Uwe Post

Eine Spiele-App von A bis Z

Android-Apps entwickeln

Ideal für Programmiereinsteiger geeignet

- Schritt für Schritt zu eigenen Apps und Spielen
- Inkl. Sprachgrundlagen von Java
- Animationen, Sounds, Zeichnen, Kamera, Bewegungssensoren, Highscores u. v. m.

DVD ROM Komplettes Startpaket mit der benötigten Software und allen Beispielen

Galileo Computing

Inhalt

»Wir irren uns nie.«
(HAL 9000)

4 Die erste App

Software installieren, Java-Crashkurs ... Jetzt wird es Zeit für Ihre erste App. Starten Sie Eclipse, schließen Sie Ihr Handy an, stellen Sie Kaffee (oder Tee) und Kekse bereit. Fertig? Auf in den Kampf!

4.1 Sag »Hallo«, Android!

Üblicherweise ist das erste Programm, das Sie in Lehrbüchern kennenlernen, eines, das den Text »Hallo, Welt« auf den Bildschirm schreibt. Mit etwas Glück lässt es Sie die Mehrwertsteuer berechnen oder D-Mark in Gulden umrechnen.

Aus Sicht eines Smartphones kommt das einer tödlichen Beleidigung ziemlich nahe, finden Sie nicht? Daher werden wir eine standesgemäße App vorziehen. Als kleine Vorbereitung, die schon ziemlich viel verrät, öffnen Sie bitte den Android Market und suchen nach »Text to Speech«. Installieren Sie die App, falls Sie sie noch nicht auf Ihrem Gerät haben. Sie stellt, wie Sie unschwer erraten können, Sprachausgabe-Funktionen bereit. Auf dem Emulator funktioniert das leider nicht ohne Weiteres, halten Sie daher Ihr Handy samt USB-Kabel bereit.

Jede App entspricht im Arbeitsbereich von Eclipse einem **Project**. Als ersten Schritt legen Sie ein neues Projekt an. Dank des installierten ADT gibt es die Möglichkeit, gleich ein Android-Projekt mit den nötigen Voreinstellungen anzulegen. Wählen Sie im Eclipse-Menü NEW • OTHER, oder drücken Sie Strg + N.

Ein neues Android-Projekt erstellen

Zum Anlegen von neuen Projekten oder Dateien gibt es Wizards. Wir benötigen den Wizard mit dem Namen ANDROID PROJECT. Es genügt, wenn Sie die ersten drei Buchstaben in das obere Textfeld eintippen: Daraufhin erscheinen darunter nur noch die Wizards mit dazu passenden Namen. Einer davon ist der gesuchte Wizard. Doppelklicken Sie darauf, um ihn zu starten (Abbildung 4.1).

Geben Sie dem neuen Projekt den Namen SAGHALLO. Wählen Sie in der Liste der Build Targets ANDROID 2.2 aus. Sie erinnern sich sicher, dass Sie diese SDK-Version zuvor mit dem SDK and AVD Manager installiert haben.

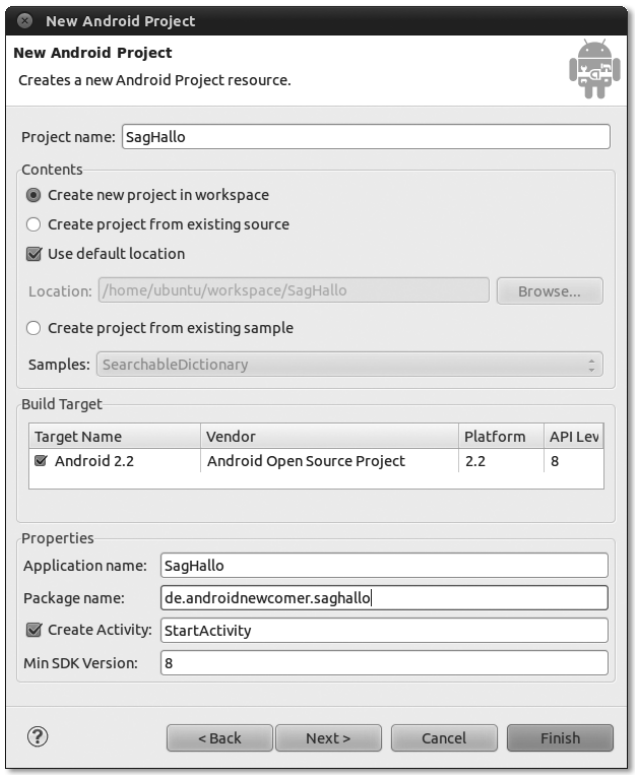

Abbildung 4.1 Der Wizard namens »New Android Project« erzeugt alle Dateien, die wir brauchen.

