

1 Ein erstes Beispiel

In diesem Abschnitt werden wir ein erstes Android-Programm erstellen. Es dient dem schnellen Einstieg in die Programmierung von Android.

Dabei handelt es sich um ein Programm zur Berechnung der Umsatzsteuer. Man gibt den Ausgangsbetrag an, wählt aus, ob man den Brutto- oder Nettobetrag angegeben hat, und wählt die Umsatzsteuer in Prozent. Abbildung 1-2 auf Seite 9 zeigt das Formular für die Eingabe. Die Berechnung wird gestartet, indem man den Optionsmenüpunkt »Umrechnen« wählt. Das Ergebnis wird auf einer zweiten Bildschirmseite angezeigt (siehe Abb. 1-3 auf Seite 17).

*Brutto-Netto-
Umrechner*

Wir werden uns bei der Erstellung des Programms nur auf die unbedingt notwendigen Elemente beschränken, um eine lauffähige Anwendung zu implementieren. Dabei werden wir noch nicht alles genau erklären. Im zweiten Teil des Buchs werden wir sehr viel tiefer ins Detail gehen und die offenen Fragen beantworten.

1.1 Projekt anlegen

Wir verwenden für die Entwicklung Eclipse mit dem Android-Plugin. Die Installation wollen wir hier nicht wiederholen. Sie ist sehr gut unter [37] beschrieben.

Mit Hilfe des Android-Plugin für Eclipse ist die Erstellung eines Projekts recht einfach (Abb. 1-1). Zu Beginn legt man den Namen des Projekts fest. Große Bedeutung kommt der Wahl des »Build Target« zu. Da es Android nun schon in mehreren Version gibt, die teilweise nicht aufwärtskompatibel sind, sollte man sich hier vorher überlegen, für welche Plattform man minimal entwickelt. Es ist sinnlos, hier »Android 2.1« zu wählen, wenn man nur Klassen und Methoden aus Android 1.5 oder Android 1.6 verwendet. Man sollte in der Praxis immer das kleinstmögliche »Build Target« wählen, damit die Anwendung auf möglichst vielen Endgeräten lauffähig ist. Bei »Min SDK Version« sollte man die Nummer des API-Levels (rechte Spalte der Tabelle »Build Target«) eintragen.

*Android-Plugin
verwenden*

*Kleinster gemeinsamer
Nenner*

Abb. 1-1
Projekt anlegen

New Android Project

Creates a new Android Project resource.



Project name:

Contents

Create new project in workspace
 Create project from existing source
 Use default location

Location:

Create project from existing sample
 Samples:

Build Target

Target Name	Vendor	Platform	API Lev
<input type="checkbox"/> Android 1.1	Android Open Source Project	1.1	2
<input type="checkbox"/> Android 1.5	Android Open Source Project	1.5	3
<input type="checkbox"/> Google APIs	Google Inc.	1.5	3
<input type="checkbox"/> Android 1.6	Android Open Source Project	1.6	4
<input type="checkbox"/> Google APIs	Google Inc.	1.6	4
<input checked="" type="checkbox"/> Android 2.0	Android Open Source Project	2.0	5
<input type="checkbox"/> Google APIs	Google Inc.	2.0	5
<input type="checkbox"/> Android 2.0.1	Android Open Source Project	2.0.1	6
<input type="checkbox"/> Google APIs	Google Inc.	2.0.1	6
<input type="checkbox"/> Android 2.1	Android Open Source Project	2.1	7
<input type="checkbox"/> Google APIs	Google Inc.	2.1	7

Standard Android platform 2.0

Properties

Application name:

Package name:

Create Activity:

Min SDK Version:

Unter »*Properties*« legt man, wie bei jedem anderen Java-Projekt auch, den Paketnamen und die zu erstellende Klasse fest, die den Programm-Eintrittspunkt darstellt. Zusätzlich ist hier der Name der Anwendung anzugeben. In unserem Fall erzeugen wir eine sogenannte *Activity*. *Activities* implementieren die Bildschirmseiten einer Android-Anwendung. Der Aufbau einer Seite wird meist in XML definiert. Die dort auszuführenden Aktionen werden in Java-Klassen implementiert. Insofern ähnelt der Aufbau von Android-Anwendungen dem von Webanwendungen.

