



# Alternative Ressourcen

# Agenda

- Einleitung
- Arten von Ressourcen
- Notwendige Bedingungen
- Vergleich von Default- und Alternative Ressourcen
- Zugriff auf Ressourcen
- Zusammenhang von Layout und Code Dateien
- Automatische Generierung von IDs
- Möglichkeiten von Alternativen Ressourcen:
  - Layouts – Orientierung und Bildschirmgröße
  - Grafiken – Auflösung
  - Zeichenketten – Mehrsprachigkeit

# Einleitung

Stand: März 2009



HTC Magic

Die gleiche Bildschirmgröße von 3.2 Zoll mit 320x480 Pixel.



T-Mobile G1

# Einleitung

Stand: Heute

android



- SmartPhone
- SmartTV
- SmartWatch
- Tablet
- Notebook
- SmartNavi??

...



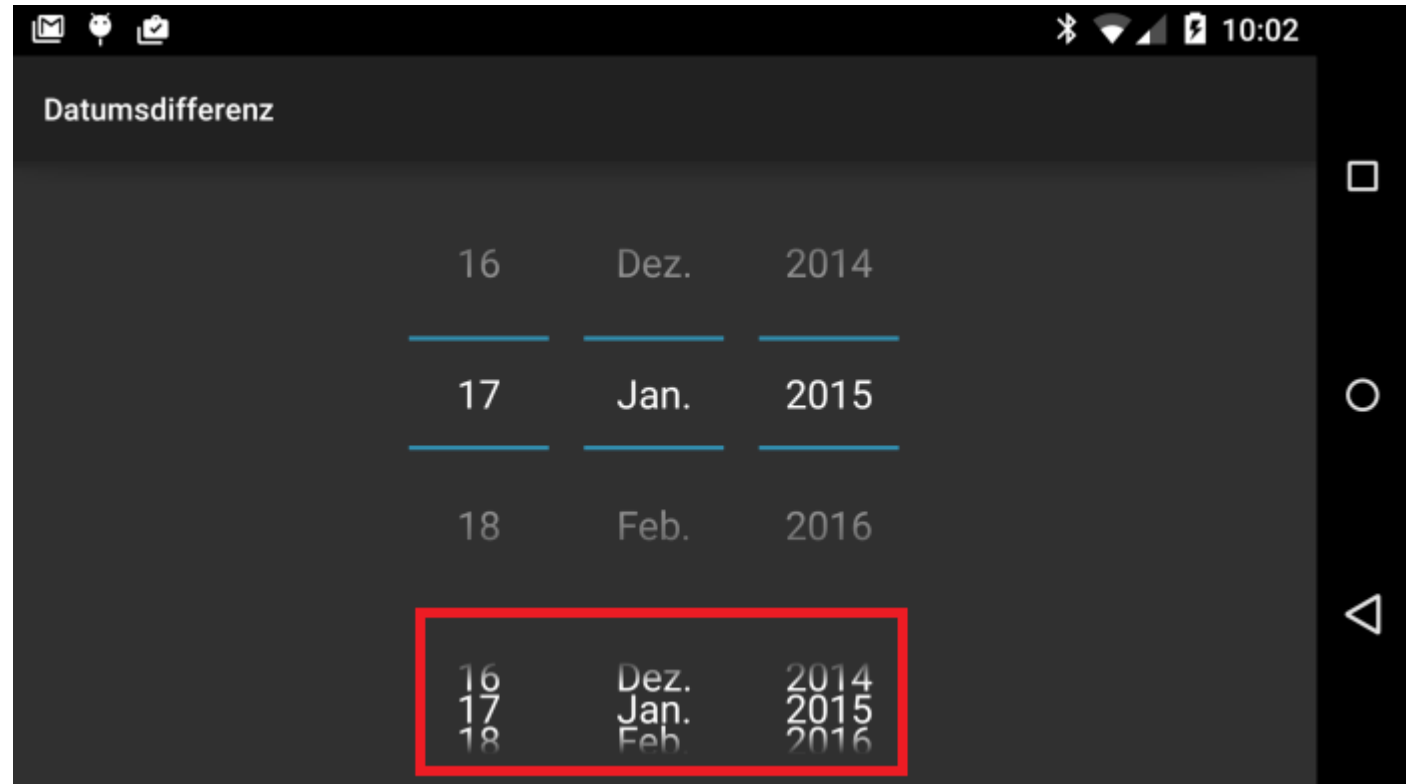
# Einleitung

Korrekte Darstellung  
in der Porträtsansicht



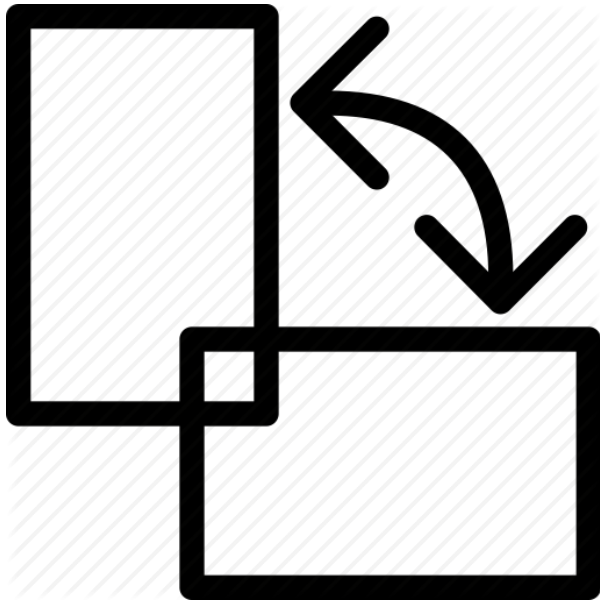
# Einleitung

Unbeabsichtigte Darstellung  
in der Landschaftsansicht



# Einleitung

Faktoren, die man bei der Entwicklung stets beachten sollte:



