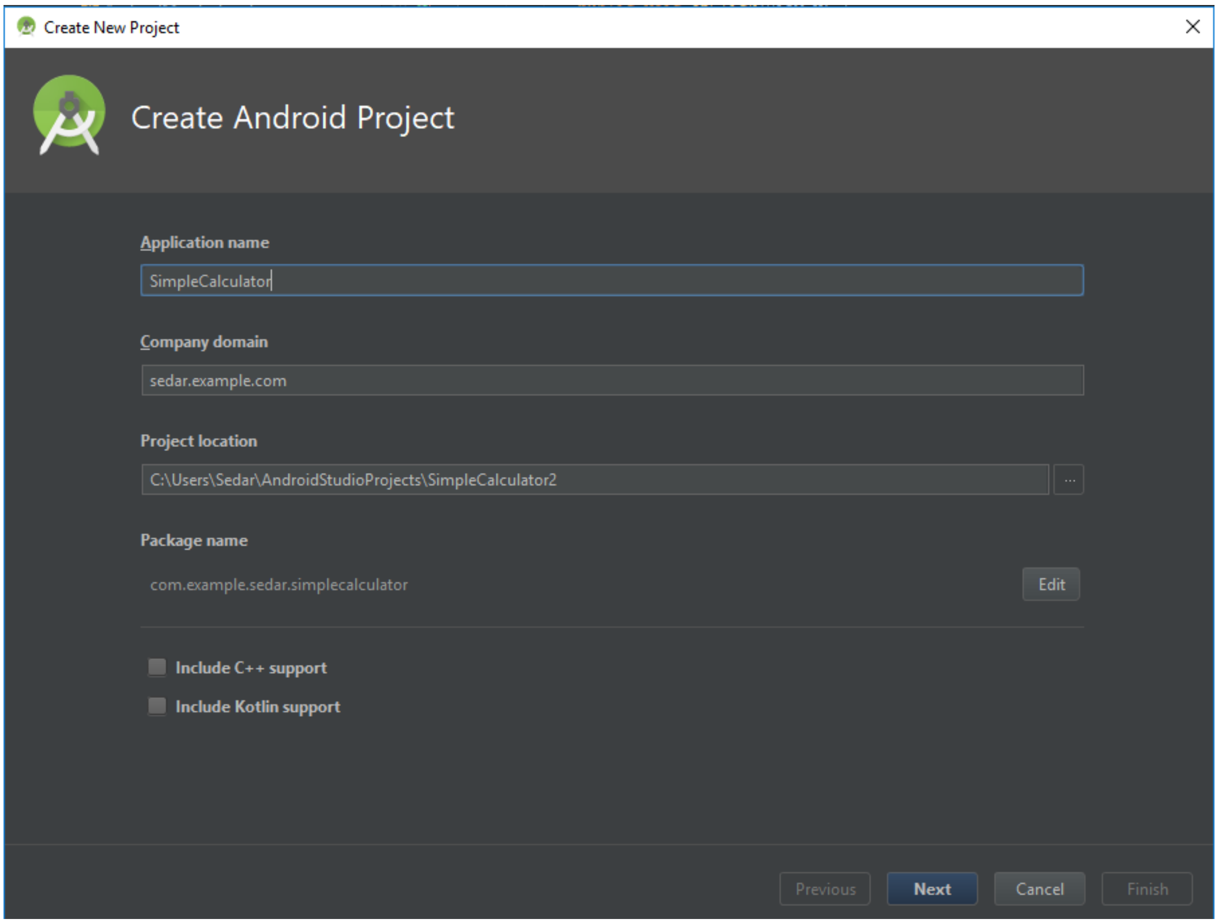
Tutorial

Table of Contents

1 Creating a Project

Create a new Project with an empty Activity



Create New Project

Create Android Project

Application name: SimpleCalculator

Company domain: sedar.example.com

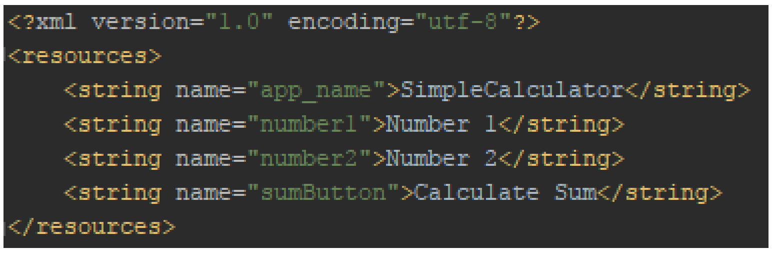
Project location: C:\Users\Sedar\AndroidStudioProjects\SimpleCalculator2