Auch als Application Name tragen Sie SagHallo ein. Der Package Name bleibt Ihnen überlassen, ich wähle in allen Beispielen `de.androidnewcomer` und hänge den Projektnamen in Kleinbuchstaben an, in diesem Fall heißt mein Package also: `de.androidnewcomer.saghallo`.

Lassen Sie den Wizard auch gleich eine Activity erzeugen. Activities sind Klassen, die jeweils App-Bildschirme verwalten, und ohne macht eine App nicht viel her. Nennen Sie die Activity `StartActivity`.

Es ist eine Konvention, den Namen jeder `Activity`-Klasse mit diesem Begriff enden zu lassen. Sie werden noch sehen, dass Activities entscheidende Bestandteile von Android-Apps sind, daher sollten die Klassen auf den ersten Blick als Activities erkennbar sein.

Tragen Sie zum Schluss als Min SDK Version eine »8« ein. Diese Versionsnummer entspricht üblicherweise der Spalte API Level des ausgewählten Build Targets. Damit bestimmen Sie, dass Ihre App nur auf Geräten ab Android 2.2 läuft.

Falls Sie ein älteres Handy haben sollten, müssen Sie hier die passende Version eingeben. Android 1.6 entspricht beispielsweise API-Level 4. Laden Sie sich mit dem AVD Manager die passende Umgebung herunter, falls nötig. Dann sehen Sie auch die zugehörige API-Level-Version.

Klicken Sie auf FINISH.

Die App, die der Wizard für uns erzeugt hat, kann natürlich noch nicht sprechen. Ihnen bleibt daher nichts anderes übrig, als es ihr beizubringen – indem Sie nun feierlich die ersten eigenen Java-Zeilen hinzufügen.

Die StartActivity

Klappen Sie im Package Explorer das dort entstandene Icon auf, das Ihr Android-Projekt **SagHallo** repräsentiert. Von den vielen Icons, die daraufhin auftauchen, ignorieren Sie zunächst alle bis auf das, neben dem SRC (Abkürzung für **Source Code,** also Quellcode) steht. In diesem Verzeichnis befinden sich alle Java-Dateien des Projekts. Der Wizard hat für den Anfang genau eine erzeugt, sie heißt *StartActivity.java* und befindet sich im Package `de.androidnewcomer.saghallo`, genau wie gewünscht.

Öffnen Sie das Package, und doppelklicken Sie auf *StartActivity.java*. Daraufhin zeigt Eclipse Ihnen im großen Fenster den vom Wizard erzeugten Java-Code (Abbildung 4.2).

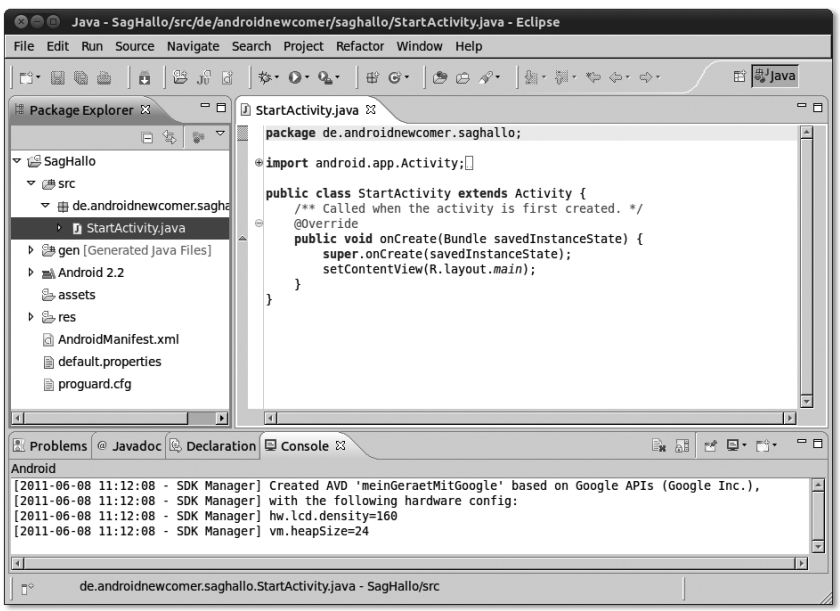

Abbildung 4.2 Der Wizard hat genau eine Java-Datei erzeugt.

Sie sehen, dass die Klasse `StartActivity` von einer Klasse namens `Activity` erbt und genau eine Methode enthält: `onCreate()`. Die Annotation `@Override` zeigt Ihnen, dass `onCreate()` offenbar eine gleichnamige Methode in der Elternklasse `Activity` überschreibt.

Wie der ebenfalls automatisch erzeugte Quellcodekommentar in freundlichem Blau erläutert, wird diese Methode beim ersten Start automatisch aufgerufen. Darum kümmert sich das Android-System ohne unser Zutun, nachdem es ein Objekt der Klasse erzeugt hat.

