



MODUL 335

MOBILE APPLIKATION



MODUL 335
MOBILE APPLIKATION

ANDROID INSTALLATION

WAS WIRD BENÖTIGT?

- ▶ Java Runtime (jre) / Java Development Kit (jdk)
 - ▶ <http://oracle.com>
- ▶ Android Studio (oder andere IDE)
 - ▶ <https://developer.android.com>
- ▶ Android Gerät (kann auch emuliert werden)
 - ▶ z.B. Nexus 5x

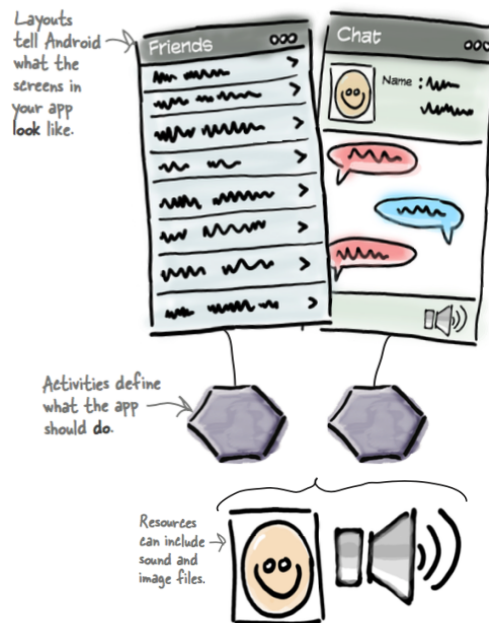


MODUL 335 - 01
MOBILE APPLIKATION

MEINE ERSTE APPLIKATION

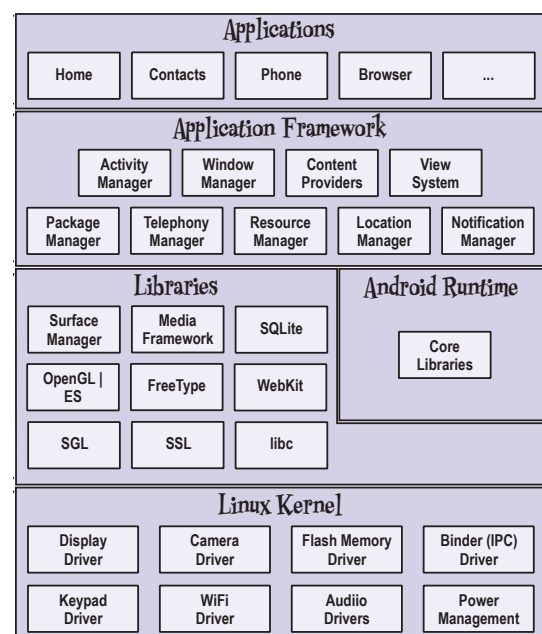
AUS WAS BESTEHT EINE APP

- ▶ Layouts definieren das Aussehen der Screens
 - ▶ Eine typische App besteht aus einem oder mehreren Screens
 - ▶ Die Layouts werden mittels xml definiert
- ▶ Activities definieren das Verhalten
 - ▶ Im Layout wird das Aussehen definiert, das Verhalten in den Activities
 - ▶ Activities werden in Java geschrieben
- ▶ Ressourcen
 - ▶ Zusätzlich zu den Layout und Activities werden manchmal weitere Ressourcen benötigt
 - ▶ z.B. Bilder oder zusätzliche Daten



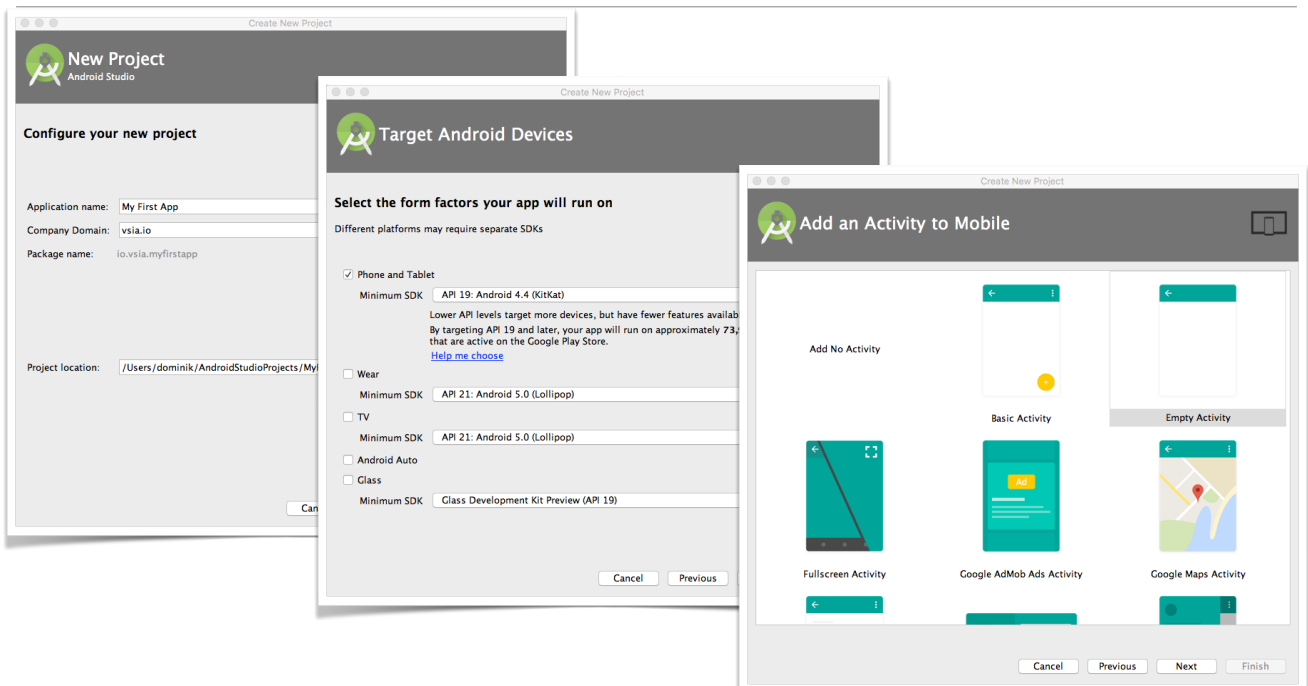
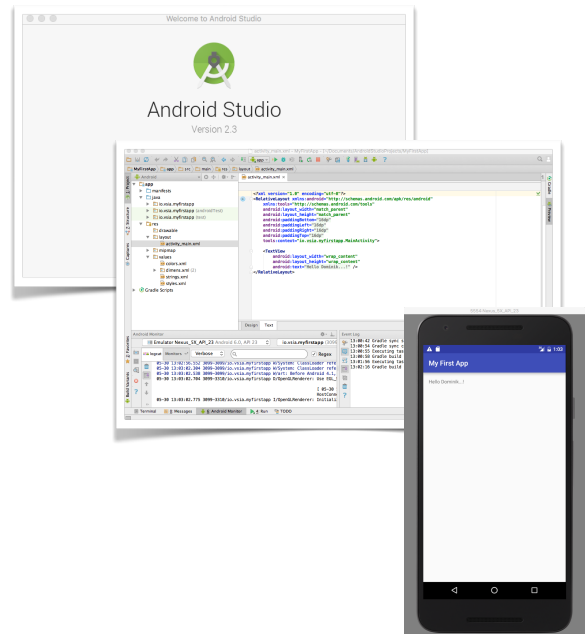
ANDROID PLATFORM

- ▶ Applications
 - ▶ Android stellt einige Core Apps bereit (z.B. Contacts, Calendar, Maps)
- ▶ Application Framework
 - ▶ Alle Apps nutzen diese APIs
- ▶ Libraries
 - ▶ Unter den APIs gibt es eine Reihe von C / C++ Bibliotheken (kein direkter Zugriff)
- ▶ Android Runtime
 - ▶ Jede App läuft in einem eigenen Prozess
- ▶ Linux Kernel
 - ▶ Die Basis bildet ein Linux Kern (Treiber, Memory Management, Sicherheit, etc.)



