Tutorial

Programmierung einer Android-Applikation

Teil 1

Autor: Oliver Matle
Datum: März 2014, Version 1.0

Inhaltsverzeichnis

Kapitel 1 – Einleitung

In diesem Tutorial wird die Programmierung einer Android-App beschrieben, mit der man persönliche Notizen verwalten kann.

Der erste Teil des Tutorial beschränkt sich auf den Bereich Navigation innerhalb einer App. Die App kann keine Notizen speichern, verarbeiten oder darstellen. Auf die Ausgestaltung der Oberfläche wurde wegen der Übersichtlichkeit weitestgehend verzichtet. In der Fortsetzung des Tutorials werden weitere Teilaspekte erklärt und die Appliaktion wird um fehlende Funktionalitäten erweitert.

Dieses Tutorial ist für Android-Anfänger, jedoch nicht für Java-Anfänger geeignet. Eine detailierter Beschreibung im Umgang mit Eclipse findet hier ebenfalls nicht statt.

Fachliche Beschreibung

Eine Notiz besteht aus einem Thema und einem Notizinhalt, also dem eigentlichen Text. Notizen können neu angelegt, gelöscht oder geändert werden. Eine Liste zeigt alle Notizenthemen an. In einer Detailansicht kann man den Notizeninhalt lesen, ändern oder löschen.

Übersicht aller Activities

Die „Start-Activity“ zeigt nur Buttons an und soll folgende Kommandos anbieten.

- Neue Notiz anlegen
- Liste der Notizen anzeigen

Die Activity „Liste der Notizen“ zeigt alle Notizen als Liste an und soll folgende Kommandos anzeigen.

- Neue Notiz anlegen
- Zurück

Die Activity „Notiz anzeigen“ zeigt den Notiztext an und soll folgende Kommandos anbieten.

- Notiz ändern
- Notiz löschen
- Zurück

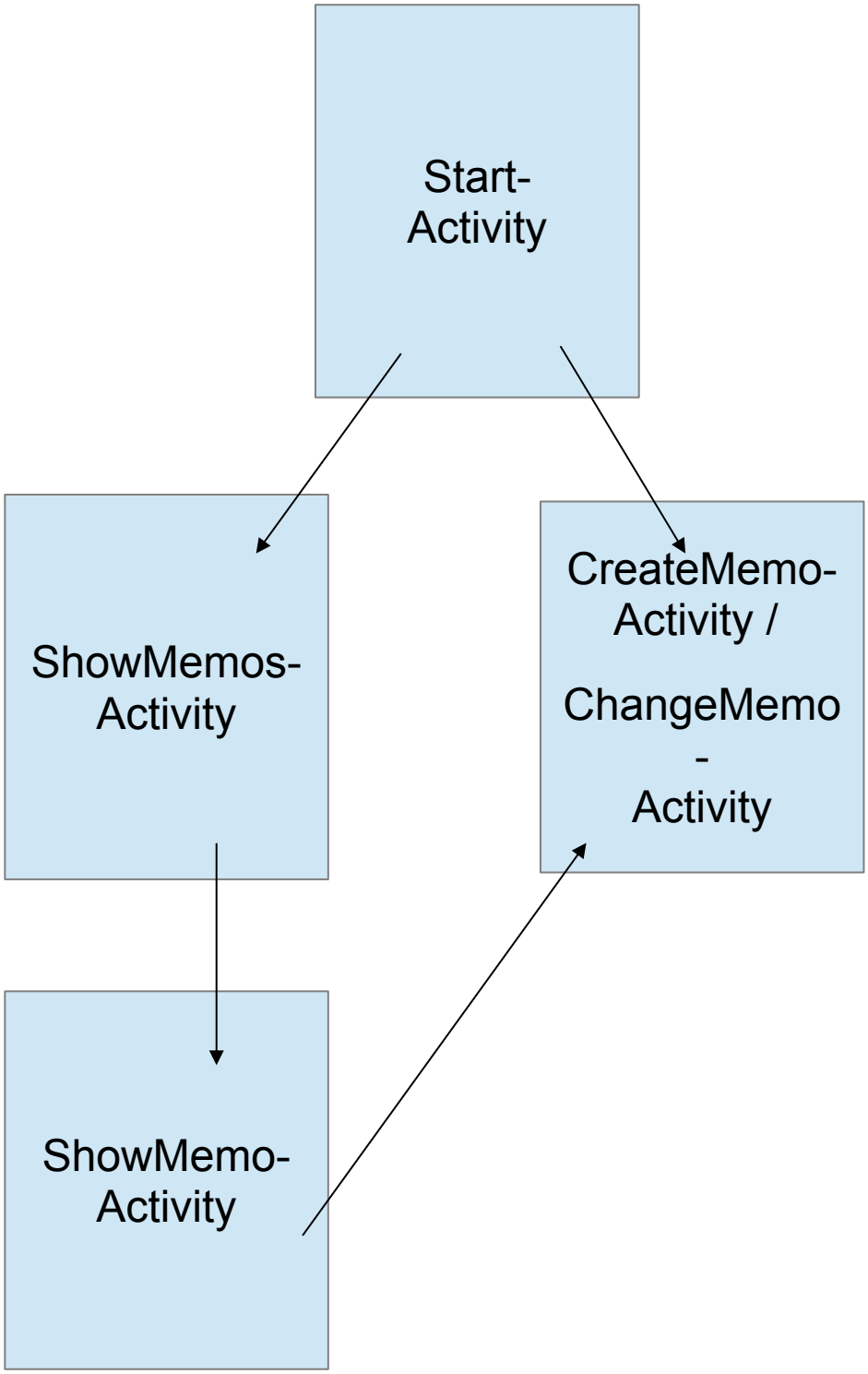
Die Activity „Neue Notiz anlegen“ zeigt jeweils ein Eingabefeld für die Überschrift und den Notiztext an und soll folgende Kommandos anbieten.

- Speichern
- Abbruch

Diese Activity dient außerdem auch zum Ändern einer Notiz. Dabei wird die aktuell ausgewählte Notiz angezeigt.

Activity	technischer Activity-Name	Controllerklasse
Start-Activity	activity_start	StartActivity
Liste der Notizen	activity_show_memos	ShowMemosActivity
Notizen anzeigen	activity_show_memo	ShowMemoActivity
Neue Notiz anlegen	activity_create_memo	CreateMemoActivity

Aufrufstruktur der verschiedenen Activities



Kapitel 2 – Activity anlegen

Jeder Bildschirminhalt ist eine sogenannte Activity. Diese besteht aus einer View, also einer grafischen Oberfläche und einem Controller, der die Daten bereitstellt und auf Ereignisse wie das Drücken auf einen Button reagiert.

In diesem Kapitel werden alle Activities beschrieben.