Package name: com.example.sedar.simplecalculator

2 Resources

2.1 Values

First, we will fill the strings.xml file in the values directory

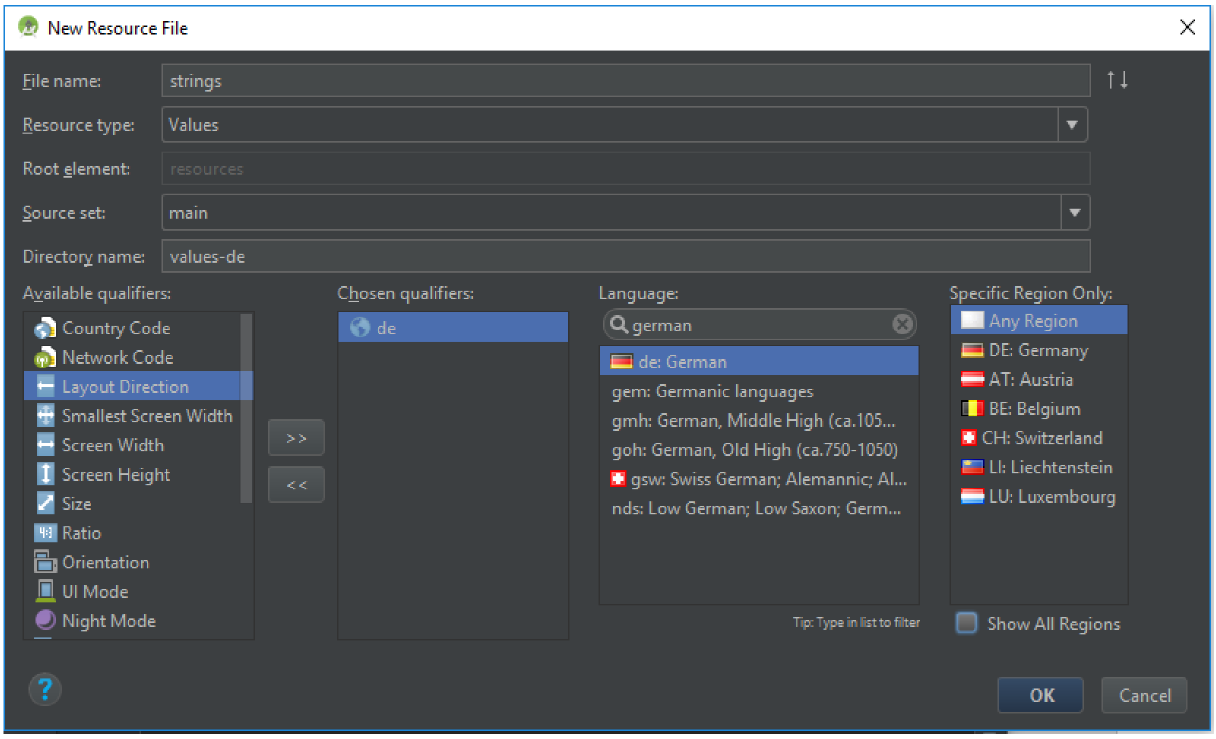


```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">SimpleCalculator</string>
    <string name="number1">Number 1</string>
    <string name="number2">Number 2</string>
    <string name="sumButton">Calculate Sum</string>
</resources>
```

These Strings can be referenced from Layout Files by identifying them by their name.

At the second step we will create a string.xml with a language qualifier.

