Android Studio – Googles alternative Entwicklungsumgebung

Testballon

Mark Zimmermann, Danny Fürniß

Mit Android Studio stellt Google eine alternative IDE zu den bislang alleinherrschenden Android Development Tools für Eclipse vor. Ein Blick auf die Betaversion.

Anwendungen für Android werden in Java entwickelt, und es gab dafür lange Zeit genau eine Entwicklungsumgebung: das ADT-Plug-in (Android Development Tools) für Eclipse. Nun hat Google auf der Entwicklerkonferenz I/O 2013 eine – ebenfalls frei verfügbare – Alternative vorgestellt: Android Studio, basierend auf der kostenfreien Community Edition von IntelliJ IDEA und durch ein entsprechendes Plug-in um Android-Features ergänzt.

Android Studio (AS) verfügt neben den von IntelliJ bereits bekannten Android-spezifischen Refaktorisierungs- und Codegenerierungs-Funktionen über zusätzliche Features. Dazu zählen die Unterstützung von Java EE, Ant, JUnit, ein grafischer GUI-Editor, Werkzeuge zur Versionskontrolle sowie verschiedene Möglichkeiten zum automatischen Erstellen von Code.

IntelliJ gilt unter Entwicklern als schnell und komfortabel. Unter anderem ernten die Code-Vervollständigung, die auch eigene Klassen integriert, und die Möglichkeit, viele Aktionen statt durch Mausklicks via Tastenkombination aufzurufen, viel Lob.

Zum Android Studio gehört die jeweils aktuelle Android-Version sowie eine Toolbox, bestehend aus Android SDK Tools, Platform Tools und Build Tools. Da man in der Regel für mehr als eine Android-Version entwickelt, lassen sich über den Android-immanenten SDK-Manager weitere Versionen nachinstallieren. Das knapp 400 MByte große Paket, herunterzuladen auf Googles Git-Plattform (die URL sowie weitere Onlineverweise sind unter „Alle Links“ zu finden), entspricht in der Größe etwa dem Eclipse ADT. Aktualisiert wird es nach der Installation automatisch. Derzeit existiert wegen des Vorschau-Status noch kein Stable Build für den produktiven Einsatz.

Kick-off: Die Projekte

Zur Installation gibt es ausführliche Hinweise online. Nach dem Start besteht neben der Option, ein neues Projekt anzulegen, die Möglichkeit, ein vorhandenes Eclipse-Projekt oder eines aus einer Sourcecode-Verwaltung zu importieren. Wählt man beim Anlegen eine ältere Android-Version als 4.0 (Ice Cream Sandwich) aus, lassen sich im „Support Mode“ Bausteine wie Fragments oder NavigationDrawer für die älteren Plattformen aktivieren.

Einen definierten Workspace wie bei Eclipse gibt es nicht. Während in Eclipse alle Projekte im Workspace lagen, können Projekte und Unterprojekte („Module“) nun in jedem Verzeichnis liegen und mehrere Module enthalten. Ein Doppelklick auf eine Java- oder XML-Datei im Projektbaum aktiviert entweder den Java-Editor oder alternativ den XML-Editor beziehungsweise GUI-Designer von Android Studio.

Über das sogenannte Tool-Window „Project“ lässt sich die Baumstruktur des Projektverzeichnisses einblenden. Im Unterschied zu einem Eclipse/ADT-Projekt werden alle relevanten Inhalte unterhalb eines SRC-Knotens abgelegt. Aufgrund der neuen Build-Umgebung (Gradle) unterscheidet sich auch die Verzeichnisstruktur von der bei Eclipse. Im Verzeichnis *src/main* befinden sich sowohl der Quellcode der App als auch die Ressourcen, Letztere in entsprechenden Unterverzeichnissen nach dem Schema der sogenannten Resource Qualifiers, also der Zuordnung der Ressourcen zu unter-

Eclipse-Projekte importieren

Um ein Projekt aus Eclipse in Android Studio zu konvertieren, muss der Entwickler es in Eclipse über „File export – Generate Gradle build files“ auswählen. Diese Funktion wird seit dem ADT PlugIn Version 22.0 geboten. Nach dem Start von Android Studio lässt sich im Hauptfenster über das Menü „File“ diese exportierte Datei „build. gradle“ auswählen und importieren.

