entwickler.press


# Android

## Schnelleinstieg

Stephan Elter, Sven Haiges

Stephan Elter, Sven Haiges

# Android

## Schnelleinstieg

entwickler.press

Stephan Elter, Sven Haiges
Android. Schnelleinstieg

ISBN: 978-3-86802-285-8



Ein Imprint der Software & Support Media GmbH

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

Ihr Kontakt zum Verlag und Lektorat:
Software & Support Media GmbH
entwickler.press
Darmstädter Landstraße 108
60598 Frankfurt am Main
Tel.: +49 (0)69 630089-0
Fax: +49 (0)69 630089-89
lektorat@entwickler-press.de
http://www.entwickler-press.de

Lektorat: Theresa Vögle
Korrektorat: Frauke Pesch
Satz: Dominique Kalbassi
Umschlaggestaltung: Maria Rudi
Titelbild: Der Android Robot steht unter der Creative-Commons-Lizenz.

# Inhaltsverzeichnis

# Vorwort

## Ein Vorwort vor dem Vorwort

Die Entwicklung des Betriebssystems Android schreitet rasant voran, und so hat sich seit der ersten Ausgabe dieses Buches sehr viel verändert. Es wurde Zeit für eine überarbeitete Version. Neben reinen Aktualisierungen, von denen es eine ganze Menge gab (selbst während des Schreibens hat sich noch einiges geändert), wurden mehr Themen und Punkte aufgenommen.

## Jetzt mit noch mehr Einstieg

Android ist nicht nur bei den Anwendern sehr beliebt und erfolgreich, es erfreut sich auch unter Entwicklern einer immer größeren Beliebtheit. Waren es beim ersten Erscheinen dieses Buches nur relativ wenige, die sich mutig mit dieser damals noch neuen Technik auseinandergesetzt haben, interessieren sich immer mehr Entwickler für Android. Darunter auch viele, die vielleicht ursprünglich gar nicht aus der Java-Welt kommen. Wir haben deshalb versucht, dieses Buch breiter aufzustellen und ein klein wenig mehr Einstieg in dem Schnelleinstieg unterzubringen.

## Es gibt kein „zu schwer“ bei Android

Egal, ob Sie erfahrener Java-Profi sind oder vielleicht aus einer ganz anderen Richtung kommen – als Entwickler, der (natürlich) objektorientierte Programmierung kennt und mit Java umgehen kann (oder sich vielleicht gerade parallel einarbeitet) –, werden Sie sehen, dass Android Spaß macht. Es gibt nichts, was wirklich schwer ist. Es gibt viele neue Konzepte, die man kennen lernen und verstehen muss, aber es ist nichts wirklich Kompliziertes dabei – Android macht Spaß! Versprochen!

## Die wunderbare Welt der Beispiele

Wir haben versucht (soweit möglich), mit möglichst kurzen Beispielen zu arbeiten, die man leichter erfassen und somit besser verstehen kann. Sehen Sie es mir deshalb bitte nach, wenn nicht jeder Code einer „strengeren Lehre" der Java-Entwicklung entsprechen sollte. So wurden beispielsweise auch für Bezeichner weitgehend „einprägsamere" (manche würden sagen „seltsame") deutsche Namen gewählt. Nicht jeder mag das, es hilft aber tatsächlich, den Code besser zu verstehen, da man so tatsächlich schneller erfassen kann, welches Android-eigene Funktionen und Methoden sind und was „selbst gebastelt" ist. Willkommen in der wunderbaren Welt der Beispiele!

## Vielen Dank!

Ein solches Buch ist immer die Arbeit von vielen, die zum Teil unerkannt im Hintergrund arbeiten und die genannten Autoren unterstützen. Ich möchte mich deshalb bei allen bedanken, die zu diesem Buch beigetragen haben – sei es durch Hilfe, Ratschläge oder einfach durch Geduld! Bedanken möchte ich mich auch bei Sony, die mir freundlicherweise ein sehr gutes Android-Testgerät zur Verfügung gestellt hatten.

## Viel Spaß und bis dann!

Ich hoffe, Sie werden mit Android und mit diesem Buch genauso viel Spaß haben wie Sven Haiges und ich. Bei Fragen und Kritik finden Sie mich immer über mein Blog unter *www.punktuelles-im-web.net*. Sicher werden Sie dort im Laufe der Zeit die eine oder andere Aktualisierung oder weitere Anregungen zu den Themen in diesem Buch finden.

Vielen Dank und viel Spaß!

