakete vorgestellt wird. Es steht Ihnen aber auch frei, erst mit Kapitel 2 dieses Buchs zu beginnen und das Tutorium-Kapitel 4 dann parallel zu Kapitel 2.3 zu lesen.

■ 1.7 Frage und Antworten

Die Versionsangaben zu Android sind ziemlich verwirrend. Gibt es da irgendwelche Regeln, die Ihnen helfen, besser durchzublicken?

Wichtig ist, zwischen Android-Betriebssystem und Android-API zu unterscheiden. Betriebssystemversionen werden üblicherweise mit Nachkommastelle angegeben (wie z. B. Android 4.4), die API-Nummern sind ganzzahlig. Leider gibt es keine Regel, wie man aus der Betriebssystemversion die zugehörige API ableiten kann. Man muss sich die Zuordnung einfach merken oder irgendwo nachschlagen (beispielsweise durch Aufruf des SDK-Managers).

■ 1.8 Übungen

1. Falls Sie es noch nicht getan haben, sollten Sie jetzt Ihre Android-Entwicklungsumgebung einrichten.
2. Falls dies für Sie auch gleichzeitig der Einstieg in die Java-Programmierung ist, sollten Sie jetzt – sofern Sie es nicht schon getan haben –, wie im Abschnitt 1.6 vorgeschlagen, mit dem Durcharbeiten der ersten Kapitel des Java-Tutoriums beginnen.

2

Auf die Plätze, fertig ... App!

Um eine eigene App zu schreiben, müssen Sie sich mit Ihrer Entwicklungsumgebung auskennen, Sie müssen die Grundlagen der Programmierung mit Java beherrschen und nicht zuletzt müssen Sie natürlich auch noch mit den speziellen Erfordernissen und Techniken der Android-Programmierung vertraut sein. Dies sind gewaltige Anforderungen.

So schwierig, wie es sich jetzt vielleicht anhört, ist es allerdings auch nicht. Wer einen Berg abtragen will, muss mit dem ersten Spatenstich beginnen. Blicken Sie also nicht auf den Berg, der noch vor Ihnen liegt, sondern immer nur auf den nächsten Spatenstich. Wir werden es mit unseren Erläuterungen ebenso halten.



Da Android Studio ständig aktualisiert wird, werden Sie von Anfang an Meldungen erhalten, dass es Updates für diverse Android Studio-Komponenten gibt. Solange Sie sich noch mithilfe dieses Buches einarbeiten, sollten Sie Android Studio möglichst nicht aktualisieren, da es sonst unnötige Abweichungen zwischen den Beschreibungen im Buch und der Software geben kann.

■ 2.1 Die Ruhe vor dem Sturm

Gerade der Einstieg in die App-Erstellung konfrontiert den Anfänger mit vielen neuen und ungewohnten Konzepten. Sie mit all diesen Konzepten direkt vertraut zu machen, wäre langwierig, ermüdend und nur wenig lehrreich.

Die gute Nachricht ist: Sie müssen sich nicht erst intensiv mit allen diesen Konzepten auseinandersetzen, um eine funktionierende App zu schreiben. Wir werden es daher im Folgenden so halten, dass wir uns jeweils nur auf die Konzepte, Techniken und Syntaxformen konzentrieren, die für den jeweils nächsten Schritt auf dem Weg zur angestrebten App wichtig sind. So wird niemand überfordert, es stellen sich schnell Erfolge ein, der Spaß an der App-Programmierung bleibt erhalten und wir dürfen dennoch sicher sein, dass sich nach und nach alles zu einem kompletten Gesamtbild zusammenfügt.

Holen Sie also noch einmal tief Luft ... und los geht's.

Die App, die wir in diesem Kapitel schreiben werden, soll nichts weiter tun, als uns auf dem Smartphone mit einem freundlichen „Hallo App-Entwickler!“ zu begrüßen. Das ist nicht gerade viel, aber es geht auch gar nicht darum, im ersten Versuch gleich eine sinnvolle und perfekte App zu erstellen. Es geht darum, einen Überblick über den App-Entwicklungsprozess zu bekommen. Und es geht darum, uns selbst zu beweisen, dass wir fähig sind, eigene Apps zu schreiben. Wenn wir diese App meistern, dann können wir auch jede andere App programmieren.



ACHTUNG

Vergewissern Sie sich vor der Arbeit mit Android Studio immer, dass Sie mit dem Internet verbunden sind.