*Programm-
Eintrittspunkt
festlegen*

Nach Fertigstellung legt der Projekt-Wizard die folgende Ordnerstruktur für Android-Projekte an:

- *src*: Java-Quelltexte (u.a. auch unsere Activity *FormularActivity*)
- *gen*: Automatisch generierte Klassen, wie z.B. die *R*-Klasse mit den Indizes der Ressourcen (Kapitel 5.3)
- *res*: Ressourcen, d.h. alle Nicht-Java-Texte und alle Nicht-Programmelemente, wie zum Beispiel Bibliotheken. Hier werden u.a. die Dateien zur Definition der Oberflächen, Bilder oder Textdefinitionen abgelegt.
- *asset*: Zusätzlicher Ordner zur Ablage von Ressourcen

Im Wurzelverzeichnis befindet sich die zentrale Datei zur Definition von Metadaten der Anwendung: das *AndroidManifest.xml*.

1.2 Die erste Activity

Wir implementieren nun unsere erste Activity, die die Startseite unserer Anwendung anzeigen wird.

*Startseite
implementieren*

```
package de.androidbuch.rechner;

import android.app.Activity;
import android.os.Bundle;

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

Listing 1.1
*Eingaben erfassen:
FormularActivity*

Für den Anfang reicht es uns zu wissen, dass unsere eigenen *Activities* von der Android-API-Klasse *Activity* abgeleitet werden müssen. *Activi-*

ties implementieren die sichtbaren Bestandteile einer Anwendung und interagieren mit dem Anwender.

1.3 Layout definieren

XML GUI

Wenden wir uns nun der Erstellung unserer Eingabemaske zu. Die Maskelemente werden in einer XML-Datei definiert. Der Vollständigkeit halber sei noch erwähnt, dass die Masken auch via Programmcode erstellt werden können. Dies ist aber, wie im Falle von Webanwendungen (JSP vs. Servlets), aus Gründen der Übersichtlichkeit und Wartbarkeit stets die zweite Wahl und wird daher nicht Thema dieses Buches sein.

Das Android-Plugin hat bereits eine solche XML-Datei `res/layout/main.xml` erstellt, die in Listing 1.2 dargestellt ist.

Listing 1.2

Eine einfache `main.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/
    apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

Es gibt verschiedene
Layout-Typen.

Ähnlich wie bei Swing-Anwendungen können verschiedene Layouts für den Aufbau der Maske verwendet werden. Beim Erstellen eines Android-Projekts wird automatisch ein *LinearLayout* mit vertikaler Ausrichtung gewählt, das wir zunächst übernehmen wollen.

Das XML-Element `TextView` enthält ein Attribut `android:text`. Hier handelt es sich um einen Verweis auf eine Zeichenkettendefinition. Sie befindet sich in der Datei `strings.xml` im Ordner `/res/values`. Die Datei hat folgenden Inhalt:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World,
        FormularActivity</string>
    <string name="app_name">Umsatzsteuerrechner</string>
</resources>
```

Text wird ausgelagert.

Der Schlüssel für den Text, der in dem Anzeigeelement `TextView` dargestellt werden soll, lautet »hello«. Er wird in der `TextView` zur Laufzeit

durch den Wert aus `strings.xml` ersetzt.

Der nächste Schritt ist nun, dieses Layout für unsere Zwecke anzupassen. Dazu überlegen wir uns, welche Oberflächenelemente für den Umsatzsteuerrechner nötig sind (siehe Tabelle 1-1).

Feldname	Funktion	Darstellung
-	Funktionsbeschreibung	Text
betrag	Fließkommazahl (Euro)	Texteingabe
art	»Brutto«, »Netto«	Radiobutton
umsatzsteuer	»19 Prozent«, »16 Prozent«, »7 Prozent«	Auswahlliste

Tab. 1-1
Feldliste »Eingabe erfassen«

Nun passen wir die Oberfläche an unsere Anforderungen an. Dazu definieren wir die Formularelemente aus Tabelle 1-1 in XML. Die Datei `main.xml` sieht nach der Erweiterung wie folgt aus:

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

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Geben Sie einen Brutto- oder Nettobetrag
            ein und lassen Sie sich die Umsatzsteuer errechnen:" />

    <EditText android:id="@+id/edt_betrag"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal" />

