5 Oberflächengestaltung

Die Qualität der Programmoberfläche ist ein wichtiges Kriterium für die Akzeptanz durch den Anwender. Schon durch den geschickten Einsatz der bereitgestellten Oberflächenelemente und deren optisch ansprechende Positionierung auf dem Bildschirm kann man viel erreichen.

Wir wollen uns daher im ersten Teil unseres Projekts mit den Grundlagen der Oberflächengestaltung in Android befassen. Dieses Kapitel führt in die Erstellung von Bildschirmseiten und Menüs ein. Wir werden alle notwendigen Komponenten und deren Zusammenspiel am Beispiel der Amando-Anwendung vorstellen. Auf Basis dieser Grundlagen können Sie später komplexere und optisch anspruchsvollere Oberflächen erzeugen.

5.1 Ziel

Erstes Ziel: die Startseite

Ziel dieses Kapitels ist es, den Startbildschirm von Amando zu implementieren. Auf diesem werden, neben einem Hinweistext, Schaltflächen für die Operationen *Position senden*, *Geokontakte verwalten*, *Karte anzeigen* und *Simulation starten* dargestellt. Über die Menütaste des Geräts können die Funktionen *Einstellungen*, *Hilfe* und *Beenden* aufgerufen werden. Die Schaltflächen und das Menü werden zunächst noch ohne Funktion sein. Die Texte der Startseite werden in einer Ressourcendatei verwaltet, was eine spätere Umstellung auf Mehrsprachigkeit erleichtert. Sie werden lernen, wie Ressourcen verwaltet werden und wie man sie referenziert.

5.2 Schnelleinstieg: Activities, Layouts und Views

In diesem Abschnitt erklären wir, aus welchen Grundbausteinen sich eine Oberfläche zusammensetzt. Anschließend wird beschrieben, wie man diese Bausteine zu einer lauffähigen Anwendung verknüpft.

5.2.1 Grundbausteine der Oberflächengestaltung

Wir wollen nach dem Start der Amando-Anwendung eine Bildschirmseite mit mehreren Schaltflächen, etwas Text und einem Bild angezeigt bekommen. Diese einzelnen Bausteine nennt man *Views*. Sie werden mit Hilfe eines Layouts angeordnet. Die Funktionalität, also unter anderem das Reagieren auf Oberflächenereignisse innerhalb des Layouts und der Zugriff auf die darin enthaltenen Views, erfolgt durch eine Android-Komponente namens *Activity*. Eine Activity sollte immer zur Darstellung genau einer Bildschirmseite implementiert werden. Sie besitzt zu einem Zeitpunkt nur ein Layout, das sie zur Anzeige bringt. Sie ist darüber hinaus in der Lage, auf Datenquellen und die Hardware des Android-Geräts zuzugreifen. Sie besitzt also die Kontrolle darüber, welche Daten angezeigt werden und welche Eingaben auf der Oberfläche im Programm umgesetzt werden.

Activity := Kontrolle

Jedes an der Oberfläche sichtbare Element ist von der Klasse `android.view.View` abgeleitet. Die klassischen Oberflächenelemente wie Eingabefelder, Schaltflächen und Auswahllisten werden wir ab jetzt als *Oberflächen-Komponente* oder *View* bezeichnen.

View = Darstellung

Views werden mit Hilfe eines Layouts auf dem Bildschirm angeordnet. Es gibt verschiedene Layouts, die eine jeweils unterschiedliche Anordnungsvorschrift darstellen. Es gibt ein Layout für eine lineare Anordnung der View-Elemente, ein Layout für eine tabellarische Anordnung, ein Layout für eine Anordnung relativ zueinander usw.