■ 2.2 Das Projekt

Wer wie wir Android Studio als Entwicklungsumgebung verwendet, für den beginnt die Arbeit an jeder neuen App mit dem Anlegen eines passenden Projekts. Starten wir also Android Studio und legen wir ein neues Projekt an.



PROJEKTE

Projekte sind die interne Verwaltungseinheit, in der Android Studio alle Daten und Dateien zusammenfasst, die nötig sind, um eine App zu erstellen.

1. Klicken Sie auf der Begrüßungsseite von Android Studio auf die Schaltfläche **Start a new Android Studio project**.

Es erscheint das Dialogfeld **Create New Project**, das aus mehreren Seiten besteht, über die man diverse Angaben zu dem neu anzulegenden App-Projekt festlegen kann. Sie werden gleich feststellen, dass eine ganze Menge von teils sehr technischen Angaben abgefragt wird, was für den ProgrammierEinstieg etwas verwirrend sein kann. Aber sei's drum. Sehen wir uns die notwendigen Angaben einmal an.

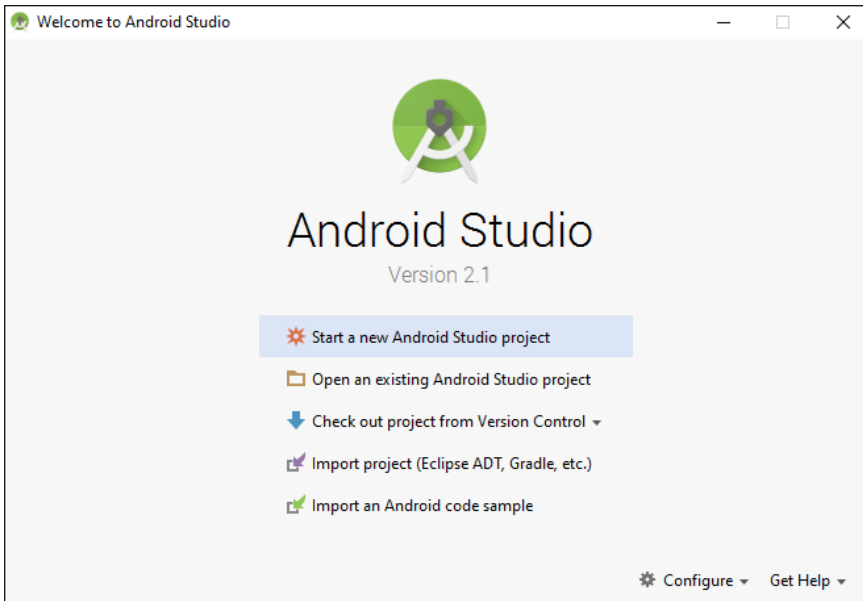


Bild 2.1 Ein neues Android-Projekt beginnen

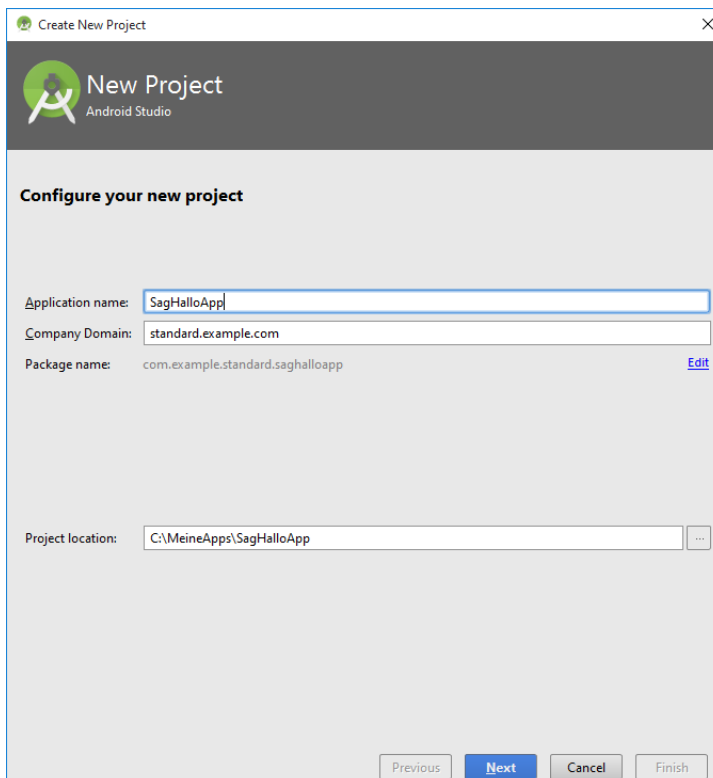


Bild 2.2 Die erste Seite des Dialogfelds zum Anlegen neuer Projekte

2. Auf der ersten Seite, **New Project**, werden verschiedene grundlegende Einstellungen abgefragt.

Da diese Einstellungen ziemlich wichtig sind, haben wir sie in Tabelle 2.1 zusammengefasst – inklusive der für unser Beispielprojekt erforderlichen Eingaben sowie Erläuterungen, was sich hinter diesen Einstellungen verbirgt.

