O'Reillys Taschenbibliothek

2. Auflage
Behandelt Android 4.3

Android Programmierung

kurz & gut

O'REILLY®

Jörg Staudemeyer

2. AUFLAGE

Android-Programmierung

kurz & gut

Jörg Staudemeyer

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Sebastopol · Tokyo

Kommentare und Fragen können Sie gerne an uns richten:
O'Reilly Verlag
Balthasarstr. 81
50670 Köln
E-Mail: kommentar@oreilly.de

1. Auflage 2012
2. Auflage 2013

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

Lektorat: Inken Kiupel, Köln
Fachliche Unterstützung: Aron Homberg, Eching
Korrektorat: Eike Nitz, Köln
Produktion: Andrea Miß, Köln
Umschlaggestaltung: Michael Oreal, Köln
Satz: Reemers Publishing Services GmbH, Krefeld; www.reemers.de
Druck: fgb freiburger graphische betriebe; www.fgb.de

ISBN 978-3-95561-463-8

Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

Inhalt

Teil II: Mit Android programmieren

Teil III: Referenz

Vorwort

Android ist eine Plattform für die Erstellung und den Betrieb von Anwendungen auf (mehr oder weniger) mobilen Geräten wie Smartphones, Tablet-Computern, E-Book-Readern, Fernsehern und vielen anderen Dingen, von denen wir heute vielleicht noch gar keine Vorstellung besitzen. Android wird als Open Source-Projekt von der *Open Handset Alliance* (*http://www.openhandsetalliance.com/*) betrieben, der neben dem Suchmaschinenbetreiber Google als treibender Kraft auch zahlreiche Mobilnetzbetreiber und Marketingfirmen sowie Software-, Chip- und Gerätehersteller angehören. Aufgrund des boomenden Marktes für die typischen Geräte, auf die Android zugeschnitten ist, bekommt diese Plattform eine immer größere Bedeutung, und dadurch wendet sich auch eine immer größer werdende Zahl von Entwicklern diesem spannenden und zukunftsträchtigen Thema zu.

Android-Programmierung – kurz & gut wendet sich an Programmierer, die sich das für die Entwicklung von Android-Apps erforderliche Grundlagenwissen schnell aneignen möchten, ohne Lehrbücher durchzuarbeiten, die Hunderte von Seiten stark sind. Das Büchlein führt Sie kurz und knapp in die wichtigsten technischen Zusammenhänge und Vorgehensweisen ein und bietet eine Kurzreferenz für die wichtigsten zur Android-Entwicklungsumgebung gehörenden Begriffe, Definitionen, APIs usw. Dünn, wie es ist, kann es keine umfassende Entwicklerdokumentation sein – um Einzelheiten und fortgeschrittene Anwendungsmöglichkeiten nachzuschlagen, werden Sie auch weiterhin auf die ausführliche Dokumentation auf der Android-Website (*http://developer.android.com*) zurückgreifen müssen. Aber dieses Buch kann Ihnen einen schnellen Einstieg in das Thema

sowie einen Überblick über die Möglichkeiten bieten, die Ihnen beim Erstellen von Anwendungen für diese Plattform zur Verfügung stehen.

Darüber hinaus kann dieses Buch auch keine Einführung in elementare Grundlagen sein. Um es erfolgreich nutzen zu können, sollten Sie daher über gewisse Vorkenntnisse verfügen (die Sie sich natürlich anhand umfangreicher verfügbarer Literatur aneignen können). In erster Linie gehört dazu ein solides Grundwissen über die Java-Programmierung (Java SE) im Speziellen und die moderne Rechner-, Netzwerk- und Datenbanktechnik im Allgemeinen. Daneben sollten Sie sich aber auch schon etwas in der Benutzung von Android-Geräten auskennen (so ist beispielsweise der Besitz eines einfachen, aber halbwegs aktuellen Android-Telefons zum Ausprobieren durchaus zu empfehlen). Und schließlich können einige Erfahrungen mit der Java-Entwicklungsumgebung Eclipse nicht schaden.

