Teil I

Mobile Betriebssystem und Entwicklung

Kapitel 1

Die Android Architektur und Technologien

Übersicht

Android ist ein Betriebssystem, das insbesondere für mobile Endgeräte eingesetzt wird. Die auf Linux basierende Architektur ist mehrschichtig aufgebaut und wird im Rahmen des Kapitels näher vorgestellt. Neben Linux als standardisierter Grundlage des Betriebssystems setzt Android mit Java und XML auf zwei weitere wichtige Basistechnologien, deren Einbindung in das System sie näher kennenlernen werden.

Lernziele
Nach Abschluss des vorliegenden Kapitels

- kennen Sie die grundlegende Architektur des Android OS,
- wissen Sie, wie Android und Java zusammenwirken
- wissen sie, mit welchen grundlegenden Ansätzen Apps entwickelt werden können.

1.1 Einleitung

1.2 Grundkonzept

In seiner Grundkonzeption setzt Android auf einige existierende Basistechnologien, modifiziert diese aber, um den Anforderungen an ein eingebettetes, Multitasking fähiges und interaktives OS gerecht zu werden:

Die Architektur des Systems ist dabei als Stack nach dem folgenden Schema aufgebaut:

.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.

1.3 Arten von Apps

App ist nicht gleich App. Je nach eingesetzter Technologien unterscheidet man drei Arten von Apps:

.
.
.
.
.
.
.
.
.
.

.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.

1.4 Java Konzept

Java ist die Standardprogrammiersprache für native Apps auf Android-Geräten. Java als beliebte Sprache mit großer Verbreitung ist damit sicherlich eine gute Grundlage. Um allerdings die Performance für mobile Geräte zu steigern. Während die Sprachelemente, die für die Programmierung von Android Apps herangezogen werden können mit den klassichen Java Sprachelementen weitestgehend ident sind, gibt es bei der Ableitung des Maschinencodes aus dem Quellcode erhebliche Unterschiede.

.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.

.
.
.

Da Smartphones in Bezug auf die Prozessorarchitektur sehr einheitlich sind, ist die Plattformunabhängigkeit wohl weniger der Nutzen von Java für Android als die Verbreitung der Programmiersprache und von entsprechenden Entwicklungstools. Mit der Dalvik VM und dem Dex-Compiler führte man bei Android ein Konzept ein, um die Ausführung von Java-Bytecode weiter zu optimieren. Die Vorteile der Dalvik VM sind dabei:

1.5 Elemente einer Android App – Architektonisches Grundkonzept

Apps wirken auf den Anwender sehr monolitisch und als geschlossene Anwendung. In Wirklichkeit steckt dahinter ein sehr modulares Entwicklungsparadigma. Ein App setze sich als sehr vielen unterschiedlichen Elementen zusammen, die je nach ihrer Funktion einer bestimmten Kategorie zugehören. Je nach Kategorie bieten sie unterschiedliche Funktionen und Schnittstellen an, auf die die eigene, aber auch andere Anwendungen zugreifen können. In Android unterscheidet man zwischen folgenden grundlegenden Elementen:

Activity

View

Intents

IntentFilter

IntentReceiver

Service

Notifications

ContentProvider

Broadcast Receiver

1.6 Life Cycle

Um bei einem Geräte mit beschränkten Ressourcen Multitasking und eine interaktive Bedienlogik zu ermöglichen, unterliegen Apps bzw. besser deren Activities einem speziellen Life Cycle, der durch den Entwickler gezielt beeinflusst werden kann.

.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.

Kapitel 2

Android - Die Entwicklerperspektive

Übersicht

Für Entwickler bietet Android ein gesamtes Ökosystem, das bei der Entwicklung von verschiedenen Typen von Apps und für unterschiedliche Plattformen unterstützt. Daneben ist es notwendig den Aufbau und Ablauf einer App sowie die Kommunikation zwischen verschiedenen Apps zu kennen, um selbst eine Entwicklung beginnen zu können. Ein einfaches Beispiel führt an die Thematik heran und liefert eine praktische Einführung in die App-Entwicklung.

Lernziele
Nach Abschluss dieses Kapitels

- haben Sie einen Einblick in die Bedeutung der Betriebssystemversion für die Entwicklung
- kennen Sie die Grundlegenden Elemente des SDK
- Sie können mit Eclipse ein Projekt anlegen und
- eine erste App entwickeln und testen.