Orientierung



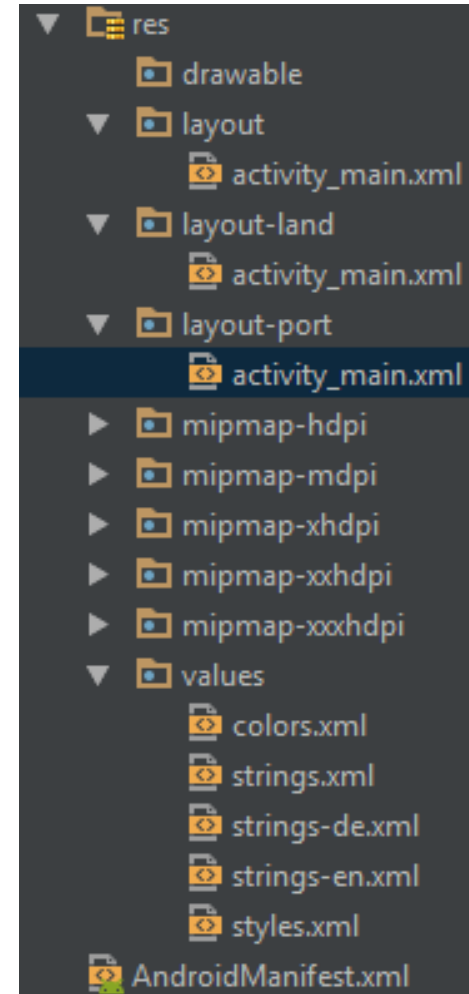
Bildschirmgröße



Sprachen

# Arten von Ressourcen

- Layouts
- Grafiken (Drawables / MIPMAPs)
- Zeichenketten (Strings)
- Farben (Colors)
- Dimensionen (Dimens)
- Styles
- Menus
- Arrays



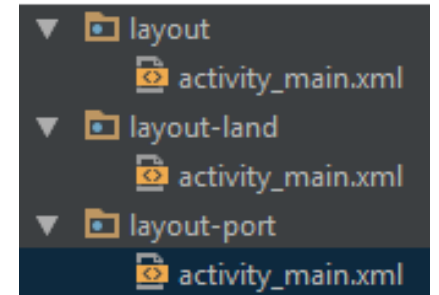


# Notwendige Bedingungen

- **Ressourcen müssen separat vom Code gespeichert werden**
  - > wenn Zeichenketten „hardgecodet“ im Quellcode stehen, kann man keine Alternativen anbieten – zumindest automatisiert
- **Ressourcen müssen sich im semantisch korrekten Verzeichnis befinden**
  - > ein Bild soll sich logischerweise nicht im „layout“ Verzeichnis befinden
- **Default-Ressourcen sollten immer vorhanden sein**
  - > dienen als Absicherung, falls man eine Konfiguration nicht explizit bedient
- **Alternative Ressourcen müssen den gleichen Namen tragen**
  - > zum Beispiel: Bilder, die sich ausschließlich in ihrer Auflösung unterscheiden

# Default- und Alternative Ressourcen

- Android wendet die „best-matching“ Ressource an
- „Matching“ erfolgt über die gegebenen Kriterien (Bildschirmkonfiguration, Orientierung, Sprache)
- Kernidee:
  - Benutzeroberfläche in unterschiedlichen Ausprägungen zur Verfügung stellen



## default resources

vs.

## alternative resources

- Für alle beliebigen Gerätekonfigurationen anwendbar
- Sollten immer vorhanden sein (für alle Konfigurationen)

Im „res“ Verzeichnis:  
<ResourceName>

- Spezifisch für unterschiedliche Gerätekonfigurationen
- Sind optional, aber tragen enorm zur UserExperience bei

Im „res“ Verzeichnis:  
<ResourceName> (- <ConfigQualifier>)+

Gleiche Ressourcen tragen dabei in jedem Ressourcenverzeichnis denselben Namen zur eindeutigen Identifikation!

# Zugriff auf Ressourcen

Code (.java Dateien)

**R.type.name**

„Dekodierungsfunktion“ notwendig:

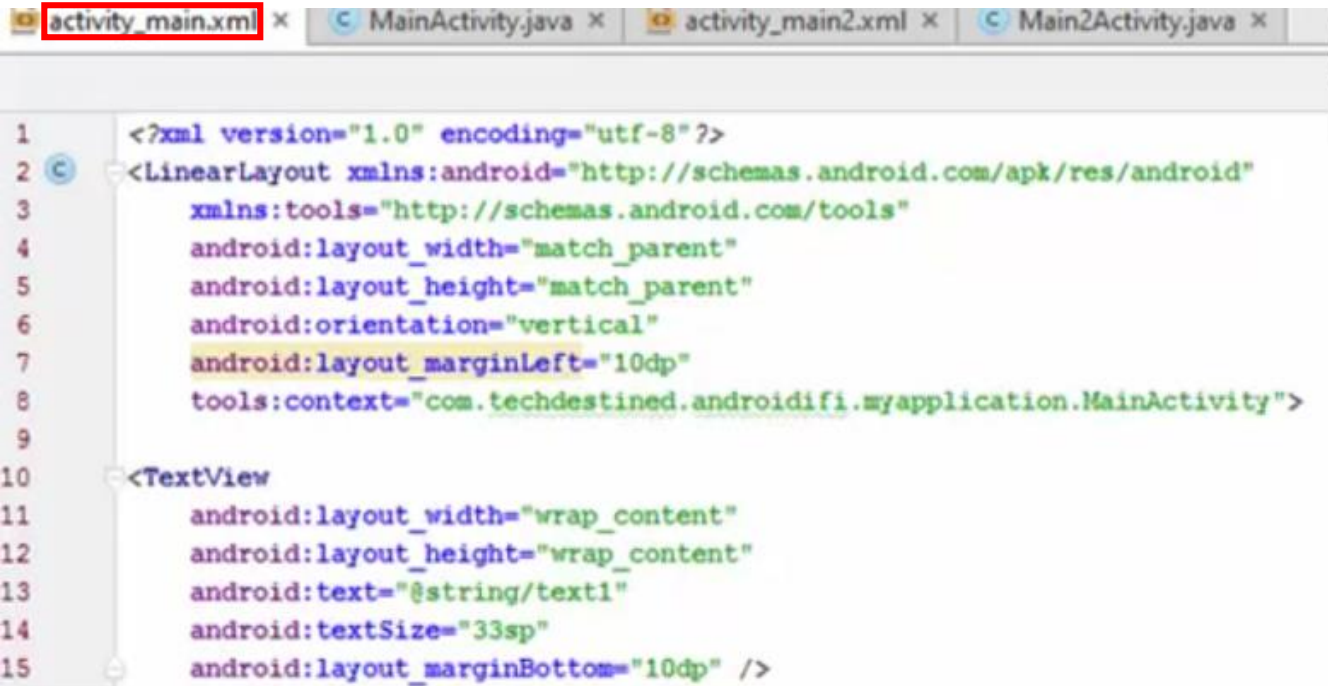
```
String helloWorld = getString(R.string.helloWorld);
```