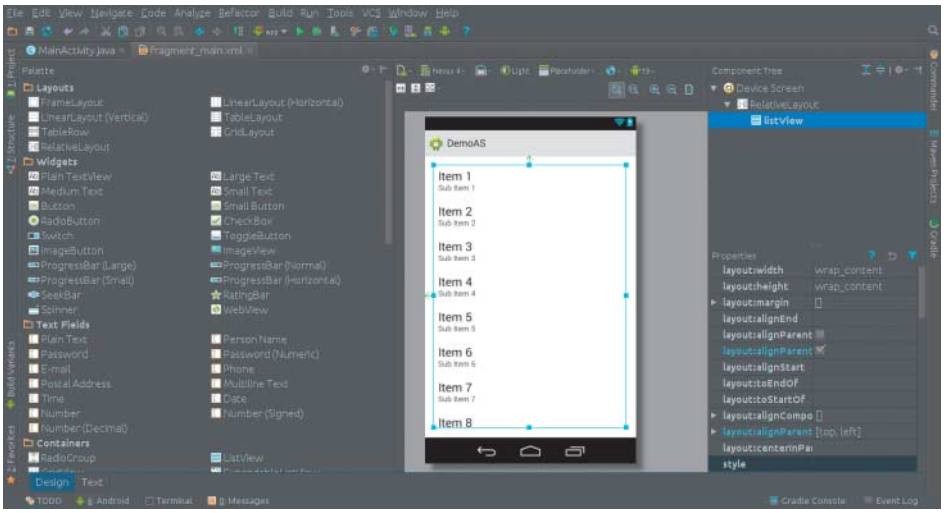
schiedlichen Displaygrößen, Sprachen, Plattformversionen et cetera.

Die Entwickler bei Google haben Android Studio mit verschiedenen Funktionen versehen, die den Programmieralltag erleichtern. So kann der Code-Editor Annotationen im Android-Quellcode auswerten oder beispielsweise feststellen, ob eine Methode des Android-Frameworks einen Nullwert zurückliefern kann – das sogar nebst Überprüfung, ob im Code dadurch die Gefahr einer NullPointer-Exception besteht. Solche Gefahrenstellen hinterlegt der Editor farbig oder versieht sie mit einem Hinweis.

„Resource Folding“ zeigt in einer separaten Spalte im Editorfenster, welche Farbe oder welche Grafik sich hinter einem Ressourcen-Verweis verbirgt. Text-Ressourcen werden direkt im Code angezeigt. Das funktioniert sowohl an der Ressourcen-Definition selbst als auch am Ort ihrer Referenzierung. Beim Umbenennen von Android-Ressourcen kann man wählen, ob der neue Name auch für verwandte Ressourcen in anderen Verzeichnissen gelten soll, etwa dieselbe Grafik mit unterschiedlicher Auflösung.

Zeichenketten (Strings) im Code lässt sich ein sogenanntes „Language Binding“ zuweisen, um beispielsweise einen String als XML-Code zu kennzeichnen, was es dem Code-Editor ermöglicht, den Inhalt des Strings auf XML-Konformität zu überprüfen. Neben XML und weiteren Sprachen sind insbesondere die File-Reference, die Überprüfung der Gültigkeit eines Dateipfads, und das Regular Expression Binding erwähnenswert. Bei Letzterem kann man sogar reguläre Ausdrücke in einem Pop-up-Fenster überprüfen.