Die Aktualität des Themas *mobile computing* bringt es mit sich, dass Android permanent um neue Fähigkeiten erweitert wird, was sich im regelmäßigen Erscheinen neuer Versionen des Betriebssystems äußert. Beim Verfassen der zweiten Auflage dieses Buchs ist die Android-Version 4.3 (»Jelly Bean«) aktuell, und auf diesen Stand beziehen wir uns auch. Dabei weisen wir aber auch auf Änderungen und Erweiterungen der Bibliotheken hin, die seit der relativ weit verbreiteten Version 2.2 (»Froyo«) eingeführt worden sind, damit Sie auch auf die Benutzer älterer Geräte Rücksicht nehmen können (siehe auch den Abschnitt »Versionsgeschichte« in Kapitel 1).

Das Buch gliedert sich in drei Teile. Der erste Teil umfasst eine Einführung in die grundlegenden Techniken von Android: die Architektur, die zugehörige Entwicklungsumgebung, die prinzipielle Funktionsweise von Android-basierten Anwendungen und den Entwicklungsprozess. Im zweiten Teil beschäftigen wir uns mit dem Android-Framework, den Möglichkeiten, die es uns zur Verfügung stellt, und damit, wie man mit diesen umgeht. Der dritte Teil enthält schließlich eine Reihe von Übersichten zu weiterführenden Möglichkeiten und zum Nachschlagen bei der täglichen Arbeit.

Wenn das Thema Android-Programmierung für Sie noch neu ist, lesen Sie am besten die beiden ersten Teile komplett, um sich das nötige Hintergrundwissen für die Arbeit mit Android zu erarbeiten. Andernfalls können Sie auf den gesamten Inhalt des Buchs entsprechend Ihren Vorkenntnissen und den Anforderungen Ihres aktuellen Projekts zugreifen.

Wenn man beginnt, sich mit der Anwendungsentwicklung unter Android zu beschäftigen, trifft man auf eine Menge Begriffe, die entweder Android-spezifisch sind oder die im Android-Umfeld abweichend von anderen Zusammenhängen verwendet werden. Um Ihnen den Einstieg zu erleichtern, finden Sie unter *http://www.oreilly.de/catalog/androidprogpr2ger/chapter* ein Glossar, in dem die wichtigsten Begriffe aufgelistet und kurz erläutert werden.

TEIL I

Android-Grundlagen

Die Plattform

Android ist eine Kombination aus Betriebssystem und Anwendungs-Framework, ausgerichtet auf die besonderen Anforderungen und Möglichkeiten von mobilen Geräten wie Tablets und Smartphones. In diesem Kapitel zeigen wir, aus welchen Schichten die Architektur dieser Plattform aufgebaut ist und wie auf ihrer Basis Anwendungen betrieben werden.

Architektur

Die Android-Plattform besteht aus eine Fülle unterschiedlicher Bestandteile, die man sich am besten als eine Reihe von Schichten mit unterschiedlichen Aufgaben vorstellt. Die meisten dieser Komponenten bestehen aus Open Source-Produkten, die zum großen Teil unabhängig von Android entstanden sind. Einige sind allerdings auch eigens für Android entwickelt worden – sei es, weil sie spezifisch für die Geräte sind, auf denen Android verwendet wird, oder weil vorhandene Open Source-Lösungen aus lizenzrechtlichen Gründen nicht eingesetzt werden können.

Abbildung 1-1 zeigt eine Übersicht der verschiedenen Schichten der Plattform.

Ebene 1: Linux-Kernel
: Die unterste Schicht des Betriebssystems bildet ein Linux-Kernel. Er stellt elementare Systemfunktionen wie die Prozessverwaltung, die Benutzer- und Rechteverwaltung, den Dateizugriff usw. zur Verfügung. Diese Schicht dient auch der Hardware-Abstraktion, hier befinden sich also die gerätespezifischen Trei-

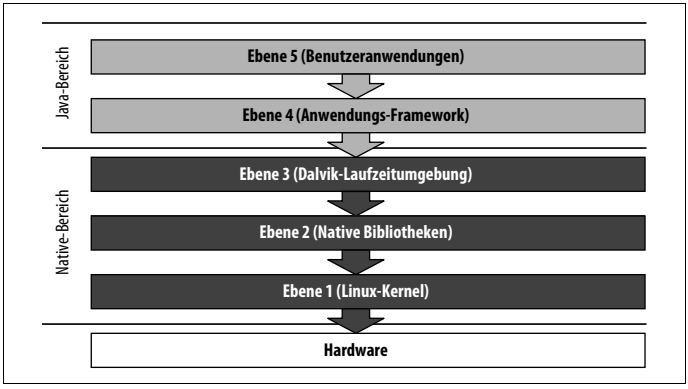

Abbildung 1-1: Schichten der Android-Plattform

ber der Hardware-Hersteller, die einen einheitlichen Zugriff auf divergierende Hardware-Plattfomen ermöglichen.

