jetzt lerne ich

Android 4

Programmierung

Markt+Technik

DIRK LOUIS / PETER MÜLLER

- Grafische Benutzeroberflächen erstellen
- Mehr als 50 fertige Beispiel-Apps entwickeln
- Mit der Eclipse-Entwicklungsumgebung arbeiten
- Alle Tools, Installationsanleitung und 300-seitiges Java-Tutorial auf DVD

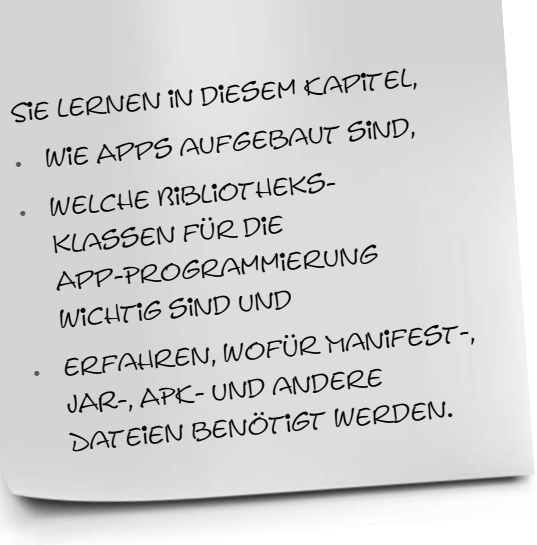
ALWAYS LEARNING

PEARSON

Inhaltsübersicht

Inhaltsverzeichnis

3 Was wann wofür



Nachdem die ersten Schritte getan sind, die erste App erstellt und hoffentlich auch erfolgreich im Emulator getestet wurde, wir also die ersten Hürden erfolgreich genommen haben – und dies gilt ganz besonders für diejenigen Leser, die nebenbei auch noch das Java-Tutorium auf der Buch-DVD durchgearbeitet haben –, werden wir in diesem Kapitel eine kurze Zwischenpause einlegen, die Entwicklerwerkzeuge und den Compiler für eine Weile ruhen lassen und diese Unterbrechung dazu nutzen, uns geistig auf die nächsten Aufgaben vorzubereiten.

Konkret werden wir uns eine Übersicht darüber verschaffen, was bei der App-Programmierung eigentlich von uns erwartet wird, mit welchen Komponenten (Android-Klassen) wir es dabei zu tun haben und wie Android-Projekte im Detail aufgebaut sind. Wir werden dabei etlichem Bekannten begegnen, aber auch viel Neues entdecken.

3.1 Was ist zu tun? – Die drei Pfeiler der App-Erstellung

Zur App-Programmierung gehört, dass Sie

- den **Code schreiben**, der festlegt, wie sich die App verhält,
- das **Layout festlegen**, das bestimmt, wie die App auf dem Anzeigegerät dargestellt wird,
- die **Ressourcen bereitstellen**, die für die Anzeige und Funktion der App benötigt werden.

Haben Sie die Aufgabenkomplexe wiedererkannt? Es sind die gleichen Aufgaben, die wir bereits in Kapitel 2.3 und 2.4 angesprochen haben.

Der Code ist naturgemäß das eigentliche, ureigene Metier des Programmierers. Wie das Codegerüst einer App aussieht, haben Sie ja bereits in Kapitel 2.3 gesehen. In Kapitel 4 werden wir uns etwas näher mit dem Code befassen, ein paar Fingerübungen machen und uns vor allem ansehen, wie uns Eclipse bei der Codebearbeitung unterstützt.

App-Benutzeroberflächen werden üblicherweise über XML-Layoutdateien definiert (siehe Kapitel 2.4). Da sie für den Erfolg und die Bedienbarkeit einer App von großer Bedeutung sind, werden wir uns in Kapitel 5 etwas ausführlicher mit ihnen beschäftigen. Wir werden uns in den XML-Code einarbeiten und uns ansehen, wie uns der Eclipse-Designer bei der visuellen Bearbeitung der Layouts unterstützt, und uns nebenbei mit Hintergrundbildern, Orientierungen und App-Symbolen befassen.