Wizards mit erweiterbaren Vorlagen erleichtern das Anlegen neuer Code-Komponenten wie Activities, Fragments, Layout-XML-Dateien und so weiter. Lint-Tools ermöglichen statische Code-Analysen. Wichtig für Ein- oder Umsteiger: Eine übergreifende Suche listet neben den gefundenen Funktionen auch deren Tastaturkürzel auf.



Der GUI-Designer von Android Studio – Designansicht (Abb. 2)

Wie bereits von Eclipse bekannt, erlaubt es das Android Studio, das Layout einer App wahlweise im Texteditor oder in einer grafischen Oberfläche, dem Design-Modus, zu erstellen. Letzterer funktioniert im Android Studio besser als im Eclipse ADT, da bereits in der Vorschau das Layout für die unterschiedlichen Android- und Länder-Versionen angezeigt wird.

GUI-Designer für Android

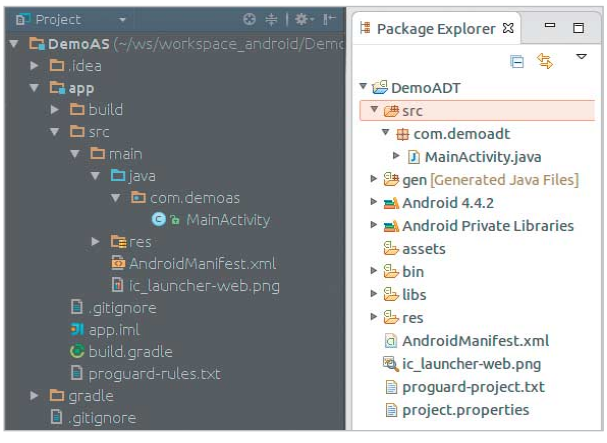
Im Design-Modus findet sich neben der Komponentenpalette ein Fenster, das die komponentenbezogene Struktur einer Activity anzeigt (Component Tree). Eigenschaften einer Komponente, zum Beispiel eines Buttons, lassen sich über ein Properties-Fenster direkt ändern. WYSIWYG, Hilfslinien und die Unterstützung von Drag & Drop unterstützen die genaue Positionierung.

Über eine Registerlasche kann man am unteren Rand auf die jeweilige XML-Codeansicht der geöffneten Activity umschalten. Diese Features geben dem Entwickler eine einfachere Arbeitsweise an die Hand, wie er es gegebenenfalls schon im Rahmen des Interface Builders von xCode für iOS Apps gewohnt ist.

Allerdings muss auch hier einschränkend gesagt werden, dass es sich bei den dargestellten Elementen nicht um eine hundertprozentige Entsprechung zur Darstellung auf dem Endgerät handelt, sondern um eine Annäherung. Genauer kann das die sogenannte Preview-Ansicht. Diese stellt eine relativ genaue Vorschau des Endergebnisses auf verschiedenen Gerätetypen dar. Zusätzlich gibt es die Möglichkeit, verschiedene Gerätetypen gleichzeitig zu betrachten.

Änderungen im XML-Editor sind umgehend innerhalb der Preview-Ansicht sichtbar. Eine große Erleichterung gegenüber dem Eclipse/ADT: Dort muss der Entwickler, um das endgültige Ergebnis begutachten zu können, jedes Mal den passenden Android-Simulator starten.