MEINE ERSTE APP

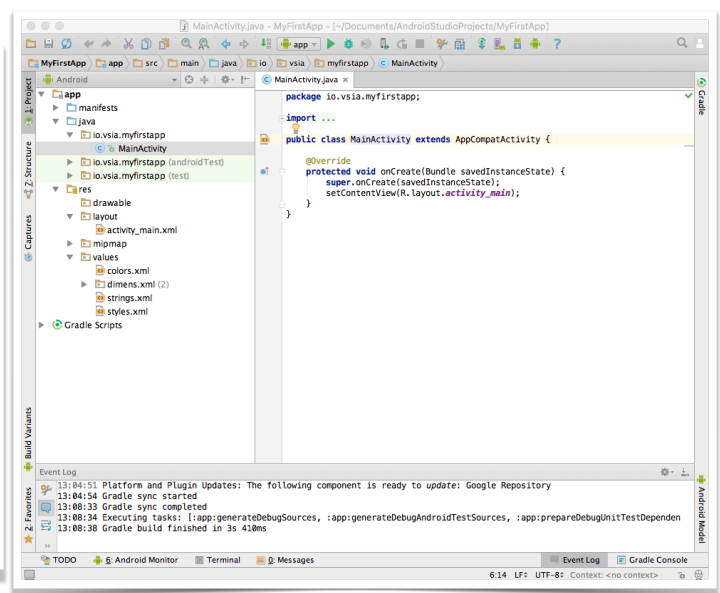
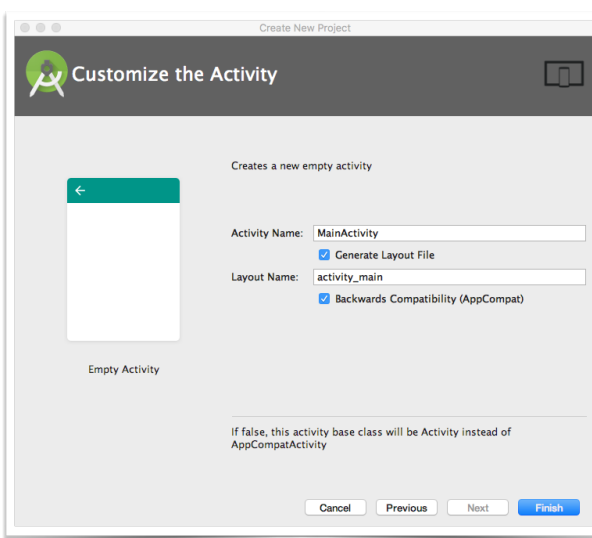
- ▶ Installation Entwicklungsumgebung
- ▶ Erstellen der ersten Applikation
- ▶ Applikation auf Gerät laufen lassen
- ▶ Applikation anpassen



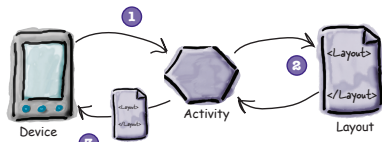
ANDROID VERSION DISTRIBUTION

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
2.3 Gingerbread	10	
4.0 Ice Cream Sandwich	15	97.4%
4.1 Jelly Bean	16	95.2%
4.2 Jelly Bean	17	87.4%
4.3 Jelly Bean	18	76.9%
4.4 KitKat	19	73.9%
5.0 Lollipop	21	40.5%
5.1 Lollipop	22	24.1%
6.0 Marshmallow	23	4.7%

MEINE ERSTE APP



SCHEMATISCHER ABLAUF

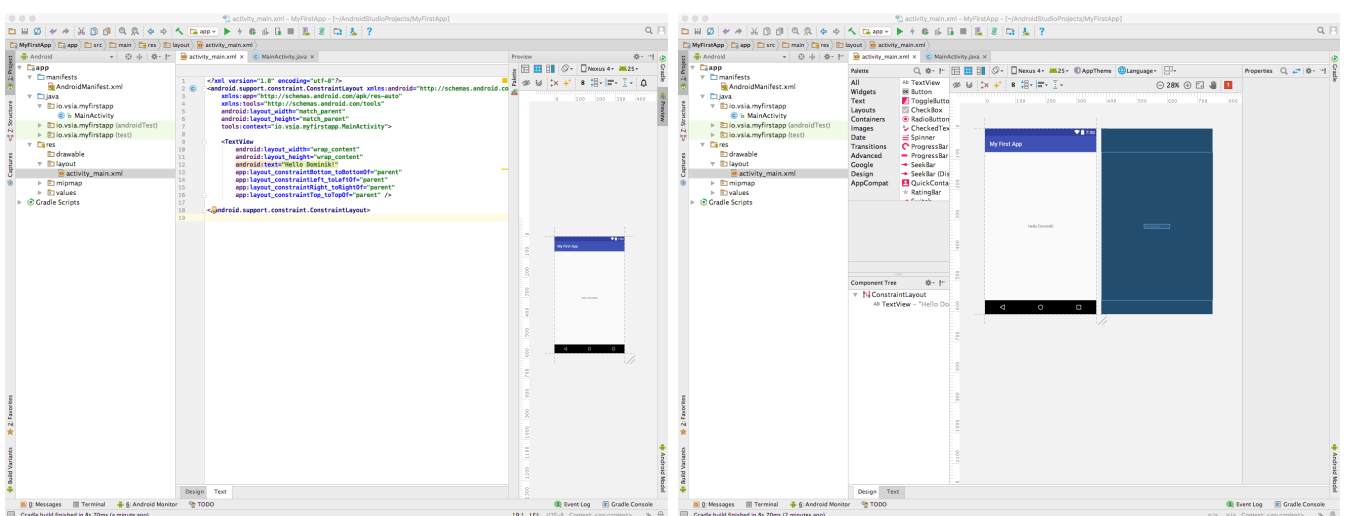


1. Gerät lädt die App und erstellt das Activity Objekt
2. Das Activity Objekt spezifiziert das Layout
3. Die Activity übergibt das Layout dem Gerät zur Anzeige



4. Der User interagiert mit dem dargestellten Layout
5. Die Activity reagiert auf die Interaktion
6. Die Activity aktualisiert die Anzeige
7. welche dem User angezeigt wird

CODE VS. DESIGN



LAYOUT: ACTIVITY_MAIN.XML

```
<?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="io.vsia.myfirstapp.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello Dominik!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

ACTIVITY: MAINACTIVITY.JAVA

```
package io.vsia.myfirstapp;

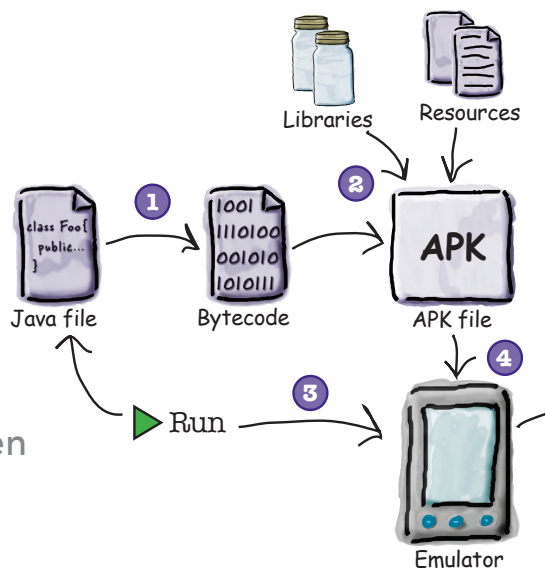
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

APPLIKATION LAUFEN LASSEN

1. Java Source wird Kompiliert
2. AndroidApplicationPackage (APK) wird erstellt
3. Sollte der Emulator noch nicht laufen, wird er gestartet
4. Das APK wird auf den Emulator geladen
5. Die Applikation wird gestartet



TEXT IN RESOURCE FILE AUSLAGERN – GENERIERTE DATEIEN

activity_main.xml

```

<?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="io.vsia.myfirstapp.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello Dominik!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>

```

strings.xml

```

<resources>
    <string name="app_name">My First App</string>
</resources>

```

TEXT IN RESOURCE FILE AUSLAGERN

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="io.vsia.myfirstapp.MainActivity">

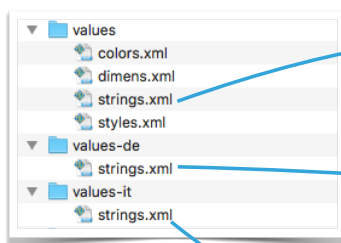
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_text" />
</RelativeLayout>
```

strings.xml

```
<resources>
    <string name="app_name">My First App</string>
    <string name="hello_text">My first Application!</string>
</resources>
```

TEXT IN RESOURCE FILE AUSLAGERN - MEHRSPRACHIGKEIT

Ein Ordner* für jede Sprache



strings.xml

```
<resources>
    <string name="app_name">My First App</string>
    <string name="hello_text">My first Application!</string>
</resources>
```

strings.xml

```
<resources>
    <string name="app_name">My First App</string>
    <string name="hello_text">Meine erste Applikation!!!!</string>
</resources>
```

strings.xml

```
<resources>
    <string name="app_name">My First App</string>
    <string name="hello_text">La mia prima applicazione!!!!</string>
</resources>
```

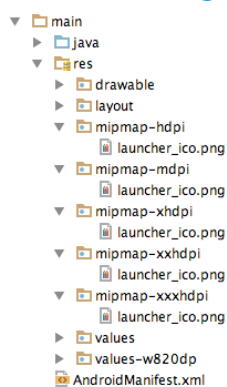
*Ordner wird beim Anlegen eines neuen string File automatisch erstellt

APP ICON

▶ ldpi	120 dpi, Low density screen	36*36 px
▶ mdpi	160 dpi, Medium density screen	48*48 px
▶ hdpi	240 dpi, High density screen	72*72 px
▶ xhdpi	320 dpi, Extra-high density screen	96*96 px
▶ xxhdpi	480 dpi, Extra-extra-high density screen	144*144 px
▶ xxxhdpi	640 dpi, Extra-extra-extra-high density screen	192*192 px

APP ICON

New => Image Asset

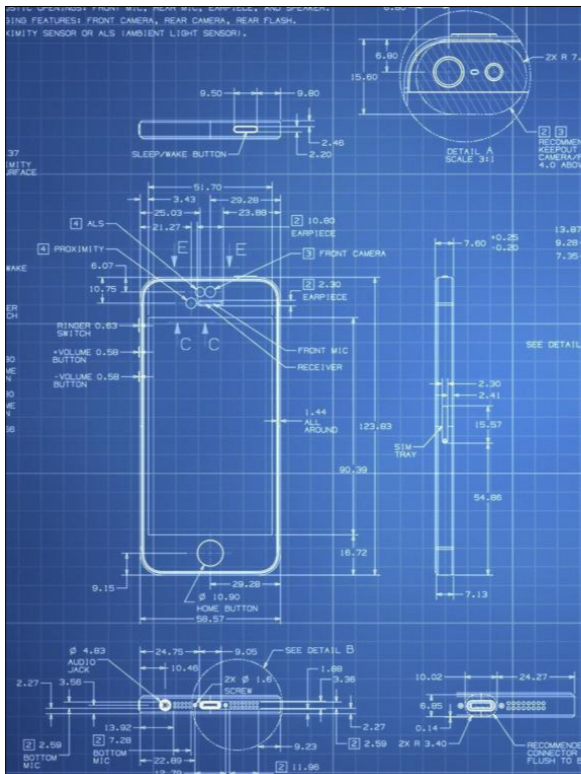