Stephan Elter *stephan.elter@googlemail.com*

**Für Andrea und Alva**

## Vorwort von Sven Haiges zur ersten Ausgabe

Das mobile Betriebssystem Android konnte 2010 einen schier unglaublichen Erfolg verzeichnen. Laut Gartner[1] stieg die Zahl der mit Android verkauften Smartphones um 888,8 Prozent. Damit hat Android Ende 2010 bereits Platz 2 der mobilen Betriebssysteme eingenommen. Apples iOS wurde bereits überholt. Auf Platz 1 befand sich 2010 noch Symbian, jedoch wird erwartet, dass 2011 Android auf Platz 1 der mobilen Betriebssysteme vorrücken wird.

Dies ist vor allem den zahlreichen High-End-Produkten von HTC, Samsung und Motorola zu verdanken. Mittlerweile gibt es weltweit kaum einen Mobilfunkanbieter, der es sich leisten kann, kein Android-basiertes Smartphone in seinem Programm zu haben.

Für Entwickler eröffnet sich mit Android eine faszinierende Welt. Dieses Buch möchte Ihnen den Einstieg in die Android-Entwicklung so einfach wie möglich machen.

Eines jedoch gleich vorweg: Android ist ein großes, recht umfassendes Thema und alle paar Monate stellt Google weitere APIs zukünftiger Versionen vor. Für dieses Buch haben wir uns deshalb Bereiche der Android-Entwicklung herausgesucht, die unserer Meinung nach elementar sind. Abgerundet wird dieser Einstieg durch Kapitel zu Maps, Widgets und NFC (Near Field Communication). Wir hoffen, dass wir damit den richtigen Mix aus Grundlangen und faszinierenden Zukunftsthemen gefunden haben.

Dieses Buch setzt allerdings auch einige Grundlagen voraus. Beispielsweise beschreiben wir nicht, wie Sie Eclipse und das Android-Development-Tools-(ADT-)Plug-in installieren[2]. Wir sind der Meinung, dass die meisten Leser diese Grundlagen bereits besitzen.

1 Gartner Mobile Devices Sales 2010: *http://www.gartner.com/it/page.jsp?id=1543014*
2 Android Eclipse Plug-in: *http://developer.android.com/sdk/eclipse-adt.html*

Feedback und Anregungen sind jederzeit willkommen – per E-Mail (*sven.haiges@gmail.com*) oder Twitter *(@hansamann)*.

Viel Spaß bei der Lektüre!

# 1 Eine App in vier Schritten

Bevor wir in den folgenden Kapiteln in die Tiefe gehen und die darin behandelten Themen ausführlich behandeln, wollen wir uns zuerst einen Überblick verschaffen, welche Schritte überhaupt notwendig sind, um eine vollständige App zu erstellen: Von der Einrichtung der notwendigen Software über die grundlegenden (aber umso wichtigeren Entwicklungsschritte), bis hin zum Export und der Veröffentlichung. Ein schneller „Ritt", um Ihnen einen Überblick zu geben, was zu tun ist, um an das Ziel zu kommen – eine App zu programmieren und öffentlich bereitzustellen. Dieses Kapitel stellt deshalb keine klassische Einleitung dar, sondern legt (zumindest gewisse) Grundlagen, die Sie in den folgenden Kapiteln immer wieder benötigen werden.

## 1.1 Java, Installation des JDK

Wir gehen in diesem Buch davon aus, dass Sie als Entwickler objektorientierte Programmierung, Java und die Bedeutung des JDKs kennen. Vermutlich werden die meisten Entwickler, die dieses Buch lesen, Java und insbesondere auch ein aktuelles JDK bereits auf ihrem Rechner installiert haben. Sollten Sie noch kein JDK auf Ihrem Rechner installiert haben, so sollten Sie dies jetzt nachholen. Eine einfache Laufzeitumgebung für Java, ein JRE, ist nicht ausreichend.

Nach der Installation des JDK kann es sinnvoll sein, einige Eintragungen in den Umgebungsvariablen von Windows vorzunehmen: Den Pfad zu dem JDK tragen Sie unter Systemvariablen unter Path in der Form *c:\Java\jdk1.7\bin* ein. Unter den Benutzervariablen erstellen Sie bitte einen neuen Eintrag *JAVA_HOME* mit einem Wert in der Form *C:\Java\jdk1.7*.

Natürlich muss dabei auf die tatsächlich verwendeten Ordner verwiesen werden, die sich von diesen beispielhaften Angaben wahrscheinlich unterscheiden. Bei Problemen kann es übrigens immer sinnvoll sein, in den Systemeinstellungen zu prüfen, ob die richtigen Pfade und Eintragungen vorgenommen wurden. Möglicherweise ist vielleicht auch noch eine wesentlich ältere, bereits vorhandene Version von Java eingetragen?