Tabelle 2.1 Einstellungen für die Seite **New Project**

Feld	Eingabe	Bedeutung
Application Name	<i>SagHalloApp</i>	Der Name der App. Dies ist gleichzeitig auch in Android Studio der Projektname. Daher sind Sonderzeichen oder Umlaute nicht erlaubt. Unter dem App-Namen wird Ihre App später intern auf dem Smartphone geführt. Und falls Sie eine App später einmal unter Google Play veröffentlichen, wird sie dort ebenfalls unter diesem Namen gelistet. Achtung! In der App-Liste des Smartphones erscheint üblicherweise statt des App-Namens der Titel der Start-Activity.
Company Domain	<i>standard.example.com</i>	Ein eindeutiger Domainname. Solange Sie nur Apps zur Übung oder für den Eigenbedarf schreiben, können Sie die Vorgabe einfach übernehmen. Falls Sie Apps über Google Play veröffentlichen, dann müssen Sie hier die Web-Domain Ihres Unternehmens/Webseite angeben.
Package Name	<i>com.example.standard.saghalloapp</i>	Packages bzw. zu Deutsch Pakete sind ein Element der Sprache Java, das zur Organisation des Codes genutzt wird. Die Angabe ist für App-Projekte zwingend erforderlich und muss eindeutig sein, um sicherzustellen, dass es auf einem Android-System keine zwei Apps mit gleichem Paketnamen gibt. Typisch ist daher die Verwendung des Domainnamens in umgekehrter Reihenfolge mit angehängtem App-Namen. Genau dies macht Android Studio bereits automatisch für Sie. Falls Sie einen anderen Paketnamen wünschen, klicken Sie auf den Link Edit an der rechten Fensterseite. Übrigens: Paketnamen werden traditionell kleingeschrieben.
Project Location	<i>c:\MeineApps\SagHalloApp</i>	Der Ort auf der Festplatte, wo das Projekt und alle zugehörigen Daten abgespeichert werden. Für alle Beispiele in diesem Buch verwenden wir hierzu das Verzeichnis <i>c:\MeineApps</i> .

3. Klicken Sie auf **Next**, um die Zielgeräte (**Target Devices**) festzulegen, auf denen Ihre App ausführbar sein soll.

Übernehmen Sie die Vorgabe **Phone and Tablet**. Als **Minimum SDK** behalten Sie die Vorgabe bei oder wählen Sie **API 16: Android 4.1**.