Layout (.xml Dateien)

**@type/name**

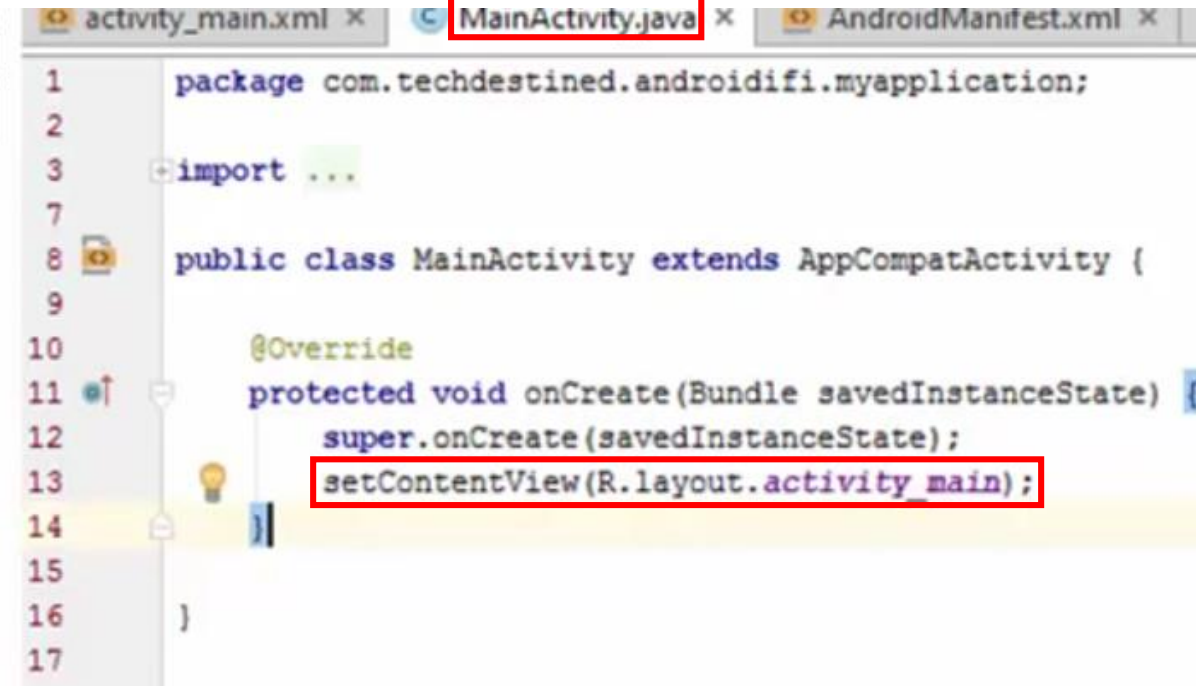
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="helloWorld">Hello World</string>
</resources>
```

# Layout und Code Dateien



```
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="vertical"
7     android:layout_marginLeft="10dp"
8     tools:context="com.techdestined.androidifi.myapplication.MainActivity">
9
10    <TextView
11        android:layout_width="wrap_content"
12        android:layout_height="wrap_content"
13        android:text="@string/text1"
14        android:textSize="33sp"
15        android:layout_marginBottom="10dp" />
```

.xml Layout-Datei (MVP -> View)



