Android - Ressource

Ressourcen

- Verwaltung von Informationen zur Benutzeroberfläche ähnlich wie Cascading Style Sheets.
- Ablage von Strings, Bildern etc. in einer Ordnerstruktur.

Beispiele

- Strings um zum Beispiel Eingabefelder zu beschriften.
- Bilder.
- Farben festlegen.
- Größenangaben als Platzhalter festlegen.
- Layouts.
- Menüs.
- Roh- und Multimediadateien.

Leibniz Universität Hannover

Speicherort

- Ablage in Unterordnern im Ordner *res*.
- Zum Beispiel werden Strings werden im Ordner *res/values* abgelegt.
- Der Name der Ordner gibt Auskunft über den Typ der Ressource, die Plattform, die Auflösung, die Sprache etc.

Strings

- Titel und Texte auf der Benutzeroberfläche.
- Speicherung in der Datei *strings.xml* in dem Ordner *res / values* .
- Ablage von String-Arrays ist möglich.

Beispiele für Strings in einer Layout-Datei

```
android:text="Bitte geben Sie eine Temperatur in Celcius ein"
android:contentDescription="Hinweis zur Eingabe der Temperatur"
```

- Strings werden durch die Anführungszeichen begrenzt.
- Die Warnung „Hardcoded Strings“ wird angezeigt.

Auslagerung in strings.xml

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

    <string name="app_name">
            Umrechnung von   Temperaturwerten
    </string>

    <string name="lblEingabeTemperatur">
             Bitte geben Sie eine  Temperatur in Celsius ein
    </string>

</resources>
```

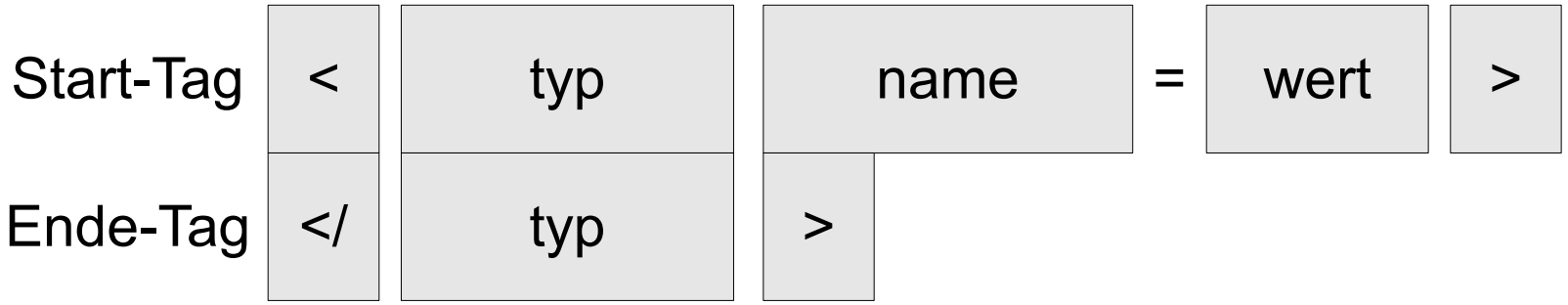
Leibniz Universität Hannover

Wurzel-Element

```
<resources>
</resources>
```

- Der XML-Tag <resources> … </resources> definiert alle eingebetteten Elemente als Ressource.
- Elemente, die in dem dem Ordner *res* und deren Unterordner abgelegt sind, nutzen den Tag als Wurzel.

Ressource als XML-Element



- Jede Ressource in einer XML-Datei hat einen bestimmten Typ. Der Typ wird als Elementname genutzt.
- Der der Name der Ressource wird durch das Attribut name festgelegt.
- Zwischen dem Start- und Ende-Tag wird der Wert der Ressource definiert.

Ressource „String“

```
<string name="app_name">
        Umrechnung von   Temperaturwerten
</string>
```

- Start-Tag: <string name="name">.
- Ende-Tag: </string>.
- Zwischen den Tags wird ein String gespeichert.
- Der Wert (der String) wird durch den Start- und Ende-Tag in einer Ressourcen-Datei angegeben.

Name der App

```
<string name="app_name">
        Umrechnung von   Temperaturwerten
</string>
```

- app_name ist ein vordefinierter Wert für das Attribut name.
- Der Name der App wird definiert.

Apostroph in Strings nutzen