```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="..."
    package="io.vsia.bmicalculator">

    <application
        ...
        android:icon="@mipmap/launcher_ico"
        ...
    </application>
</manifest>
```

Icon Generator:

<https://romannurik.github.io/AndroidAssetStudio/icons-launcher.html>



MODUL 335 - 02

MOBILE APPLIKATION

LAYOUTS

MASSEINHEITEN

- ▶ px ⇨ Pixels - entspricht den eigentlichen pixel des screens
- ▶ in ⇨ Inches - basierend auf der physische Grösse des Bildschirms (2.54cm)
- ▶ mm ⇨ Millimeters - basierend auf der physische Grösse des Bildschirms
- ▶ pt ⇨ Points - 1/72 eines Inches
- ▶ dp ⇨ Density-independent Pixels - Eine abstrakte Grösseneinheit bezogen auf einen 160 dpi Bildschirm. 1dp = 1pixel bei 160 dpi.
Dieses Verhältnis ändert bezogen auf die Pixeldichte (dpi)
- ▶ sp ⇨ Scale-independent Pixels - Wie dp, allerdings zusätzlich skaliert mit der Schriftgrössen Anpassung

GRUNDLAYOUTS

▶ RelativeLayout

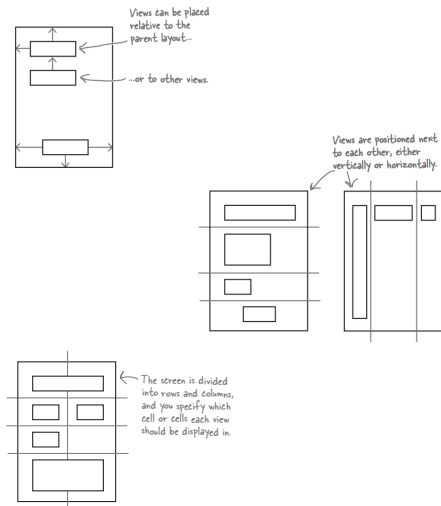
- ▶ Alle Positionen sind relative zu anderen Komponenten oder Umgebenden Komponenten

▶ LinearLayout

- ▶ Alle Komponenten werden entweder Vertikal oder Horizontal angeordnet

▶ GridLayout

- ▶ Bei einem GridLayout werden die Komponenten in „Zeilen“ und „Spalten“ angeordnet



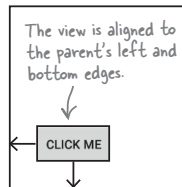
RELATIVE LAYOUT

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="io.vsia.layoutdemo.RelativeLayout">
</RelativeLayout>
```

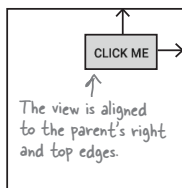
```
<resources>
    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
</resources>
```

RELATIVE LAYOUT – POSITIONIERUNG GEGENÜBER PARENT LAYOUT

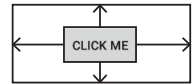
`android:layout_alignParentBottom=""`
`android:layout_alignParentLeft=""`



`android:layout_alignParentRight=""`
`android:layout_alignParentTop=""`



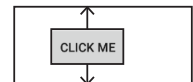
`android:layout_centerInParent=""`



`android:layout_centerHorizontal=""`

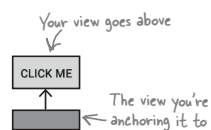


`android:layout_centerVertical=""`



RELATIVE LAYOUT – POSITIONIERUNG RELATIVE ZU ANDEREN VIEWS

`android:layout_above=""`



`android:layout_below=""`



`android:layout_alignTop=""`



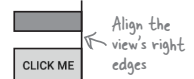
`android:layout_alignBottom=""`



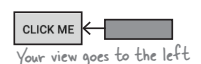
`android:layout_alignLeft=""`



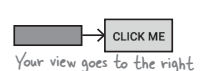
`android:layout_alignRight=""`



`android:layout_toLeftOf=""`



`android:layout_toRightOf=""`

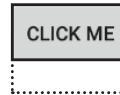


RELATIVE LAYOUT – ABSTAND ZU ANDEREN VIEWS

`android:layout_marginTop=""`



`android:layout_marginBottom=""`



`android:layout_marginLeft=""`



`android:layout_marginRight=""`

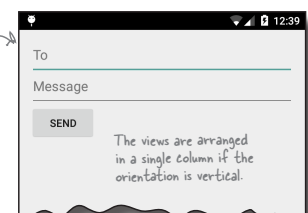


LINEAR LAYOUT

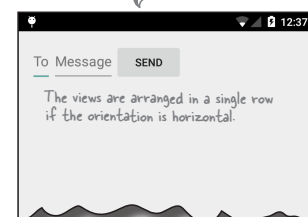
vertical oder horizontal

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="io.vsia.layoutdemo.LinearLayout">
</LinearLayout>
```

A linear layout with a vertical orientation.



A linear layout with a horizontal orientation.



LINEAR LAYOUT - VIEWS

gravity bestimmt die
Position des Inhalts

hint zeigt dem User
ein Hinweis an

```
<EditText
    android:id="@+id/to"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="To" />

<EditText
    android:id="@+id/message"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:gravity="top"
    android:hint="Message" />

<Button
    android:id="@+id/buttonSend"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="right"
    android:text="SEND" />
```

Höhe wird dynamisch bestimmt

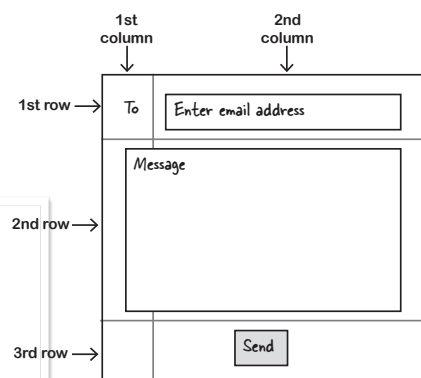
Mittels layout_weight kann sich
die View ausdehnen

layout_gravity bestimmt die
Position der View

GRID LAYOUT

definiert die Anzahl der Spalten

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="2"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="io.vsia.layoutdemo.GridLayout">
</GridLayout>
```



GRID LAYOUT

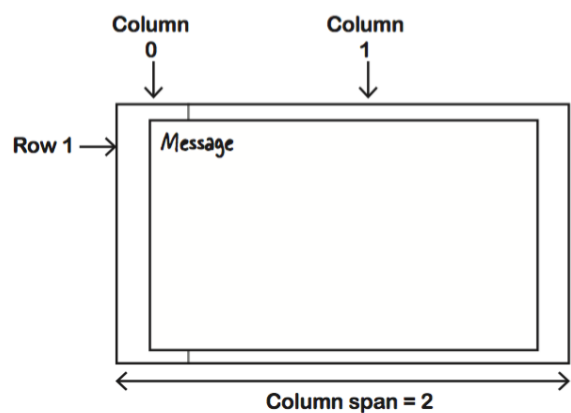
```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="0"
    android:layout_row="0"
    android:text="To" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="1"
    android:layout_gravity="fill_horizontal"
    android:layout_row="0"
    android:hint="E-Mail" />
```



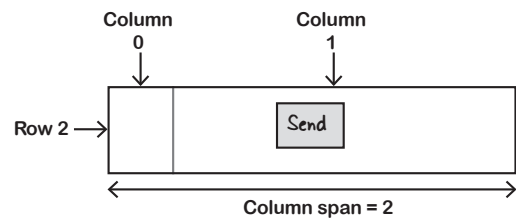
GRID LAYOUT

```
<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="0"
    android:layout_columnSpan="2"
    android:layout_gravity="fill"
    android:layout_row="1"
    android:gravity="top"
    android:hint="Message" />
```



GRID LAYOUT

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_column="0"  
    android:layout_columnSpan="2"  
    android:layout_gravity="center_horizontal"  
    android:layout_row="2"  
    android:text="Send" />
```



MODUL 335 - 03 MOBILE APPLIKATION KOMPONENTEN

KOMPONENTEN – TextView

```
<TextView
    android:id="@+id/text_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="TextView"
    android:textSize="14sp" />
```

```
//TextView
TextView textView = (TextView)
findViewById(R.id.text_view);
textView.setText("Some other string");
```

KOMPONENTEN – EditText

```
<EditText
    android:id="@+id/edit_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="EditText"
    android:inputType="number" />
```