```
1 package com.techdestined.androidifi.myapplication;
2
3 import ...
4
5
6
7
8 public class MainActivity extends AppCompatActivity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_main);
14     }
15
16 }
17
```

.java Code-Datei (MVP -> Presenter)

Jeder Code-Datei wird eine entsprechende Layout-Datei mit „setContentView(...)“ zugeordnet  
(Prinzip : Lose Kopplung)

# Automatische Generierung von IDs

- jedes Element im Layout besitzt eine ID, die automatisch generiert wird

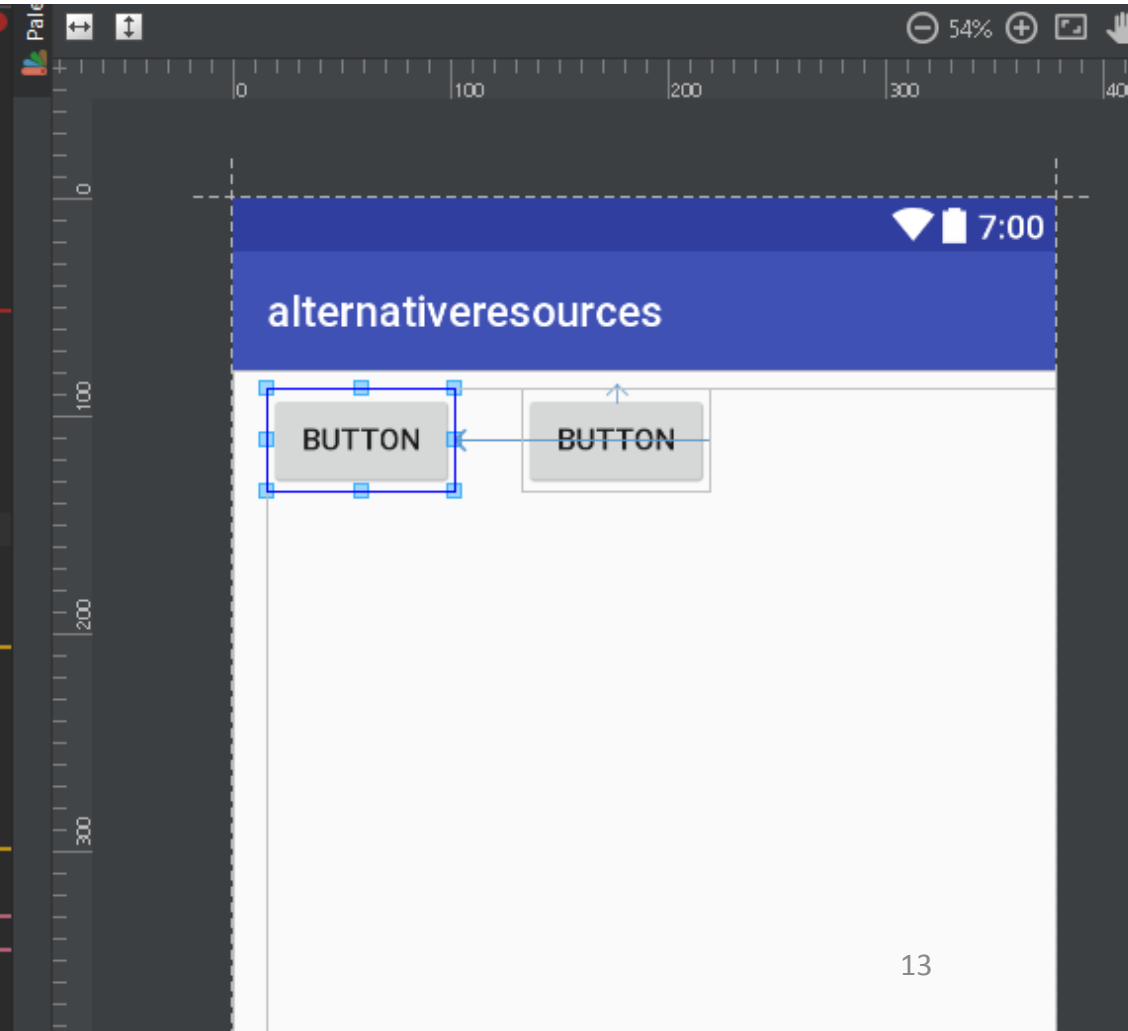
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.boss.patrick.alternativeresources.MainActivity">

    <RelativeLayout
        android:layout_width="368dp"
        android:layout_height="495dp"
        app:layout_constraintRight_toRightOf="parent"
        tools:layout_editor_absoluteY="8dp">

        <Button
            android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button" />

        <Button
            android:id="@+id/button3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button"
            android:layout_alignParentTop="true"
            android:layout_toRightOf="@+id/button2"
            android:layout_toEndOf="@+id/button2"
            android:layout_marginLeft="31dp"
            android:layout_marginStart="31dp" />

    </RelativeLayout>
</android.support.constraint.ConstraintLayout>
```



# R.java – IDs der Ressourcen

The screenshot displays the Android Studio IDE. On the left, the 'Project' view shows the file structure of the 'alternativeresources' project. The 'app' directory is expanded, showing subdirectories like 'build', 'source', and 'r'. The 'R' class is highlighted under the 'r' directory. On the right, the 'R.java' file is open, showing a list of resource IDs. The search bar at the top of the editor is set to 'button2'. The list of IDs includes various static final integers, with 'button2' and 'button3' highlighted by a red box.

Project structure (left):

- alternativeresources
  - .gradle
  - .idea
  - app
    - build
      - generated
        - res
      - source
        - aidl
        - apt
        - buildConfig
        - r
          - androidTest
          - debug
            - android.support
            - com.boss.patrick.alternativeresources

R.java content (right):

```
2314 public static final int action_image=0x7f0b0061;
2315 public static final int action_menu_divider=0x7f0b0002;
2316 public static final int action_menu_presenter=0x7f0b0003;
2317 public static final int action_mode_bar=0x7f0b004c;
2318 public static final int action_mode_bar_stub=0x7f0b004b;
2319 public static final int action_mode_close_button=0x7f0b002e;
2320 public static final int action_text=0x7f0b0062;
2321 public static final int actions=0x7f0b0070;
2322 public static final int activity_chooser_view_content=0x7f0b002f;
2323 public static final int add=0x7f0b001b;
2324 public static final int alertTitle=0x7f0b0043;
2325 public static final int action_menu_divider=0x7f0b0002;
2326 public static final int action_menu_presenter=0x7f0b0003;
2327 public static final int basic=0x7f0b000f;
2328 public static final int beginning=0x7f0b0022;
2329 public static final int bottom=0x7f0b002a;
2330 public static final int button2=0x7f0b005e;
2331 public static final int button3=0x7f0b005f;
2332 public static final int buttonPanel=0x7f0b0036;
2333 public static final int cancel_action=0x7f0b0064;
2334 public static final int chains=0x7f0b0010;
2335 public static final int checkbox=0x7f0b0046;
2336 public static final int chronometer=0x7f0b006c;
2337 public static final int collapseActionView=0x7f0b0026;
2338 public static final int contentPanel=0x7f0b0039;
2339 public static final int custom=0x7f0b0040;
2340 public static final int customPanel=0x7f0b003f;
2341 public static final int decor_content_parent=0x7f0b004d;
```

# Relation von DPI und DIP (DP)

Idee: Der Abstand von **x dp** soll auf einem **hoch** aufgelöstem Bildschirm genau gleich aussehen wie auf einem **niedrig** aufgelöstem.