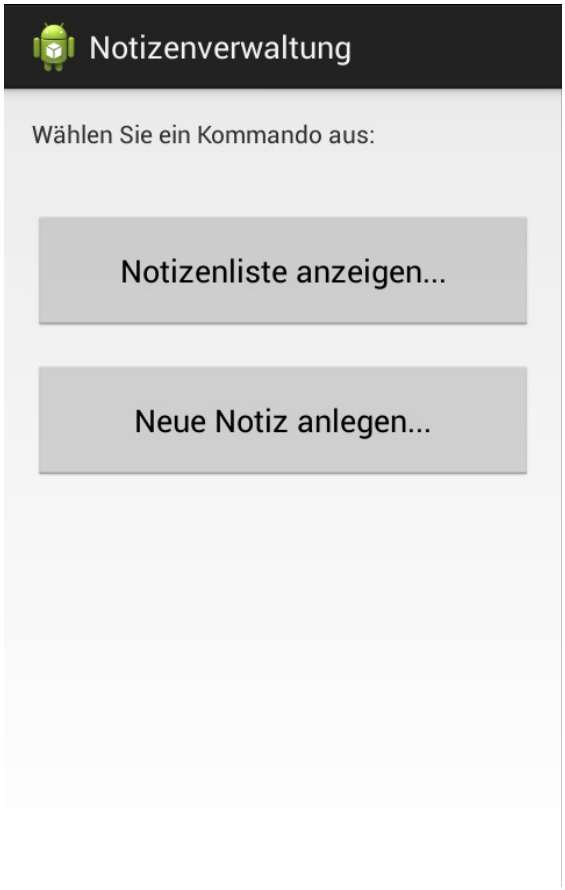
Anlegen der Start-Activity

Die erste Activity wurde bereits beim Anlegen des Projekts erstellt. Wir haben ihr den Namen activity_start gegeben. Diese befindet sich im Zweig **res/layout/activity_start.xml** und hat folgenden Inhalt.

```
<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=".StartActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/label" />

    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="70dp"
        android:layout_alignLeft="@+id/button2"
        android:layout_below="@+id/button2"
        android:layout_marginTop="16dp"
        android:text="@string/label" />

    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="70dp"
        android:layout_alignLeft="@+id/textView1"
        android:layout_below="@+id/textView1"
        android:layout_marginTop="34dp"
        android:text="@string/label" />

</RelativeLayout>
```

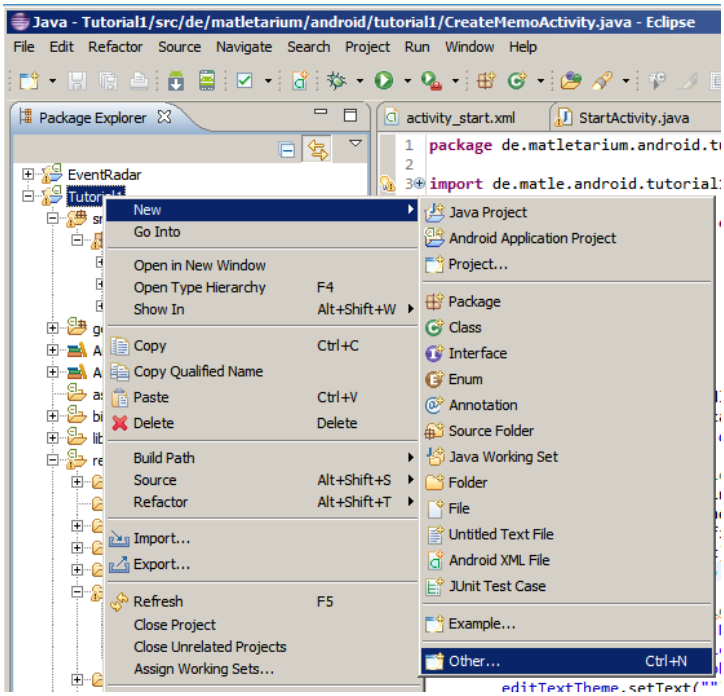
Es wird eine TextView sowie zwei Button definiert. Über die ID lassen sich diese Objekte später aus dem Controller heruas ansprechen. Die Funktion der beiden Buttons ist hierbei noch offen, weshalb die technischen Namen auch allgemein gehalten werden. Sollte sich nämlcih später daran

etwas ändern, wäre es verwirrend wenn die technischen Namen nicht mehr der Funktionalitäten entsprächen.
Das Layout entspricht ungefähr dem folgenden Bild, wobei zum Zeitpunkt der Bilderstellung die Inhalte der TextView und der Buttons bereits durch den Controller mit richtigen Werten bzw. Zeichenkettten belegt wurden. Doch dazu später mehr.