```
//EditText
EditText editText = (EditText)
findViewById(R.id.edit_text);
String text = editText.getText().toString();
```

z.B.
phone | textPassword | textCapSentences | textAutoCorrect

KOMPONENTEN – Button

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onButtonClicked"
    android:text="Button" />
```

```
// Called when the button is clicked
public void onButtonClicked(View view) {
    //Do something in response to button click
}
```

KOMPONENTEN – ToggleButton

```
<ToggleButton
    android:id="@+id/toggle_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onToggleButtonClicked"
    android:textOff="Off"
    android:textOn="On" />
```

```
// Called when the toggle button is clicked
public void onToggleClicked(View view) {
    // Get the state of the toggle button.
    boolean on = ((ToggleButton)
view).isChecked();
    if (on) {
        // On
    } else {
        // Off
    }
}
```

KOMPONENTEN - Switch

```
<Switch
    android:id="@+id/switch_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onSwitchClicked"
    android:textOff="Off"
    android:textOn="On" />
```

```
// Called when the switch is clicked
public void onSwitchClicked(View view) {
    // Is the switch on?
    boolean on = ((Switch) view).isChecked();
    if (on) {
        // On
    } else {
        // Off
    }
}
```

KOMPONENTEN - CheckBox

```
<CheckBox
    android:id="@+id/checkbox_1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="CheckBox_1" />
```

```
//CheckBox
CheckBox checkbox = (CheckBox)
findViewById(R.id.checkbox_1);
boolean checked = checkbox.isChecked();
if (checked) {
    //do something
}
```

KOMPONENTEN – CheckBox – onClick

```
<CheckBox
    android:id="@+id/checkbox_1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onCheckboxClicked"
    android:text="CheckBox_1" />

<CheckBox
    android:id="@+id/checkbox_2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onCheckboxClicked"
    android:text="CheckBox_2" />
```

```
public void onCheckboxClicked(View view) {
    // Has the checkbox that was clicked been checked?
    boolean checked = ((CheckBox) view).isChecked();

    // Retrieve which checkbox was clicked
    switch (view.getId()) {
        case R.id.checkbox_1:
            if (checked) {
                // do something!!
            } else {
                // do something!!
            }
            break;
        case R.id.checkbox_2:
            if (checked) {
                // do something!!
            } else {
                // do something!!
            }
            break;
    }
}
```

KOMPONENTEN – RadioGroup – RadioButton

```
<RadioGroup
    android:id="@+id/radio_group"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <RadioButton
        android:id="@+id/RadioButton_1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="RadioButton_1" />

    <RadioButton
        android:id="@+id/RadioButton_2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="RadioButton_2" />
</RadioGroup>
```

```
//RadioButton
RadioGroup radioGroup = (RadioGroup)
findViewById(R.id.radio_group);

int id = radioGroup.getCheckedRadioButtonId();

if (id == -1) {
    //no item selected
} else {
    RadioButton radioButton = (RadioButton)
    findViewById(id);
}
```

KOMPONENTEN – RadioGroup – RadioButton – onClick

```
<RadioGroup
    android:id="@+id/radio_group"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <RadioButton
        android:id="@+id/RadioButton_1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onRadioButtonClicked"
        android:text="RadioButton_1" />

    <RadioButton
        android:id="@+id/RadioButton_2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onRadioButtonClicked"
        android:text="RadioButton_2" />
</RadioGroup>
```

```
public void onRadioButtonClicked(View view) {

    RadioGroup radioGroup = (RadioGroup)
    findViewById(R.id.radio_group);

    int id = radioGroup.getCheckedRadioButtonId();

    switch (id) {
        case R.id.RadioButton_1:
            // do something!!
            break;
        case R.id.RadioButton_2:
            // do something!!
            break;
    }
}
```

KOMPONENTEN – Spinner

```
<Spinner
    android:id="@+id/spinner"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:entries="@array/spinner_values" />
```

```
//Spinner
Spinner spinner = (Spinner)
findViewById(R.id.spinner);
String string =
String.valueOf(spinner.getSelectedItem());
```

KOMPONENTEN – ImageView

```
<ImageView
    android:layout_width="200dp"
    android:layout_height="100dp"
    android:contentDescription="ImageView"
    android:src="@drawable/android" />
```

```
ImageView photo = (ImageView)
findViewById(R.id.ImageView);
int image = R.drawable.android;
String description = "This is the logo";
photo.setImageResource(image);
photo.setContentDescription(description);
```

KOMPONENTEN – ImageButton

```
<ImageButton
    android:id="@+id/ImageButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/android_ico" />
```

```
// Called when the button is clicked
public void onButtonClicked(View view) {
    //Do something in response to button click
}
```

KOMPONENTEN – ListView

```
<ListView
    android:id="@+id/listView"
    android:layout_weight="1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentTop="true" />
```

```
ListView listView = (ListView) findViewById(R.id.listView);
ArrayList<E> entries = databaseHelper.getAllEntries();

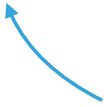
if (!entries.isEmpty()) {
    ArrayAdapter<E> listAdapter = new ArrayAdapter<E>(this, android.R.layout.simple_list_item_1, entries);
    listView.setAdapter(listAdapter);
} else {
}
```

KOMPONENTEN – ProgressBar

```
<ProgressBar
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="match_parent"
    android:layout_height="40dp"
    android:id="@+id/progressBar"
    android:layout_gravity="center_horizontal" />
```

```
ProgressBar progressBar = (ProgressBar) findViewById(R.id.progressBar);
progressBar.setProgress(50);
```

Wert 0-100



KOMPONENTEN – Toast

```
//Toast
String toastText = "Hello, I'm a Toast!";
int duration = Toast.LENGTH_SHORT;
Toast toast = Toast.makeText(this, toastText, duration);
toast.show();
```

KOMPONENTEN – ScrollView

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="io.vsia.layoutdemo.ScrollView">

    .....
    .....
    .....

</ScrollView>
```



MODUL 335 - 04 MOBILE APPLIKATION

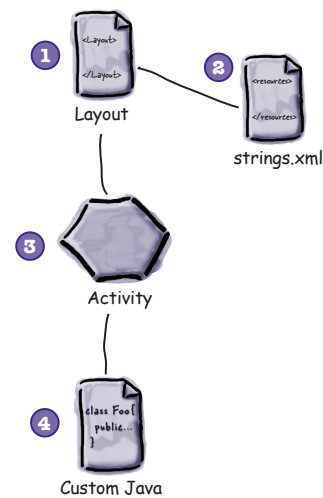
EINGABE UND AUSGABE

BIER RATGEBER



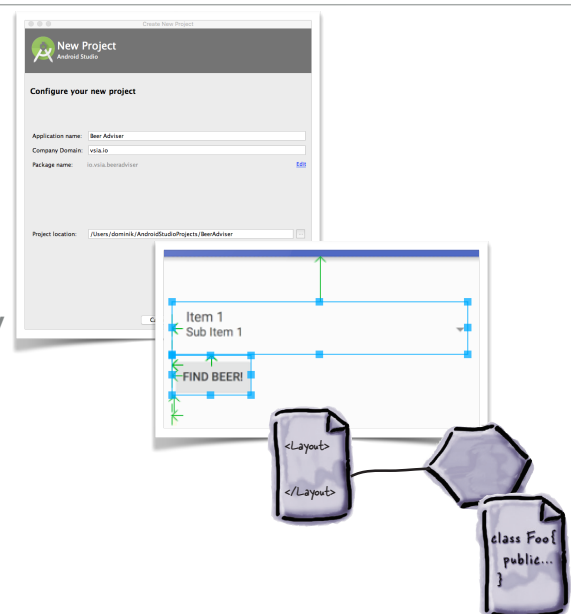
AUS WAS BESTEHT DIE APP

1. Das Layout definiert alle GUI Komponenten
 - ▶ Drop-Down/Button/Textfeld
2. Hier werden alle Strings für die App definiert
3. Die Activity definiert die Interaktion
4. Hier wird die BusinessLogik implementiert



SCHRITT FÜR SCHRITT

1. Erstelle ein neues Projekt
2. Aktualisiere das Layout
3. Verbinde das Layout mit der Activity
4. Implementiere die BusinessLogik



2. AKTUALISIERE DAS LAYOUT

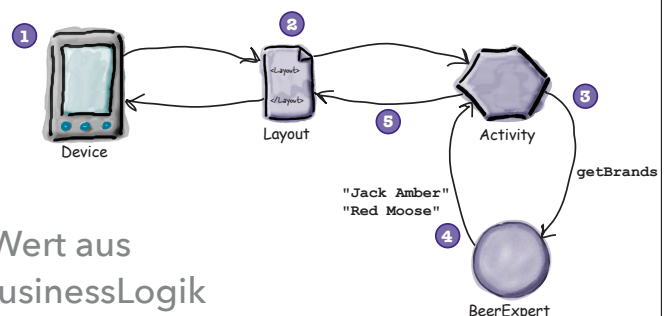
```
<Spinner
    android:id="@+id/color"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="false"
    android:layout_marginTop="37dp" />
```

```
<Button
    android:id="@+id/find_beer"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignLeft="@id/color"
    android:layout_below="@id/color"
    android:text="Button" />
```