DPI = Dots per inch = Anzahl der Punkte pro Zoll = Pixeldichte

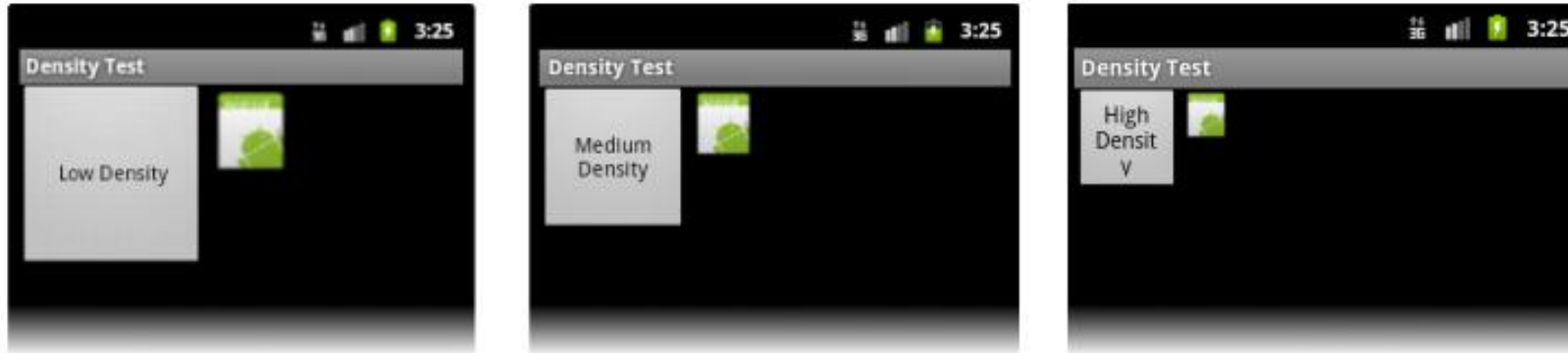
DIP = Density Independent Points = geräteunabhängige Punkte = universelle Einheit

Wenn das Gerät eine physikalische Bildschirmgröße von 160dpi hat, dann ist **1px = 1dp**

$$\text{px} = \text{dp} * (\text{dpi} / 160)$$

**dpi/160** kann abgefragt werden mit `getResources().getDisplayMetrics().density`

# Geräteunabhängigkeit (Auflösung)

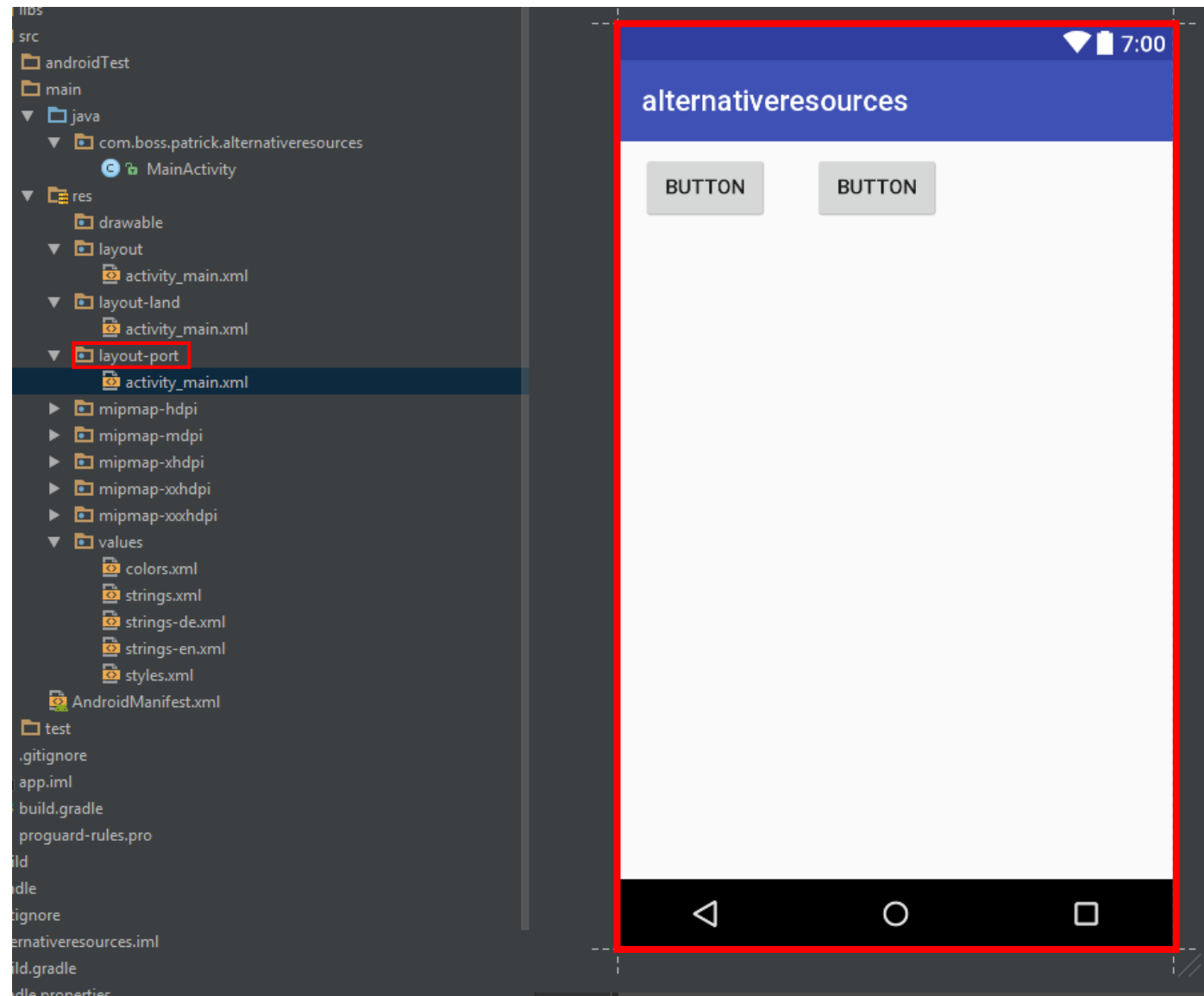


Oben wurden **gewöhnliche Pixel (px)** und unten **geräteunabhängige Punkte (dp)** genutzt