Create a new **Android Resource File** in your res Directory



New Resource File

File name: strings

Resource type: Values

Root element: resources

Source set: main

Directory name: values-de

Available qualifiers: Country Code, Network Code, Layout Direction, Smallest Screen Width, Screen Width, Screen Height, Size, Ratio, Orientation, UI Mode, Night Mode

Chosen qualifiers: de

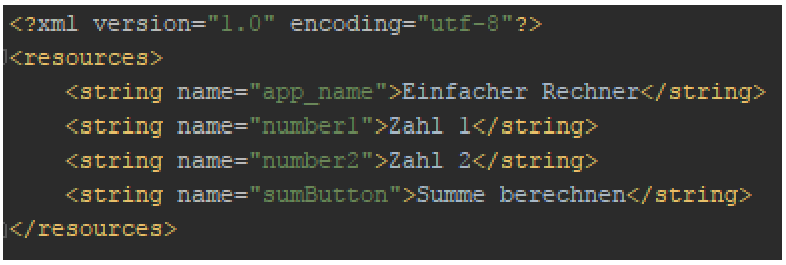
Language: german

de: German
gem: Germanic languages
gmh: German, Middle High (ca.105...
goh: German, Old High (ca.750-1050)
gsw: Swiss German; Alemannic; Al...
nds: Low German; Low Saxon; Germ...

Specific Region Only:

Any Region
DE: Germany
AT: Austria
BE: Belgium
CH: Switzerland
LI: Liechtenstein
LU: Luxembourg

We will copy the values of **values\strings.xml** into **de\strings.xml** and translate the values into German but keep the names unchanged.

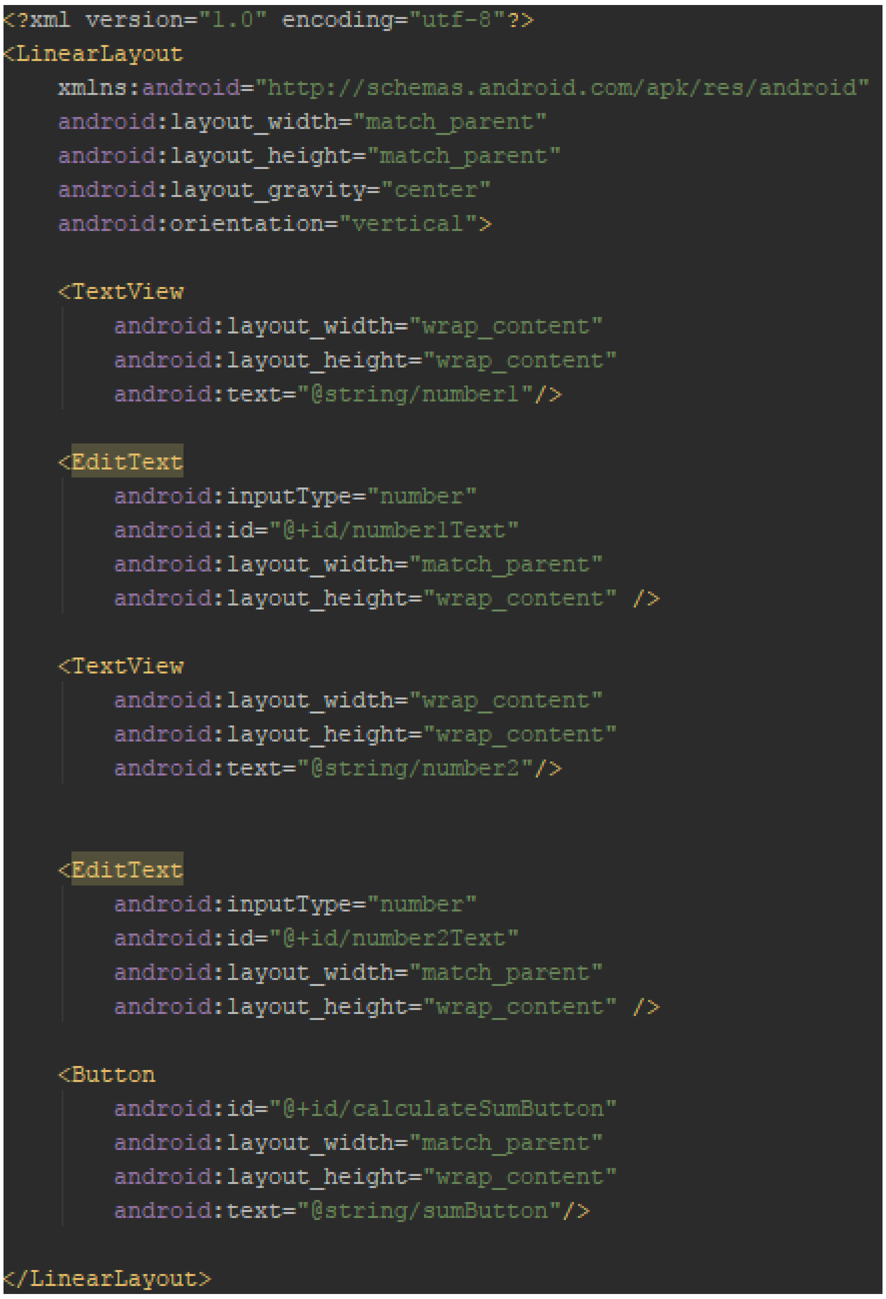


```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Einfacher Rechner</string>
    <string name="number1">Zahl 1</string>
    <string name="number2">Zahl 2</string>
    <string name="sumButton">Summe berechnen</string>
</resources>
```