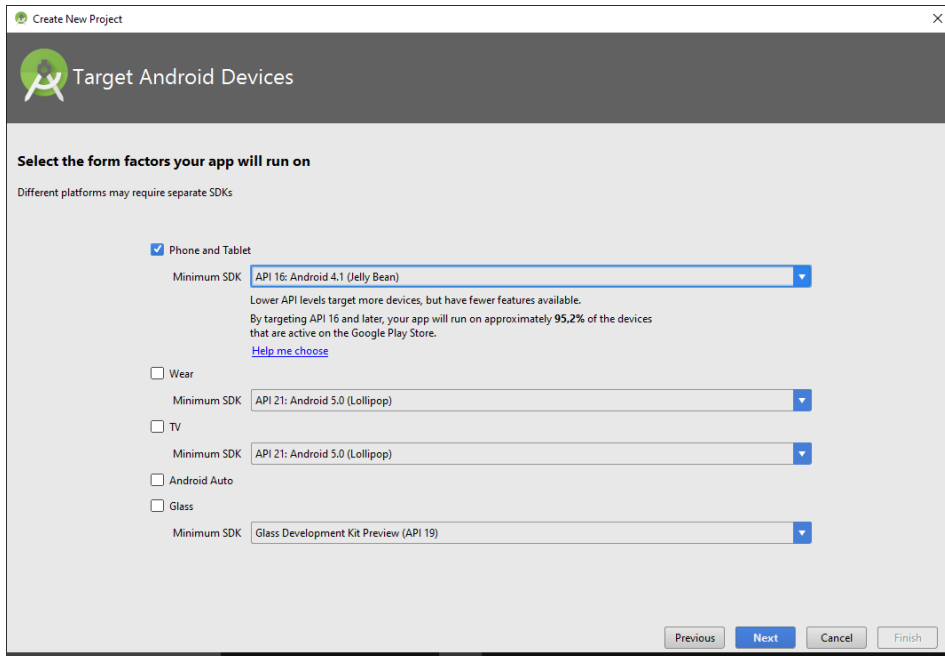


Bild 2.3 Die Zielplattform festlegen



Minimum SDK ist die Mindestversion des Android SDK, die auf einem Gerät vorhanden sein muss, damit Ihre App auf diesem Gerät installiert und ausgeführt werden kann. Ferner gibt es noch Target SDK Version sowie Compile SDK Version. Diese beiden werden automatisch von Android Studio auf die neueste vorhandene Android API-Version gesetzt.

4. Klicken Sie auf **Next**. Auf der Seite **Add an activity** können Sie festlegen, ob und mit welchem Grundgerüst das App-Projekt erzeugt werden soll. Wählen Sie hier den Eintrag **EMPTY Activity** und klicken Sie auf **Next**.



„Activities“ repräsentieren in einer App im Wesentlichen die einzelnen Bildschirmseiten der App. Alle Apps, die wir in diesem Buch erstellen, besitzen mindestens eine Activity. Die hier ausgewählte *Empty Activity* macht nichts anderes, als eine leere Bildschirmseite anzuzeigen.

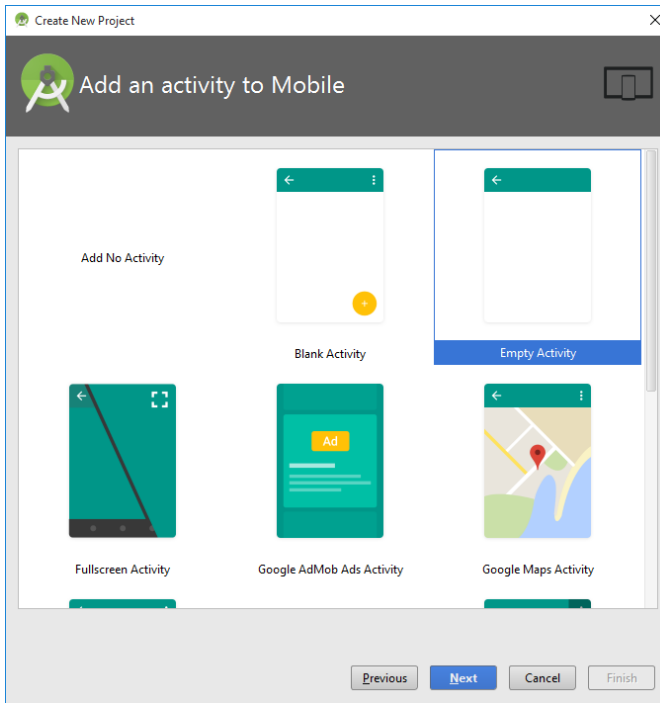


Bild 2.4
Eine leere Activity
hinzufügen

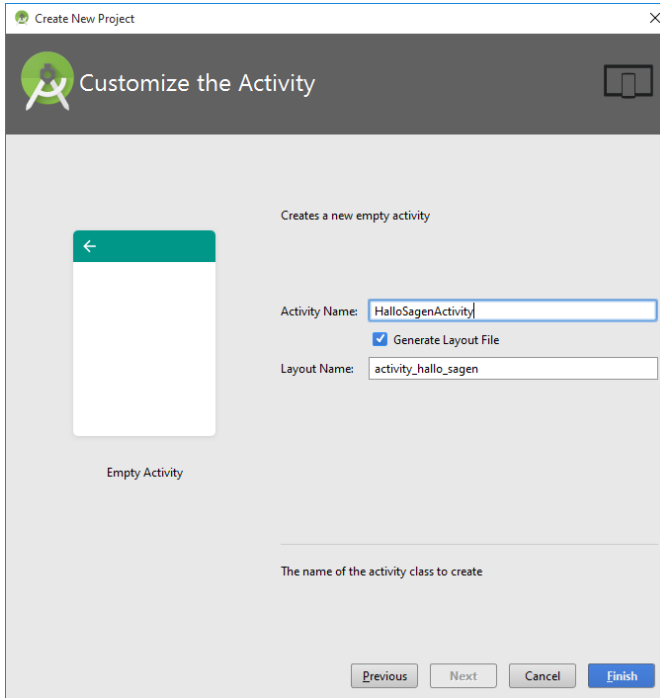


Bild 2.5
Die Activity konfigurieren

5. Nachdem wir uns entschlossen haben, mit einer leeren Activity (Empty Activity) zu starten, müssen wir noch nähere Angaben zu dieser Activity machen, damit Android Studio alles richtig anlegen kann.