Layout = Anordnung

Layouts sind von der Klasse `android.view.ViewGroup` abgeleitet. Mit dieser Klasse hat man als Programmierer in der Regel selten bis gar nicht zu tun, aber es ist wichtig, sich zu vergegenwärtigen, dass Layouts (= `ViewGroup`) Gruppen von Views sind, die auch ineinandergeschachtelt werden können. Views selbst sind schon recht komplexe Objekte, Layouts demzufolge sehr komplexe Objekte, und geschachtelte Layouts können so umfangreich werden, dass sie die Performanz und den Speicherverbrauch einer Anwendung spürbar beeinflussen und sogar verhindern, dass eine Activity lauffähig ist. Wir werden in Abschnitt 5.4.1 darauf zurückkommen.

Oberfläche := Baum von Views

Die Oberfläche einer Anwendung, die gerade angezeigt wird, nennen wir Bildschirmseite. Sie setzt sich zusammen aus dem Layout (mit den darin enthaltenen Views oder weiteren Layouts) und einer Activity-Klasse, die auf die Oberflächenereignisse reagiert und Daten in View-Elementen zur Anzeige bringen kann. Die Bildschirmseite ist also der Teil der gesamten Oberfläche, mit der ein Anwender in Kontakt mit dem Programm tritt. Über sie übt er die Kontrolle über die Anwendung aus.

Bildschirmseite := Activity + Layout

5.2.2 Oberflächen implementieren

Beispiel: Startseite

Wir werden nun die oben genannten Begriffe auf unsere Aufgabenstellung, die Implementierung von Amando, anwenden. Am Beispiel der Startseite des Programms werden wir uns sowohl theoretisch wie praktisch mit dem Erstellen von Activities befassen. Abbildung 5-1 zeigt die Bildschirmseite, die angezeigt wird, wenn man Amando startet.

Abb. 5-1
Bildschirmaufbau

Je nachdem, welche Android-Version im Emulator oder auf dem Android-Gerät läuft, sieht die Oberfläche unterschiedlich aus. Die Abbildungen zeigen ein Android-4.1-Gerät. Man sieht auf der Startseite eine Textzeile, vier Schaltflächen und ein Bild. Dies sind unsere Views, die mit Hilfe eines Layouts innerhalb der Activity angeordnet sind.

Oben rechts im Bildschirm sind drei graue Punkte zu sehen. Drückt man darauf, öffnet sich das Optionsmenü. Android-2.x-Geräte haben eine eigene Taste für das Optionsmenü. Wenn es geöffnet wurde, werden die Menüoptionen *Einstellungen*, *Hilfe* und *Beenden* angezeigt. Dieses Menü ist nicht Teil des Layouts, sondern wird von der Activity beigesteuert.

Zunächst wird wieder, wie im Einstiegsbeispiel in Kapitel 1, ein Android-Projekt angelegt. Tabelle 5-1 zeigt die notwendigen Einstellungen, die wir für das Amando-Projekt brauchen.

Tab. 5-1 *Projekteinstellungen für das Amando-Projekt*

Application Name:	Amando
Project Name:	amando5
Package Name:	de.visionera.androidbuch.amando5
Minimum Required SDK:	API 14: Android 4.0.3
Target SDK:	API 21: Android 5.0
Compile With:	API 21: Android 5.0
Theme:	Material Light
Create Activity:	Blank Activity
Activity Name:	Startseite
Layout Name:	startseite

Android Studio generiert uns das gewünschte Projekt. Um nun dem Startbildschirm das gewünschte Aussehen zu geben, füllen wir die Datei `startseite.xml` mit dem Inhalt aus Listing 5-1.

Tipp

Das gesamte Projekt finden Sie auf der Webseite www.androidbuch.de im Bereich *Download* zum Herunterladen.