Kommentare

Es gibt zwei Möglichkeiten, Kommentare im Programmcode zu kennzeichnen, damit der Java-Compiler nicht versucht, den Text zu übersetzen.

Die eine Schreibweise schließt den Kommentar in Schrägstriche und Sterne ein:

```
/* Dies ist ein Kommentar.
   Er kann sich über mehrere Zeilen erstrecken. */
```

Wie Sie sehen, können Sie auf diese Weise längere Kommentare schreiben. Oft verwenden Programmierer diese Notation, um kurze Codeschnipsel *auszukommentieren*.

```
if( bedingung1  /* && bedingung2 */ ) {
```

Diese Strategie dient dazu, auszuprobieren, wie sich ein Programm ohne einen bestimmten Teil verhält.

Die zweite Möglichkeit, Kommentare im Programmcode zu kennzeichnen, ist der doppelte Schrägstrich `//`.

```
Auto cabrio = new Auto(); // erzeugt mein neues Auto
```

Der Java-Compiler ignoriert alles, was hinter dem `//` steht, bis zum Ende der Zeile. Folglich können Sie damit keine einzelnen Elemente innerhalb einer Zeile auskommentieren, ebenso erfordert jede weitere Kommentarzeile einen weiteren einleitenden `//`.

Der Doppelschrägstrich wird gern am Zeilenanfang verwendet, um eine ganze Zeile auszukommentieren. Eclipse hilft Ihnen sogar dabei: Wenn Sie mehrere Zeilen markieren und Strg + ⇧ + 7 drücken (also sozusagen Strg + /), werden alle Zeilen mit `//` auskommentiert oder, wenn sie schon auskommentiert sind, wieder einkommentiert (d. h. die `//` entfernt). Natürlich klappt das auch über das Menü: Wählen Sie Source • Toggle Comment.

Eclipse hebt Kommentare zwar farblich hervor, aber allzu viele kleine Kommentare verbessern nicht gerade die Übersicht.

Die Faustregel für Kommentare lautet: Schreiben Sie welche, wenn Sie es für möglich halten, dass Sie oder andere Programmierer eine Stelle sonst nicht auf Anhieb verstehen. Und überschätzen Sie dabei niemanden …

Derzeit erledigt die `onCreate()`-Methode zwei Dinge: Erstens ruft sie die gleichnamige Methode der Elternklasse auf. Damit sie sich nicht selbst aufruft, steht

`super` davor. Selbstverständlich besitzt auch die Elternklasse eine Methode namens `onCreate()`, und sie erledigt wichtige organisatorische Aufgaben. Deshalb muss sie unbedingt aufgerufen werden.

Die zweite Codezeile in der Methode `onCreate()` ruft die Methode `setContentView()` auf. Da diese Methode offensichtlich nicht in `StartActivity` zu finden ist, können Sie davon ausgehen, dass sie in einer Elternklasse definiert ist. Die Methode erhält als Parameter einen Wert, über den später noch zu sprechen sein wird. Für den Moment genügt es, zu wissen, dass diese Zeile dafür sorgt, dass ein anderswo unter dem Namen `main` definierter Bildschirminhalt angezeigt wird.

Zur Erinnerung: Private Methoden

In einer Methode können Sie alle Methoden derselben Klasse aufrufen, außerdem alle Methoden der Elternklasse, die nicht mit dem Modifizierer `private` Ihren Blicken entzogen sind:

```
class Elternklasse {
   private void eineMethode() {
      ...
   }
}
...
class Kindklasse extends Elternklasse {
   public void testMethode() {
      eineMethode(); // Fehler
   }
}
```

Die Programmierer der Klasse `Activity` stellen Ihnen eine Menge hilfreicher Methoden zur Verfügung, die Sie in eigenen von `Activity` abgeleiteten Klassen verwenden können.

Lassen Sie uns nun die `StartActivity` um die gewünschte Sprachausgabe erweitern. Das wird dann dazu führen, dass die App beim ersten Start zu uns spricht.

Fügen Sie zunächst der Klasse ein Attribut hinzu:

```
private TextToSpeech tts;
```

Diese Eingabe quittiert Eclipse mit einer roten Unterstreichung, weil `TextToSpeech` unbekannt ist. Rote Unterstreichung heißt: Syntaxfehler, der Compiler kann dies nicht übersetzen, folglich kann er kein lauffähiges Programm erzeugen. Also müssen Sie `TextToSpeech` importieren.

Die gesuchte Klasse befindet sich in einem Paket des Android SDK, daher müssen Sie sie importieren. Glücklicherweise nimmt Ihnen Eclipse diese Arbeit ab, weil es im Gegensatz zu Ihnen schnell nachschauen kann, welches das zu importie-

Index

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z