App Entwicklung mit dem Android SDK

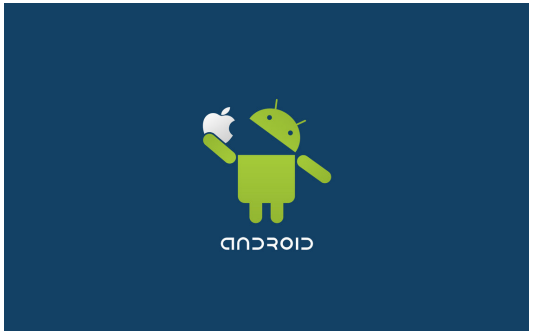
Grafische Benutzeroberfläche

Dominik Wagner

Informatik Sommercamp 2012

23.7.2012

Kurze Übersicht



- Um ein *Layout* für eine Android-Anwendung zu definieren schreibt man im ersten Schritt eine `XML`-Datei.
- Diese `XML`-Datei wird automatisch in `Java`-Code übersetzt.
- In der `onCreate`-Methode unserer Anwendung wählen wir die Layout-Datei aus.
- Mit `findViewById(...)` können wir uns ein Element der GUI nehmen und z.B. einen *Listener* anhängen.

XML - Einführung

XML steht für *eXtensible Markup Language*.
Wer HTML kann, kann XML. Für alle anderen gilt:

- XML verwendet so genannte *Tags*.
- Ein Tag kann aufgemacht werden: `<xml>`
- Ein Tag kann zugemacht werden: `</xml>`
- Ein Tag kann auf- und zugemacht werden: `<xml />`
- Ein Tag hat Attribute: `<xml attr="Text"...>`
- Ein Tag hat Tags und Text: `<a><b><c>Text</c></b></a>`
- . . . mehr später

XML - Beispiel

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:id="@+id/Game">
  <LinearLayout android:layout_margin="15dp">
    <include layout="@layout/row" />
  </LinearLayout>
  <RelativeLayout>
    <ImageView android:src="@drawable/circle" />
  </RelativeLayout>
</LinearLayout>
```

Android-Layout mit XML 1/2

Wie ein HTML-Dokument besteht auch eine Layoutdatei zum einen aus gruppierenden Elementen (ViewGroups) und aus sichtbaren Objekten (Views). Zu den Gruppen gehören:

- LinearLayout
- RadioGroup
- RelativeLayout
- ...

Zu den Views gehören Elemente wie

- Button
- ImageView
- RadioButton
- ...

Android-Layout mit XML 2/2

Gruppen können beliebig ineinander verschachtelt werden. Irgendwann muss man natürlich innerhalb einer Gruppierung Elemente anordnen. Beispiele hierzu findet man bei Hello, Views Dort wird auch erklärt, welche Bedeutung die Attribute der einzelnen Tags haben.

Mehr XML ...

Es gibt noch ein paar Sachen, die per XML definiert werden können:

- Texte (res/values/strings.xml)
- Styles (res/values/styles.xml)
- Graphiken (res/values/drawable/...)
- ...

Auch hierzu gibt es auf den Webseiten des Android-SDKs Beispiele.

findViewById 1/3

Um auf ein Objekt des Layouts zugreifen zu können gibt es die Funktion `findViewById()`. Die ID eines Objekts ist in der XML-Datei durch das Attribute `android:id` angegeben.

```
android:id="@+id/CurrentColumn3"
```

`@id` gibt an, dass es sich hierbei um eine ID handelt. Das "+" fügt man zusätzlich ein, wenn eine ID neu ist. Um nun auf das Objekt mit der ID `CurrentColumn3` zugreifen zu können, schreibt man folgendes in Java:

```
View v = findViewById(R.id.CurrentColumn3);
```

findViewById 2/3

Da wir einen Button von einer ImageView unterscheiden möchten, müssen wir an vielen Stellen passend casten:

```
Button b = (Button) findViewById(R.id.button);
ImageView i = (ImageView) findViewById(R.id.image);
```

findViewById 3/3

Manchmal gibt es eine ID auch mehrfach. Um zu entscheiden, welches Objekt gemeint ist, rufen wir `findViewById` auf der `ViewGroup` auf, in der sich das Objekt befindet:

```
ViewGroup c = (ViewGroup) findViewById(R.id.Row1);
ImageView i = (ImageView) c.findViewById(R.id.image);
ViewGroup d = (ViewGroup) findViewById(R.id.Row2);
ImageView j = (ImageView) d.findViewById(R.id.image);
```

android:, xmlns:, mm:

Die Attribute der XML-Tags haben verschiedene Präfixe. In den meisten Fällen android:, manchmal gar keine und dann gibt es noch xmlns: und mm:. Eine ausführliche Erklärung hierzu liefert http://de.wikipedia.org/wiki/XMLNS. Die einfache Erklärung ist:

android:, xmlns:, mm:

Die Attribute der XML-Tags haben verschiedene Präfixe. In den meisten Fällen android:, manchmal gar keine und dann gibt es noch xmlns: und mm:. Eine ausführliche Erklärung hierzu liefert http://de.wikipedia.org/wiki/XMLNS. Die einfache Erklärung ist: Macht euch keine Gedanken darüber und verwendet es so, wie es in unseren Beispieldateien zu sehen ist oder es in den Dokumentationen des SDKs beschrieben ist.

Weiterführende Informationen

- Hello, Views*
- User Interface*
- Application Resources*
- Layout Tricks: Creating Reusable UI Components
- User Interface Guidlines
- Declaring a custom android UI element using XML

Die mit * markierten Seiten sind hilfreich für die App-Entwicklung, die anderen könnt ihr euch anschauen, wenn ihr euch detailierter informieren möchtet.