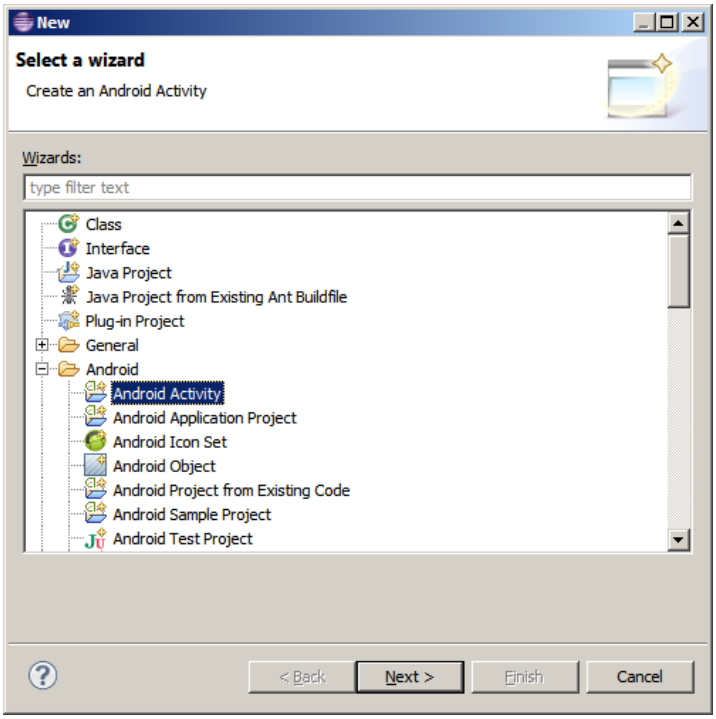


Anlegen der Activity Create Memo

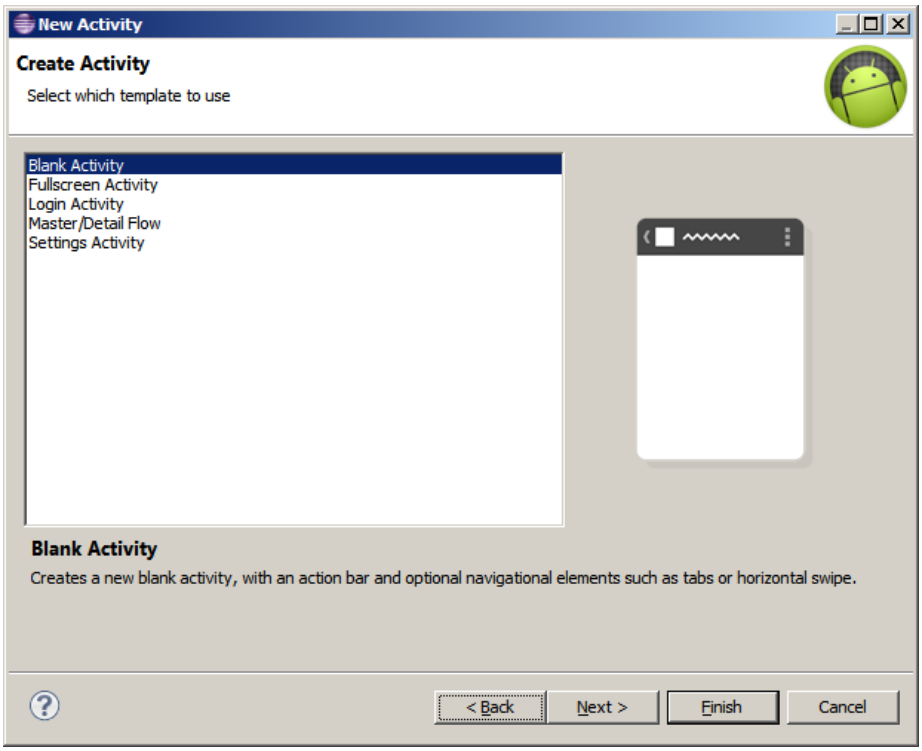
Die zweite Activity wird mit dem Wizzard von Eclipse angelegt. Dazu wählt man im Kontextmenü des Projektes Neu/Other.



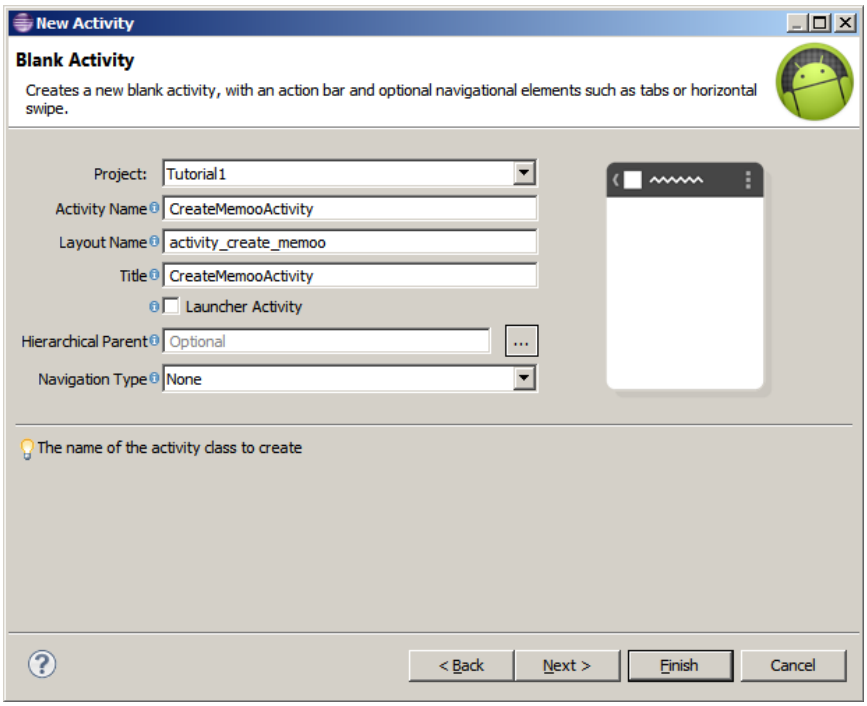
Hier wählt man Android Activity aus.



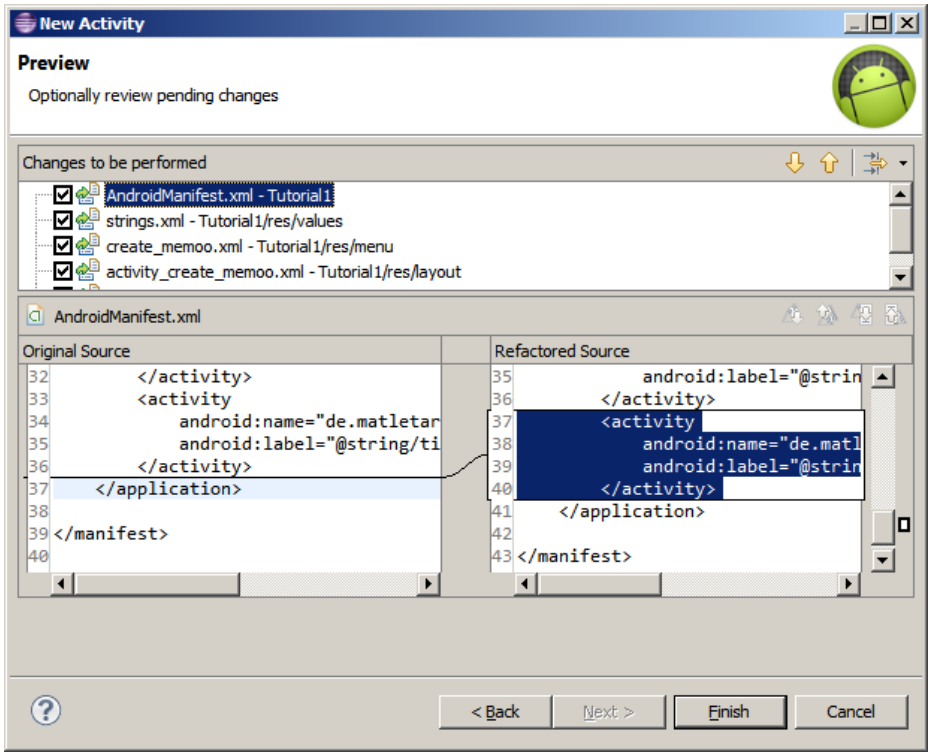
Die folgenden Eigenschaften einfach mit dem Button Next/Weiter übernehmen.



Hier wählt man einen sinnvollen Namen wie CreateMemoActivity.



Mit dem Button Next gelangt man zum letzten Wizzard-Dialog. Er zeigt alle Änderungen an, die jetzt durchgeführt werden.



Mit dem Button Finish werden alle Dateien angelegt und die Konfigurationen in die entsprechenden Dateien eingetragen.