```
<TextView
    android:id="@+id/brands"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@id/find_beer"
    android:layout_below="@id/find_beer"
    android:layout_marginTop="18dp"
    android:text="TextView" />
```

3. VERBINDE DAS LAYOUT MIT DER ACTIVITY

1. Der User wählt eine Kategorie aus
2. Das Layout definiert die Methode bei Klick auf Button
3. Die Methode in der Activity liest den Wert aus dem DropDown und gibt ihn an die BusinessLogik
4. Die BusinessLogik gibt das Resultat aufgrund des übergebenen Parameter zurück
5. Die Aktivität aktualisiert das Layout



2. AKTUALISIERE DAS LAYOUT - STRING WERTE

Zugriff mit: @array/beer_colors

```
<resources>
  <string name="app_name">Beer Adviser</string>
  <string name="find_beer">Finde ein Bier!</string>
  <string name="brands"></string>
  <string-array name="beer_colors">
    <item>hell</item>
    <item>amber</item>
    <item>dunkel</item>
  </string-array>
</resources>
```

Zugriff mit: @string/brands

STRINGS AUSLAGERN & DROPDOWN WERTE EINFÜGEN

```
<Spinner
  android:id="@+id/color"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_alignParentTop="true"
  android:layout_centerHorizontal="false"
  android:layout_marginTop="37dp"
  android:entries="@array/beer_colors" />
```

```
<Button
  android:id="@+id/find_beer"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_alignParentStart="true"
  android:layout_alignLeft="@id/color"
  android:layout_below="@id/color"
  android:text="@string/find_beer" />
```

```
<TextView
  android:id="@+id/brands"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_alignLeft="@id/find_beer"
  android:layout_below="@id/find_beer"
  android:layout_marginTop="18dp"
  android:text="@string/brands" />
```

WAS SOLL DER BUTTON MACHEN? - VERSION 1 OHNE BUSINESS LOGIK

```
<Button
    android:id="@+id/find_beer"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignLeft="@id/color"
    android:layout_below="@id/color"
    android:text="@string/find_beer"
    android:onClick="onClickFindBeer" />
```

```
public void onClickFindBeer(View view) {

    //Get a reference to the TextView
    TextView brands = (TextView) findViewById(R.id.brands);

    //Get a reference to the Spinner
    Spinner color = (Spinner) findViewById(R.id.color);

    //Get the selected item in the Spinner
    String beerType = String.valueOf(color.getSelectedItem());

    //Display the selected item
    brands.setText(beerType);

}
```

4. BUSINESSLOGIK

```
private BeerExpert expert = new BeerExpert();

//Call when the button gets clicked
public void onClickFindBeer(View view) {

    //Get a reference to the TextView
    TextView brands = (TextView) findViewById(R.id.brands);

    //Get a reference to the Spinner
    Spinner color = (Spinner) findViewById(R.id.color);

    //Get the selected item in the Spinner
    String beerType = String.valueOf(color.getSelectedItem());

    //Get recommendations from the BeerExpert class
    List<String> brandsList = expert.getBrands(beerType);
    StringBuilder brandsFormatted = new StringBuilder();
    for (String brand : brandsList) {
        brandsFormatted.append(brand).append('\n');
    }

    //Display the beers
    brands.setText(brandsFormatted);

}
```

```
public class BeerExpert {

    List<String> getBrands(String color) {
        List<String> brands = new ArrayList<String>();

        switch (color) {
            case "hell":
                brands.add("Burgdorfer Hell");
                brands.add("Feldschlösschen Original");
                break;
            case "amber":
                brands.add("Bärner Amber Bier");
                brands.add("Feldschlösschen Amber");
                break;
            case "dunkel":
                brands.add("Burgdorfer Aemme");
                brands.add("Feldschlösschen Dunkle Perle");
                break;
            default:
                brands.add("kein Bier gefunden");
        }

        return brands;
    }

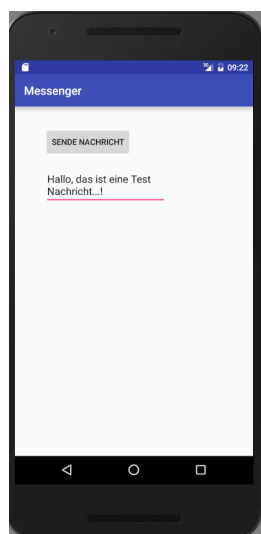
}
```



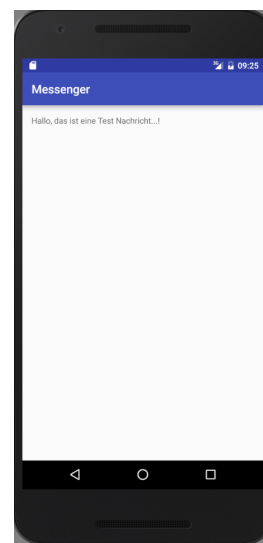
MODUL 335 - 05 MOBILE APPLIKATION

MEHRERE ACTIVITIES

MESSENGER



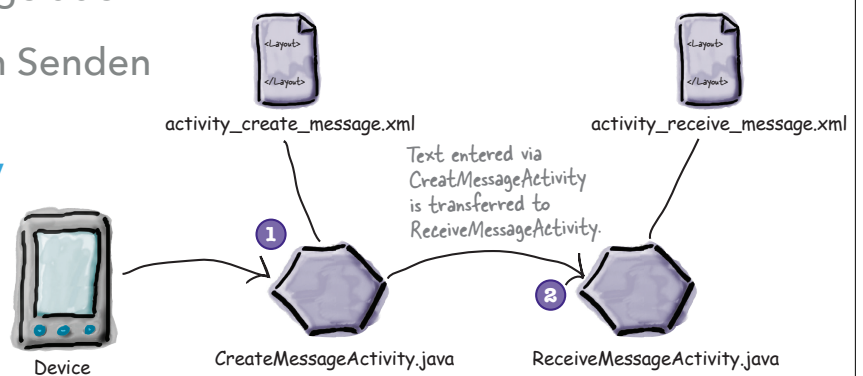
CreateMessageActivity



ReceiveMessageActivity

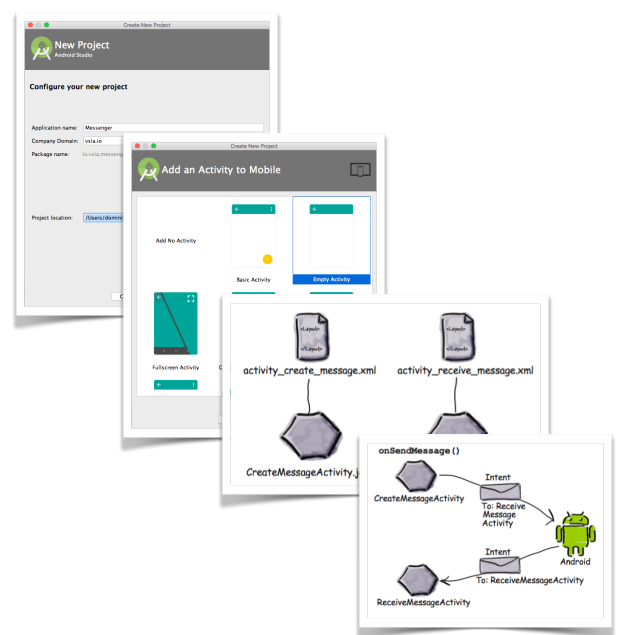
AUS WAS BESTEHT DIE APP

1. Beim Start der Applikation wird **CreateMessageActivity** geladen
2. Sobald der User auf den Senden Button klickt, wird die **ReceiveMessageActivity** geladen



SCHRITT FÜR SCHRITT

1. Erstelle ein neues Projekt mit einer Activity und Layout
2. Füge eine zweite Activity und Layout hinzu
3. Lasse die erste Activity die zweite aufrufen
4. Übertrage Daten von der ersten zur zweiten Activity



1.1 NEUES PROJEKT ERSTELLEN

- ▶ Project
 - ▶ Application name: Messenger
 - ▶ Company Domain: vsia.io
- ▶ Target / SDK
 - ▶ Phone and Tablet: API 19 Android 4.4 (KitKat)
- ▶ Activity
 - ▶ Template: Empty Activity
 - ▶ Activity Name: CreateMessageActivity
 - ▶ Layout Name: activity_create_message

1.2 AKTUALISIERE DAS LAYOUT