## 1.2 Download des ADT Bundles – das Android SDK und Eclipse

Musste man früher (nun, allzu lang ist das noch gar nicht her) die Entwicklungsumgebung, in den meisten Fällen Eclipse, für sich alleine herunterladen und danach das Android SDK und das ADT-Plug-in getrennt installieren, so erhält man jetzt auf den Google-Seiten für Entwickler[1] ein komplettes Paket, dass fast alles beinhaltet, was man sich – zumindest für die Android-Entwicklung – wünschen kann. Dafür ist das Downloadpaket aber auch auf beachtliche 500 MB angewachsen. Das so genannte ADT Bundle beinhaltet eine fertig angepasste Version der Entwicklungsumgebung Eclipse sowie das eigentliche Android SDK mit der jeweils aktuellsten Android-Version, dem SDK und dem AVD Manager.

Die Installation beschränkt sich darauf, das gesamte Paket mit einem leistungsfähigen Entpacker auf dem Rechner zu entpacken. Im einfachsten Fall öffnen Sie die Zip-Datei und ziehen den darin enthaltenen Ordner, beispielsweise *adt-bundle-windows-x86-20131030*, auf ein lokales Laufwerk. Wer nicht gerne wartet, sollte die Downloadzeit des Zip-Pakets nutzen, um zumindest unter Windows einen etwas schnelleren Entpacker wie beispielsweise 7-Zip zu installieren – ansonsten sollte man für das Entpacken durchaus etwas mehr Zeit einrechnen.

1 *http://developer.android.com*

Wer das ADT Bundle nicht in dieser Form mit Eclipse verwenden möchte, kann natürlich auch andere IDEs wie beispielsweise NetBeans verwenden. Hinweise zu Konfiguration und Support finden Sie in den meisten Fällen direkt auf Webseiten der jeweiligen IDE oder in den dortigen Foren. Wer hingegen Eclipse bereits installiert hat, kann diese Version natürlich auch weiterhin für die Entwicklung verwenden und das Android SDK und ADT-Plug-in nach wie vor getrennt herunterladen und installieren.

## Ein Wort zum kommenden Star – dem Android Studio

Im Mai 2013 wurde im Rahmen der Google I/O eine neue Entwicklungsumgebung für die Android-Entwicklung vorgestellt: Android Studio basiert auf der Entwicklungsumgebung IntelliJ IDEA der Firma JetBrains. Seit dieser ersten Ankündigung verharrt das Android Studio mit langsam steigenden Versionsnummern nach wie vor recht hartnäckig im Stadium einer „Early Access Preview“. Wann eine erste finale Version erscheint, ist bis heute leider nicht bekannt, Google selbst empfiehlt, im Zweifelsfall noch das ADT Bundle zu verwenden.

Nach Murphys Gesetz wird das Android Studio nach über einem Jahr vermutlich genau mit dem Erscheinen dieses Buchs (oder kurz danach) als finale Version veröffentlicht. Da sich nach Murphys Gesetz und den bisherigen Erfahrungen mit Google aber auch noch sehr viel beim Android Studio bis zu einer finalen Version verändern wird, haben wir uns entschlossen, noch auf Eclipse bzw. das ADT Bundle als Grundlage für dieses Buch zu setzen. Aber keine Angst, auch wenn es natürlich Unterschiede in der Bedienung gibt, man muss tatsächlich nicht ganz von vorne beginnen. Und man darf auch nicht vergessen, dass Eclipse sehr weit verbreitet ist. Auch wenn Eclipse vielleicht nicht das Herz jedes Entwicklers im Sturm erobert hat, hat es dennoch eine sehr breite Basis treuer User. Es ist deshalb auch mit dem Erscheinen des Android Studios nicht davon auszugehen, dass die Android-Entwicklung mit Eclipse am Ende angelangt ist.

## Ein erster Blick in das große Paket – was bietet das ADT Bundle?

Im Kern besteht das ADT Bundle aus drei für den Entwickler wichtigen Paketen: dem SDK Manager, mit dem für die Entwicklung notwendige Android-Versionen heruntergeladen und verwaltet werden können, dem AVD Manager, um virtuelle Android-Geräte erstellen und starten zu können, und natürlich Eclipse. Aus unerfindlichen Gründen darf man sich die Programme zum Starten selbst aus der Ordnerstruktur zusammensuchen. Der SDK Manager ist zumindest anfangs das wichtigste Paket und tatsächlich auch am einfachsten zu finden: direkt im Stammordner des ADT Bundles als ausführbare Datei.