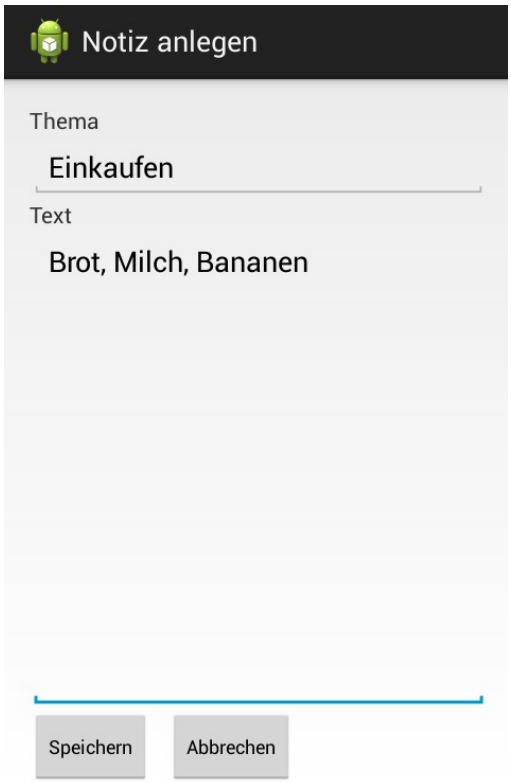
Die Datei **activity_create_memo.xml** muss folgenden Inhalt erhalten.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    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=".CreateMemoActivity" >

    <TextView
        android:id="@+id/label1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

    <EditText
        android:id="@+id/editTextTheme"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:lines="1"
        android:maxLines="1"
        android:maxLength="28"
        android:ems="10"
        android:gravity="top"
        android:inputType="text">

        <requestFocus />
    </EditText>

        <TextView
        android:id="@+id/label2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

    <EditText
        android:id="@+id/editTextMemotext"
        android:layout_width="match_parent"
        android:layout_height="0dip"
        android:layout_weight="0.25"
        android:ems="10"
        android:lines="10"
        android:gravity="top"
        android:inputType="textMultiLine" />

   <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal" >

            <Button
                android:id="@+id/button1"
```

```
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Button"
                android:textSize="12sp" />

            <Button
                android:id="@+id/button2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Button"
                android:layout_marginLeft="10dp"
                android:textSize="12sp" />

        </LinearLayout>

</LinearLayout>
```

In dieser Activity ist eine TextView definiert, die später einen kleinen Beschreibungstext anzeigen soll. Die beiden EditText-Objekte nehmen das Thema sowie den Notiztext auf. Mit den beiden Buttons steuern speichern wir die Notiz ab oder verlassen die Activity wieder. Das Layout ist so gestaltet, dass die Textview und die beiden Edittext-Felder untereinander angeordnet sind (vertical) und die beiden Buttons nebeneinander (horizontal). Siehe dazu das TAG **android:orientation.**

Das Layout sieht ungefähr wie folgt aus. Wobei auch hier durch den Controller bereits die korrekten Beschriftungen hinterlegt wurden. Damit man sich die Ansicht besser vorstellen kann, wurde eine Notiz „Einkaufen“ sowie ein passender Text erfasst.



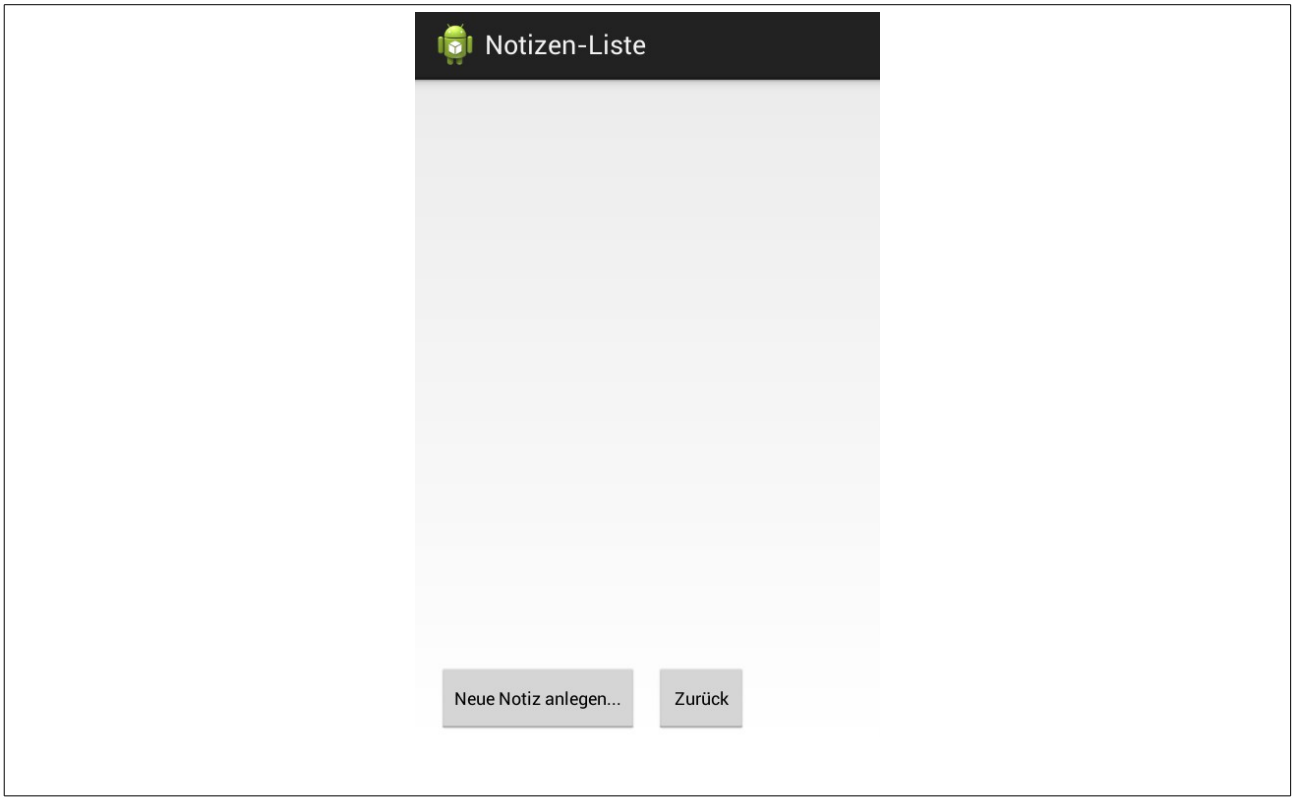
Anlegen der Activity Show Memos

Analog zum Anlegen der Activity Create Memo verwendet man wieder den Eclipse-Wizzard. Der Inhalt der Datei **activity_show_memos.xml** ist wie folgt.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    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=".ShowMemosActivity" >

    <ListView
        android:id="@+id/listView1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="0.30" >
    </ListView>

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <Button
            android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button"
            android:textSize="12sp" />

        <Button
            android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp"
            android:text="Button"
            android:textSize="12sp" />
    </LinearLayout>

</LinearLayout>
```

In dieser Activity wird neben den bereits verwendeten und bekannten Objekten wie Buttons und Textfelder auch eine Liste verwendet, die alle Notizen untereinander anzeigt.