```
<Button
    android:id="@+id/send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginLeft="38dp"
    android:layout_marginTop="20dp"
    android:onClick="onSendMessage"
    android:text="@string/send" />
```

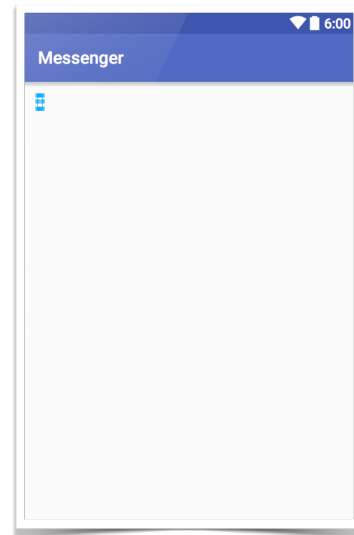
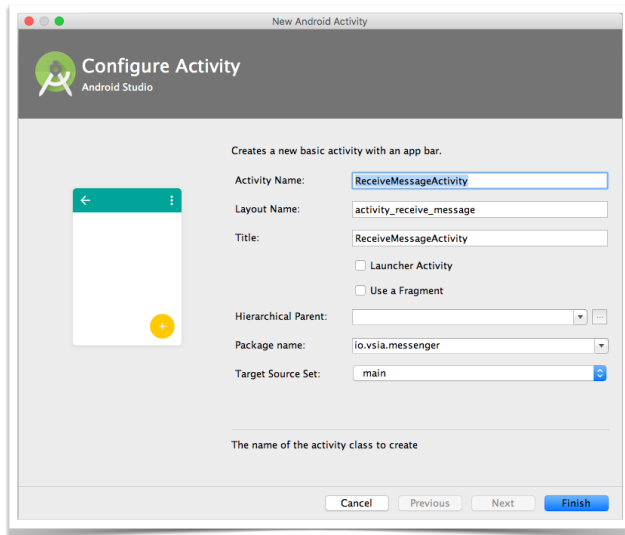
```
<EditText
    android:id="@+id/message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/send"
    android:layout_below="@+id/send"
    android:layout_marginTop="18dp"
    android:ems="10" />
```

1.3 AKTUALISIERE DIE STRINGS

- ▶ Auf dem Button soll „Sende Nachricht“ stehen

1.4 ERSTELLE IN DER ACTIVITY EINE LEERE METHODE

2.0 ERSTELLE EINE ZWEITE ACTIVITY

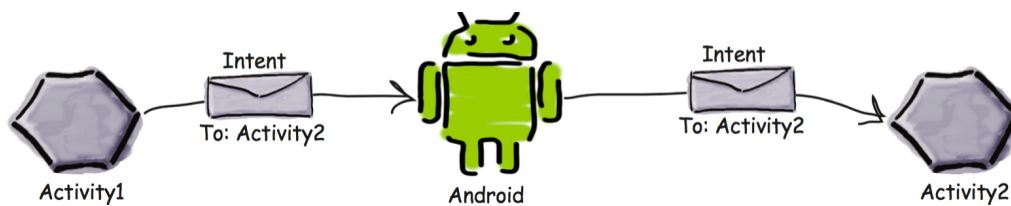


DAS ANDROID MANIFEST



3 LASSE DIE ERSTE ACTIVITY DIE ZWEITE AUFRUFEN

- ▶ Activities können lediglich mittels sogenannten Intent's andere Activities aufrufen:
 - ▶ Intent erstellen: `Intent intent = new Intent(this, Activity2.class);`
 - ▶ Intent absenden: `startActivity(intent);`



3 LÖSUNG

```
//Call onSendMessage() when the button is clicked  
public void onSendMessage(View view) {  
  
    Intent intent = new Intent(this, ReceiveMessageActivity.class);  
    startActivity(intent);  
}
```

4 DATEN VON DER ERSTEN ZUR ZWEITEN ACTIVITY ÜBERTRAGEN

- ▶ Einem Intent können Daten (primitive, String, etc) mitgegeben werden.
 - ▶ Daten mitsenden: `intent.putExtra("message", value);`
 - ▶ Daten auslesen: `intent.getStringExtra("message");`
`intent.getIntExtra("message");`

4 DATEN AN DIE ZWEITE ACTIVITY ÜBERTRAGEN (CreateMessageActivity)

```
// onSendMessage() wird ausgeführt sobald der Senden Button geklickt wird  
public void onSendMessage(View view) {  
  
    // EditText Komponente holen  
  
    // Text aus der EditText Komponente auslesen  
  
    // neuen Intent erstellen  
  
    // Text (aus der EditText-Komponente) dem Intent hinzufügen  
  
    // Intent absenden  
  
}
```

4 DATEN AN DIE ZWEITE ACTIVITY ÜBERTRAGEN (CreateMessageActivity) – LÖSUNG

```
// onSendMessage() wird ausgeführt sobald der Senden Button geklickt wird
public void onSendMessage(View view) {

    // EditText Komponente holen
    EditText messageView = (EditText) findViewById(R.id.message);

    // Text aus der EditText Komponente auslesen
    String messageText = messageView.getText().toString();

    // neuen Intent erstellen
    Intent intent = new Intent(this, ReceiveMessageActivity.class);

    // Text (aus der EditText-Komponente) dem Intent hinzufügen
    intent.putExtra("message", messageText);

    // Intent absenden
    startActivity(intent);

}
```

4 DATEN AUS DEM INTENT AUSLESEN (ReceiveMessageActivity)

```
// wird ausgeführt sobald die Activity erstellt wird
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_receive_message);

    // Intent holen

    // Text aus Intent Komponente auslesen

    // TextView Komponente holen

    // Text der TextView Komponente setzen

}
```

4 DATEN AUS DEM INTENT AUSLESEN (ReceiveMessageActivity) - LÖSUNG

```
// wird ausgeführt sobald die Activity erstellt wird
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_receive_message);

    // Intent holen
    Intent intent = getIntent();

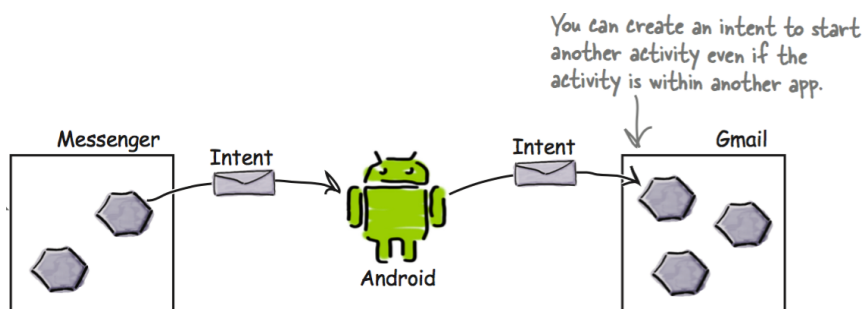
    // Text aus Intent Komponente auslesen
    String messageText = intent.getStringExtra("message");

    // TextView Komponente holen
    TextView messageView = (TextView) findViewById(R.id.message);

    // Text der TextView Komponente setzen
    messageView.setText(messageText);
}
```

DATEN AN ANDERE APPLIKATIONEN SENDEN

- ▶ Intents können auch verwendet werden um Activities anderer Applikationen zu starten (inkl. Datenübertragung)
- ▶ Der Mechanismus ist dabei, weitestgehend, derselbe



DATEN AN ANDERE APPLIKATIONEN SENDEN - IMPLICIT INTENT

- ▶ Da wir nicht wissen welche Applikationen auf den Geräten installiert sind, definieren wir lediglich welche Aktion wir auslösen wollen und überlassen es dem User mit welcher Applikation er diese Aktion durchführen will.
- ▶ z.B. `Intent.ACTION_DIAL`, `Intent.ACTION_WEB_SEARCH`, `Intent.ACTION_SEND`
- ▶ Intent erstellen: `Intent intent = new Intent(Intent.ACTION_SEND);`
- ▶ Daten mitsenden: `intent.setType("text/plain");`
`intent.putExtra(Intent.EXTRA_TEXT, messageText);`

WOHER WEISS ANDROID WELCHE APP FÜR WELCHE ACTION GEEIGNET IST?

AndroidManifest.xml

```
<activity android:name="ShareActivity">
  <intent-filter>
    <action android:name="android.intent.action.SEND"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="text/plain"/>
    <data android:mimeType="image/*"/>
  </intent-filter>
</activity>
```

definiert, dass diese Activity
mit ACTION_SEND umgehen
kann

Definiert die zulässigen MIME-Typen

Wird benötigt um überhaupt
implicit-Intents zu empfangen

DATEN AN DIE ZWEITE ACTIVITY ÜBERTRAGEN (CreateMessageActivity)

```
// onSendMessage() wird ausgeführt sobald der Senden Button geklickt wird
public void onSendMessage(View view) {

    // EditText Komponente holen
    EditText messageView = (EditText) findViewById(R.id.message);

    // Text aus der EditText Komponente auslesen
    String messageText = messageView.getText().toString();

    // neuen Intent erstellen

    // Datentyp für den Intent definieren

    // Text (aus der EditText-Komponente) dem Intent hinzufügen

    // Intent absenden
    startActivity(intent);
}
```

DATEN AN DIE ZWEITE ACTIVITY ÜBERTRAGEN (CreateMessageActivity) - LÖSUNG

```
// onSendMessage() wird ausgeführt sobald der Senden Button geklickt wird
public void onSendMessage(View view) {

    // EditText Komponente holen
    EditText messageView = (EditText) findViewById(R.id.message);

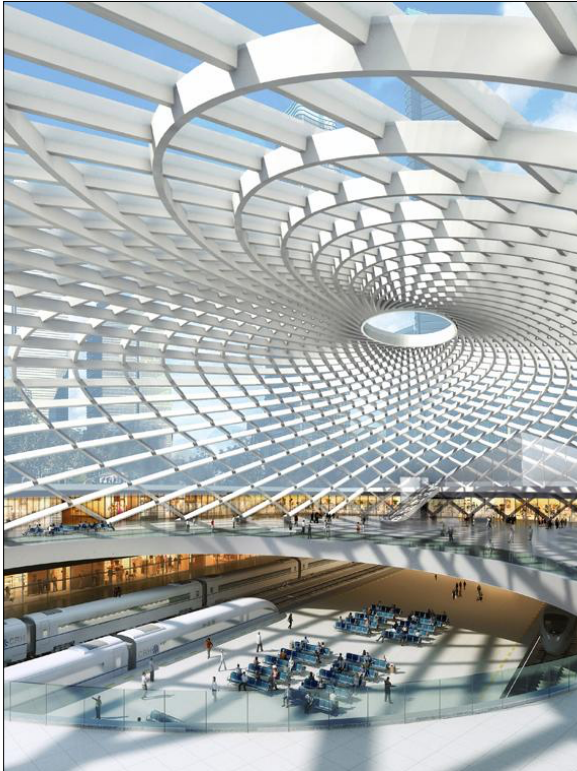
    // Text aus der EditText Komponente auslesen
    String messageText = messageView.getText().toString();