**Abbildung 1.1:** Wenigstens der dickliche SDK Manager ist direkt im Stammordner des ADT Bundle leicht zu finden

Eclipse ist als ausführbare Datei erwartungsgemäß im Unterordner *eclipse* zu finden. Nur der AVD Manager macht es uns da schon etwas schwerer, er ist etwas besser versteckt zu finden unter *sdk\tools\lib*. Glücklicherweise können sowohl der SDK Manager als auch der AVD Manager direkt über Eclipse gestartet werden.

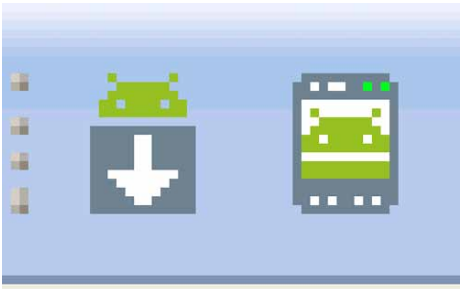

**Abbildung 1.2:** Vielleicht nicht unbedingt schick, aber zumindest gut zu finden: Der SDK Manager und der AVD Manager in der Icon-Leiste von Eclipse

**PROFITIPP:** Sollte es Probleme beim Start eines Managers geben, vielleicht auch nachdem bereits andere Versionen des ADT Bundles installiert waren, hilft hier im Zweifelsfall ein passender Eintrag in den Umgebungsvariablen von Windows in der Form *ANDROID_SDK_HOME* mit der Angabe des aktuellen Pfads wie beispielsweise *C:\adt-bundle-windows-x86*.

## Der SDK Manager und der AVD Manager – ein eingespieltes Team

Der SDK Manager ist für das Herunterladen und die Installation der jeweils gewünschten (oder benötigten) Android-Versionen zuständig, während der AVD Manager für die Einrichtung und Verwaltung unserer virtuellen Smartphones benötigt wird.

## Der SDK Manager – Verwalter der Versionen

Der SDK Manager kann, wie bereits erwähnt, über das entsprechende Icon in Eclipse oder direkt im Stammordner des ADT Bundles als *SDK Manager.exe* gestartet werden. Der SDK Manager ist für den Download

und auch für die Updates aller Komponenten zuständig, die Sie für die Android-Entwicklung benötigen. Im Normalfall sind im ADT Bundle zumindest die allerwichtigsten Komponenten und das aktuellste API bereits enthalten, sodass man eigentlich sofort ohne einen weiteren Download starten könnte. Wer allerdings nicht nur für die jeweils letzte API-Version entwickeln, sondern auch ältere Android-Versionen berücksichtigen und testen möchte, sollte sich die entsprechenden Pakete noch herunterladen. Diese Pakete werden für die Entwicklung und insbesondere für die AVDs, die Android Virtual Devices, benötigt.

Der Download neuer Pakete kann durchaus einige Zeit in Anspruch nehmen, und es kann auch passieren, dass der eine oder andere Download nicht auf Anhieb klappt. Man sollte also immer einen Blick darauf werfen, ob alle gewünschten Pakete tatsächlich heruntergeladen und als *installed* gekennzeichnet sind.

**PROFITIPP:** Grundsätzlich sollte man sich gut überlegen, für welche Versionen man entwickelt. Zu dem Zeitpunkt, zu dem dieses Buch überarbeitet wurde, hatte die Version Gingerbread mit der API-Version 10 noch immer eine Verbreitung von fast 20 Prozent. Google berücksichtigt bei seiner Zählung nur Geräte, die sich tatsächlich im Google Play Store einbuchen und damit als aktivere User angesehen werden. Die genannten 20 Prozent beziehen sich also nicht auf irgendwelche Altgeräte, die vielleicht nur noch zum Telefonieren verwendet werden (dafür sollen Android-Handys ja auch ganz gut geeignet sein) oder unbenutzt in irgendeiner Schublade ihr Dasein fristen.

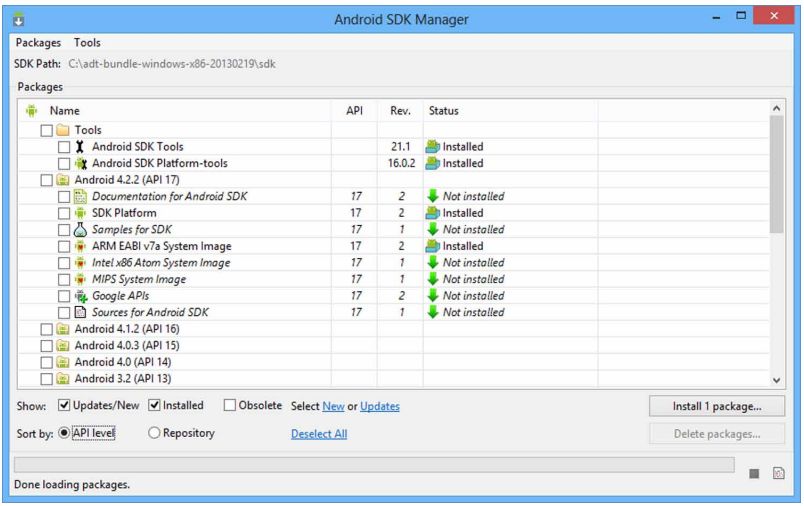


**Abbildung 1.3:** Der SDK Manager – der Download neuer Pakete kann einige Zeit in Anspruch nehmen und klappt auch nicht immer auf Anhieb