Füllen Sie die Seite gemäß Bild 2.5 aus.

Tabelle 2.2 Einstellungen für die Seite **New empty Activity**

Feld	Eingabe	Bedeutung
Activity Name	<i>HalloSagenActivity</i>	Wir nennen die Haupt-Activity unserer App einfach <code>HalloSagenActivity</code> . Achtung! Falls Sie einen eigenen Namen vergeben, beachten Sie, dass dieser keine Leer- und keine Sonderzeichen enthalten darf, da das Android Studio unter diesem Namen eine Klasse definiert. (In Java dürfen Namen von im Code definierten Elementen keine Leer- oder Sonderzeichen enthalten.)
Layout Name	<i>activity_hallo_sagen</i>	Grundsätzlich wird zu jeder Activity ein Layout definiert, das angibt, wie die Bildschirmseite der Activity aufgebaut ist. Übernehmen Sie einfach die Vorgabe. Dann kann man leichter ableiten, welches Layout zu welcher Activity gehört.

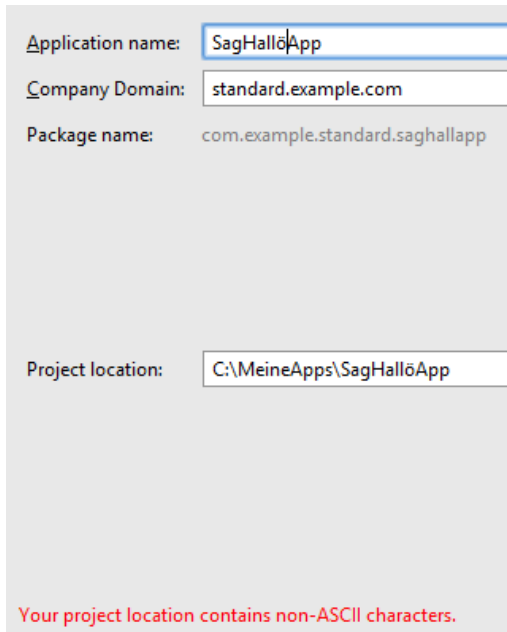


Wenn Sie Namen für Verzeichnisse oder Programmelemente vergeben, verzichten Sie auf Leerzeichen, Sonderzeichen und möglichst auch auf Umlaute (obwohl Letztere mittlerweile von den meisten Systemen und Programmiersprachen korrekt verarbeitet werden).

6. Klicken Sie auf **Finish**, um das Projekt anlegen zu lassen.

Automatische Fehlerkontrolle

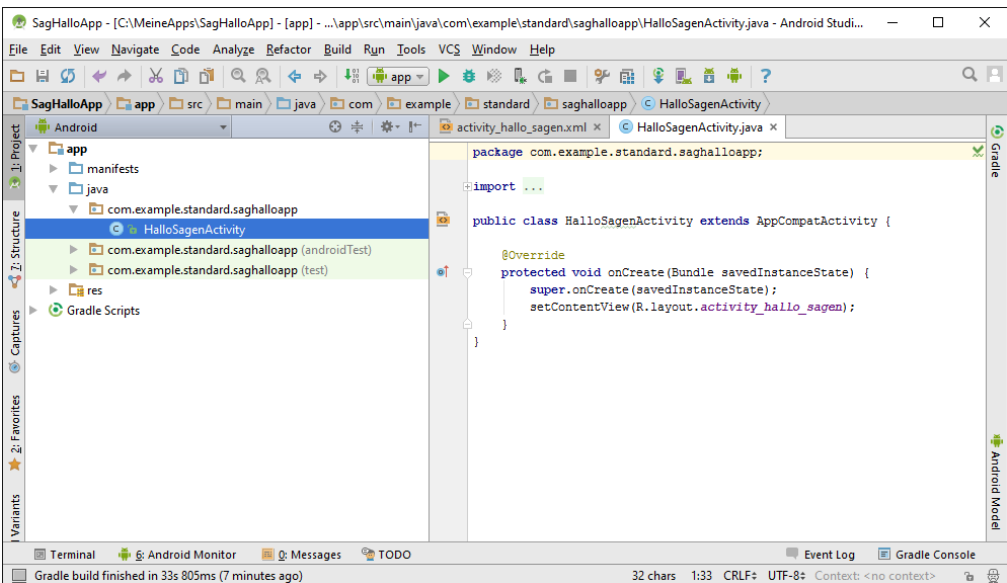
Gerade für Einsteiger sehr hilfreich ist die automatische Fehlerkontrolle des **Create New Project**-Dialogs, die eine ganze Reihe von fehlerbehafteten oder widersprüchlichen Einstellungen abfängt. Beispielsweise sind an vielen Stellen wie dem App-Namen keine Sonderzeichen oder Umlaute erlaubt. Falls Sie dennoch Leerzeichen einbauen, weist Sie das Dialogfeld mit einer roten Meldung im unteren Bereich darauf hin. Zusätzlich wird die Schaltfläche **Next** oder **Finish** deaktiviert, sodass Sie gezwungen sind, den Fehler zu beheben, bevor Sie weitermachen können.