Das Layout sieht ungefähr wie folgt aus, wobei die Liste deshalb nicht dargestellt wird, weil keine Notizen vorhanden sind.



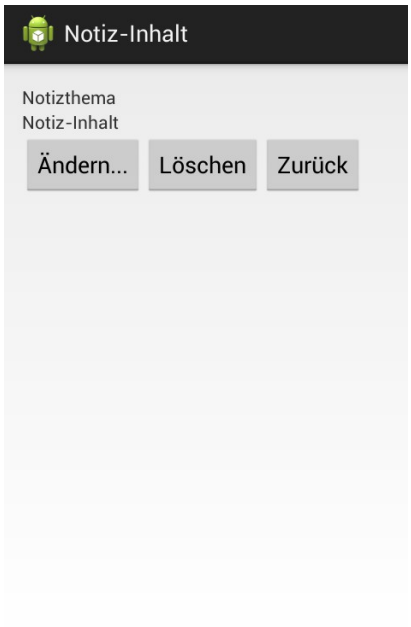
Anlegen der Activity Show Memo

Analog zum Anlegen der Activity Create Memo verwendet man wieder den Eclipse-Wizzard. Der Inhalt der Datei **activity_show_memo.xml** ist wie folgt.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    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=".ShowMemoActivity" >
   <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
          <TextView
              android:id="@+id/textView1"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:text="@string/hello_world" />
          <TextView
              android:id="@+id/textView2"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="TextView" />
      </LinearLayout>
    <TextView android:layout_height="0dip"
          android:layout_width="fill_parent"
          android:layout_weight="1" />
      <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_gravity="bottom">

        <Button
            android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button"
            android:textSize="12sp"/>

        <Button
            android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp"
            android:text="Button"
            android:textSize="12sp"/>
```

```
        <Button
            android:id="@+id/button3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp"
            android:text="Button"
            android:textSize="12sp"/>

    </LinearLayout>

</LinearLayout>
```

Das Layout sieht ungefähr wie folgt aus.



Kapitel 3 – Anlegen der Controller

Die Controller sind Java-Klassen und befnden sich im Zweig **src** und darunter in einem Java-Package. Die Controller stellen die Daten zur Verfügung, kümmern sich um die Beschriftung der View-Objekte und reagieren auf die Benutzereingaben.

Anlegen des Controllers für die Activity_Start

Die Controllerklasse dieser Activity befindet sich in der Datei **src/de.matletarium.android.tutorial1.StartActivity**.

Zunächst werden die Objekte der GUI definiert und mit Null initialisiert.

```
public class StartActivity extends Activity {

	TextView textView1=null;
	Button buttonNewMemo=null;
	Button buttonShowMemoList=null;
```

In der Methode onCreate lesen wir die Referenzen der Objekte aus der View und weisen sie unseren lokalen Variablen zu. Die View ist dem Controller mit der Anweisung **setContentView** bekannt gemacht worden. Siehe dazu auch die Datei **gen/de.matletarium.android.tutorial1.R.java.**

```
protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_start);

		// 1. Objekte aus der View lesen
		textView1= (TextView) findViewById(R.id.textView1);
		buttonNewMemo= (Button) findViewById(R.id.button1);
		buttonShowMemoList= (Button) findViewById(R.id.button2);
```

Anschließend setzt man die gewünschten Texte und Startwerte. Hier werden die Buttons und Labels beschriftet. Der Einfachheit halber werder die Texte hier „hart verdrahtet“ und nicht wie sonst üblich nach dem Prinzip i18n aus einer Sprachendatei gelesen.

```
		// 2. Objekte initialisieren
		textView1.setText("Wählen Sie ein Kommando aus:");
		buttonNewMemo.setText("Neue Notiz anlegen...");
		buttonShowMemoList.setText("Notizenliste anzeigen...");
```

Zuletzt registriert man die ActionListener und weißt ihnen Methoden zu.

```
		// 3. ActionListener registrieren
		buttonNewMemo.setOnClickListener(new View.OnClickListener() {
			@Override
			public void onClick(View v) {
				handleButtonNewMemo();
			}
		});
```

```
        buttonShowMemoList.setOnClickListener(new View.OnClickListener() {
              @Override
              public void onClick(View v) {
                    handleButtonShowMemoList();
              }
        });
```

Die beiden Handler-Methoden rufen jeweils eine andere Activity auf, einmal die Activity CreateMemo und einmal ShowMemos.

```
  private void handleButtonNewMemo(){
        Intent i= new Intent(StartActivity.this,CreateMemoActivity.class);
        startActivity(i);
  }
  private void handleButtonShowMemoList(){
        Intent i= new Intent(StartActivity.this,ShowMemosActivity.class);
        startActivity(i);
  }
```

Anlegen des Controllers für die Activity_CreateMemo

Auch hier erfolgt die Initialisierung auf die gleiche Weise.

Datei: **CreateMemoActivity.java**

```
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_create_memo);

		// 1. Objekte aus der View lesen
		textView1 = (TextView)findViewById(R.id.label1);
		textView2 = (TextView)findViewById(R.id.label2);
		buttonSave= (Button) findViewById(R.id.button1);
		buttonCancel= (Button) findViewById(R.id.button2);
		editTextTheme= (EditText) findViewById(R.id.editTextTheme);
		editTextMemo= (EditText) findViewById(R.id.editTextMemotext);

		// 2. Objekte initialisieren
		textView1.setText("Thema");
		textView2.setText("Text");
		buttonSave.setText("Speichern");
		buttonCancel.setText("Abbrechen");
		editTextTheme.setText("");
		editTextMemo.setText("");

		// 3. ActionListener registrieren
		buttonSave.setOnClickListener(new View.OnClickListener() {
			@Override
			public void onClick(View v) {
				handleButtonSave();
			}
		});
		buttonCancel.setOnClickListener(new View.OnClickListener() {
			@Override
			public void onClick(View v) {
				handleButtonCancel();
			}
		});
	}
```

Weil noch keine Daten angezeigt oder verarbeitet werden, wird über die GUI mit der Methode **Toast** nur darüber informiert, dass der Button angeklickt wurde. Der Button Cancel sorgt per **finish** dafür, dass wieder die vorherige Activity aufgerufen wird und verlässt die aktuelle Activity.