The Android can load, based on qualifies and system properties the corresponding files. In this example the **de/strings.xml** would be loaded instead of the **values/strings.xml** if the system language is German, and the sources which references the string names would get the translated Strings.

2.2 Layout

After creating the **de/strings.xml** we will define our Layout. We will us **activity_main.xml**, which should be an open tab already.



```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/number1"/>

    <EditText
        android:inputType="number"
        android:id="@+id/number1Text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/number2"/>


    <EditText
        android:inputType="number"
        android:id="@+id/number2Text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/calculateSumButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/sumButton"/>

</LinearLayout>
```

Linear Layout: A Linear Layout is a layout which lists its components either horizontally or vertically.

TextView: A UI Element which shows the linked/written text value

EditText: A UI Element which can be edited by the user and receive the users input.

Button: A UI Element which shows a button with the inked/written text value

layout_width / layout_height: They determine the width and height of the of the UI Element.

match_parent: The UI Element takes on the same dimensions as its parent Element.

wrap_content: The UI Element takes up as much space as the content demands

android:text: The text can either be hardcoded or referenced from a strings.xml via the "@string/[string-name]" syntax.

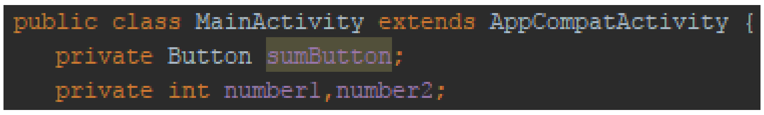
android:id: the unique ID which identifies the object in whole resource collection. It is declared by using "@+id/[unique name]"

android:inputType: The type of input which is accepted.

3 Source Code

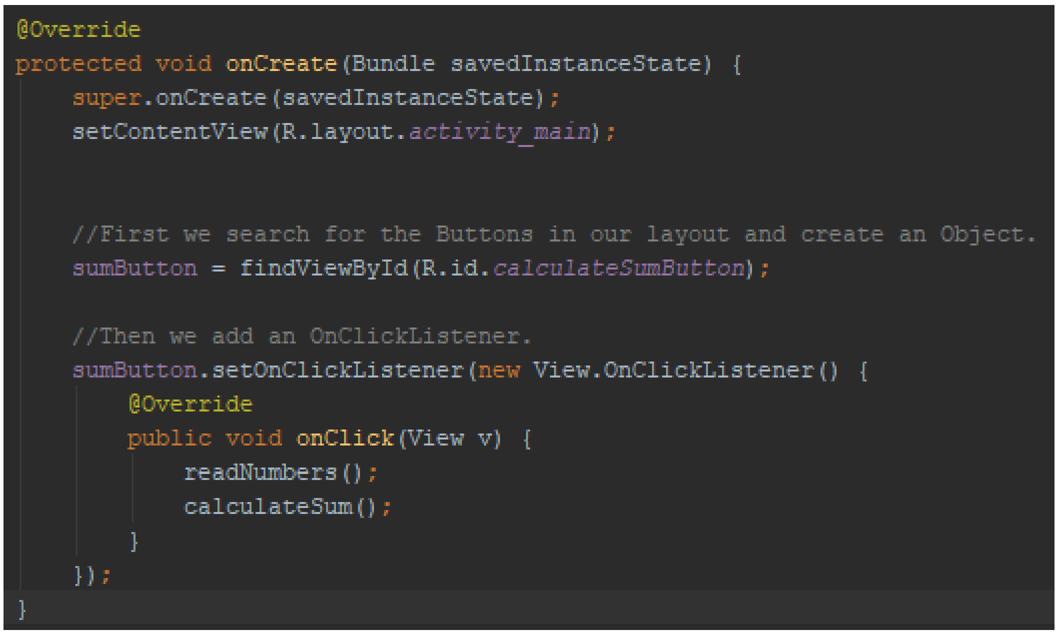
After configuring the resources, we will write our Activity.
Open the MainActivity.java file.