Eine Besonderheit des in Android eingesetzten Linux-Kernels besteht darin, dass sich dort anstelle der standardmäßig verwendeten GNU-C-Systembibliothek eine eigene Bibliothek befindet, die besser auf die Erfordernisse mobiler Geräte und das Android-Lizenzmodell abgestimmt ist.

Ebene 2: Native Bibliotheken

Die nächste Schicht besteht aus einer Reihe von C- bzw. C++-Bibliotheken. Sie bieten Basisfunktionen, die über die Funktionen des Linux-Kernels hinausgehen und von den darüberliegenden Schichten verwendet werden. Dazu gehören die folgenden Bibliotheken:

- *SQLite* ist ein kompaktes, aber leistungsfähiges relationales Datenbanksystem, dessen Datenbanken jeweils aus einer einzigen Datei bestehen.
- *LibWebCore* ist eine sehr schnelle HTML-Rendering-Engine, die beispielsweise auch im Webbrowser *Google Chrome* eingesetzt wird.

- Das *Media Framework* umfasst Bibliotheken auf der Basis von OpenCORE, mit deren Hilfe zahlreiche Bild-, Video- und Audioformate wiedergegeben und aufgenommen werden können.
- *SGL* und *OpenGL* sind Engines für die Darstellung von 2-D- und 3-D-Grafiken.
- *FreeType* ist eine Bibliothek für die Darstellung von Bitmap- und Vektorzeichensätzen.

Ebene 3: Dalvik-Laufzeitumgebung

Die Laufzeitumgebung für die in Java geschriebenen Programme der oberen Systemschichten besteht aus einer speziellen virtuellen Maschine und den zugehörigen Plattformbibliotheken.

Die virtuelle Maschine Dalvik läuft ebenfalls auf der Basis des Linux-Kernels und dient dazu, die in einer speziellen Bytecode-Variante kodierten Java-Programme der Android-Apps auszuführen.

Die Dalvik-VM unterscheidet sich in technischer Hinsicht von der standardmäßigen Java-VM und verarbeitet nur eine spezielle Form von Maschinenprogrammen, die sich von normalem Java-Bytecode grundsätzlich unterscheiden. Das *Dalvik Executable Format* (DEX) wird nicht mithilfe eines Compilers direkt aus Java-Quellprogrammen erzeugt. Vielmehr übersetzt man die Java-Programme zunächst in normalen Java-Bytecode, aus dem dann vom Hilfsprogramm *dx* des Android-Toolkits der DEX-Code generiert wird.

Der Vorteil dieses Vorgehens besteht darin, dass die im Laufe der Zeit immer wieder vorgenommenen Änderungen und Erweiterungen der Java-Sprachsyntax nicht nachvollzogen werden müssen, solange sich der Java-Bytecode nicht verändert (was tatsächlich über lange Zeit hinweg der Fall war). Prinzipiell könnte man sogar eine andere Programmiersprache verwenden, sofern deren Compiler gültigen Java-Bytecode erzeugt.

Die Plattformbibliotheken basieren auf den Ergebnissen des (inzwischen eingestellten) Projekts *Apache Harmony* (*http://*

harmony.apache.org/) und entsprechen weitgehend den Standardbibliotheken der Java-SE-Version (nicht Java-ME). Ein wesentlicher Unterschied zu Java-SE besteht darin, dass beispielsweise die Bibliotheken für Benutzerschnittstellen (AWT und Swing) entfernt und durch Android-spezifische APIs ersetzt worden sind.