```
private void handleButtonSave(){
	Toast.makeText(CreateMemoActivity.this,"Speichere Notiz.",
				Toast.LENGTH_SHORT).show();
}
private void handleButtonCancel(){
	finish();
}
```

Anlegen des Controllers für die Activity_Showmemos

Auch hier erfolgt die Initialisierung auf die gleiche Weise.

Datei: **ShowMemosActivity.java**

```
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_show_memos);

		// 1. Objekte aus der View lesen
		aListView= (ListView) findViewById(R.id.listView1);
		buttonNewMemo= (Button) findViewById(R.id.button1);
		buttonGoBack= (Button) findViewById(R.id.button2);

		// 2. Objekte initialisieren
		buttonNewMemo.setText("Neue Notiz anlegen...");
		buttonGoBack.setText("Zurück");

		// 3. ActionListener registrieren
		buttonNewMemo.setOnClickListener(new View.OnClickListener() {
			@Override
			public void onClick(View v) {
				handleButtonNewMemo();
			}
		});
		buttonGoBack.setOnClickListener(new View.OnClickListener() {
			@Override
			public void onClick(View v) {
				handleButtonGoBack();
			}
		});

	}
```

Die durch den Listener aufgerufenen Methoden sorgen für die Anzeige und Aktvierung (**Intent**)der jeweiligen Activities oder aber das zurückspringen (**finish**)in die vorherige Ansicht.

```
	private void handleButtonNewMemo(){
		Intent i= new Intent(ShowMemosActivity.this,CreateMemoActivity.class);
		startActivity(i);
	}
	private void handleButtonGoBack(){
		finish();
	}
```

Anlegen des Controllers für die Activity_ShowMemo

Der Inhalt des letzten Controllers zeigt altbekanntes. Dasgleiche Spiel mit den GUI-Objekten, den ActionListenern und den Händlern.

Datei: **ShowMemoActivity.java**

```
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_show_memo);

		// 1. Objekte aus der View lesen
		textViewTheme = (TextView)findViewById(R.id.textView1);
		textViewMemotext = (TextView)findViewById(R.id.textView2);
		buttonGoBack= (Button) findViewById(R.id.button1);
		buttonChange= (Button) findViewById(R.id.button2);
		buttonDelete= (Button) findViewById(R.id.button3);

		// 2. Objekte initialisieren
		textViewTheme.setText("Notizthema");
		textViewMemotext.setText("Notiz-Inhalt");
		buttonChange.setText("Ändern...");
		buttonDelete.setText("Löschen");
		buttonGoBack.setText("Zurück");

		// 3. ActionListener registrieren
		buttonChange.setOnClickListener(new View.OnClickListener() {
			@Override
			public void onClick(View v) {
				handleButtonChange();
			}
		});
		buttonDelete.setOnClickListener(new View.OnClickListener() {
			@Override
			public void onClick(View v) {
				handleButtonDelete();
			}
		});
		buttonGoBack.setOnClickListener(new View.OnClickListener() {
			@Override
			public void onClick(View v) {
				handleButtonGoBack();
			}
		});
	}
```

```
	private void handleButtonChange(){
		Intent i= new Intent(ShowMemoActivity.this,CreateMemoActivity.class);
		startActivity(i);
	}
	private void handleButtonDelete(){
		Toast.makeText(ShowMemoActivity.this,
				"Lösche Notiz.",Toast.LENGTH_SHORT).show();
	}
	private void handleButtonGoBack(){
		finish();
	}
```

Kapitel 4 – Projektkonfiguration

Das Zusammenspiel aller Activities, Controllern sowie der Stringkonstanten erfolgt in anderen XML-Dateien.

In der Datei **res/values/strings.xml** werden die Stiringkonstanten definiert. Hier stehen die Titel der einzelnen Activities sowie der App-Name.

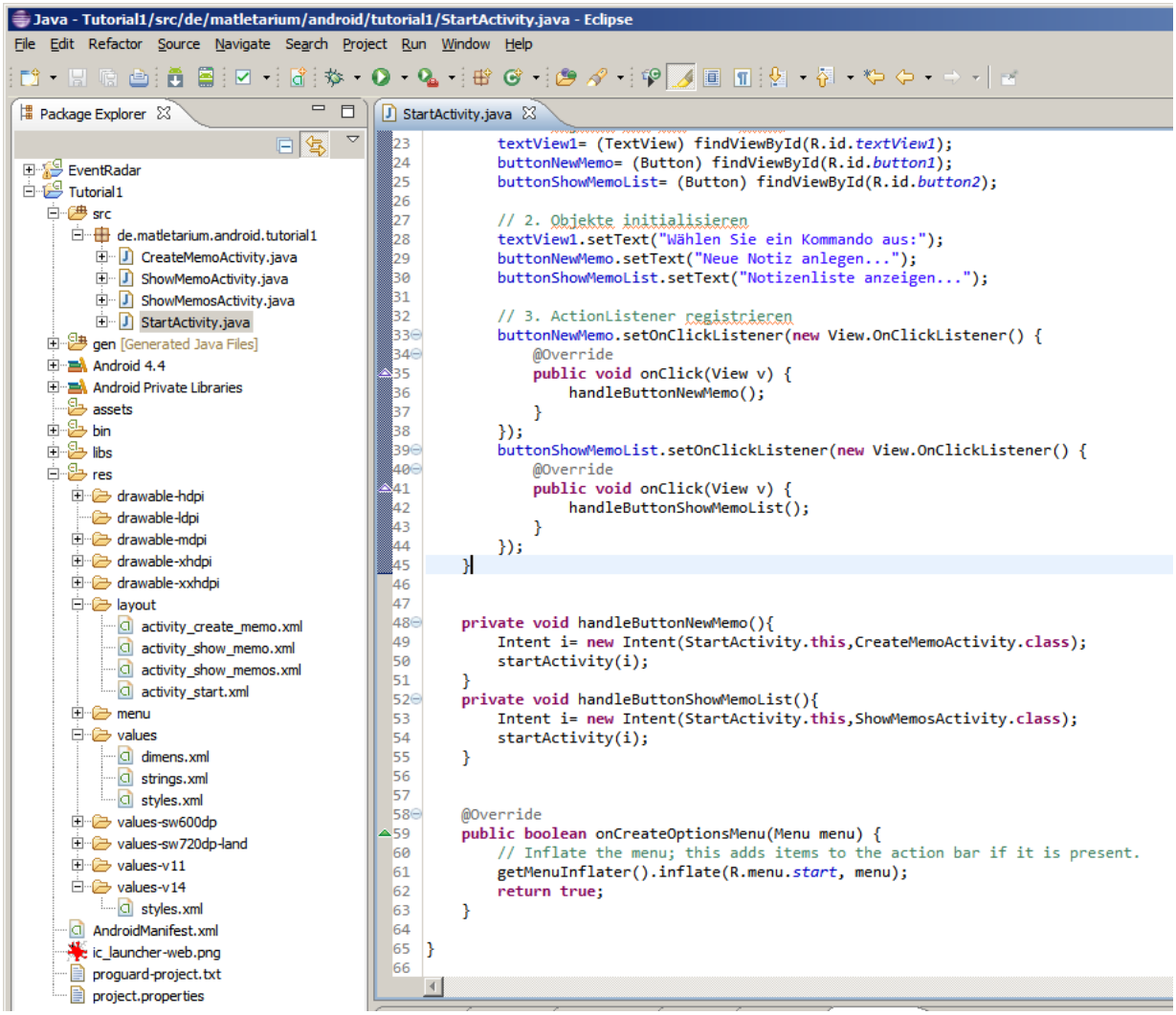
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Notizenverwaltung</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="label">label</string>
    <string name="title_activity_show_memos">Notizen-Liste</string>
    <string name="title_activity_create_memo">Notiz anlegen</string>
    <string name="title_activity_show_memo">Notiz-Inhalt</string>
</resources>
```