## Der AVD Manager – Smartphones auf dem Rechner

Um während der Entwicklung eine App vernünftig testen zu können, benötigt man verschiedene Smartphones und idealerweise auch unterschiedliche Tablets. Da nicht jeder in der glücklichen Lage ist, ein gutes Dutzend Smartphones und Tablets in den Schubladen herumliegen zu haben, kann man wenigstens mithilfe des AVD Managers die unterschiedlichsten Modelle erzeugen und verwalten. Dabei können für jedes AVD, Android Virtual Device, die Android-Version, die Auflösung und die vorhandene (simulierte) Hardware getrennt festgelegt werden. Hilfreich sind dabei die so genannten *Device Definitions*, vorkonfigurierte Basismodelle, aus denen schnell die entsprechenden „Geräte“ in Form tatsächlicher AVDs generiert werden können – eine Art Vorlage. Wenig verwunderlich ist, dass vor den allgemeinen Gerätedefinitionen die Google-eigenen Nexus-Modelle zuerst angeboten werden.

Wenn Sie über den AVD Manager ein neues „Gerät" anlegen (was Sie tatsächlich auch gleich tun sollten), dann können Sie diesen Geräten unabhängig von den Device Definitions beliebige APIs zuordnen – vorausgesetzt, die gewünschte Version ist auch tatsächlich über den SDK Manager installiert worden. Wer sich also wundern sollte, warum er bestimmte APIs nicht zur Auswahl hat, der sollte dies im SDK Manager überprüfen und gegebenenfalls etwas Zeit zum Herunterladen und Nachinstallieren einkalkulieren. Es ist durchaus sinnvoll, AVDs mit verschiedenen APIs (also Android-Versionen) anzulegen, um die eigene App eben nicht nur mit der aktuellsten Version sehen zu können.

Damit das Display des Smartphones (oder Tablets) unabhängig von der Auflösung und Größe des eigenen Monitors in der richtigen Größe angezeigt wird, kann die Darstellungsgröße beim Start unter den Launch Options angegeben und skaliert werden. Auf diesem Weg kann man natürlich auch große Displays ganz bewusst kleiner darstellen, beispielsweise beim Arbeiten an einem Laptop, dessen Display vielleicht nicht so viel Platz bietet.

**PROFITIPP:** Sinnvoll kann es sein, den Snapshot zu aktivieren: Damit startet das virtuelle Gerät zukünftig wesentlich schneller, da es in einer Art Stand-by abgespeichert wird. Bei dem ersten Start eines neuen virtuellen Geräts kann es auch sinnvoll sein, einen eventuell laufenden Virenscanner oder andere ressourcenhungrige Programme kurzzeitig zu deaktivieren. Je nach System kann dies erfahrungsgemäß die doch „etwas" längere Wartezeit verkürzen.

Sollte später im Laufe der Entwicklung ein AVD nicht mehr korrekt funktionieren oder seltsame Ergebnisse liefern, kann es notwendig sein, das virtuelle Gerät zurückzusetzen, indem bei einem erneuten Start des AVDs bei den Launch Options die Option Wipe User Data gewählt wird. Dadurch werden alle Daten und Apps entfernt, und das Gerät ist wieder in seinem Urzustand.

## Smartphones ohne Touchscreen – Bedienung der Android Virtual Devices

Während des Testens und Ausprobierens ist es empfehlenswert, die wichtigen Tastenkombinationen für die Bedienung des virtuellen Smartphones zur Hand zu haben. Es mag im ersten Moment wenig sinnvoll erscheinen, an einem Emulator die Telefonfunktion aufzurufen, doch wer sich die unterschiedlichen Zustände einer Activity (also einer App) ansehen möchte, der kann so problemlos eine Unterbrechung bzw. ein *onPause()* simulieren, dazu aber später mehr.

| Taste, Tastenkombination | Funktion |
|---|---|
| HOME | Homescreen |
| F2 | Linker Softkey, Menü |
| SHIFT + F2 | Rechter Softkey |
| ESC | Zurück |
| F3 | Telefon, wählen |
| F4 | Telefon, beenden |
| F7 | An-, Ausschalter |
| linke STRG + F12 | Drehen, jeweils Wechsel Portrait – Landscape |

Eine gute, vollständigere Übersicht ist unter *http://www.shortcutworld.com/en/win/Android-Emulator.html* oder unter *http://developer.android.com/tools/help/emulator.html* zu finden.