**Bild 2.6**

Das Dialogfeld weist Sie darauf hin, dass es in den Eingaben eine Unstimmigkeit gibt.

Das neu angelegte Projekt in Android Studio

Zurück im Hauptbildschirm von Android Studio sollten Sie nun das neu angelegte Projekt sehen. Die Oberfläche von Android Studio besteht aus verschiedenen Unterfenstern und Bereichen, die Sie einzeln verbergen können (Klick auf das Pfeil-Icon in der jeweiligen rechten oberen Ecke) und über das Menü **View/Tool Windows** wieder sichtbar machen.

**Bild 2.7** Das neu angelegte Projekt in Android Studio

In der linken Hälfte sehen Sie die Projektansicht, der wir uns gleich näher widmen. Im rechten Bereich wird in der Regel der Editor angezeigt: Für jede geöffnete Datei wird ein eigenes Register angelegt und der Inhalt kann betrachtet oder bearbeitet werden. Je nach Dateityp kann der Editor dabei eine rein textbasierte Sache oder aber auch ein spezifischer Designer mit grafischer Oberfläche sein.

Die Projektansicht auf der linken Seite zeigt alle Ordner und Dateien des Projekts und hat zwei Hauptzweige:

- *app*: Hier liegt der eigentliche Quellcode der App, inklusive der notwendigen Bilder, Texte, Bibliotheken usw. Hier dürfen Sie sich als App-Entwickler austoben.
- *Gradle Scripts*: Beinhaltet Skripte und Informationen für das Build-System von Android Studio, um aus den verschiedenen Artefakten eine funktionierende App zu erzeugen. Hier brauchen und sollten Sie als Einsteiger nichts anfassen!

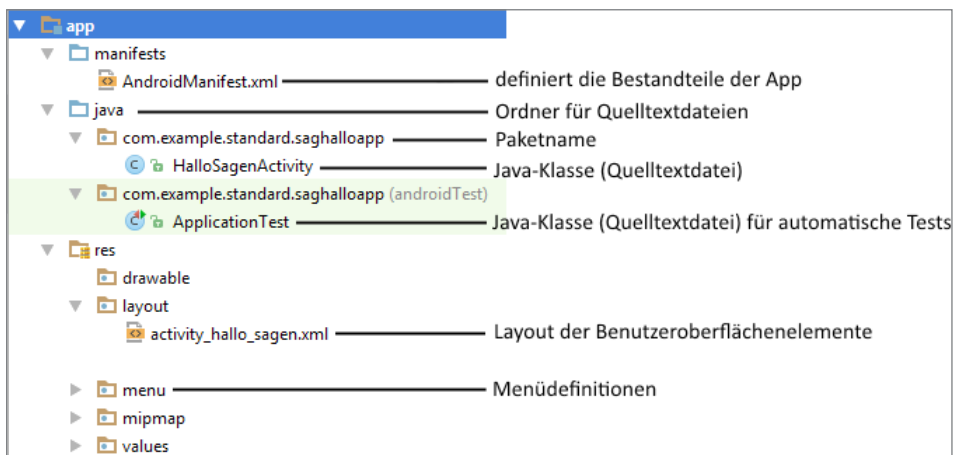


Bild 2.8 Der app-Ordner in der Projektansicht

Das Projekt auf der Festplatte

Die Ordnerstruktur des Projekts in Android Studio finden Sie fast identisch auch auf der Festplatte wieder. Aus dem vorangehenden Abschnitt wissen Sie ja bereits, dass Android Studio zu jedem Projekt ein gleichnamiges Projektverzeichnis auf der Festplatte anlegt. Aber auch die Unterordner der Projektstruktur finden Sie als Unterordner im Projektverzeichnis wieder.