Ebene 4: Anwendungs-Framework
Das Anwendungs-Framework von Android stellt eine Fülle von Grundfunktionen zur Verfügung, mit deren Hilfe Anwendungen die Features ihrer Geräte nutzen sowie eine einheitliche Darstellung und ein Höchstmaß an Effizienz und Wiederverwendung erreichen können. Dazu gehören unter anderem die folgenden Dinge:

- Manager für verschiedenste aktive Elemente des Frameworks, z.B. Activities, Datenbanken, Accounts usw.
- Eine große Anzahl von Bildschirmelementen (Views) für die Darstellung von Informationen in unterschiedlicher Form und Anordnung
- Basisklassen für die verschiedenen Arten von Komponenten, aus denen Sie Ihre Anwendungen zusammensetzen können
- Diverse APIs für die Kommunikation zwischen Anwendungen untereinander und mit der Außenwelt sowie für die Nutzung der verschiedenen Gerätefunktionen wie Fotografie und Telefonie.

Alle Teile des Anwendungs-Frameworks sind in Java programmiert und liegen im DEX-Format vor, auch wenn die eigentliche Funktionalität häufig durch den Linux-Kernel oder die nativen Bibliotheken zur Verfügung gestellt wird.

Ebene 5: Benutzeranwendungen (Apps)
Die oberste Schicht des Android-Systems bilden die Benutzeranwendungen; sie umfassen die gesamte für die Anwender sichtbare Funktionalität. Dazu gehört eine Reihe eingebauter Apps, durch die die Grundfunktionen des Geräts zur Verfügung gestellt werden, sowie alle Erweiterungen, die über ein App-Repository wie *Google Play* oder auch manuell installiert

worden sind. Alle Benutzeranwendungen sind im Prinzip gleichberechtigt und als lose gekoppelte Pakete installiert, können also im Prinzip durch alternative Pakete ersetzt werden.

Android-Apps werden grundsätzlich in Java programmiert und »sehen« nur die Java-Klassen des Anwendungs-Frameworks, das alle unterliegenden Systemschichten komplett kapselt. Es gibt allerdings auch eine Möglichkeit, mithilfe des *Native Development Kit* von Android (NDK) in C bzw. C++ (oder einer anderen Sprache) programmierten Binärcode einzubetten.

Die Installation einer Android-App erfolgt in Form eines sogenannten *Android Application Package* (APK). Dabei handelt es sich um eine JAR-Archivdatei mit der Dateiendung *.apk*, die eine genau festgelegte Struktur aufweisen muss. Die APK-Dateien werden komplett von der Android-Entwicklungsumgebung zusammengebaut, daher müssen wir uns nicht mit den Einzelheiten in ihrem Inneren auseinandersetzen.

In diesem Buch werden wir uns ausschließlich auf den Ebenen 4 und 5 bewegen: Wie Benutzeranwendungen im Einzelnen funktionieren, wie man sie programmiert und welche Unterstützung durch das Framework es gibt.

Versionsgeschichte

Android hat in seiner kurzen Geschichte bereits eine Reihe von Release-Ständen erreicht, die mit jeweils neuen oder veränderten Funktionalitäten verbunden sind. Da ein Versionsupdate nicht bei allen Geräten möglich ist, muss man damit rechnen, dass auch die alten Android-Versionen noch vielfältig genutzt werden. Anwendungsprogrammierer müssen sich daher immer überlegen, inwieweit sie Features der neueren Plattform nutzen wollen, wenn sie dadurch Nutzer alter Android-Versionen ausschließen.

Jede Android-Plattform wird wie üblich durch eine Versionsnummer (wie beispielsweise »2.3«) gekennzeichnet, und in der Regel zusätzlich auch durch einen Codenamen wie »Froyo« oder »Gingerbread« (seit der Version 1.5 sind dies Bezeichnungen beliebter

amerikanischer Süßigkeiten). Daneben gibt es eine gesonderte, durchgängig ganzzahlige Nummerierung der zu jeder Version gehörenden API-Levels, also die Versionsstände der Bibliotheken, auf die sich deren Dokumentationen häufig beziehen.

Tabelle 1-1 führt die wichtigsten noch relevanten Android-Versionen auf. Genannt sind jeweils die Versionsnummer, das API-Level, der Codename (entsprechend den Konstanten in `android.os.Build.VERSION_CODES`) sowie der Freigabemonat.