2.1 Android OS Versionen

Seit seiner Entstehung hat Android zwischenzeitlich zahlreiche Erweiterungen, Ergänzungen und Korrekturen erfahren. Dies betrifft sowohl den Betriebssystemkern (Linux basiert), als auch die funktionalen Möglichkeiten, verfügbaren Standardbibliotheken, Visualisierung, Performance etc.

Da zwischenzeitlich zahlreiche Versionen des Betriebssystems auf dem Markt sind, ist es dringend notwendig, sich zuerst Gedanken zu machen, auf welche Version des Betriebssystems man die Entwicklung abstellt. Dabei ist wichtig zu beachten, dass man bei den meisten Änderungen des

Android-Betriebssystems darauf geachtet hat, dass das System bei den Apps abwärts kompatibel ist. Dies bedeutet, dass Apps, die für ältere OS Varianten entwickelt wurden meist auch in den neueren Versionen funktionieren.

Konsequenzen der Entscheidung für eine Version:

Folgende Kernversionen von Android sind wichtig, da sie wesentliche Änderungen im OS Konzept oder der Funktionalität mit sich brachten.

Version	Name	API	Veröff.	Inventionen
1.0	Base	1	Sep 2008	Android Market (heute Play-Store), Google Maps, Gmail, YouTube, automatische Synchronisierung des Adressbuchs und des Kalenders, ...
1.5	Cupcake	3	Apr 2009	automatischer Wechsel zwischen Hoch- und Querformat, Bildschirm-Tastatur Aufnahme und Wiedergabe von Videos, automatisches Verbinden und Stereo für Bluetooth, weitere Sprachen neben Englisch und Deutsch
2.0.x, 2.1	Éclair	5, 6, 7	Okt 2009	Digitalzoom und Unterstützung von Blitzlicht, Unterstützung von Microsoft Exchange, Bluetooth 2.1 (WPAN),
2.3.x	Ginger-bread	9, 10	ab Dez 2010	neuer Linux Kernel, Unterstützung von HTML5 Elementen, Verbesserte Dual Core Unterstützung, Unterstützung neuer Sensoren Gyor, Barometer, ...
3.x.x	Honey-comb	11, 12, 13	ab Feb 2011	eigene Version nur für Tablets, USB Host Modus, ...
4.0.x	Ice Cream Sandwich	14, 15	Okt 2011	Zusammenführung von Smartphone und Tablet OS Versionen, NFC Funktionalität, ...
4.1.x - 4.3.x	Jelly Bean	16, 17, 18	Jun 2012	neue UI Konzepte, viele Verbesserungen und Bugfixes
4.4.x	KitKat	19	Okt 2013	viele Verbesserungen und Bug Fixes
4.4. W	Wearable	Funktionen für Wear-ables		

2.2 IDE und das SDK

Die Entwicklung von kompletten und komplexen Anwendungen erfordert in der Regel mehr als nur einen Texteditor und einen Compiler, als Minimum machen weitere Tools die Entwicklung einfacher:

Diese einzelnen Programme verknüpft in über eine Anwendung, meist mit einer graphischen Benutzeroberfläche, werden als IDE (Integrated Development Environment) bzw. bei Ausrichtung auf spezifische Systeme oder Programmiersprachen als SDK (Software Development Kit) bezeichnet.

2.3 Android Developer Tools - Bundle

Google bzw. die Open Handset Alliance als Kernentwickler von Android als mobilem Betriebssystem bieten ein SDK zum Download und zur freien Nutzung für die Entwicklung eigener Applikationen an. Dieses besteht aus folgenden Komponenten:

Das fertige SDK lässt sich als Bundle (ADT Bundle) downloaden. Es stehen Versionen für unterschiedliche Plattformen und Betriebssysteme zur Verfügung. Direkt nach dem Entpacken, kann Eclipse durch öffnen der Datei unter

```
1
```

gestartet werden.

2.4 Blick in Eclipse

Eclipse als IDE besteht aus verschiedenen Elementen. Die wichtigsten sind:

Wichtig in Eclipse sind die Konzepte der Erweiterbarkeit und Perspektiven. Eclipse ist im Kern eine Entwicklungsplattform für Java, die in Java implementiert ist. Sie zeichnet sich aber dadurch aus, dass

2.5 Das ADT und seine Bestandteile

Das ADT ist eine solche Erweiterung für Eclipse. Da wir das ADT Bundle installiert haben, haben wir alles bereits fertig konfiguriert und können mit der App-Entwicklung für Android starten. Das ADT lässt sich auch als eigenes Bundle in eine bestehende Installation von Eclipse integrieren (siehe hierzu Tutorials im Internet).