    <RadioGroup android:id="@+id/rg_art"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <RadioButton android:id="@+id/rb_art_netto"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Netto"
            android:checked="true" />
```

Listing 1.3
main.xml für den
Umsatzsteuerrechner

```

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

    <Spinner android:id="@+id/sp_umsatzsteuer"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:drawSelectorOnTop="true"
        android:entries="@array/ust_anzeige"
        android:entryValues="@array/ust_werte" />

</LinearLayout>

```

Spinner = Auswahlliste

Wir haben das Layout um ein Texteingabefeld (EditText) und eine RadioGroup, bestehend aus zwei RadioButtons, ergänzt. Zusätzlich ist ein Spinner dazugekommen. Ein Spinner ist eine Auswahlliste. In unserem Beispiel ist die Wertebelegung für den Spinner statisch, so dass wir sie in eine weitere XML-Datei im values-Verzeichnis auslagern können (Listing 1.4). Wir geben ihr den Namen arrays.xml.

Listing 1.4
res/arrays.xml

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <array name="ust_anzeige">
        <item>19 Prozent</item>
        <item>16 Prozent</item>
        <item>7 Prozent</item>
    </array>
    <array name="ust_werte">
        <item>19</item>
        <item>16</item>
        <item>7</item>
    </array>
</resources>

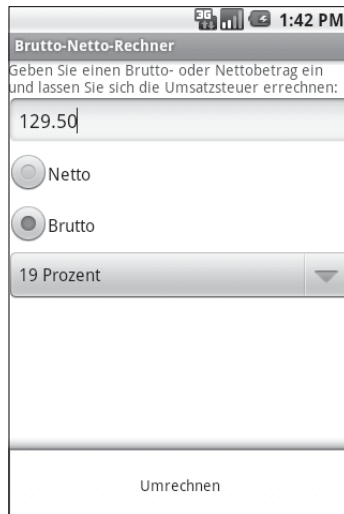
```

*Eine Anwendung
starten*

Geschafft! Unser Formular zur Erfassung der Eingabewerte ist fertig. Nun muss die Anwendung nur noch im Emulator gestartet werden. Auch dazu bedienen wir uns der vorhandenen Eclipse-Umgebung (*Run* → *Run* → *Android Application*). Nach einer Wartezeit sollte die folgende Bildschirmseite erscheinen (Abb. 1-2).

Tipp!

Der Emulator braucht recht lange zum Starten. Starten Sie ihn daher zu Beginn einmal und schließen Sie das Emulatorfenster nicht. Jeder weitere Start der Anwendung erfolgt dann sehr schnell.

**Abb. 1-2**

Beispielanwendung im Emulator

1.4 Activities aufrufen

Beschäftigen wir uns nun mit Interaktionen zwischen Activities. Wenn die Menüoption »Umrechnen« gedrückt wird, soll eine neue Seite erscheinen, auf der das Ergebnis der Berechnung angezeigt werden soll. Abhängig davon, ob die eingegebene Zahl einen Brutto- oder Nettobetrag darstellt, soll entsprechend der Prozentzahl der korrekte Umsatzsteuerbetrag angezeigt werden. Wir brauchen dafür eine weitere Bildschirmseite, die folgende Ergebnisfelder enthalten soll:

- Nettobetrag: Betrag ohne Umsatzsteuer
- Umsatzsteuerbetrag
- Bruttobetrag: Betrag inkl. Umsatzsteuer

Für die zweite Bildschirmseite werden wieder ein Layout und eine Activity benötigt. Anhand dieser Seite demonstrieren wir

- die Verwendung von dynamischen Inhalten in Activities und
- die Interaktion zwischen Activities.

Interaktionen zwischen Activities

Mehr Aktivität

Beginnen wir mit der Definition der Oberfläche. Hierzu erzeugen wir das Layout (Seiten-Beschreibungsdatei) `ergebnis_anzeigen.xml` und legen sie unterhalb von `/res/layout` ab.

Hinweis

Der Name von Dateien unterhalb des Ordners `/res` darf nur aus Ziffern und Kleinbuchstaben sowie dem Unterstrich bestehen. Der eingängigere Name `ergebnisAnzeigen.xml` (die Java-übliche »Camel-Case«-Schreibweise) wäre daher nicht erlaubt. Wir verwenden die Schreibweise generell in den Ressourcendateien (vgl. Listing 1.4) zur Namensgebung, auch wenn dort die Camel-Case-Schreibweise erlaubt ist.