    // neuen Intent erstellen
    Intent intent = new Intent(Intent.ACTION_SEND);

    // Datentyp für den Intent definieren
    intent.setType("text/plain");

    // Text (aus der EditText-Komponente) dem Intent hinzufügen
    intent.putExtra(Intent.EXTRA_TEXT, messageText);

    // Intent absenden
    startActivity(intent);
}
```

MODUL 335 - 06

MOBILE APPLIKATION

DATABASE

SQLite

- ▶ Leichtgewichtig
 - ▶ Die meisten DBS brauchen einen eigenen Datenbank Server, bei SQLite ist dies nicht der Fall. SQLite ist „nur“ ein File. Wenn die Datenbank nicht benötigt wird, braucht sie auch keine Prozessorleistung.
- ▶ Optimierte für Einzelbenutzer
 - ▶ Die Applikation ist das einzige was auf die DB zugreift, wir müssen uns also nicht um Dinge wie User Namen und Passwörter kümmern.
- ▶ Stabil und Schnell
 - ▶ SQLite Datenbanken sind sehr stabil und schnell. Sie können mit Transaktionen umgehen und dementsprechend Rollbacks durchführen.

SQLite

- ▶ Android erstellt automatisch einen Ordner für die Datenbank(en).
- ▶ /data/data/io.vsia.xyz/databases
- ▶ Eine Applikation kann mehrere Datenbanken speichern/benutzen.
- ▶ Eine Datenbank besteht aus zwei Dateien:
 - ▶ database file ⇨ Hier werden die Daten gespeichert.
 - ▶ journal file ⇨ Alle Änderungen werden hier gespeichert.

SQLite

Android stellt drei Klassen zur Verwaltung der Datenbanken zur Verfügung:

1. SQLiteDatabase

- ▶ Ermöglicht Zugriff auf die DB (SQLConnection in JDBC)

2. SQLiteOpenHelper

- ▶ Dient zur Erstellung und Verwaltung der DB

3. Cursor

- ▶ Ermöglicht das Schreiben und Lesen auf der DB (ResultSet in JDBC)

SQLite Helper

- ▶ SQLiteOpenHelper Klasse wird benötigt um die Datenbank zu erstellen und zu verwalten.
- ▶ Erstellen der Datenbank
 - ▶ Helper stellt sicher, dass bei der ersten Installation die Datenbank und deren Tables erstellt werden.
- ▶ Wartung der Datenbank
 - ▶ Sollte die Struktur der Datenbank ändern, wird mittels Helper die DB aktualisiert.
- ▶ Auf Datenbank zugreifen
 - ▶ CRUD Operationen $\Rightarrow$ (DA)O/R Mapper

SQLite Datentypen

- ▶ INTEGER integer type
 - ▶ TEXT character type
 - ▶ REAL floating-point number
 - ▶ NUMERIC Booleans, dates, and date-times
 - ▶ BLOB Binary Large Object
-
- ▶ Die Grösse muss nicht definiert werden.

SQLite Helper

Name der DB

DB Version

Wird beim Erstellen
der Datenbank aufgerufenWird ausgeführt sobald
die Versionsnummer höher*
ist als die der Datenbank*Version kleiner
⇒ onDowngrade()

```
public class BMIDatabaseHelper extends SQLiteOpenHelper {

    private static final String DB_NAME = "bmi";
    private static final int DB_VERSION = 1;