Wichtig sind für uns nun drei Aspekte des ADT:

2.6 Hello Android

Mit einem klassischen "Hello World"Projekt wollen wir uns einen Überblick über den Entwicklungsprozess und die Ressourcen verschaffen. Hierzu durchlaufen wir folgende erste Schritte:

Durch das Anlegen eines neuen Projektes wird automatisch von Eclipse/ADT eine Ordner- und Dateistruktur erzeugt, in der die wichtigesten Ressourcen für eine Anwendunge mit einem Fenster bereits vorgegegeben sind. Die Anwendung ist bereits ausführbar und kann in einem virtuellen Device getestet werden, ergibt aber funktional noch keinen großen Mehrwert. Hierzu müssen wir in einem ersten Schritte mehrere Ressourcen verändern und kleine Einfügungen vornehmen.

Die wichtigesten Ressourcen eines Android-Projektes sind:

Um ein erstes Projekt bearbeiten und testen zu können, benötigen wir noch ein Geräte zum Test. Neben der Möglichkeit echte Geräte anzuschließen und direkt auf diesen physikalischen Devices testen zu können, gibt es auch die Option, verschiedene virtuelle Geräte zu definieren und diese emulieren zu lassen. Wir wollen ein erstes Geräte definieren und starten:

Wir ergänzen nun unsere App um einen weiteren Textview, um zu sehen, wie das Zusammenspiel zwischen Layout, Strings und Activity funktioniert.

Wir passen das Layout wie folgt an:

```
1
2
3
4
5
```

Darunter fügen wir nun einen zweiten TextView ein:

```
1
2
3
4
5
6
```

Durch die Benennung beider TextViews mit einer id können wir nun die Views ansprechen und dadurch auch die Positionierung regeln (RelativeLayout).

Um nun im zweiten Textfeld einen Text zu erhalten müssen wir noch die strings.xml erweitern und darin unseren String hello_ message aufnehmen:

```
1
2
```

Die Ansprache eines TextViews durch eine ID ist aber auch aus Sicht der Anwendungslogik wichtig. Wir können nun den Textview aus unserem Programm heraus manipulieren, dessen Erscheinungsbild ändern oder eben den Inhalt, also den Text, beeinflussen. Der Textview ist ein Objekt, auf das wir in unserer Programmierung zugreifen können:

```
1
2
3
4
5
6
7
8
9
10
11
```

2.7 Erste Funktionen

Wir wollen nun erste kleine Anpassungen an unserer App vornehmen und dadurch das Android Grundkonzept und das Zusammenwirken von Activity, Layout und Values näher kennenlernen.

Hierzu wollen wir zwei Activities miteinander verbinden. In der ersten Activity können wir Werte in ein Textfeld eingeben und diese werden nach Klick auf eine Schaltfläche in einer zweiten Activity dargestellt. Hierzu werden wir eine zweite Activity erzeugen und beide Activities über ein Intent verbinden.

Ändern Sie das existierende Layout für die MainActivity wie folgt ab:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:orientation="horizontal">
7     <EditText android:id="@+id/edit_message"
8         android:layout_weight="1"
```

```
9           android:layout_width="0dp"
10          android:layout_height="wrap_content"
11          android:hint="@string/edit_message" />
12      <Button
13          android:layout_width="wrap_content"
14          android:layout_height="wrap_content"
15          android:text="@string/button_send" />
16  </LinearLayout>
```

Wir müssen nun noch die Strings registrieren, da wir im Layout keine Texte direkt eintragen, sondern immer auf res/strings verweisen. Hier sind nun folgende Einträge notwendig:

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <string name="app_name">My First App</string>
4      <string name="edit_message">Enter a message</string>
5      <string name="button_send">Send</string>
6      <string name="action_settings">Settings</string>
7      <string name="title_activity_main">MainActivity</string>
8  </resources>
```

Führen Sie das Programm im Emulator aus, um zu überprüfen, ob alles korrekt angezeigt wird. Sie werden feststellen, dass der Button noch ohne Leben ist und keine weitere Interaktion möglich ist mit Ausnahme der Eingabe eines Textes in des entsprechende Feld.