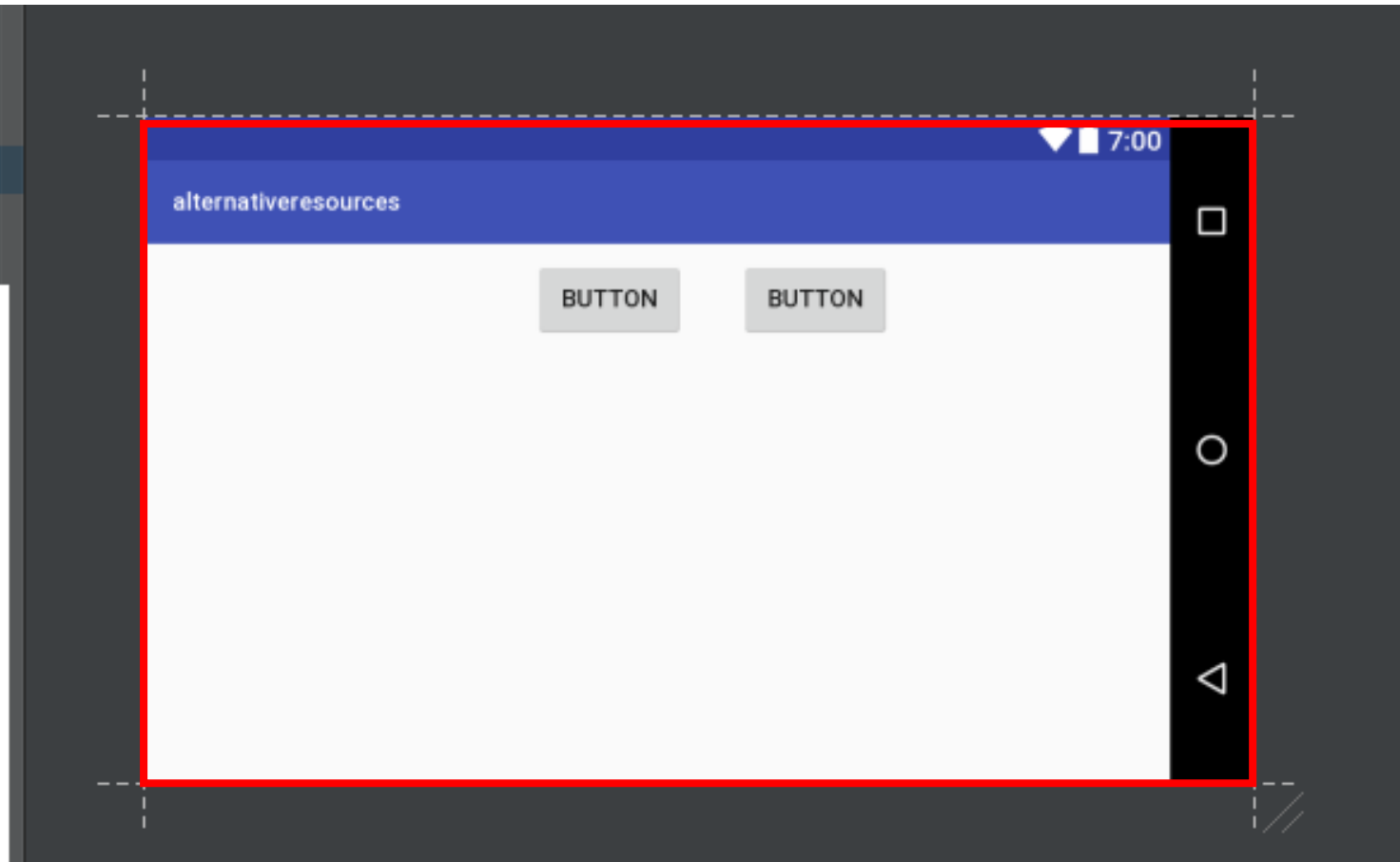
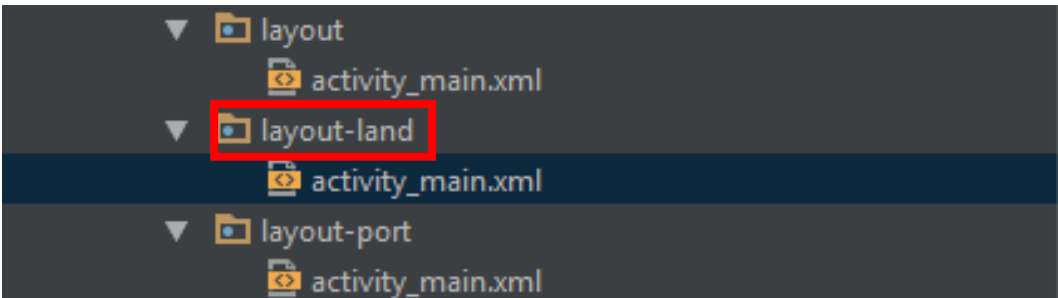




# Layout – Porträt – Orientierung



# Layout – Landschaft – Orientierung



# Erzwungene Orientierung

AndroidManifest.xml

```
<activity android:name=".YourActivity"  
    android:configChanges="orientation"  
    android:screenOrientation="(portrait|landscape)"/>
```

