Freie Universität Berlin

Google Android API

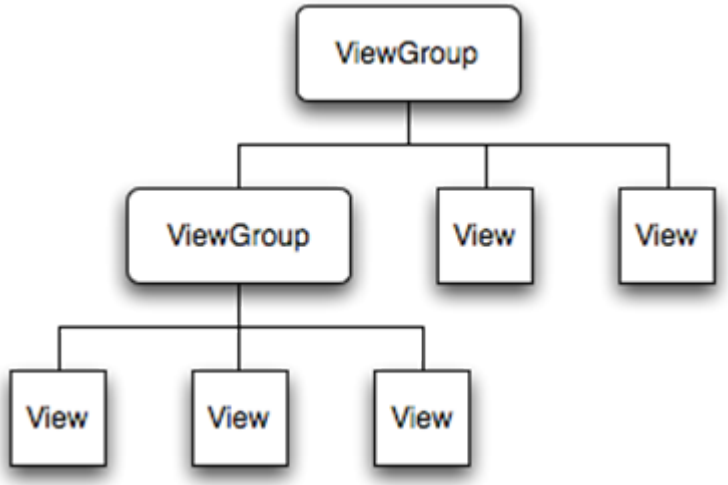
User Interface

Eine knappe Einführung

Google Android API – User Interface

Designelement	Wofür	Klasse / Package
View	Basisklasse für alle Designelemente	*public class android.view.View*
Viewgroup	Container für Views und Layouts	*public abstract class android.view.ViewGroup*
Layout	Die Art der Anordnung von Designelementen auf dem Bildschirm	*package android.widget*
Widget	Designelemente wie Button, Textbox usw.	*package android.widget*

Views und Viewgroups



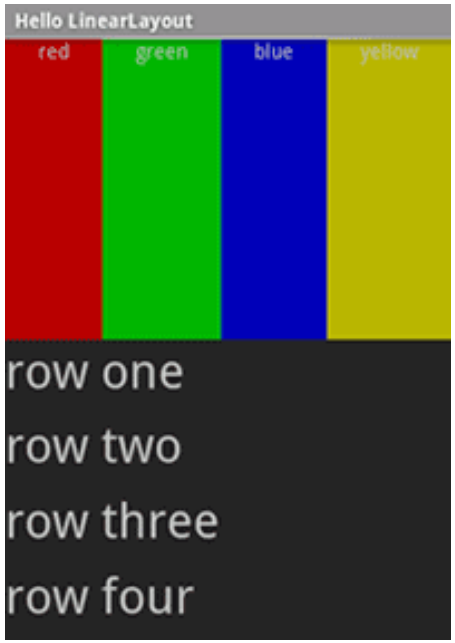
- Viewgroups enthalten Views oder Viewgroups
- So entsteht eine klare „parent“ „children“ Hierarchie
- Viewgroups legen dabei die Art der Anzeige fest
- Views sind die tatsächlich angezeigen Designelemente
- Vergleich:
  - java.awt.Component <-> android.view.View
  - java.awt.Container <-> android.view.ViewGroup

Layouts

- Layouts definieren die Andordnung der Views innerhalb einer Viewgroup
- Layouts können in XML beschrieben werden
  - Zugriff auf Layout-Elemente aus dem Code über die generierte **„R“-Klasse**
- XML Tags innerhalb des Layout tragen den Namen der View-Klasse
  - **<Button>** erstellt einen Button
  - **<LinearLayout>** ein Lineares Layout
- Ein paar Beispiele:

Layout	Anordnung der Kinder
LinearLayout	Vertikal oder horizontal
TableLayout	Reihen und Spalten
AbsoluteLayout	Absolute x/y-Koordinaten
RelativeLayout	Relativ zu Eltern, oder Geschwistern