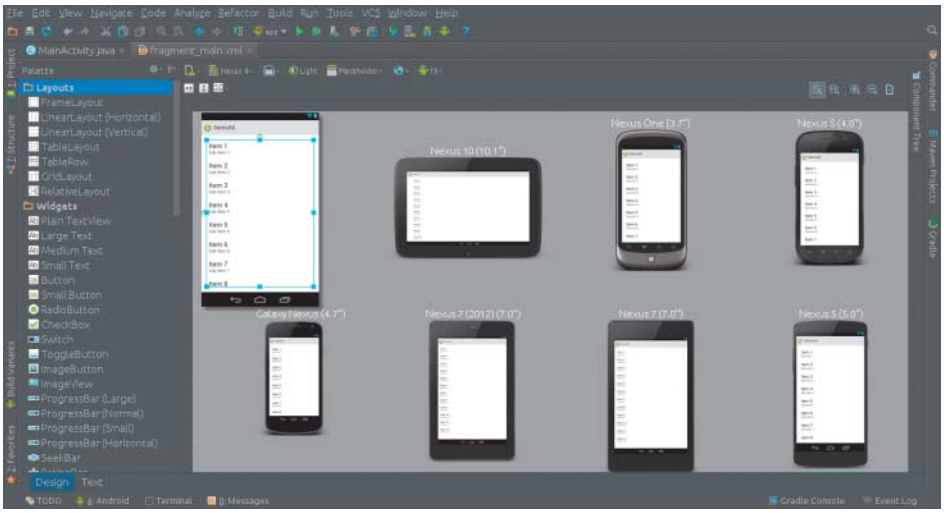
Aus dem Designer heraus lassen sich Screenshots nicht nur speichern, sondern auch mit einem Geräte-Rahmen, Schatten- und Spiegelungseffekt aufhübschen.



Unterschiedliche Projektstruktur: links Android Studio, rechts Eclipse ADT (Abb. 1)

Unter der Haube – ein neues Build-System

Das integrierte neue Build-System auf Gradle-Basis bringt signifikante Veränderungen mit sich. Gradle ist ein auf Groovy basierendes, durch Plug-ins erweiterbares Build-System. Im Gegensatz zu Maven-Projektdefinitionen sind Gradle-Skripte ausführbarer Code. Gradle entstand ursprünglich für den Bau von Softwaresystemen, die aus einer Vielzahl von Modulen bestehen. Basierend auf der Philosophie „Erwarte das Unerwartete“ haben die Gradle-Schöpfer versucht, das in Maven etablierte „Build-by-convention“-Prinzip (eine Variante von „Kon-



Mehrere Zielplattformen auf einen Blick (Abb. 3)

vention vor Konfiguration“) mit der Flexibilität von Ant zu vereinen.

In der Konfigurationsdatei *build.gradle* kommt eine domänenspezifische Projektbeschreibungssprache zum Einsatz:

```
buildscript { repositories { mavenCentral() }
dependencies { classpath 7
               com.android.tools:gradle:0.8+' }
apply plugin:  android'
android {     compileSdkVersion 19 }
```

Im Bereich *buildscript { … }* werden Konfigurationen für den Build selbst gesetzt. Im verwendeten Beispiel ist dies der Zugriff auf das Maven Central Repository sowie das Einbinden des Android-Gradle-Plug-ins. Mit der *apply plugin*-Direktive wird das Plug-in im Build angewendet. Der Bereich *android { … }* schließlich konfiguriert die Parameter für den eigentlichen Bau der Android-Anwendung.

Gradle ermöglicht dem Entwickler das Anlegen unterschiedlicher Konfigurationen, sodass er leicht verschiedene App-Versionen auf Basis desselben Codes produzieren kann, etwa eine kostenlose und eine Bezahlversion. Generell verbessert Gradle die Integration auf einem Build-Server, weil der Gradle-Build bereits auf dem Entwicklungssystem zum Einsatz kommt.

Den Umstieg auf Gradle erleichtert der sogenannte Gradle Wrapper, ein generiertes Skript im Projektverzeichnis, das automatisch für den Download und die Installation von Gradle sorgt.

