Android – Eine Einführung

Layouts, Views & Adapter

Andreas Wilhelm

Institut für Informatik
Georg-August-Universität Göttingen

www.avedo.net

Contents

Layouts

Contents

Allgemeines

- Oberflächen für Aktivitäten, Menüs, Dialogen und Widgets
- Grundelemente Views und ViewGroups (Layouts & AdapterViews)
- Deklaration im XML-Format
- Formatierung eines Elements über Attribute
- Alternativ View-Objekte während der Laufzeit im Quellcode erzeugen und ändern
- Weitestgehend einheitliche Namensgebung für XML-Attribute und Objekt-Methoden

XML-Layouts

Deklaration von Layouts in XML verschlankt den Code, ermöglicht Unterstützung verschiedener Oberflächen, Sprachen und Auflösungen und erleichtert den Debugging-Prozess.

Deklaration

- Erstellung von Layouts erinnert an HTML-Code
- Jedes layout darf nur ein Wurzelelement enthalten (View, ViewGroup, Layout, Merge)
- Darunter beliebig verschachtelte Layout-Hierarchie
- Speichern von Layouts unter */res/layouts*
- Zugriff im Quellcode über automatisch generierte Klasse *R*
- Zugriff in anderen Layouts über *@[package:]layout/filename*

Allgemeines

- Struktur-Elemente, die andere Views anordnen
- LinearLayout, das FrameLayout oder das RelativeLayout
- Weitere ViewGroups ohne Kindelemente (Bsp: AdapterView)
- Basis-Attribute ID (optional), sowie Breite und Höhe

Android-IDs

Eindeutige ID eines Layout-Elements wird über *android:id* in der Form *@+id/name* deklariert. + signalisiert dabei, dass es sich um eine neue ID handelt. Diese ID kann im Quellcode mit *R.id.name* referenziert werden.
Alternativ kann eine ID als Resource deklariert und mit *@id/name* referenziert werden. Die Deklaration einer solchen Resource wird in XML vorgenommen und unter einem beliebigen Dateinamen unter */res/values* abgelegt.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <item type="id" name="id_name" />
</resources>
```

Listing : Deklaration von IDs

Bemaßungen

- Bemaßungen eines ViewGroups mit den Attributen *android:layout_width* und *android:layout_height* entweder explizit oder implizit
- Explizite Angabe erfolgt dabei als Wert oder als Resource
- Implizite über die Schlüsselwörter *fill_parent*, *match_parent* oder *wrap_content*

Zur Bemaßungen können verschiedene Einheiten (px, dp, sp, pt, in, mm) oder Schlüsselwörter verwenden.

Schlüsselwort	Beschreibung
match_parent	Weist aktuellem Objekt die Größe des Eltern-Elements zu (löst *fill_parent* ab)
fill_parent	Weist aktuellen Objekt die Größe des Eltern-Elements zu
wrap_content	Sorgt für Anpassung der Größe, sodass Inhalt umschlossen wird

Alternativ kann man auch eine Resource im Quellcode über die Klasse *R* oder in einer XMl-Datei mit dem Schlüssel *@[package:]dimen/dimension_name* referenziert.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="imagebox_height">64dp</dimen>
    <dimen name="imagebox_width">32dp</dimen>
</resources>
```

Allgemeines

- Allein stehendes Element einer grafischen Oberfläche
- Nimmt rechteckige Fläche des Bildschirms ein
- Kümmert sich um die Ausgabe und das Verarbeiten von Ereignissen
- Textfelder, Buttons und Eingabefelder, wie beispielsweise *TextView*
- Deklaration per XML oder im Quellcode
- Verwaltung der Basis-Attribute, wie bei ViewGroups

RequestFocus

- Gibt View bzw. ViewGroup direkt bei Initialisierung den Fokus
- XML-Deklaration als leeres Element *<requestFocus>*
- Zuweisung eines solchen Elements nur einmal pro Datei

Include

- Verbinden bereits existierender Layouts
- Element <*include*>
- Einziges eigenes Attribut Name des einzubinden Layouts
- Optional eine ID, sowie Breite und Höhe des Layouts
- Werte überschreiben die Werte des Wurzelknotens im eingebundenen Layout
- Änderungen an den Bemaßungen mit *android:layout_width* und *android:layout_height* nur für beide Attribute möglich

ViewStub

Alternativ kann anstatt des <*include*>-Elements auch ein *ViewStub* verwendet werden.

Merge

- Wurzelelement um Layouts zu verbinden
- Sinnvoll wenn Layouts mit *<include>* verbunden werden sollen
- Flacherer und dadurch einfacherer Aufbau der View-Hierarchie