```
<string name="hinweis">Guten Tag \'Teilnehmer\'</string>
<string name="hinweis">"Guten Tag 'Teilnehmer' "</string>
```

- Ein Apostroph muss mit Hilfe des Backslash maskiert werden.
- In einem String, begrenzt durch Anführungszeichen, kann ein Apostroph vorkommen.

Anführungszeichen in Strings nutzen

```
<string name="hinweis">Guten Tag \"Teilnehmer\"</string>
<string name="hinweis">'Guten Tag "Teilnehmer" '</string>
```

- Anführungszeichen müssen mit Hilfe des Backslash maskiert werden.
- In einem String, begrenzt durch Apostrophs, können Anführungszeichen vorkommen.

Leibniz Universität Hannover

Nutzung von Unicode-Zeichen

```
<string name="lblUmrechnung">
                    &#8230; in &#8230;  umwandeln
</string>

 <string name="lblUmrechnungOhneUnicode">
                  ... in ... umwandeln
</string>
```

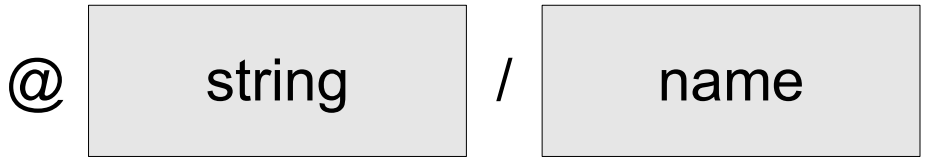
- Unicode-Zeichen werden mit Hilfe einer vierstelligen, hexadezimalen Zahl codiert (&#nnnn).
- Siehe https://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references

Ressource „String“ in der Layout-Datei

```
android:text="@string/lblEingabeTemperatur"
android:contentDescription="@string/hinweisLabel"
```

- Das Attribut muss dem Typ der Ressource entsprechen.
- Dem Attribut android:text kann nur ein String zugewiesen werden, aber kein Bild.

Definition der Ressource



- Jeder Ressourcen-ID wird in XML ein Klammeraffen vorangestellt.
- Rechts vom Schrägstrich wird der Name der Ressource angegeben.
- Links vom Schrägstrich wird der Ressourcen-Typ angegeben. In diesem Beispiel wird auf einen String verwiesen.

Name der Ressource in R.java

@ string / name

- Deklaration in der Datei:

  public static final int hinweisLabel=0x7f030003;
- Der Kasten „name“ symbolisiert einen x-beliebigen Variablenamen aus der Datei *R.java*.
- Der Variablenname beschreibt ein Attribut einer bestimmten Klasse.
- Die Variable ist entsprechend ihres Typs in einer Datei in einem Unterordner des Ordners *res* definiert.

Typ der Ressource in R.java

@ | string | / | name |

- Deklaration der Klasse in der Datei:

  public static final class string { }
- Der Ressourcen-Typ entspricht einem Klassennamen in der Datei *R.java*.
- Die Klasse bezieht sich auf eine bestimmte Datei in einem Unterordner des Ordners res. In diesem Beispiel *res / values / strings.xml*.

Nutzung im Code

```
package de.example.Android_Example04;
import android.app.Activity;
import android.os.Bundle;
import android.content.res.Resources;

public class MainActivity extends Activity{
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Resources res = getResources();
        String ausgabeText = res.getString(R.string.lblUmrechnung);
    }
}
```

import-Anweisung

```
import android.content.res.Resources;
```

- Die Klasse android.content stellt Subklassen für den Zugriff und die Veröffentlichung von Daten in einer App bereit.
- Die Klasse android.content.res ist eine Subklasse für das Ressourcen-Management einer App.
- Die Klasse android.content.res.Resources regelt den Zugriff auf Dateien im Ordner res.

Ressource holen

```
Resources res = getResources();
```

- Die Methode getResource() liefert das Ressource-Objekt der Applikation zurück.

String-Ressouce im Code nutzen

```
String ausgabeText = res.getString(R.string.lblUmrechnung);
```

- Das Ressource-Objekt liefert mit Hilfe der Methode getString() einen String aus der entsprechenden Ressource-Datei zurück.
- Der Methode wird die ID der Ressource übergeben. R.string verweist auf die entsprechende Klasse in der Datei *R.java*. In dieser Klasse ist eine konstantes Attribut lblUmrechnung definiert.

Leibniz Universität Hannover

Größenangaben

- Definition in der Datei *dimensions.xml* in dem Ordner *res / values*.
- Standardgrößen für Widgets.
- Standard-Schriftgrößen für Texte.

Leibniz Universität Hannover

Auslagerung in dimensions.xml

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

<resources>
    <dimen name="labelSchrift">12sp</dimen>
    <dimen name="abstandToTitel">10dp</dimen>
    <dimen name="abstandToControl">3dp</dimen>
    <dimen name="btnBreite">150dp</dimen>

</resources>
```