In der Manifest-Datei **res/AndroidManifest.xml** werden alle Activities aufgelistet.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.matletarium.android.tutorial1"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="de.matletarium.android.tutorial1.StartActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="de.matletarium.android.tutorial1.ShowMemosActivity"
            android:label="@string/title_activity_show_memos" >
        </activity>
        <activity
            android:name="de.matletarium.android.tutorial1.CreateMemoActivity"
            android:label="@string/title_activity_create_memo" >
        </activity>
```

```
        <activity
            android:name="de.matletarium.android.tutorial1.ShowMemoActivity"
            android:label="@string/title_activity_show_memo" >
        </activity>
    </application>

</manifest>
```

Insgesamt ergibt sich jetzt folgendes Bild der Eclipse-Entwicklungsumgebung mit allen besprochenen Dateien.



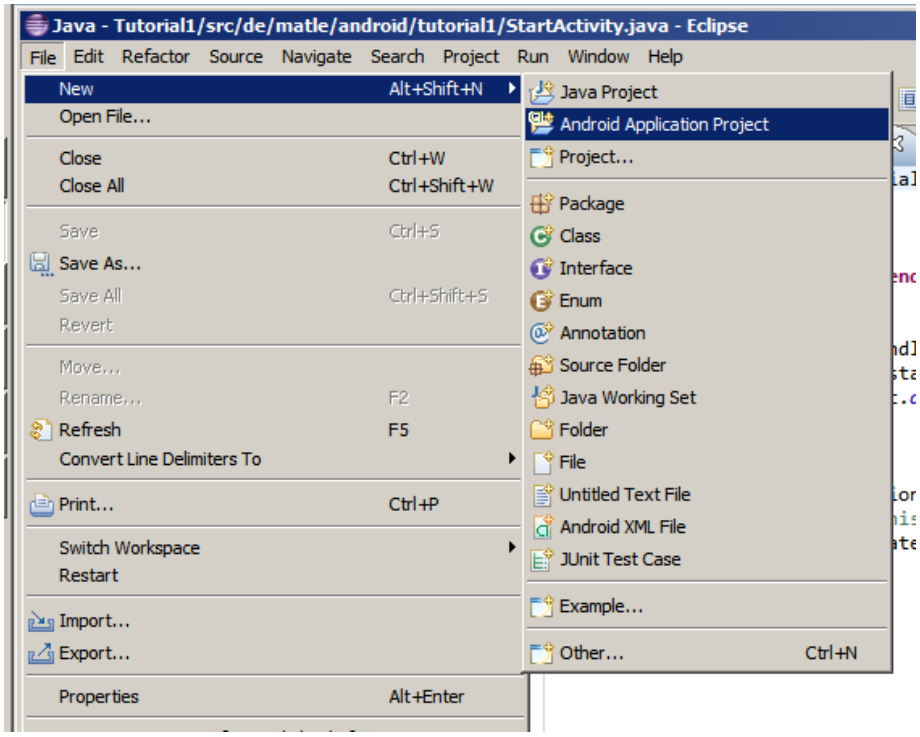
Kapitel 5 – Zusammenfassung und Ausblick zum Teil 2

In diesem Tutorial wurde gezeigt, wie man mehrere View, sogenannte Activities, erstellt und mit Hilfe von Buttons dorthin navigiert. Auf schöne Oberflächen wurde verzichtet. Es fehlen auch Funktionen zum Umgang und der Anzeige von Daten.

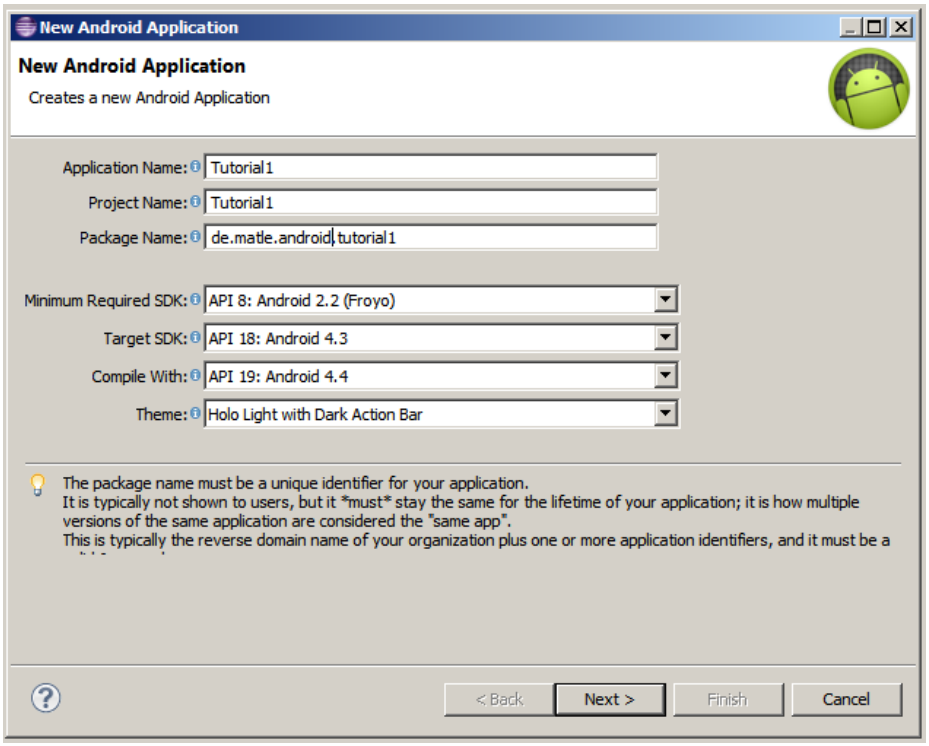
Im nächsten Teil soll die Oberfläche etwas schöner gestaltet werden. Außerdem sollen sich Notizen erfassen und speichern lassen.

Anhang A - Neues Android-Projekt anlegen

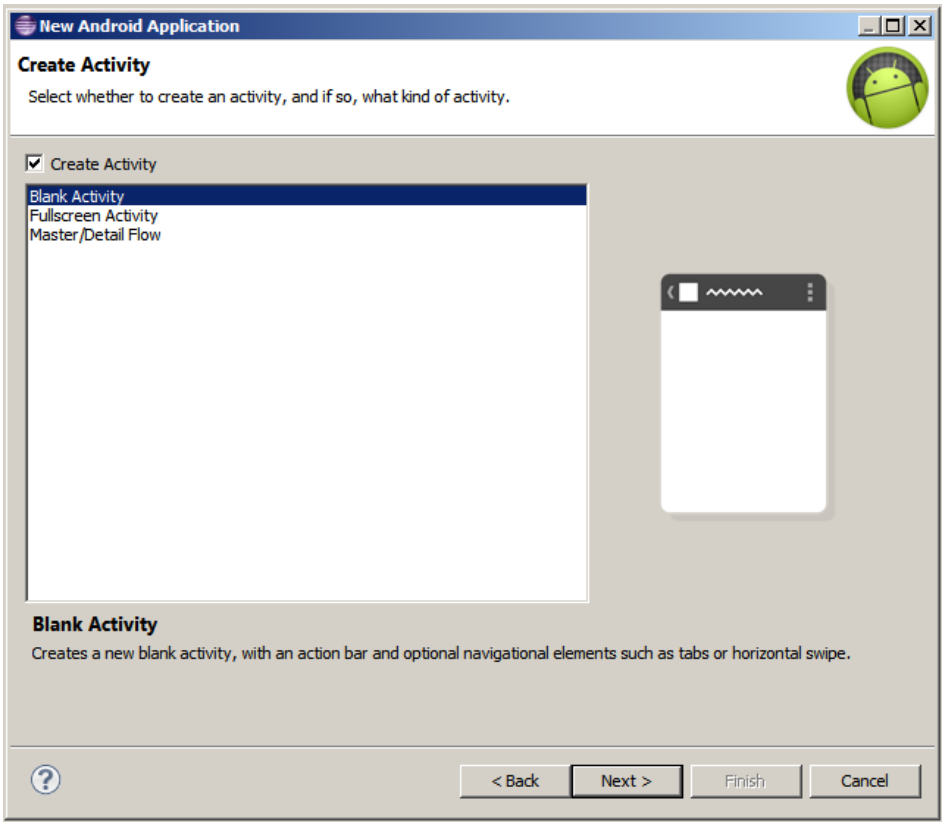
Diese Projekt kann ganz einfach selbst erstellt werden.

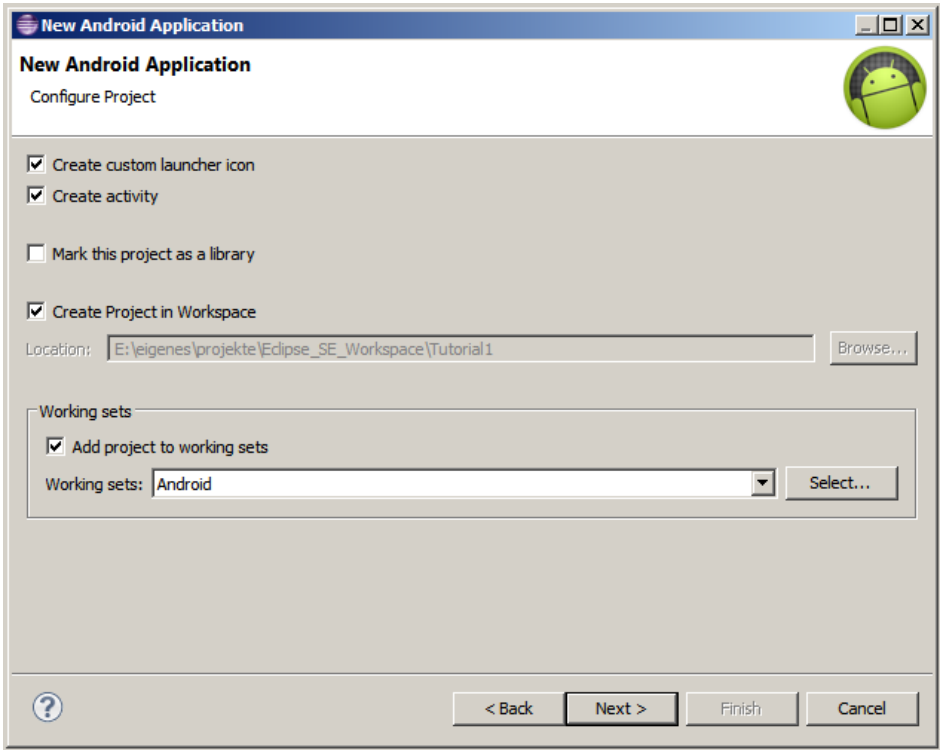


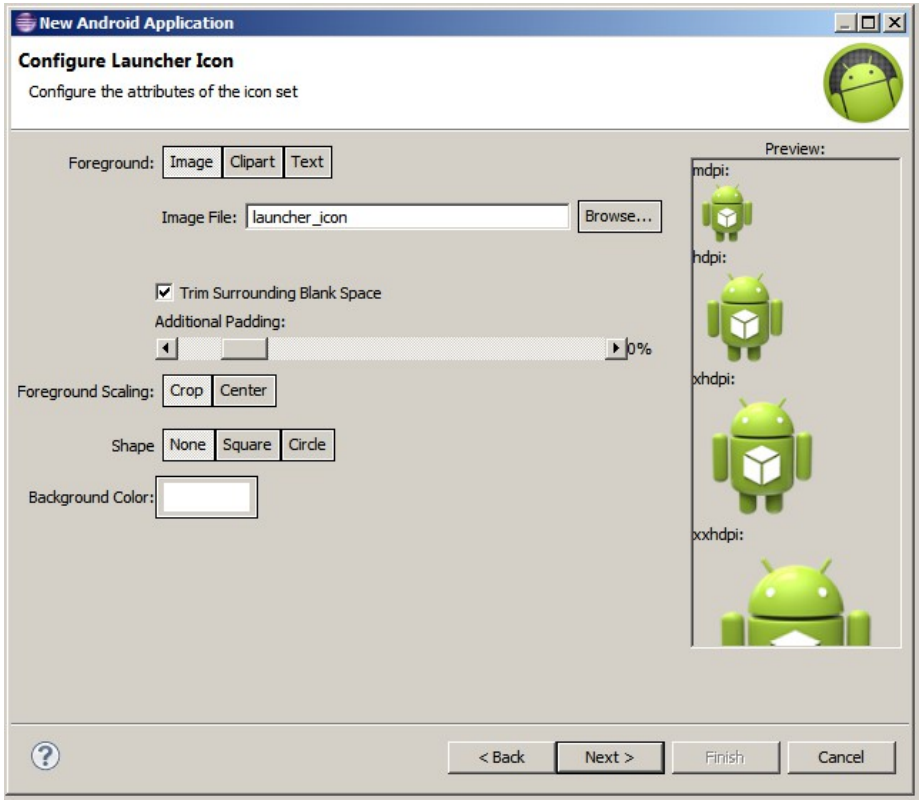
Projektnamen vergeben...



auf Next klicken...







Mit Finish wird das Projekt angelegt.