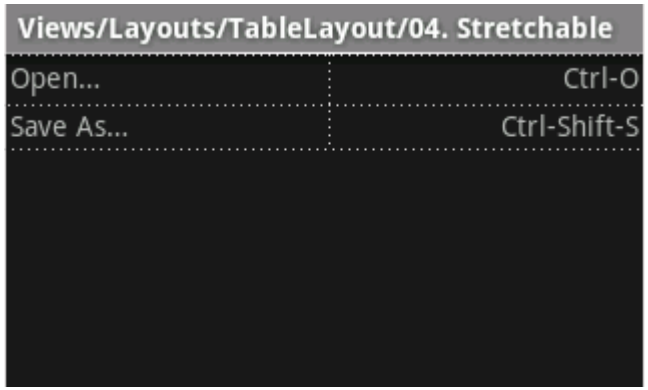
Layouts – Linear Layout



Vereinfachter Code:

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout android:orientation="vertical">

   <LinearLayout android:orientation="horizontal">

            <TextView ... />
            <TextView ... />
            <TextView ... />
            <TextView ... />

   </LinearLayout>

   <LinearLayout android:orientation="vertical">

            <TextView ... />
            <TextView ... />
            <TextView ... />
            <TextView ... />

   </LinearLayout>

</LinearLayout>
```

Layouts – Table Layout



Vereinfachter Code:

```
<?xml version="1.0" encoding="utf-8"?>

<TableLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:stretchColumns="1">

   <TableRow>

      <TextView ... />
      <TextView ... />

   </TableRow>
   <TableRow>

      <TextView ... />
      <TextView ... />

   </TableRow>

</TableLayout>
```

Widget

- Ist ein View Object, d.h. Unterklasse von *android.view.View*
- Ist für die Interaktion mit dem Benutzer zuständig

- Widgets liegen im Widget Package *android.widget.**
- Es sind aber auch eigene Widgets erstellbar

- Beispiele:
  - TextView
  - CheckBox
  - Button
  - DatePicker
  - ZoomControls

UI Events

Jedes Widget hört auf verschiedene Events
Diese werden von Eventlistenern gesteuert
onClick()
View.OnClickListener.
Press Enter Key oder Touch
onLongClick()
View.OnLongClickListener.
Press Enter Key & Hold oder Touch & Hold
onFocusChange()
View.OnFocusChangeListener.
wenn der Fokus mit den Richtungsknöpfen von einem View genommen wird
onKey()
View.OnKeyListener.
Wenn der Focus auf einem View liegt und irgendeine Taste gedrückt wird
onTouch()
View.OnTouchListener.
Alle Touch Events (press, release, gesture)
onCreateContextMenu()
View.OnCreateContextMenuListener.
Wenn ein ContextMenu aufgebaut wird

Code Beispiel #1: Java

```
private TextView mDateDisplay;
private Button mPickDate;
private int mYear, mMonth, mDay;
static final int DATE_DIALOG_ID = 0;

protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main); // capture our View elements
        mDateDisplay = (TextView) findViewById(R.id.dateDisplay);
        mPickDate = (Button) findViewById(R.id.pickDate); // add a click
                                                          listener to the button
        mPickDate.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                        showDialog(DATE_DIALOG_ID);
                }
        }); // get the current date
        final Calendar c = Calendar.getInstance();
        mYear = c.get(Calendar.YEAR);
        mMonth = c.get(Calendar.MONTH);
        mDay = c.get(Calendar.DAY_OF_MONTH); // display the current date
        updateDisplay();
}
```

Code Beispiel #2: XML-Layout

Dateiname: **main**.xml

```
<?xml version="1.0" encoding="utf-8"?>
   <LinearLayout              xmlns:android="http://schemas.android.com/apk/res/android"
          android:layout_width="wrap_content"    android:layout_height="wrap_content"
          android:orientation="vertical">
      <TextView     android:id="@+id/dateDisplay"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text=""
      />
      <Button android:id="@+id/pickDate"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="Change the date"
      />
   </LinearLayout>
```

Einführung in das UI:
http://developer.android.com/guide/topics/ui/index.html

Eine handvoll weiterer Beispiele, **Hello, Views**:
http://developer.android.com/guide/tutorials/views/index.html