## Oder

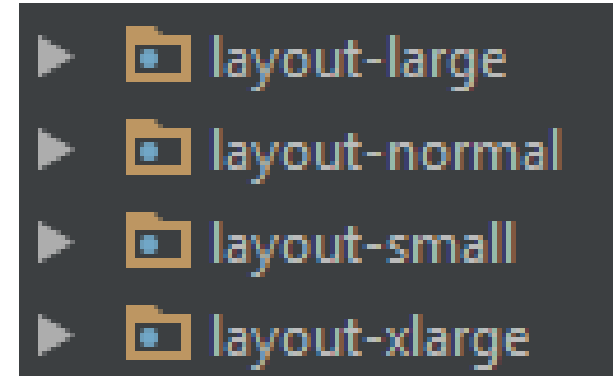
Activity.java

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_xyz);  
    this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE |  
                                ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);  
}
```

# Layout – Bildschirmgröße

## 4 verschiedene Bildschirmkategorien

- |           |   |     |              |                 |
|-----------|---|-----|--------------|-----------------|
| 1. small  | – | 2   | bis 3.2 Zoll | (320dp x 426dp) |
| 2. normal | – | 3.2 | bis 4 Zoll   | (320dp x 470dp) |
| 3. large  | – | 4   | bis 7 Zoll   | (480dp x 640dp) |
| 4. xlarge | – |     | ab 7 Zoll    | (720dp x 960dp) |



(optional: **–land** oder **–port** Suffix)

Grobe Einteilung ist allerdings nicht ausreichend  
für Oberflächen von Tablets geeignet

# Layout – Bildschirmgröße – neuer Mechanismus

## Idee:

Eine Mindestgröße in vertikaler oder horizontaler Richtung angeben.

Größenangabe bezieht sich auf den Bereich unterhalb der Actionbar, die Größe dieser wird **zuzüglich** drauf gerechnet.

## Breite:

- 320 dp für gängige Smartphones
- 480 dp für kleine Tablets
- 600 dp für 7-Zoll-Tablets
- 720 dp für 10-Zoll-Tablets

Neues Layout nach Orientierungswechsel:

**layout-w...dp**

Beispiel:

**res/layout-w320dp**

Mindestbreite von 320dp

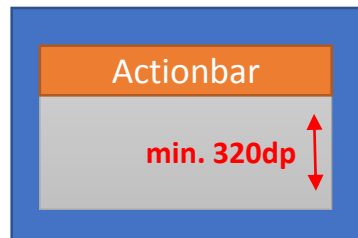


**layout-h...dp**

Beispiel:

**res/layout-h320dp**

Mindesthöhe von 320dp



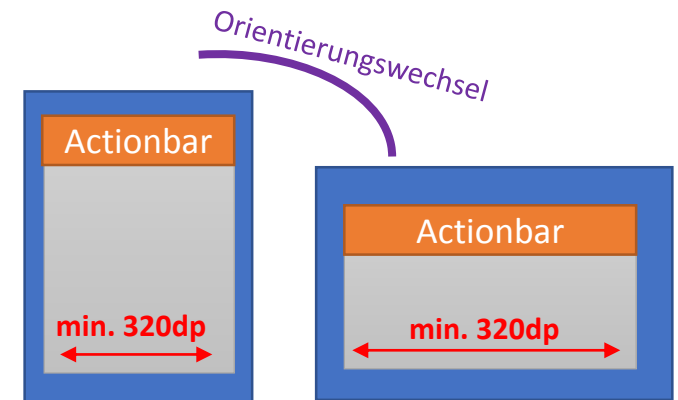
Orientierungswechsel bewirkt keine Neuwahl:

**layout-sw...dp**

Beispiel:

**res/layout-sw320dp**

Mindestbreite von 320dp  
unabhängig vom  
Orientierungswechsel



Manchmal reicht eine **ScrollView** aus, um kleinere Bildschirme zu bedienen

# Layout – Bildschirmgröße - Anordnung

Default Ressource

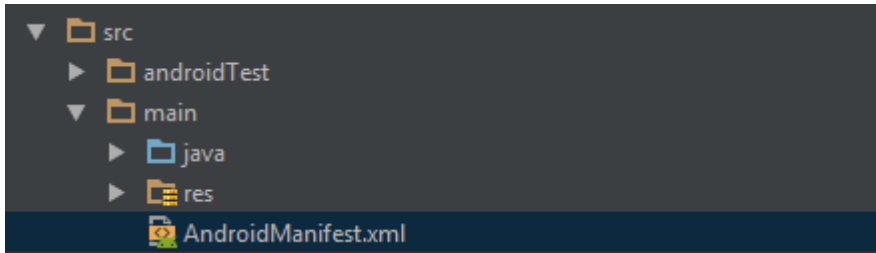


Für unterschiedliche Bildschirmgröße gibt es unterschiedliche Layouts.

Alternative Ressourcen



# Layout – Bildschirmgröße - Manifestdatei



## Download:

Google Play zeigt die App nur den Geräten mit den unterstützten Bildschirmgrößen an.