Eine kleine Abweichung gibt es lediglich bei den Dateien, die den Java-Teil einer App beinhalten. Der Basisordner ist nicht wie in der Projektansicht `app\java`, sondern `app\src\main\java`. Darunter liegen dann weitere Unterverzeichnisse, die den Namensteilen des Paketnamens entsprechen: Java verlangt nämlich, dass für jeden der durch Punkte getrennten Namensteile eines Paketnamens ein eigenes Unterverzeichnis erstellt wird und dass die Quelldateien aus einem Paket in dem durch den Paketnamen spezifizierten Verzeichnis stehen. Aus diesem Grund finden Sie die Quelltextdatei `HalloSagenActivity.java` aus unserem Paket `com.example.standard.saghalloapp` nicht direkt im Unterverzeichnis `app\src\main\java`, sondern unter `app\src\main\java\com\example\standard\saghalloapp`.

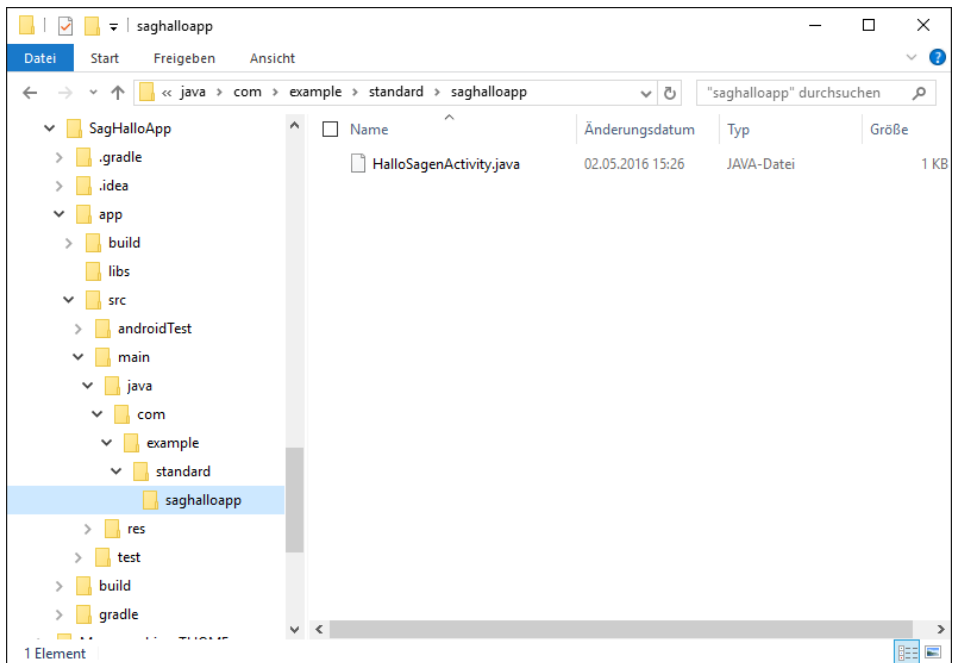


Bild 2.9 Das Projekt SagHalloApp auf der Festplatte



TIPP

Sie erinnern sich nicht mehr, wo auf der Festplatte Ihr Projekt liegt, oder Sie wollen einfach mal direkt dorthin gehen? Klicken Sie nur in der Projektansicht den gewünschten Ordner mit der rechten Maustaste an und wählen Sie aus dem Kontextmenü den Eintrag **Show in Explorer**. Der registrierte Dateimanager (z. B. Windows Explorer) wird automatisch geöffnet und Sie befinden sich im gewünschten Verzeichnis.

■ 2.3 Das vorgegebene Codegerüst

Lassen Sie uns nun einen Blick in den Code werfen, der bereits für uns erzeugt wurde.



MATERIAL ZUM BUCH

Auch wenn der Code des für uns angelegten Codegerüsts nur wenige Zeilen umfasst, nutzt er bereits eine Vielzahl objektorientierter und Java-typischer Syntaxelemente. Programmieranfänger und Umsteiger von anderen Sprachen sollten daher jetzt Kapitel 4 des Java-Tutoriums öffnen und das Tutorium vorab oder parallel zu den folgenden Ausführungen lesen.