Apps arbeiten viel mit Ressourcen: Bilder, anzuzeigende Texte und Beschriftungen, Farben, Stile, Mediendateien etc. In Kapitel 6 werden wir die verschiedenen Ressourcentypen vorstellen. Vor allem die Arbeit mit Strings und Bildern, inklusive der Unterstützung unterschiedlicher Geräteauflösungen, werden wir etwas eingehender betrachten.

3.2 Wer hilft uns? – Bausteine und Klassen

Android-Apps werden in der Umgebung eines Android-Betriebssystems ausgeführt. Das Betriebssystem stellt, unterstützt von passender Hardware, den Apps viele interessante Optionen zur Verfügung (wie z.B. das Abspielen von Sounddateien, das Aufnehmen von Fotos, das Starten fremder App-Komponenten, das Speichern von Dateien etc.), stellt umgekehrt aber auch Anforderungen an die Apps (vorgegebener Aufbau, vorgegebene Kommunikationswege, DEX-Bytecode etc.).

Um dem Programmierer die Arbeit zu erleichtern, sodass er mit möglichst geringem Aufwand Android-konforme Apps erstellen und die vielen technischen Möglichkeiten nutzen kann, stellt uns Google die Klassen der Android-Bibliothek zur Verfügung.

3.2.1 Bausteine für den App-Aufbau

Apps bestehen aus diversen Bausteinen, hinter denen naturgemäß Klassen aus der Android-Bibliothek stehen. Sehen wir uns einige dieser Bausteine etwas genauer an.

Activities

Während Windows-Anwendungen üblicherweise **ein** Hauptfenster besitzen, in dem der Anwender **eine Vielzahl** von Aktionen durchführen kann, bestehen Apps aus **einer oder mehreren** Bildschirmseiten, die jede **einer** bestimmten Aktivität gewidmet sind. Womit wir beim Thema Aktivitäten, oder, wie der Android-Programmierer auch sagt, »Activities« wären.

Eine Activity ist eine Kombination aus Bildschirmseite und zugehörigem Code. Eine Activity sollte einer in sich abgeschlossenen Aufgabe (Aktivität) gewidmet sein, sie wird als Klasse implementiert, die von der Bibliotheksklasse `android.app.Activity` abzuleiten ist, und sie muss in der Manifestdatei der App aufgeführt werden.

Eine Besonderheit der App-Activities ist, dass sie in sich geschlossene Bausteine darstellen und nur lose an ihre App gebunden sind. Dies hat zwei Konsequenzen:

- Eine Activity kann grundsätzlich von jeder App auf dem Android-Gerät aufgerufen werden.
- Für den Aufruf von Activities gibt es einen globalen Aufrufmechanismus: der Aufruf über Intents. Diesen Mechanismus müssen Sie verwenden, gleichgültig, ob Sie eine Activity der eigenen oder einer fremden App aufrufen möchten.

Intents

Betrachtet man Apps als lose verbundene Activities, so sind es die Intents, zu Deutsch »Absichten«, die die lose Verbindung zwischen den Activities herstellen.

Wiederverwendbare Komponenten

Ist es nicht seltsam, eine Anwendung als eine Sammlung eigenständiger binärer Software-Komponenten zu definieren? Ganz und gar nicht! Microsoft betreibt seit Jahren einen enormen Aufwand, um über diverse Technologien (COM, DCOM, COM+, .NET Framework) seiner Windows-Entwicklergemeinde das zu bieten, was Android von vornherein mitbringt: die einfache anwendungsübergreifende Wiederverwendung von auf dem System installierten Software-Bausteinen.