Um dies zu ändern müssen wir nun einige Schritte machen, um unsere App zu erweitern.

- Wir benötigen für unsere MainActivity eine Methode, um den Button zu beleben.
- Um die Eingabe zu Verarbeiten und anderswo darzustellen, sind eine zweite Activity erforderlich
- sowie ein Intent, der beide Activities verbindet.

Ergänzen Sie den Button im Layout unserer MainActivity wie folgt:

```
1  <Button
2      android:layout_width="wrap_content"
3      android:layout_height="wrap_content"
4      android:text="@string/button_send"
5      android:onClick="sendMessage" />
```

Öffnen Sie nun die MainActivity Class und ergänzen Sie diese um die sendMessage Methode und den entsprechenden Intent, der unsere MainActivity mit einer neuen Activity, die es noch anzulegen gilt, verbindet.

```
1  public void sendMessage(View view) {
2      Intent intent = new Intent(this, DisplayMessageActivity.class);
3      EditText editText = (EditText) findViewById(R.id.edit_message);
4      String message = editText.getText().toString();
5      intent.putExtra(EXTRA_MESSAGE, message);
6      startActivity(intent);
7  }
```

Diese Methode benötigt die View Class. Diese können Sie manuell hinzufügen:

```
1 import android.view.View;
```

oder automatisch von Eclipse mit Ctrl + Shift + O ergänzen lassen. Durch das automatische Ergänzen werden ebenfalls die Klassen für Intent und EditText hinzugefügt.

Auch unsere Message müssen wir noch als Konstante in der Activity ergänzen:

```
1 public class MainActivity extends Activity {
2     public final static String EXTRA_MESSAGE =
3         "com.example.myhelloworld.MESSAGE";
4     ...
5 }
```

Wir benötigen nun noch eine Activity, mit der wir unsere Nachricht anzeigen lassen können. Diese legen wir wie folgt an:

- New → Other Android Activity
- Auswahl Blank Activity und Eintrag der Details in den folgenden Fenstern:
- Project: MyHelloWorld
- Activity Name: DisplayMessageActivity
- Layout Name: activity_ display_ message
- Navigation Type: None
- Hierarchial Parent: com.example.myhelloworld.MainActivity
- Title: My Message
- Click Finish.

Öffnen Sie nun den Quellcode der neuen Activity und ergänzen Sie ihn wie folgt:

```
1  public class DisplayMessageActivity extends Activity {
2
3      @SuppressLint("NewApi")
4      @Override
5      protected void onCreate(Bundle savedInstanceState) {
6          super.onCreate(savedInstanceState);
7          setContentView(R.layout.activity_display_message);
8
9          // Make sure we're running on Honeycomb or higher to use ActionBar APIs
10         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
11             // Show the Up button in the action bar.
12             getActionBar().setDisplayHomeAsUpEnabled(true);
13         }
14     }
15
```

```
16      @Override
17      public boolean onOptionsItemSelected(MenuItem item) {
18          switch (item.getItemId()) {
19          case android.R.id.home:
20              NavUtils.navigateUpFromSameTask(this);
21              return true;
22          }
23          return super.onOptionsItemSelected(item);
24      }
25  }
```

Damit in Android eine Activity zugelassen wird, muss sie im Manifest der App deklariert werden:

```
1  <application ... >
2      ...
3      <activity
4          android:name="com.example.myhelloworld.DisplayMessageActivity"
5          android:label="@string/title_activity_display_message"
6          android:parentActivityName="com.example.myfirstapp.MainActivity" >
7          <meta-data
8              android:name="android.support.PARENT_ACTIVITY"
9              android:value="com.example.myhelloworld.MainActivity" />
10     </activity>
11 </application>
```

In der onCreate() Methode der neuen Activity greifen wir nun die Daten des Intents auf, erzeugen einen TextView und zeigen die Daten darin an:

```
1  @Override
2  public void onCreate(Bundle savedInstanceState) {
3      super.onCreate(savedInstanceState);
4
5      // Get the message from the intent
6      Intent intent = getIntent();
7      String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
8
9      // Create the text view
10     TextView textView = new TextView(this);
11     textView.setTextSize(40);
12     textView.setText(message);
13
14     // Set the text view as the activity layout
15     setContentView(textView);
16 }
```

Sie sollten nun in der Lage sein, die App mit "Run As ßu generieren, an den Emulator zu übertragen und dort auszuführen.