```
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android">
  ...
  <supports-screens
    android:smallScreens="true"
    android:normalScreens="true"
    android:largeScreens="true"
    android:xlargeScreens="false" />
  ...
</manifest>
```

Kleinstmögliche Breite mit **android:requiresSmallestWidthDp** festlegen

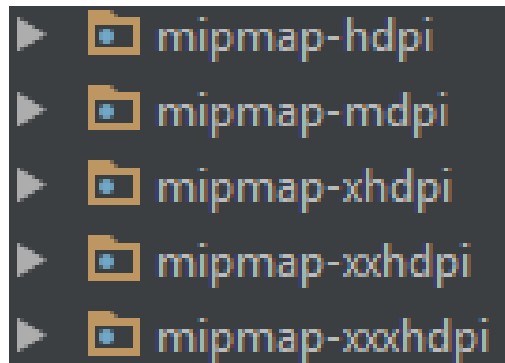
Man sollte immer alle passenden Bildschirmgrößen explizit angeben, sonst kann es evtl. zu Komplikationen aufgrund von unterschiedlichen SDKs führen!

# Grafiken

- werden in unterschiedlicher Auflösung in den entsprechenden Verzeichnissen gespeichert (drawable-xxx, mipmap-xxx)

- Idee:

Grafiken sollen bei jeder Oberfläche, unabhängig von technischen Details gleich wirken



1.	ldpi	niedrige Auflösung	120dpi
2.	mdpi	mittlere Auflösung	160dpi
3.	hdpi	hohe Auflösung	240dpi
4.	xhdpi	sehr hohe Auflösung	320dpi
5.	xxhdpi	sehr sehr hohe Auflösung	480dpi
6.	xxxhdpi	sehr sehr sehr hohe Auflösung	640dpi

Bilder im **drawable** bzw. **mipmap** (Icon) Verzeichnis ohne Spezifikationsuffix (alternative Ressourcen) werden automatisch skaliert, Bilder im **drawable-nodpi** Verzeichnis hingegen nicht!



# Erstellung von Grafiken

## 160dpi als Referenz

Gruppe	Auflösung	Skalierung
1. ldpi	120dpi	75%
2. mdpi	160dpi	<b>100%</b>
3. hdpi	240dpi	150%
4. xhdpi	320dpi	200%
5. xxhdpi	480dpi	300%
6. xxxhdpi	640dpi	400%

automatische Skalierung im **res/drawable** :

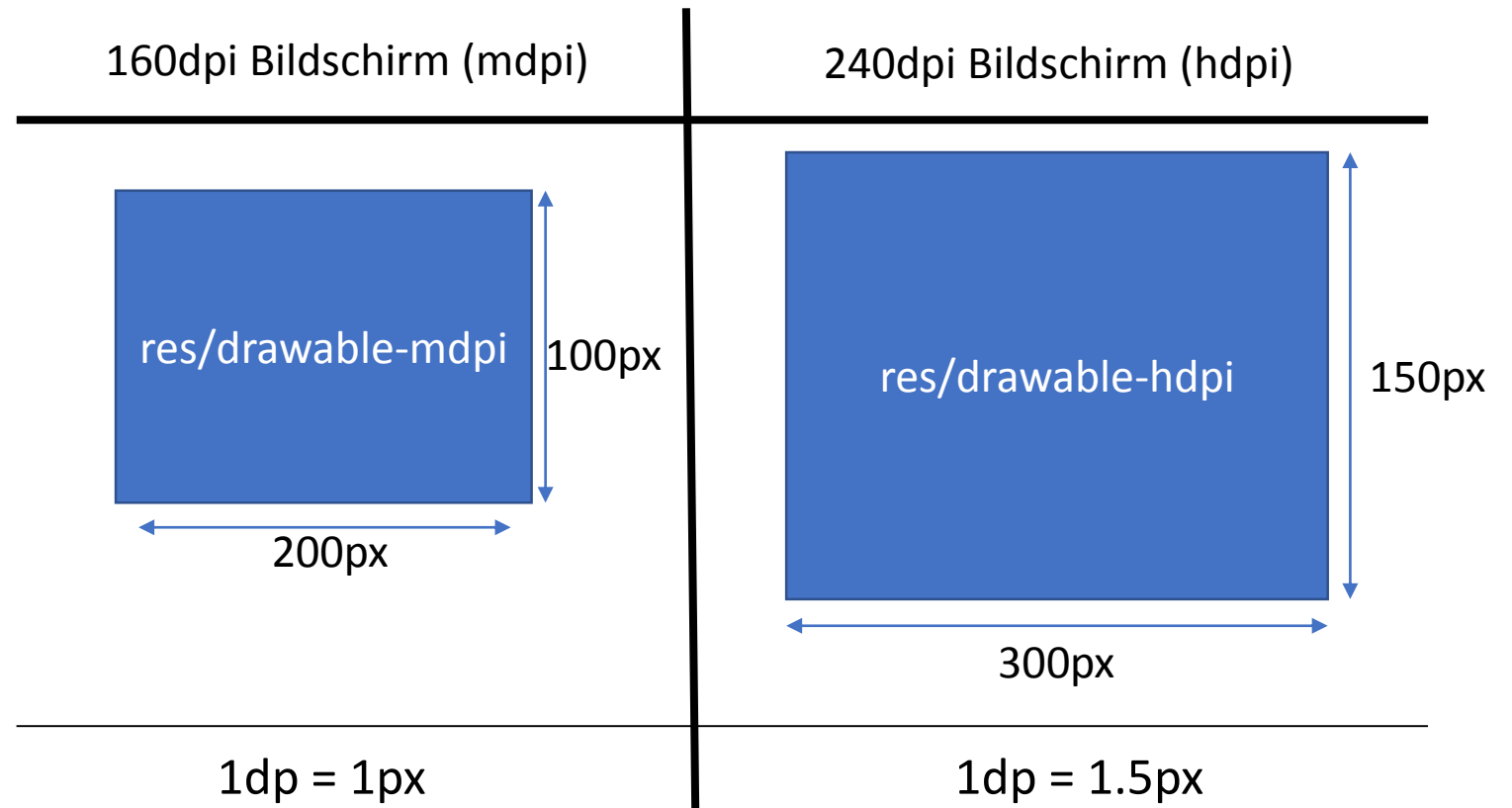


160dpi



320dpi

Bei einer **160dpi** Bildschirmgröße gilt: **1dp = 1px**



# Strings - Mehrsprachigkeit

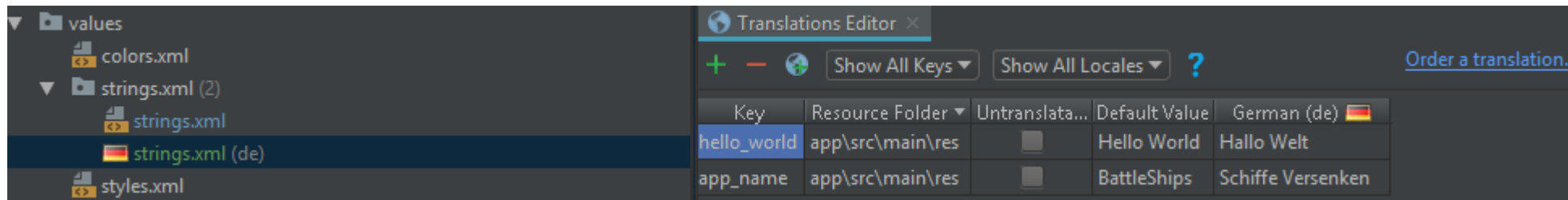
- zentrale Stelle für die Verwaltung aller Zeichenketten nicht im Code „hardcoden“!
- Default Ressource/Value: Englisch

manuell:

**strings-[länderspezifisches Suffix].xml**

Bspw: strings-de.xml , strings-fr.xml ...

Translation Editor:



Habt ihr noch Fragen  
???

# Quellen

- <http://www.satzgewinn.com/wp-content/uploads/2015/05/Sprachen.jpg>
- <http://s.hswstatic.com/gif/monitor-lcd-diagonal.jpg>
- <http://www.aremobil.de/b/1782-fuenf-jahre-android-diese-geraete-praegten-das-beliebte-system>
- <https://yt3.ggpht.com/-tUnSh4hL1b0/AAAAAAAAAAI/AAAAAAAAAA/AlY5-05CyFk/s900-c-k-no-mo-rj-c0x0000000/photo.jpg>
- <https://developer.android.com/guide/topics/resources/overview.html>
- <https://www.youtube.com/watch?v=WpiLCQX30pA&list=WL&index=11>
- [https://developer.android.com/guide/practices/screens\\_support.html](https://developer.android.com/guide/practices/screens_support.html)
- <https://support.google.com/l10n/answer/6341304?hl=en>
- Android Studio 2.0 Screenshots
- Android 7 : Das Praxisbuch für Entwickler von Thomas Künneth