Doppelklicken Sie in der Projektansicht auf den Knoten *HalloSagenActivity*. An dem **C** (wie Class) vor dem Knoten erkennen Sie übrigens, dass es sich um den Quelltext einer Java-Klasse handelt.

Android Studio lädt den Inhalt der Datei in den Editor und zeigt ihn in der rechten Fensterhälfte in einem neuen Editor-Register an.

Listing 2.1 Inhalt der Klasse *HalloSagenActivity*

```
package com.example.standard.saghalloapp;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class HalloSagenActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hallo_sagen);
    }
}
```

Leser, die bereits über Erfahrung in der Java-Programmierung verfügen, werden die grundlegende Struktur sicher schnell entschlüsselt haben ... und sich vielleicht wundern, dass keine `main()`-Methode definiert wird.

Auf Leser, die noch über keinerlei Erfahrung in der Java-Programmierung verfügen, dürfte dieses Codebeispiel eher abschreckend wirken, doch der Aufbau ist gar nicht so kompliziert, wie es auf den ersten Blick aussieht. Blendet man die Details aus, schälen sich nur wenige Komponenten heraus (siehe Bild 2.10). Klicken Sie hierzu im Editor am linken Rand auf das Minus-Symbol neben dem Methodennamen `onCreate()`. Dadurch wird der Methodenrumpf ausgeblendet und das Ganze ist etwas übersichtlicher. Lassen Sie uns nun von oben nach unten die Bestandteile näher betrachten.

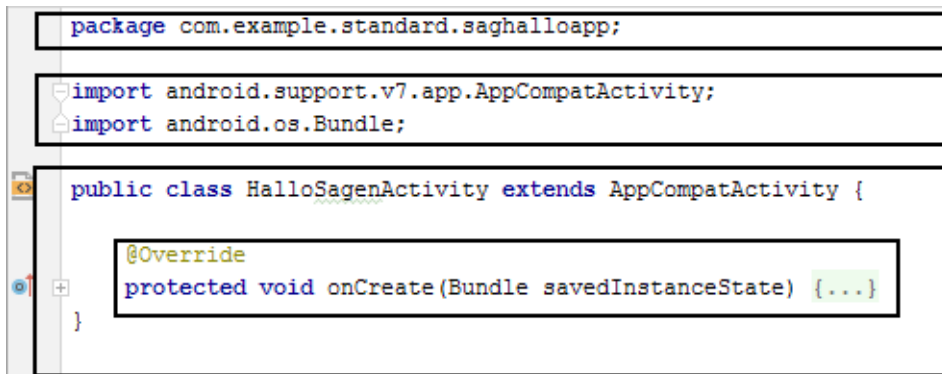


Bild 2.10 Struktur des Code-Grundgerüsts

2.3.1 Die package-Anweisung

In der obersten Zeile treffen wir den Paketnamen wieder, den wir im Dialogfeld **Create New Project** eingegeben hatten.

Die Anweisung

```
package com.example.standard.saghalloapp;
```

sorgt dafür, dass alle nachfolgend definierten Codeelemente – in unserem Grundgerüst wäre dies die Klasse `HalloSagenActivity` – untergeordnete Elemente des Pakets `com.example.standard.saghalloapp` sind.



PAKETE

Die Android-Programmierung verlangt, dass alle Klassen einer App in einem gemeinsamen Paket zusammengefasst werden. Außerdem sollte der verwendete Paketname, der beim Anlegen des Android-Projekts angegeben wird, eindeutig sein, damit es auf den Smartphones nicht zu Namenskonflikten durch Klassen aus gleichlautenden Paketen kommt.

Die gute Nachricht ist, dass Android Studio-Anwender mit der Paketverwaltung kaum aktiv zu tun haben. Sie geben den Paketnamen einfach bei der Einrichtung des Projekts an (siehe Abschnitt 2.2) und um alles Weitere kümmert sich Android Studio.

Mehr zur Codeorganisation mit Paketen erfahren Sie in Kapitel 4 des Java-Tutoriums.

2.3.2 Die import-Anweisungen

Unter der Paketangabe stehen diverse `import`-Anweisungen.

```
import android.support.v7.app.AppCompatActivity  
import android.os.Bundle;
```

Die Namen `android.support.v7.app` und `android.os` sind Pakete aus der Android-Bibliothek. In dieser Bibliothek gibt es zahlreiche Klassen, die wir für die Programmierung unserer Apps verwenden können. Zwei dieser Klassen sind `AppCompatActivity` und `Bundle`: Letztere ist im Paket `android.os` definiert, `AppCompatActivity` im Paket `android.support.v7.app`.