## Eclipse, Start und Konfiguration

Im Ordner des ADT Bundles finden Sie den Unterordner *eclipse*, in dem sich unsere Entwicklungsumgebung in Form einer ausführbaren Datei *eclipse.exe* befindet (zumindest unter Windows in dieser Form). Natürlich kann (sollte) man das Programm dauerhaft für einen schnelleren Zugriff in dem Programmstarter der eigenen Wahl integrieren. Auf Eclipse selbst und seine Bedienung soll an dieser Stelle nicht weiter eingegangen werden, da dies ein zu umfangreiches Thema wäre und wir

einen Schnelleinstieg in die Android-Entwicklung, nicht aber in die Bedienung von Eclipse präsentieren wollen. Aber keine Angst, selbst wenn Sie Eclipse noch nicht kennen sollten – alles Schlechte, was Sie darüber gehört haben, stimmt, und alles Gute übrigens auch. Den einen oder anderen Tipp zur Bedienung werden wir aber dennoch ansprechen. Und bei allen anfangs etwas unüberschaubaren Funktionen ist Eclipse im Kern immer noch ganz gut zu bedienen, wenn Sie bereits mit anderen IDEs oder leistungsfähigen Editoren gearbeitet haben.

**PROFITIPP:** Falls Sie sich doch einmal in Eclipse „verlaufen" haben sollten: Über die Menüleiste Fenster | Perspektive Öffnen | Java gelangen Sie immer wieder zum eigentlichen Bearbeitungsansicht zurück. Und egal, was Sie geschlossen oder verschoben haben, über Fenster | Perspektive zurücksetzen... wird alles wieder auf den ursprünglichen Zustand gesetzt.


**Abbildung 1.4:** Auch wenn „Android Developer Tools" draufsteht – drin ist trotzdem Eclipse

## Die erste App in weniger als fünf Minuten – versprochen

Nachdem wir jetzt die grundlegende Software installiert und konfiguriert haben, wollen wir uns jetzt endlich ansehen, wie man sich die erste eigene App in etwas weniger als fünf Minuten generiert. Natürlich kann

man dabei nicht erwarten, mehr als ein kurzes „Hallo Welt“ in so kurzer Zeit zu erstellen, aber es geht dabei auch nicht um eine bestimmte Funktionalität der App, sondern um den Weg dorthin.

Wenn Sie nach dem ersten Start der Android Developer Tools (trotz des Splash-Screens bleibt es Eclipse) den Workspace (der Bereich, in dem all Ihre Dateien von Eclipse gespeichert werden) auf Ihrem Computer festgelegt haben, können Sie auch schon direkt das erste Projekt beginnen. Mit STRG + N lassen Sie sich alle vorhandenen Assistenten anzeigen und wählen unter dem Punkt Android bitte das Android Application Project. In diesem Assistenten gehen Sie durch die einzelnen Schritte. Geben Sie Ihrer ersten App einen tollen, aussagekräftigen Namen und benennen auch das Projekt eindeutig. Um in der versprochenen Zeit eine erste App zu bekommen, lassen Sie bitte den Punkt Create Activity angewählt und klicken sich dann durch bis zum *finish*.

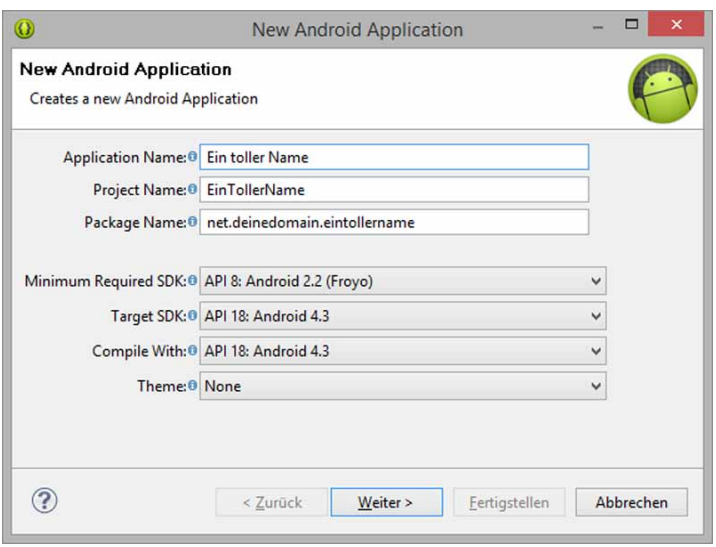


**Abbildung 1.5:** Der Start mit dem „Android Application Project“ – hier können Sie Namen und die unterstützten Android-Versionen festlegen; „Application Name“ und „Project Name“ müssen nicht identisch sein