Konkret bedeutet dies: Wenn Sie aus einer Activity heraus eine andere Activity starten möchten, müssen Sie

- einen Intent erzeugen, der angibt, welche Activity aufzurufen ist und welche Daten dieser Activity gegebenenfalls übergeben werden sollen,
- den Intent mit einer passenden Android-Methode abschicken.

Das Android-System empfängt den Intent und sucht nach der auszuführenden Activity. Gibt es auf dem System eine Activity, die zur Beschreibung in dem Intent passt, wird die Activity gestartet.

Tauchen wir noch ein wenig tiefer in den Intent-Mechanismus ein. Grundsätzlich gibt es zwei Wege, einen Intent zu adressieren:

- als expliziten Intent – in diesem Fall wird als Adressat die Activity angegeben, die aufgerufen werden soll.
- als impliziten Intent – in diesem Fall wird als Adressat keine konkrete Activity angegeben. Stattdessen werden bestimmte Informationen über die gewünschte Aktion mitgeliefert (*action*, *type* und *category*), und das Android-System bestimmt, welche der auf dem System vorhandenen Activities zu den Informationen passt. (Zur Unterstützung dieses Mechanismus definieren die Activities in der Manifestdatei sogenannte Intent-Filter, deren Daten mit den Informationen im Intent-Objekt abgeglichen werden, siehe auch weiter unten die Ausführungen zur Manifestdatei.)

Aufsteiger Activities sind nicht die einzigen Komponenten, die über Intents aufgerufen werden können. Auch Services und Broadcast Receiver sind ausführbare binäre Komponenten, die über Intents gestartet werden.

Broadcast Intents

Nicht nur Apps können Intents abschicken. Auch das Android-System selbst kann Intents versenden, die von interessierten Apps über Broadcast Receiver abgefangen werden können. Wir sprechen in diesem Fall von Broadcast Intents.

Views

Kommen wir noch einmal auf die grafischen Benutzeroberflächen der Apps zurück. Diese sind, wie Sie wissen, auf Bildschirmseiten verteilt und werden üblicherweise über XML-Layoutdateien definiert (siehe auch Kapitel 2.4).

Aufgebaut werden diese Bildschirmseiten aus Views. Eine View, zu Deutsch »Ansicht«, ist einfach ein rechteckiges Element einer UI-Oberfläche, das sich selbst zeichnet und grundsätzlich mit dem Anwender interagieren kann.

Drei Arten von Views sind für uns besonders interessant:

- Zeichenflächen – Instanzen der Klassen `View`, `ImageView` oder `SurfaceView`, die rechteckige Bereiche auf einer Bildschirmseite repräsentieren, in die wir zeichnen können.
- Widgets – spezialisierte Views, die einer bestimmten Aufgabe gewidmet sind. So sind z.B. die typischen Steuerelemente wie Eingabefelder, Schaltflächen, Listenfelder etc. als Widgets im Paket `android.widget` definiert.
- Viewgroups – Container-Views, die andere Views in sich aufnehmen können. Viewgroups, die andere Views nicht nur aufnehmen, sondern auch noch nach bestimmten Regeln anordnen, bezeichnen wir als Layout-Views (siehe Kapitel 5.3).

Sonstige Bausteine

Es gibt noch eine Reihe weiterer App-Bausteine, die in diesem Buch zwar keine besondere Rolle spielen, von denen Sie aber dennoch schon einmal gehört haben sollten.

- Services

  Ein Service, zu Deutsch »Dienst«, ist eine Arbeit, die im Hintergrund erledigt wird. Im Gegensatz zu Activities besitzen Services daher keine eigene Benutzeroberfläche. Mithilfe von Services kann man Aufgaben parallel zur laufenden App ausführen (beispielsweise eine Hintergrundmusik abspielen) oder langwierige Aktionen, wie z.B. das Herunterladen großer Multimediadateien aus dem Internet, im Hintergrund erledigen, ohne dass die App dadurch lahmgelegt wird. Services werden als Unterklassen der Klasse `Service` implementiert.