Integrierte Versionsverwaltung

Um dem Entwickler die Arbeit mit der Versionsverwaltung möglichst effizient zu gestalten, integriert IntelliJ eine Reihe verschiedener Systeme. Android Studio kann dadurch sowie durch eigene Erweiterungen beispielsweise bearbeitete Dateien farblich kennzeichnen. Wird Android Studio im „Dracula“-Theme verwendet, stellt es bearbeitete Dateien blau dar. Neu hinzugefügte sind rot, sofern sie noch nicht über den „Add“- Befehl in die Versionsverwaltung aufgenommen wurden, danach grün. Ein „Commit“ lässt alle betroffenen Dateien wieder in der Standardfarbe grau erscheinen.

Die Versionsverwaltung bietet unterschiedliche Ansichten auf lokale und entfernte Änderungen. Über die Funktion „Cherry Pick“ lassen sich einzelne Änderungen entfernter Commits übernehmen, was das Zusammenführen verschiedener Quellen erleichtert. Mit GIT als Versionsverwaltung kann man eine Datei via Push in entfernte Repositories einspielen. Unterstützt wird zusätzlich die Arbeit mit Branches.

Google hat in das Android Studio eine Reihe von neuen Diensten und Services integriert. Die Developer Console gibt dem Entwickler Tipps zur Optimierung seiner App und hilft durch den „App Translation Service“, Anwendungen leichter zu lokalisieren, der Entwickler kann seine zu übersetzenden Strings auf eine zentrale Plattform hochladen und nach der Übersetzung in die Zielsprache wieder in die IDE einfügen (dieser Dienst ist allerdings kostenpflichtig).

Google Cloud Messaging (GCM) erlaubt es, ein Cloud-basiertes Backend zum Nachrichtenaustausch zwischen einem Server und Endgeräten in die App zu integrieren. Die Benutzung von GCM ist für Entwickler kostenlos und mengenmäßig nicht limitiert.

Zu Android Studio gehört auch die Integration der App Engine zum Generieren von Server-Projekten. Damit lassen sich Endpunkte generieren, das fertige Projekt hochladen, eine Testseite erzeugen oder auch eine Client-Bibliothek generieren.

Fazit

Zurzeit liegt die IDE noch in einer sehr frühen Version vor, das bedeutet, dass sich noch viel ändern kann und wird. Entwickler sollten berücksichtigen, dass zum Wissensaufbau notwendige Internetquellen und Dokumentationen sich im Moment noch primär mit Eclipse/ADT beschäftigen. Auch Google selbst weist darauf hin, dass es sich um eine frühe Vorschau handelt, die noch nicht für produktives Arbeiten geeignet ist.

Ohnehin gibt es keinen Grund zur Eile, da Google das Eclipse-Plug-in ADT weiterhin unterstützen will. Interessant ist auf jeden Fall auch das andere Build-System – wer sich bisher nur mit Ant beschäftigt hat, kann durch Android Studio jetzt Gradle kennen lernen. Das könnte sich langfristig lohnen, da Gradle sehr flexibel ist und auch eine Verwaltung von Abhängigkeiten zum Beispiel über Maven mitbringt.

Darum empfehlen die Autoren: die neue IDE mal ausprobieren und ihre Entwicklung zumindest mitverfolgen. (js)

Danny Fürniß

ist bei der EnBW Energie Baden-Württemberg AG in Karlsruhe als Softwarearchitekt tätig und beschäftigt sich intensiv mit der Android-Plattform. Als Community Manager der Google Developer Group Karlsruhe organisiert er außerdem ehrenamtlich Developer Events zum Thema Google und Android.

Mark Zimmermann

ist bei der EnBW Energie Baden-Württemberg AG in Karlsruhe als Teamleiter tätig. Das Aufgabenfeld in dem von ihm verantworteten Team umfasst unter anderem die Entwicklung von mobilen Applikationen für den internen Einsatz beziehungsweise für Endkunden der EnBW, in Abstimmung mit dem zentralen Bereich der IT-Sicherheit.

Literatur

[1] Peter Falck; Jetzt starten mit Android Studio (Mobile Entwicklung); Verlag Barbara Hohensee; Januar 2014

Alle Links: www.ix.de/ix1404088