**PROFITIPP:** Wenn Sie dabei auch den Punkt Create Custom Launcher Icon angewählt lassen, können Sie sehr schnell ein eigenes Icon für Ihre erste App erstellen. Ansonsten wird ein Standard-Icon verwendet. Die Bedienung dieses Assistenten ist tatsächlich recht intuitiv und es macht durchaus Spaß, sich ein eigenes Icon zurechtzubasteln – mit den versprochenen fünf Minuten wird es dann allerdings schon etwas knapp. Das Icon wird automatisch für unterschiedliche Displays in jeweils passenden Größen generiert und im Projekt gespeichert. Falls Sie schon einmal einen Blick darauf werfen wollen (was Sie auch tun sollten): In der Projektstruktur unter *res* finden Sie alle erzeugten Icons in entsprechenden Unterordnern. Bei der Kompilierung werden diese Elemente alle in der erzeugten Installationsdatei gespeichert und das Icon in der passenden Größe später von dem jeweiligen Gerät automatisch zur Darstellung herausgesucht.

Nach dem Abschluss des Assistenten Android Application Project wird eine vollständige App mit einer einzelnen Activity erzeugt. Vereinfacht gesagt besteht eine App aus einer oder mehreren Activities. Activities wiederum sind relativ unabhängige Komponenten, die eine eigene darstellende Oberfläche in Form einer XML-Datei und eine direkt zugehörige Java-Datei für eine mögliche Logik haben (egal, ob das jetzt schon verständlich ist: Das ist eine der wichtigsten Grundlagen von Android, also bitte gedanklich schon einmal vormerken).

**KOMPAKT:** Sie haben auch die Möglichkeit, mit STRG + N einen Assistenten für ein Android Sample Project aufzurufen. Damit können Sie sich aus (meist) funktionstüchtigen Beispielen ein Projekt mit einer vollständigen App, samt aller notwendigen Elemente wie Grafiken, Activities und Quellcode, generieren lassen. Dazu müssen Sie allerdings über den Android SDK Manager für das jeweilige API das zugehörige Paket „Samples for SDK“ installiert haben. Jedes API hat dabei ein eigenes, zugehöriges Paket, das nicht von anderen APIs verwendet werden kann. Und bitte nicht verzweifeln: nicht alle Beispiele sind fehlerfrei!

Links im Paket-Explorer sehen Sie die Dateien und Elemente Ihrer App in einer neu angelegten Projektstruktur. Ihre App ist zu diesem Zeitpunkt bereits funktional, kann kompiliert und auf einem AVD gestartet werden. Dazu müssen Sie in der Symbolleiste nur auf das bekannte grüne Startsymbol für *run* klicken (leicht zu finden neben dem kleinen Käfersymbol) oder alternativ Ihre App über STRG + F11 starten. Eclipse kompiliert Ihre App, erzeugt eine Installationsdatei in der Form *DerNameDerApp.APK*, startet eine AVD, installiert und startet dann dort Ihre App. Wenn Sie noch kein Android Virtual Device angelegt oder zumindest noch nie gestartet hatten, dann kann es mit den versprochenen fünf Minuten je nach Rechnerleistung allerdings schon sehr knapp werden.

**KOMPAKT:** Nach der Erzeugung eines neuen Projekts in Eclipse (bzw. den Android Developer Tools) wird normalerweise nur die XML-Datei geöffnet, die für das Layout der App, genauer gesagt ihrer aktuellen Activity, zuständig ist. Damit Eclipse die App aber tatsächlich kompiliert und dann startet, sollten sie die zugehörige Java-Quelldatei öffnen und im aktiven Editor-Fenster sehen – oder zumindest rechts im Explorer markiert haben. Ansonsten tut sich nicht viel bei dem Versuch, die App zu starten.