    BMIDatabaseHelper(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }

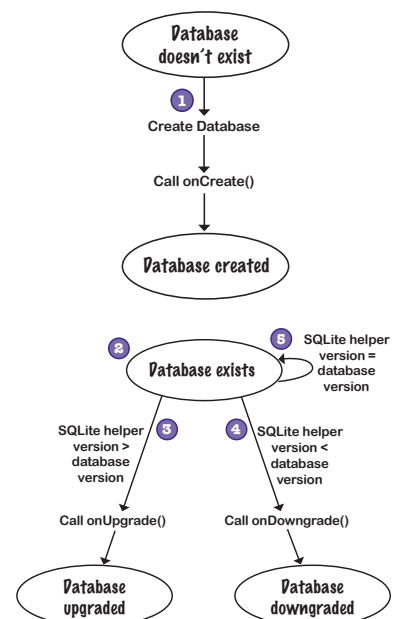
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE BMI_ENTRY ("
            + "id INTEGER PRIMARY KEY AUTOINCREMENT, "
            + "NAME TEXT, "
            + "WEIGHT REAL, "
            + "HEIGHT REAL, "
            + "BMI REAL);");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }
}
```

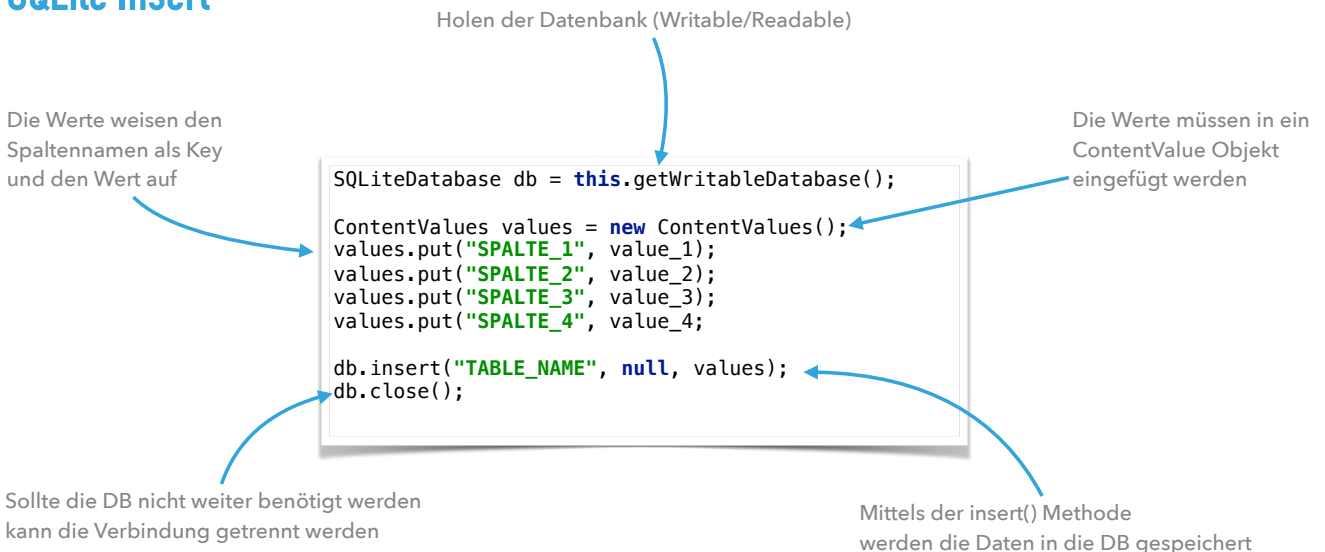
für erweiterte
Funktionen

SQLite Versionierung

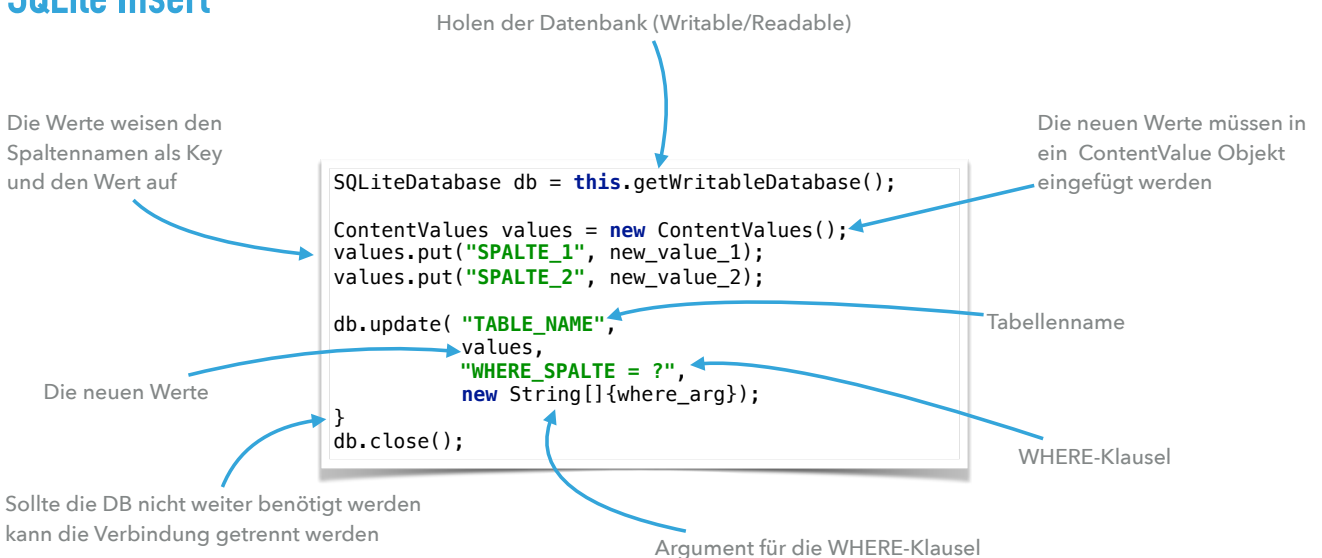
1. Existiert die DB nicht, wird sie mittels onCreate() erstellt.
2. Existiert die DB bereits, wird die Version überprüft.
3. Sollte die Version des Helper höher sein als die der DB, wird on Upgrade ausgeführt.
4. Sollte die Version des Helper kleiner sein als die der DB, wird onDowngrade ausgeführt.
5. Ist die Version des Helpers und der DB identisch wird keine Methode ausgeführt



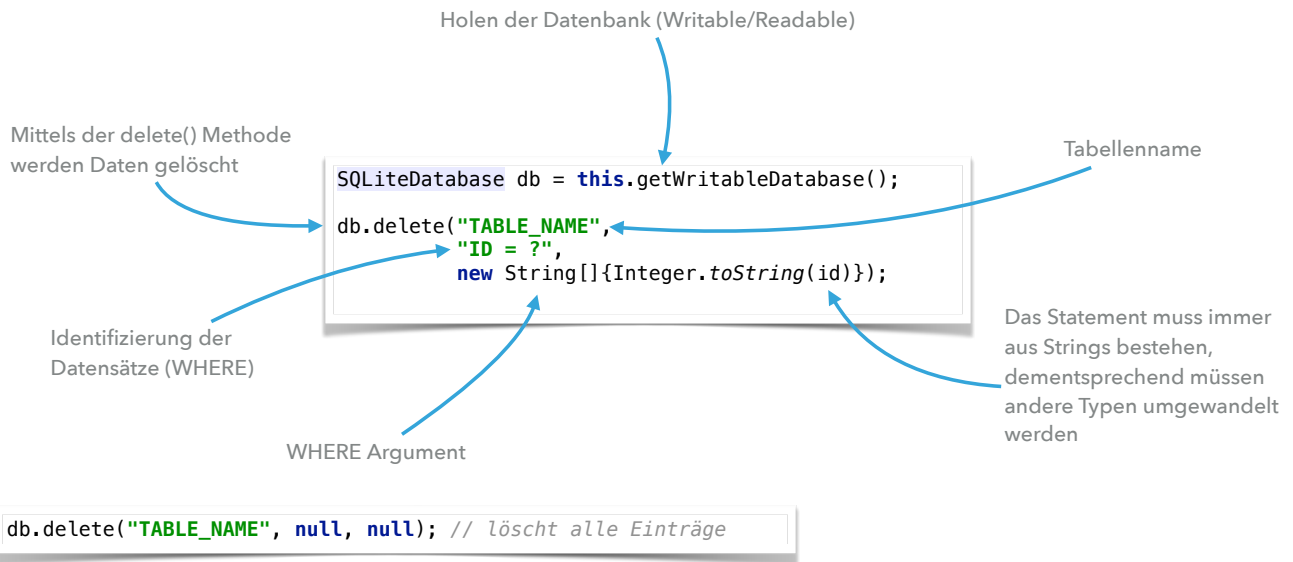
SQLite Insert



SQLite Insert



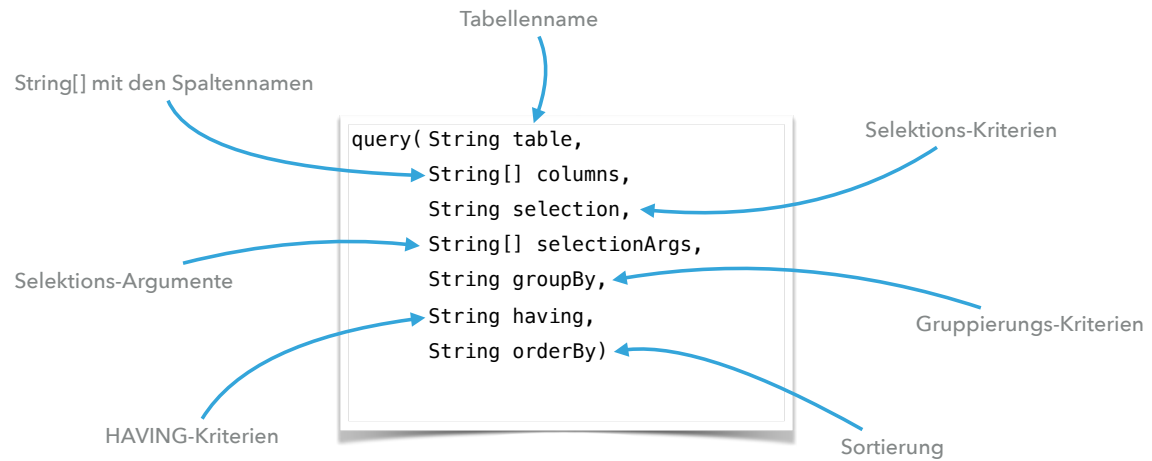
SQLite Delete



SQLite Select

- ▶ Mittels `query()` Methoden kann auf Daten zugegriffen werden.
- ▶ `query()` geben das Resultat als `Cursor` zurück
- ▶ Dabei gibt es zwei grundsätzliche Methoden:
 - ▶ `query()`
 - ▶ `rawQuery()`

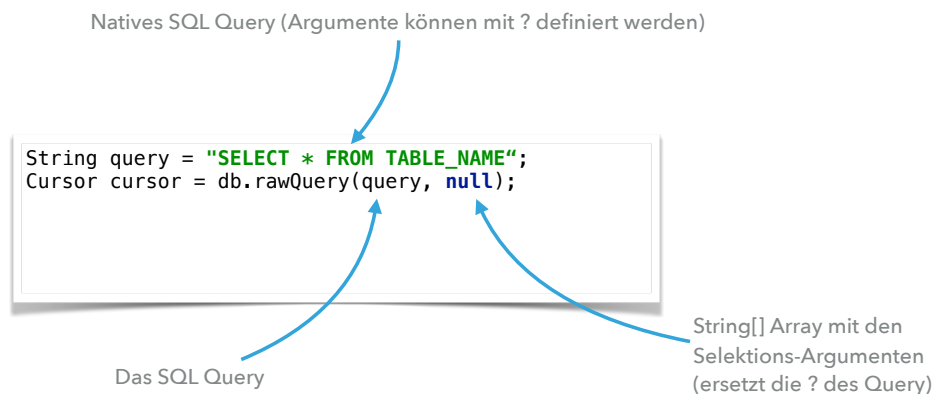
SQLite Select db.query()



Alle Parameter müssen angegeben werden!

Wird ein Parameter nicht verwendet, muss ein null-Value mitgegeben werden.

SQLite Select db.rawQuery()



SQLite Select db.query() – db.rawQuery() – BMI Beispiel

Variante 1: db.query()

```
SQLiteDatabase db = this.getReadableDatabase();  
Cursor cursor = db.query("BMI_ENTRY", null, null, null, null, null, null);
```

Variante 2: db.rawQuery()

```
SQLiteDatabase db = this.getReadableDatabase();  
String query = "SELECT * FROM BMI_ENTRY";  
Cursor cursor = db.rawQuery(query, null);
```

SQLite Cursor auswerten O/R – BMI Beispiel

```
if (cursor.moveToFirst()) {  
    do {  
        BMI bmi = new BMI();  
        bmi.name = cursor.getString(cursor.getColumnIndex("NAME"));  
        bmi.weight = cursor.getDouble(cursor.getColumnIndex("WEIGHT"));  
        bmi.height = cursor.getDouble(cursor.getColumnIndex("HEIGHT"));  
        bmi.bmi = cursor.getDouble(cursor.getColumnIndex("BMI"));  
        bmiEntries.add(bmi);  
    } while (cursor.moveToNext());  
}
```


SQLite Cursor auswerten O/R – BMI Beispiel

```
if (cursor.moveToFirst()) {  
    do {  
        BMI bmi = new BMI();  
        bmi.name = cursor.getString(cursor.getColumnIndex("NAME"));  
        bmi.weight = cursor.getDouble(cursor.getColumnIndex("WEIGHT"));  
        bmi.height = cursor.getDouble(cursor.getColumnIndex("HEIGHT"));  
        bmi.bmi = cursor.getDouble(cursor.getColumnIndex("BMI"));  
        bmiEntries.add(bmi);  
    } while (cursor.moveToNext());  
}
```

KOMPONENTEN – ListView

```
<ListView  
    android:id="@+id/listView"  
    android:layout_weight="1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignParentStart="true"  
    android:layout_alignParentTop="true" />
```

- ListView ruft die toString() Methode der Elemente auf, um die Daten anzuzeigen.

```
ListView listView = (ListView) findViewById(R.id.listView);  
ArrayList<E> entries = databaseHelper.getAllEntries();  
  
if (!entries.isEmpty()) {  
    ArrayAdapter<E> listAdapter = new ArrayAdapter<E>(this, android.R.layout.simple_list_item_1, entries);  
    listView.setAdapter(listAdapter);  
} else {  
}
```