Listing 1.5

Table Layout, Ergebnis anzeigen

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android=
    "http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TableRow>
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Nettobetrag:" />
        <TextView
            android:id="@+id/txt_nettobetrag"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />
    </TableRow>

    <TableRow>
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Umsatzsteuer:" />
        <TextView
            android:id="@+id/txt_umsatzsteuer"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />
    </TableRow>

    <TableRow>
        <TextView
            android:layout_width="fill_parent"
```

```

        android:layout_height="wrap_content"
        android:text="Bruttobetrag:" />
    <TextView
        android:id="@+id/txt_bruttobetrag"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
</TableRow>

</TableLayout>

```

Für diese View verwenden wir ein `TableLayout`, da die Ergebnisse in Tabellenform dargestellt werden sollen. Ansonsten gleicht diese Bildschirmseitendefinition der vorherigen.

TableLayout

1.5 Das Android-Manifest

Die mit diesem Layout verknüpfte Activity `ErgebnisActivity` muss dem System erst noch bekannt gemacht werden. Dazu wird sie im `AndroidManifest.xml` des Projektes registriert. Listing 1.6 zeigt das vollständige Android-Manifest der Einführungsanwendung.

```

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

    <uses-sdk android:minSdkVersion="5" />

    <application android:icon="@drawable/icon"
        android:label="@string/app_name"
        android:theme="@android:style/Theme.Light"> (1)

        <activity android:name=".FormularActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name=
                    "android.intent.action.MAIN" />
                <category android:name=
                    "android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

```

Listing 1.6

AndroidManifest.xml

```
<activity android:name=".ErgebnisActivity" />           (2)
</application>
```

```
</manifest>
```

Das Android-Manifest liegt im Wurzelverzeichnis des Projekts. Es wurde automatisch generiert, als wir unser Projekt angelegt haben. Wir führen nun die notwendigen Erweiterungen durch.

Innerhalb des `<application>`-Elements findet die Deklaration der Activities statt. Für die Activity `FormularActivity` wird ein sogenannter Intent-Filter definiert. Mit Intents und Intent-Filtern werden wir uns in Kapitel 7 ausführlicher befassen. Ein Intent repräsentiert einen konkreten Aufruf einer anderen Activity, eines Hintergrundprozesses oder einer externen Anwendung. Wir können den englischen Begriff mit »*Absichtserklärung*« übersetzen. Der hier verwendete Intent-Filter sorgt dafür, dass die Umsatzsteuerrechner-Anwendung gestartet wird, indem die Activity `FormularActivity` angezeigt wird. Der Intent selbst wird vom Android-System verschickt, sobald man die Anwendung startet.

Intent ruft Activity auf.

Wir haben im Manifest zwei Änderungen durchgeführt. Zunächst haben wir durch die Verwendung eines sogenannten »Themes« dafür gesorgt, dass unsere Anwendung etwas freundlicher und heller aussieht, indem unter anderem ein heller Hintergrund verwendet wird statt einem schwarzen (1). Dazu haben wir das `<application>`-Element um ein XML-Attribut erweitert. Anschließend haben wir die Activity zur Darstellung des Ergebnisses eingefügt (2).

Ergebnis berechnen

Zum Berechnen des Ergebnisses benötigen wir einen Menüeintrag in der Bildschirmseite `FormularActivity`. Dazu erweitern wir die Activity um eine zusätzliche Methode. Sie erzeugt einen Menüeintrag »*Umrechnen*« (siehe Listing 1.7).

Listing 1.7
*Implementierung
Standardmenü*

```
public static final int AUSRECHNEN_ID = Menu.FIRST;
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    menu.add(0, AUSRECHNEN_ID, Menu.NONE, "Umrechnen");
    return super.onCreateOptionsMenu(menu);
}
```

Werte übergeben

Als nächsten Schritt lassen wir die beiden Activities miteinander kommunizieren. Konkret soll `FormularActivity` nach Auswahl der Menüoption »*Umrechnen*« die Activity `ErgebnisActivity` aufrufen. Dabei sollen die Formularwerte übertragen werden, das Ergebnis berechnet und auf dem Bildschirm dargestellt werden.

In unserem Fall muss `FormularActivity` einen `Intent` erzeugen, ihn mit Übergabeparametern versehen und anschließend ausführen, damit `ErgebnisActivity` aufgerufen wird. Listing 1.8 zeigt den dafür erforderlichen Code aus `FormularActivity`.