## 1.3 Die App signieren, exportieren und veröffentlichen

Jede App muss signiert sein, um auf einem Android-Gerät laufen zu dürfen. Das gilt nicht nur für echte Smartphones, sondern auch für die jedes virtuelle Gerät. Während der Entwicklung müssen wir uns darüber nicht den Kopf zerbrechen, unser SDK vergibt bei der Entwicklung automatisch ein Testzertifikat. Die bei der Entwicklung erzeugte Installationsdatei, eine Datei mit der Endung *apk*, finden Sie in der Projektstruktur in dem Ordner *bin*. Diese APK-Datei könnten Sie direkt verwenden und auf einem belie-

bigen Smartphone oder Tablet installieren. Eine sehr einfache Möglichkeit (ohne ein Kabel zu verwenden) ist es, sich diese Installationsdatei an die eigene Google-Adresse zu mailen. Aus der E-Mail heraus kann man die Installation auf dem jeweiligen Gerät direkt starten, ohne die entsprechende APK erst im Dateisystem des Geräts suchen zu müssen.

**PROFITIPP:** Wichtig dabei ist, dass Sie auf Ihrem Gerät auch die Installation für Anwendungen zulassen, die nicht aus Google Play[2] stammen. Dazu müssen Sie in den Einstellungen Ihres Android-Handys unter ANWENDUNGEN die Installation aus *unbekannten Quellen* zulassen. Nach der Installation können Sie den Haken wieder entfernen, müssen ihn dann aber bei den nächsten Installationen erneut setzen.

## Richtig signieren – der Stempel vom Amt

Wollen Sie Ihre App in einem Market wie Google Play veröffentlichen oder auf anderem Weg einem größeren Publikum zur Verfügung stellen, dann kommen Sie nicht daran vorbei, Ihre App selbst korrekt zu signieren. Sie können alles dafür Notwendige entweder auf der Kommandozeile erledigen oder alternativ (und vor allem wesentlich komfortabler) die Exportfunktion des SDK nutzen, mit der Sie sehr einfach einen notwendigen Keystore anlegen und darin die Schlüssel Ihrer Apps verwalten können.

**PROFITIPP:** Wichtig ist, dass Sie den so erzeugten Schlüssel nicht (nie, niemals!) verlieren. Eine App kann nur dann von Ihnen als Entwickler upgedatet werden, wenn der vergebene Schlüssel bei allen Versionen gleich ist. Unter anderem über diesen Schlüssel erfolgt die eindeutige Identifizierung einer App. Erzeugen Sie – aus welchen Gründen auch immer – einen anderen Schlüssel, dann wird Ihre App nicht mehr als dieselbe erkannt und auch nicht als Update behandelt.

2 Zugang zu Google Play: *https://play.google.com/apps/publish*

Der schnellste Weg (es gibt in Eclipse meist mehrere Wege, etwas zu tun), um eine korrekt signierte App bereitstellen zu können, ist tatsächlich denkbar einfach. Klicken Sie links im Paket-Explorer mit der rechten Maustaste auf den Stammordner des jeweiligen Projekts und wählen aus dem Kontextmenü Android tools und dort Export Signed Application Package.

Schließlich muss entweder ein vorhandener Keystore ausgewählt oder ein neuer erzeugt werden. Ein Keystore ist eine passwortgeschützte Datei, die einen oder mehrere Schlüssel speichern kann. Natürlich sollten Sie diese Datei an einem sicheren Ort und auf einem zuverlässigen Medium speichern (oder zumindest ein Backup davon machen).

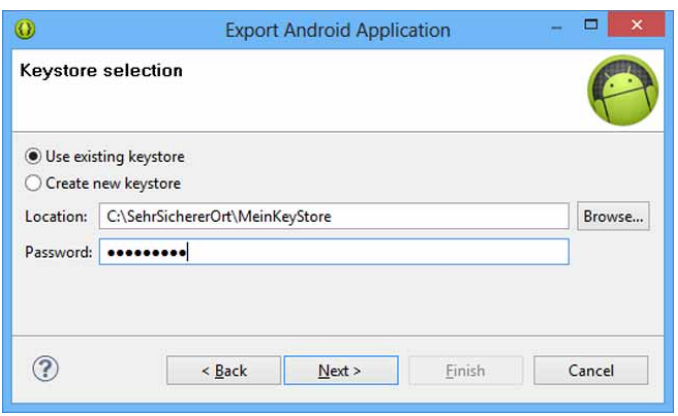


**Abbildung 1.6:** Wählen Sie einen sicheren Ort für Ihren Keystore – und vor allem: Vergessen Sie das Kennwort nicht

Im nächsten Schritt erzeugen Sie einen neuen Schlüssel oder wählen den passenden aus, wenn Sie zuvor bereits einen Schlüssel für diese App erzeugt haben. Falls Sie einen neuen Schlüssel erzeugen, müssen Sie im Folgenden einige Angaben zu diesem Schlüssel bzw. zu Ihrer App machen. Sie sollten einen passenden Alias für Ihren Schlüssel wählen, damit Sie diesen auch nach längerer Zeit noch korrekt zuordnen können. Die