Tabelle 1-1: Android-Versionen

Plattform	Level	Codename	Freigabe
1.0	1	BASE	10.2008
1.1	2	BASE_1_1	02.2009
1.5	3	CUPCAKE	05.2009
1.6	4	DONUT	09.2009
2.0	5	ECLAIR	11.2009
2.0.1	6	ECLAIR_0_1	12.2009
2.1	7	ECLAIR_MR1	01.2010
2.2	8	FROYO	06.2010
2.3	9	GINGERBREAD	11.2010
2.3.3	10	GINGERBREAD_MR1	02.2011
3.0	11	HONEYCOMB	02.2011
3.1	12	HONEYCOMB_MR1	05.2011
3.2	13	HONEYCOMB_MR2	06.2011
4.0	14	ICE_CREAM_SANDWICH	10.2011
4.0.3	15	ICE_CREAM_SANDWICH_MR1	11.2011
4.1	16	JELLY_BEAN	06.2012
4.2	17	JELLY_BEAN_MR1	10.2012
4.3	18	JELLY_BEAN_MR2	07.2013

Die jeweils aktuelle Android-Entwicklungsumgebung ermöglicht Ihnen, auch Anwendungen für frühere Versionen zu erstellen. Beispielsweise können Sie festlegen, dass Ihre Anwendung mindestens das API-Level 8 (d. h. Android 2.2, »Froyo«) voraussetzt. Sie sollte dann auch auf neueren Versionen laufen, da diese in der Regel weitgehend abwärtskompatibel sind; auf älteren Geräten ist sie jedoch noch nicht einmal installierbar. Mithilfe der zur Entwicklungsumgebung gehörenden *Android Support Library* ist es allerdings auch möglich, bestimmte Funktionen, die erst mit neueren Plattformen eingeführt worden sind, auf Systemen mit einem früheren Versionsstand zu nutzen (siehe den Abschnitt »Hinweise zur Programmierung« in Kapitel 4).

In jedem Fall sollten Sie Ihre Anwendungen immer auch mit der neuesten Android-Plattform testen, um sicherzustellen, dass sie auch auf aktuellen Geräten funktionieren und benutzbar sind.

Um Ihnen die Einschätzung zu erleichtern, auf welche früheren Android-Plattformen Sie noch Rücksicht nehmen sollten, finden Sie unter *http://developer.android.com/about/dashboards/index.html* eine Übersicht der aktuellen Marktanteile, die regelmäßig anhand der Zugriffe auf Google Play ermittelt werden. Gegenwärtig (August 2013) zeigt sie, dass Anwendungen, die mit Android 2.2 und höher kompatibel sind, fast 99 Prozent des gesamten Gerätebestands abdecken.

Unter *http://developer.android.com/guide/appendix/api-levels.html* können Sie die Release-Notes der wichtigeren aktuellen und früheren Plattformversionen abrufen, um zu erfahren, in welchen Versionen welche neuen Funktionen eingeführt worden sind. Eine aktuelle Zusammenfassung der Versionsgeschichte finden Sie auch in der Wikipedia unter *http://de.wikipedia.org/wiki/Android_(Betriebssystem)*.

Android-Apps

Es gibt grundlegende Unterschiede zwischen einem normalen Java-Programm, für das es außer der obligatorischen `main()`-Methode

wenig Verpflichtendes gibt, und einer Android-Anwendung. Android definiert eine Fülle von Regeln und Konventionen, die eingehalten werden müssen, damit sich ein Programm überhaupt auf einem Gerät installieren und starten lässt. So ist auch genau festgelegt, aus welchen Bestandteilen eine Android-Anwendung bestehen kann und wie diese Bestandteile untereinander kommunizieren.

Bestandteile

Die Grundbestandteile einer Android-App sind die Java-Programme, in denen die Funktionalität definiert ist, die Ressourcen mit ergänzenden Daten und das Manifest, in dem die ganze Anwendung beschrieben ist.

Java-Programme

Eine Android-Anwendung wird standardmäßig durch Java-Programme realisiert, die in der Dalvik-VM ausgeführt werden. Sie enthält mindestens eine – meistens aber mehrere – zentrale Klassen, die als *Komponenten* bezeichnet werden. Diese implementieren die Kernfunktionalität der Anwendung und kommunizieren mit dem Benutzer und der Systemumgebung.

Daneben kann es in einer Android-App natürlich auch weitere Java-Klassen geben, die Daten oder Funktionen bündeln oder für die Kommunikation mit dem System benötigt werden. Ihre Verwendung unterscheidet sich prinzipiell nicht von der anderer Java-Anwendungen.