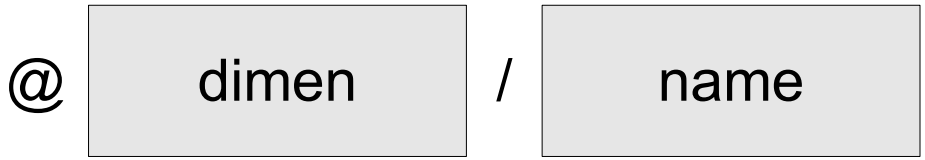
XML-Element „Dimension“

```
<dimen name="labelSchrift">12sp</dimen>
```

- <dimen name="name"> wert </dimen>.
- Zwischen dem Start- und Ende-Tag wird der Wert - die Dimension - definiert.
- Eine Dimension besteht aus einer Ganzzahl und der Maßeinheit.

Ressource „Dimension“ in der Layout-Datei

```
android:layout_width="@dimen/btnBreite"
```



Bilder

- Bilddateien, die als Hintergrund oder Symbol genutzt werden, werden in den Ordner *res / drawble* abgelegt.
- Bilder, in dem Format „png“, „jpg“ oder „gif“. Das Format „png“ sollte bevorzugt werden.

Leibniz Universität Hannover

Dateiname

- Der Dateiname darf nur aus den Kleinbuchstaben a..z und den Ziffern 0..9 bestehen.
- In dem Dateinamen werden keine Umlaute, Sonderzeichen oder Leerzeichen verwendet.
- Der Dateiname spiegelt den Inhalt der Datei wieder.

Bildgröße und -auflösung

- Die Bildgröße sollte entsprechend des Ausgabegerätes gewählt werden.
- In Abhängigkeit der genutzten Geräteauflösung sollten die Bilder in verschiedenen Auflösungen vorhanden sein. Die Auflösung spiegelt sich im Namen der Ordner wieder.

Ressource „Bild“ in der Layout-Datei

```
android:background="@drawable/thermometer"
```

@ drawable / dateiname

Ressourcen als Standard

- Die Angaben werden als Standard für die verschiedensten Geräte genutzt.
- Texte, Größenangaben etc. in dem Verzeichnis *res / values*.
- Layouts in dem Verzeichnis *res / layout*.
- Bilder in dem Verzeichnis *res / drawable*.

Alternative Ressourcen

- Die Ressourcen werden in Abhängigkeit des Bildschirms, des Formates, der Sprache der Auflösung etc. gespeichert.
- Layouts in dem Verzeichnis *layout-land* werden für die Darstellung im Querformat genutzt.
- Bilder in dem Verzeichnis *res / drawable-hdpi*. Bilder, Grafiken in einer Auflösung von ca. 240 dpi.

Beispiel

Alternative für	Beschreibung	Kürzel
Orientierung	Hochformat (senkrecht)	port
	Querformat (waagerecht)	land
Bildschirmdichte	Geringe Bildschirmdichte. Ca. 120 dpi	ldpi
	Mittlere Bildschirmdichte. Ca. 160 dpi.	mdpi
	Hohe Bildschirmdichte. Ca. 240 dpi.	hdpi
	Extra hohe Dichte. Ca. 320 dpi.	xhppi
Sprache	deutsch	de
	englisch	en
	französich	fr

- Siehe http://developer.android.com/guide/topics/resources/providing-resources.html.

Beispiel: Hoch- / Querformat

- Rechtsklick auf den Ordner *res*.
- *New* – *Folder* im Kontextmenü des Ordners *res*.
- Eingabe des Foldernamen *layout-land* für Querformat. Klick auf *Finish*.
- Klick auf die Standard-Layoutdatei in dem Ordner *layout.* <STRG>+<C> kopiert die Datei in die Zwischenablage.
- Klick auf den Ordner *layout-land*. <STRG>+<V> fügt die Datei aus der Zwischenablage in den Ordner *layout-land* ein.
- Die Layout-Datei für das Querformat wird geöffnet, verändert und gespeichert.
- <STRG>+<F12> stellt die Emulation zum Testen von Hoch- auf Querformat und umgekehrt um.