```
public static final String BETRAG_KEY = "betrag";
public static final String BETRAG_ART = "art";
public static final String UST_PROZENT = "ust";

@Override
public boolean onOptionsItemSelected(
    MenuItem item) { // (1)
    switch (item.getItemId()) { // (2)
        case AUSRECHNEN_ID:
            // Betrag:
            final EditText txtBetrag =
                (EditText) findViewById(R.id.edt_betrag); // (3)
            final float betrag = Float.parseFloat(
                txtBetrag.getText().toString()); // (4)

            // Art des Betrags (Brutto, Netto):
            boolean isNetto = true;
            final RadioGroup rg =
                (RadioGroup) findViewById(R.id.rg_art);
            switch (rg.getCheckedRadioButtonId()) {
                case R.id.rb_art_netto:
                    isNetto = true;
                    break;
                case R.id.rb_art_brutto:
                    isNetto = false;
                    break;
                default:
            }

            // Prozentwert Umsatzsteuer:
            final Spinner spinner =
                (Spinner) findViewById(R.id.sp_umsatzsteuer);
            final int pos = spinner.getSelectedItemPosition();
            final int[] prozentwerte =
                getResources().getIntArray(R.array.ust_werte);
            final int prozentwert = prozentwerte[pos];

            final Intent intent = new Intent(this, // (5)
                ErgebnisActivity.class);

            intent.putExtra(BETRAG_KEY, betrag); // (6)
```

Listing 1.8

*Aufruf anderer
Activities per Intent*

```

        intent.putExtra(BETRAG_ART, isNetto);
        intent.putExtra(UST_PROZENT, prozentwert);

        startActivity(intent); // (7)

        default:
    }

    return super.onOptionsItemSelected(item);
}

```

Programmablauf

Wir wollen hier im Einstiegsbeispiel den Quellcode noch nicht im Detail erklären, dies geschieht später ausführlich in den entsprechenden Kapiteln. Wir geben lediglich einen Überblick über die Funktionsweise des Programmcodes.

- Methode `onOptionsItemSelected` wird ausgeführt, wenn Menüeintrag »Umrechnen« gewählt wurde (1).
- Switch-Case-Anweisung wertet die Menüeinträge aus (wir haben hier nur einen) (2).
- `findViewById` liefert Zugriff auf ein Oberflächenelement (View genannt, z.B. das Texteingabefeld) (3).
- Es wird der Wert aus dem View-Element des Formulars geholt (4).
- Es wird ein Intent zum Aufruf der Folge-Activity erzeugt (5).
- Der Wert aus dem Formular wird dem Intent als Übergabeparameter hinzugefügt (6).
- Die Folge-Activity wird mit Hilfe des Intents aufgerufen (7).

Ergebnis berechnen

Auf der Gegenseite muss nun die Activity `ErgebnisActivity` erstellt werden. In ihrer `onCreate`-Methode wird der Intent entgegengenommen, und seine Daten werden verarbeitet. Zuvor implementieren wir jedoch eine Hilfsklasse, die die Berechnung der Ergebniswerte übernimmt. Ihr kann man die Formularwerte übergeben und den Umsatzsteuerbetrag berechnen lassen. Die entsprechenden Attribute der Klasse liefern die Ergebnisse (Netto-, Brutto und Umsatzsteuerbetrag), welche in der `ErgebnisActivity` zur Anzeige gebracht werden sollen.

Listing 1.9
*Hilfsklasse zur
 Berechnung des
 Ergebnisses*