Listing 5-1 *Bildschirmlayout startseite.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent" >

  <TextView
    style="@style/TextStyleUeberschrift"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/schwarz"
    android:text="@string/txt_startseiteanzeigen_intro" />

  <Button
    android:id="@+id/sf_position_senden"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/sf_position_senden"
    android:onClick="onClickPositionSenden" />
```

```
  <Button
    android:id="@+id/sf_starte_geokontakte"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/sf_geokontakte_verwalten"
    android:onClick="onClickGeokontakteVerwalten" />

  <Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/sf_karte_anzeigen"
    android:onClick="onClickKarteAnzeigen" />

  <Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/sf_simulation"
    android:onClick="onClickSimulationStarten" />

  <ImageView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center_horizontal"
    android:src="@drawable/amando_logo" />

</LinearLayout>
```

Das Layout können wir nun der Start-Activity hinzufügen. Die Activity `Startseite` lädt das Layout während ihrer Erzeugung (Listing 5-2). Dies geschieht in der `onCreate`-Methode, die Teil der Activity ist und bei ihrer Erzeugung automatisch aufgerufen wird. Mit weiteren Methoden, die während der Erzeugung einer Activity automatisch durchlaufen werden, beschäftigen wir uns in Kapitel 15 über Lebenszyklen von Komponenten.

Eintrittspunkt einer Activity

Listing 5-2
Verknüpfung Layout mit Activity

```
package de.visionera.androidbuch.amando5.gui;

public class Startseite extends Activity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.startseite);
  }
}
```

Mehr müssen wir zunächst einmal nicht tun, um Layout, Views und Activity zusammenzubringen. Bei dem Layout (siehe Listing 5-1) handelt es sich um ein `LinearLayout`, das seine View-Elemente hintereinander oder untereinander anordnet. Es enthält drei Arten von View-Elementen: `TextView`, `Button` und `ImageView`. Wir werden in den folgenden Abschnitten darüber hinaus die wichtigsten Bestandteile, die für den Bau von Android-Oberflächen notwendig sind, kennenlernen.

Drei Arten von Views

In Android ist es möglich, Oberflächen einerseits in XML zu definieren, andererseits entspricht jedem in XML definierten Element ein konkretes Java-Objekt. Man könnte auf XML verzichten und die Oberflächen komplett in Java implementieren. Dies empfehlen wir jedoch nicht, da die Definition in XML wesentlich übersichtlicher ist und den Java-Quelltext schlanker hält. Ein weiterer Grund ist, dass Werkzeuge zur Oberflächenerstellung in Android XML als Ausgabe produzieren. Das im Android-Plugin eingebaute Werkzeug zur Oberflächengestaltung arbeitet z. B. mit XML.

Oberflächen nicht in Java erstellen!

Zurück zu unserem Layout. Bei der Definition der Views fällt auf, dass einige der XML-Attributwerte ein @-Zeichen enthalten. Beispielsweise findet man dort:

```
@style/TextStyleUeberschrift
@string/txt_startseiteanzeigen_intro
@color/schwarz
@drawable/amando_logo
```

Verweise auf Ressourcen definieren

Die durch @ eingeleiteten Attributwerte verweisen auf sogenannte *Ressourcen*. Dabei handelt es sich um Texte, Farbdefinitionen, Schlüsselwerte, Bilder oder ähnliche Nicht-Java-Bestandteile der Anwendung. Ressourcen spielen bei der Oberflächendefinition eine wichtige Rolle. Die nächsten Abschnitte befassen sich mit der Definition von Ressourcen und stellen die wichtigsten Ressourcenarten und ihre Anwendungsgebiete vor.

5.3 Ressourcen

Ein Programm besteht meist nicht nur aus Quellcode, sondern nutzt beispielsweise Grafiken und Piktogramme oder bringt eigene Audiodateien für bestimmte Aktionen mit. Wenn eine Anwendung mehrere Landessprachen unterstützen soll, müssen alle sichtbaren Texte mehrsprachig vorliegen. Auch Farbdefinitionen, Menüeinträge, Listen mit Daten oder Layouts werden nicht im Java-Quellcode definiert, sondern in XML als sogenannte *Ressourcen* hinterlegt. Je nach Format und Verwendungszweck unterscheiden wir folgende *Ressourcenarten*:

Definition in XML