We will create sumButton, number1 and number2 variables in our MainActitvity class.
sumButton will be our calculateSumButton, number 1 our first number and number 2 our second number.



```java
public class MainActivity extends AppCompatActivity {
    private Button sumButton;
    private int number1,number2;
```

In our **onCreate** method we will search for our **sumButton** from our Layout and add an **onClickListener**, which calls the void methods **readNumbers** and **calculateSum**.

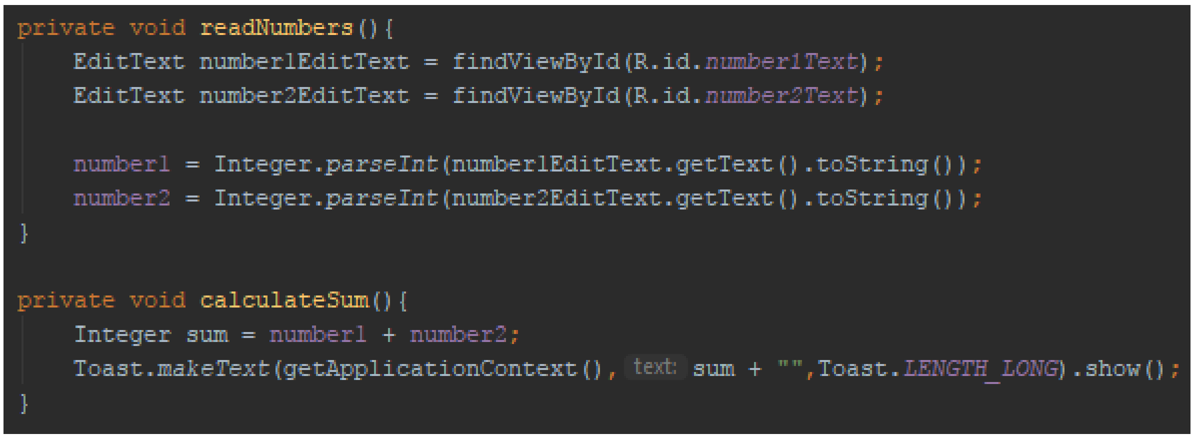


```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);


    //First we search for the Buttons in our layout and create an Object.
    sumButton = findViewById(R.id.calculateSumButton);

    //Then we add an OnClickListener.
    sumButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            readNumbers();
            calculateSum();
        }
    });
}
```

We can search for our **sumButton** by using the method **findViewById(Int ID)**, it searches the View for an element which has the same ID. **R** is the collection of all our Resources and with **.id** we are asking for all saved ids. Our **calculateSumButton** has the ID name **calculateSumButton**. By adding **.calculateSumButton** to our **R.id.** we get the **IntegerValue** of that ID name.

After finding the Button we can add an **OnClickListener** with the **setOnClickListener** method and an Anonymous Inner Class as a parameter. In this Class we call the readNumbers and calculateSum methods.

Now we will write our **readNumbers** and **calculateSum** methods. **readNumbers()** will read the values from the EditText fields and parse them. **calculateSum()** adds those numbers and pushes out the result as a toast.



```java
private void readNumbers(){
    EditText number1EditText = findViewById(R.id.number1Text);
    EditText number2EditText = findViewById(R.id.number2Text);

    number1 = Integer.parseInt(number1EditText.getText().toString());
    number2 = Integer.parseInt(number2EditText.getText().toString());
}

private void calculateSum(){
    Integer sum = number1 + number2;
    Toast.makeText(getApplicationContext(), text: sum + "",Toast.LENGTH_LONG).show();
}
```

In **readNumbers()** we create two EditText Objects for our two EditText elements in our View. Then we read their Strings and parse them.

In **calculateSum()** the result of the addition is written into a Toast, which shows the result to the user

4 Conclusion

The Application is done and works now. You can test it on the Android Studio Emulator or on your Phone. This gives you the first Impression how an Android App is built.

If you are interested you can expand the app by adding subtraction, multiplication and division. Or you could work through the documentation on developer.android.com or other tutorials on the internet to expand your knowledge.