```

public class Ergebnis {

    public float betrag;
    public boolean isNetto;
    public int ustProzent;

    public float betragNetto;
    public float betragBrutto;
}

```

```
public float betragUst;

public void berechneErgebnis() {
    // Berechne Bruttobetrag aus Nettobetrag:
    if (isNetto) {
        betragNetto = betrag;
        betragUst = betrag * ustProzent / 100;
        betragBrutto = betrag + betragUst;
    } else {
        // Berechne Nettobetrag aus Bruttobetrag:
        betragBrutto = betrag;
        betragUst = betrag * ustProzent /
            (100 + ustProzent);
        betragNetto = betrag - betragUst;
    }
}
```

Für Java-erfahrene Programmierer ist der Verzicht auf Getter- und Setter-Methoden für die Attribute der Klasse ungewohnt. Man verzichtet bei der Android-Programmierung üblicherweise aus Performanzgründen darauf. Es gibt eine Reihe weiterer Tipps für effiziente Programmierung unter Android. Kapitel 21 oder die Online-Dokumentation ([17]) bieten einen guten Überblick.

Keine Getter- und Setter-Methoden

Nun können wir die Activity zur Anzeige des Ergebnisses implementieren. Listing 1.10 zeigt den Quellcode.

```
public class ErgebnisActivity extends Activity {

    @Override
    public void onCreate(Bundle icle) {
        super.onCreate(icle);
        setContentView(R.layout.ergebnis_anzeigen);

        final Bundle extras = getIntent().getExtras();

        if (extras != null) {
            final Ergebnis ergebnis = new Ergebnis();

            ergebnis.betrag =
                extras.getFloat(FormularActivity.BETRAG_KEY);
            ergebnis.isNetto =
                extras.getBoolean(FormularActivity.BETRAG_ART,
                    true);
            ergebnis.ustProzent =
                extras.getInt(FormularActivity.UST_PROZENT);
```

Listing 1.10
*Activity zur Anzeige
des Ergebnisses*

```
        zeigeErgebnis(ergebnis);
    }
}

private void zeigeErgebnis(Ergebnis ergebnis) {
    setTitle("Ergebnis");

    ergebnis.berechneErgebnis();

    final TextView txtNettobetrag =
        (TextView) findViewById(R.id.txt_nettoebetrag);
    txtNettobetrag.setText(String.valueOf(
        ergebnis.betragNetto));

    final TextView txtUmsatzsteuer =
        (TextView) findViewById(R.id.txt_umsatzsteuer);
    txtUmsatzsteuer.setText(
        String.valueOf(ergebnis.betragUst));

    final TextView txtBruttobetrag =
        (TextView) findViewById(R.id.txt_bruttobetrag);
    txtBruttobetrag.setText(
        String.valueOf(ergebnis.betragBrutto));
}
}
```

Zugriff auf Views

In der `onCreate`-Methode holen wir uns die Formularwerte aus dem Intent, den wir von `FormularActivity` erhalten haben. Die Methode `zeigeErgebnis` führt die Berechnung in der Klasse `Ergebnis` durch. Wir holen uns die für die Ergebnisanzeige nötigen View-Elemente aus dem Layout (siehe Listing 1.5) und setzen die Ergebniswerte mittels der Methode `setText`. Abbildung 1-3 zeigt die Ergebnisseite im Emulator.

In diesem Beispiel ging es darum, das Prinzip der losen Kopplung von Komponenten, hier zwei Activities, zu verdeutlichen. Wir haben gesehen, wie eine Activity einen Intent verschickt, sobald der Menüeintrag »Umrechnen« gedrückt wurde. Die auf Intents basierende offene Kommunikationsarchitektur stellt eine der Besonderheiten von Android dar.

Das Ergebnis im Emulator

Zum Abschluss dieser Einführung schauen wir uns das Ergebnis im Emulator an (siehe Abb. 1-3). Der vollständige Quellcode steht unter <http://www.androidbuch.de> zum Herunterladen bereit. Es empfiehlt sich, dieses Beispiel auf eigene Faust zu verändern und die Resultate unmittelbar im Emulator zu betrachten.

**Abb. 1-3**

Die Ergebnisseite des Umsatzsteuerrechners

1.6 Fazit

In diesem Abschnitt gaben wir Ihnen einen kurzen Einblick in die Programmierung von Android-Geräten. Kennengelernt haben wir die Grundbestandteile

- Bildschirmseiten-Erstellung
- Formularverarbeitung
- Interaktion zwischen Bildschirmseiten
- Start der Laufzeitumgebung und des Emulators

sowie die Android-Artefakte

- Activity
- Layout
- View
- Intent
- Android-Manifest

Nun ist es an der Zeit, sich ein wenig mit der Theorie zu befassen. Der Rest dieses ersten Teils beschäftigt sich mit den Konzepten hinter Android.