- Broadcast Receiver

  Broadcast Receiver sind Komponenten, die auf Meldungen des Android-Systems reagieren, wie z.B. »niedriger Batteriestand« (`ACTION_BATTERY_LOW`) oder »Der Kameraknopf wurde gedrückt« (`ACTION_CAMERA_BUTTON`). Broadcast Receiver besitzen keine eigene Benutzeroberfläche, können aber Nachrichten und Optionen zur Reaktion auf das Ereignis in die System-Statusleiste ausgeben.

- Content Provider

  Content Provider sind Komponenten, die Ihnen bei der Verwaltung externer Daten helfen. Sie können sie für Daten benutzen, die nur von einer App verwendet werden. Sie können über einen Content Provider aber auch Daten verwalten, die von mehreren oder allen Apps auf einem Android-Gerät genutzt werden können.

Das Activity-View-Intent-Geflecht

Eine Android-App besteht aus einer oder mehreren Activities. Jede dieser Activities steht für einen in sich abgeschlossenen Aufgabenbereich inklusive zugehöriger Bildschirmseite. Apps können aus prinzipiell beliebig vielen Activities (Bildschirmseiten) bestehen. Der Wechsel von einer Activity zur anderen erfolgt stets durch Absendung eines Intents.

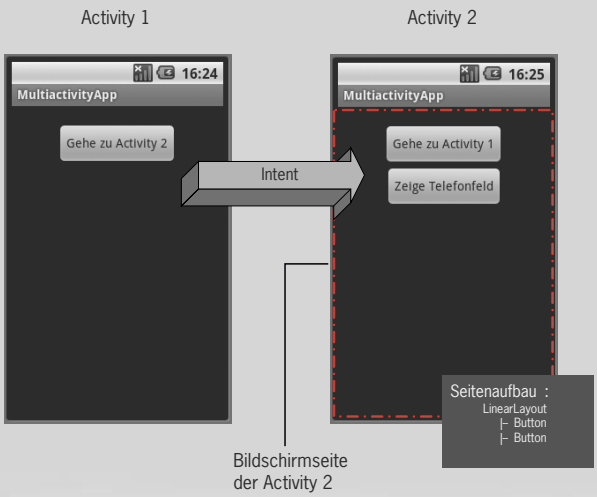


Abbildung 3.1: Drückt der Anwender den Button auf der 1. Bildschirmseite, wird ein Intent abgesetzt, der die Bildschirmseite der 2. Activity aufruft. Will der Anwender zurück zur 1. Seite, muss er auf den zugehörigen Button drücken, der natürlich ebenfalls einen Intent absetzt (hier nicht dargestellt). Die gestrichelte Linie markiert den Bereich, den die Bildschirmseite umfasst.

- Fragmente

  Mit Android 3.0 (API-Level 11) eingeführte Komponente. Fragmente erlauben dem Programmierer, den Code umfangreicher Activities aufzuteilen, indem er Teile der Activity in Fragmenten auslagert. Fragmente definieren ihre eigene Benutzeroberfläche und haben einen eigenen Lebenszyklus. Ihr wohl größter Vorzug: auf Fragmente basierende grafische Benutzeroberflächen können so aufgebaut werden, dass funktionell zusammen gehörende Fragmente auf einem Tablet-PC nebeneinander, auf einem Smartphone aber nacheinander angezeigt werden. Eine so konzipierte App bietet auf Tablet-PCs einen größeren Bedienungskomfort, ist aber auch auf Smartphones ausführbar.

App-Komponenten

Activities, Services, Content Provider, Broadcast Receiver und Fragmente werden in der Android-Terminologie als »*Android-Komponenten*« bezeichnet – essentielle Bausteine, die Aufgaben definieren, die vom Anwender oder vom System gestartet werden können.

Stichwortverzeichnis