Auch komplette Fremdbibliotheken (JARs) können in einer Anwendung enthalten sein, sofern sie kompatibel zur Android-Laufzeitumgebung sind (siehe auch Kapitel 3, *Der Entwicklungszyklus*).

Ressourcen

Mit Ressourcen sind diverse Arten unveränderlicher Daten wie Bilder, Medien und Texte gemeint, die von einer Anwendung benötigt werden, aber nicht zu den Programmen gehören.

Ressourcen haben entweder die Form von Dateien bestimmter Typen in bestimmten Verzeichnissen oder die Form von Elementen in XML-Dateien. Android stellt Methoden zur Verfügung, mit deren Hilfe die Ressourcen einfach in die Anwendung eingebunden werden. Dabei gibt es die Möglichkeit, automatisch zwischen verschiedenen *Locales* oder Geräteausstattungen zu differenzieren. So können Sie beispielsweise sprachabhängige Beschriftungen für Ihre Anwendung festlegen, die automatisch entsprechend der Spracheinstellung des Benutzers ausgewählt werden.

Näheres zum Umgang mit Ressourcen finden Sie in Kapitel 6, *Ressourcen*.

Das Manifest

Zu jeder Android-Anwendung gehört eine XML-Datei mit dem Namen *AndroidManifest.xml*, in der alle Komponenten sowie verschiedene andere Eigenschaften der Anwendung beschrieben sind. Diese Datei enthält alle Informationen, die von der Entwicklungsumgebung für die Erzeugung, von Google Play und anderen App-Repositories für die Publikation sowie vom Betriebssystem für die Installation und den Betrieb benötigt werden. Dazu gehören unter anderem die folgenden Daten:

- Bestimmte Anforderungen an das Gerät, auf dem die Anwendung installiert werden soll, z. B. die Mindestversion von Android, die Bildschirmgröße, das Vorhandensein eines GPS-Empfängers usw.
- Die Berechtigungen, die von der Anwendung benötigt werden, damit sie betrieben werden kann, z. B. Zugriff auf den aktuellen Standort, die Nutzung der Kamera usw.
- Die in der Anwendung enthaltenen Komponenten und deren spezifische Eigenschaften.

Funktionsweise

Android-Anwendungen laufen typischerweise auf mobilen Geräten, die spezielle Anforderungen an die auf ihnen betriebenen Pro-

gramme stellen. Dazu gehören begrenzte physische Ressourcen, eine Vielzahl von eng miteinander verwobenen Anwendungen aus unterschiedlichen Quellen, aber auch die Notwendigkeit, auf bestimmte Ereignisse (wie etwa eingehende Telefonanrufe) unvermittelt reagieren zu können.

Um unter diesen Bedingungen einen sicheren Betrieb zu gewährleisten, müssen die Anwendungen in einer bestimmten Art und Weise betrieben werden.

Betrieb in der Sandbox

Jede Android-Anwendung wird in einer eigenen sogenannten *Sandbox* (also in einer weitgehend isolierten Umgebung) betrieben, in der sie nur die Rechte hat, die sie für ihre spezifischen Aufgaben benötigt.

Jede Anwendung läuft in der Regel unter ihrem eigenen Linux-Benutzerkonto, von dem allerdings weder Anwender noch Anwendungsprogrammierer etwas wahrnehmen. Die Zugriffsberechtigungen aller zu einer Anwendung gehörenden Dateien im Systemspeicher werden standardmäßig so eingestellt, dass nur das eigene Benutzerkonto darauf zugreifen kann.

Sofern eine Anwendung nicht gerade in Entwicklung ist und über die Entwicklungsumgebung gestartet wird, lässt sie sich nur installieren, wenn sie mit einem Zertifikat des Herstellers signiert ist. Das Zertifikat muss nicht unbedingt von einer offiziellen Zertifizierungsstelle ausgestellt worden sein (und ist es in der Regel auch nicht), denn es dient nur dazu, Anwendungen verschiedener Herkunft sicher voneinander zu trennen und ihre Quelle zu bestimmen.

Man kann auch mehrere Anwendungen so einrichten, dass sie gemeinsam unter demselben Linux-Konto laufen und gegenseitig auf ihre Dateien zugreifen können. Das erfordert aber, dass sie mit demselben Zertifikat signiert sind – also praktisch aus derselben Quelle stammen.