Andreas Linke

Ein Code für alle

Apps für verschiedene Android-Versionen programmieren

Wer seine App für mehrere Android-Versionen gleichzeitig fit machen und trotzdem nicht auf die neuesten Funktionen verzichten möchte, muss ein paar Kniffe anwenden. Im Kern geht es darum, neue API-Klassen einzusetzen, ohne ältere Android-Versionen aus dem Tritt zu bringen.

Während Apple seine iOS-Updates regelmäßig auch Geräten der vorletzten Generation zur Verfügung stellt und so für eine recht einheitliche Versionsverbreitung sorgt, erhalten Android-Anwender nur selten Updates von ihrem Hersteller. Die Folge ist ein bunter Zoo von Android-Versionen im Markt. Wer als App-Entwickler möglichst viele Geräte unterstützen möchte, ohne sich auf den kleinsten gemeinsamen Nenner beim Funktionsumfang zurückzuziehen, kommt daher nicht umhin, über kompatible Programmierung nachzudenken.

Um neue API-Funktionen nutzen zu können, muss man zunächst in den Projekteigenschaften die entsprechende Android-Version, zum Beispiel Android 2.2, als Build-Target auswählen. Damit die App jedoch auch auf älteren Geräten installiert werden kann, setzt man im AndroidManifest unter „Manifest Extras“ bei „Uses SDK“ die „Min SDK Version“ auf die kleinste unterstützte Version.

Im Code

Die aktuell laufende Version steht in der Konstanten Build.VERSION.SDK_INT. Die Abfrage

```
if (Build.VERSION.SDK_INT >=
    Build.VERSION_CODES.FROYO)
  // ...
```

prüft beispielsweise, ob auf dem Gerät mindestens Android 2.2 (Codename Froyo, siehe Tabelle) läuft.

Leider kann man nun nicht einfach innerhalb der if-Bedingung die Klassen und Methoden der höheren Android-Version verwenden. Wer das versucht, erhält einen java.lang.VerifyError, weil der Class-Loader beim Laden der Klasse den gesamten Bytecode überprüft und alle ihm unbekannten Konstruktoren und Funktionsaufrufe bemängelt. Abhilfe schafft das Verpacken der gewünschten Funktionalität in eine gesonderte Klasse, die nur instantiiert wird, wenn die richtige Android-Version vorhanden ist. Java spezifiziert nämlich ein sogenanntes „lazy class loading“. Das heißt: Klassen werden erst dann geladen (und verifiziert), wenn sie tatsächlich verwendet (instantiiert) werden. Innerhalb der Klasse lässt sich beliebig auf neue Android-Klassen und Methoden zugreifen. Die Kapsel-Klasse darf auch als innere Klasse in der verwendenden Klasse definiert werden.

Das Folgende demonstriert die Technik am Beispiel der Apfelmännchen-App aus [1]. Wer diesen Artikel oder einen der Vorgänger aus der Android-Entwicklungsserie verpasst hat, findet alle zusammen im aktuellen Sonderheft „c’t kompakt 02/2011 Programmieren“. Den Code zum Beispielprojekt bekommen Sie wie gewohnt über den c’t-Link am Artikelende.

Android-Versionen

Version	Codename	API Level	Verbreitung
3.0	Honeycomb	11	0,20%
2.3	Gingerbread	9	2,50%
2.2	Froyo	8	63,90%
2.0/2.1	Eclair	5	27,20%
1.6	Donut	4	3,50%
1.5	Cupcake	3	2,70%

Android 2.2 bringt unter anderem erstmals Unterstützung für Multi-Touch-Gesten wie beispielsweise die bekannte Zwei-Finger-Zoom-Geste mit. Die Funktionen zum Auswerten der Geste stecken in der Android-Klasse ScaleGestureDetector, die in onTouchEvent() über alle Fingerberührungen und Bewegungen informiert wird. Über einen ScaleListener erfährt die App, ob eine Zoom-Geste erkannt wurde:

```
scaleGestureDetector =
  new ScaleGestureDetector(context,
    new ScaleGestureDetector
      .SimpleOnScaleGestureListener() {
      @Override
      public void onScaleEnd(
          ScaleGestureDetector detector) {
        // Zoom-Geste beendet, Fraktal skalieren
        startPoint.x = detector.getFocusX();
        startPoint.y = detector.getFocusY();
        Log.v("Apfel4", "Zoom by "
          + detector.getScaleFactor());
        zoom(detector.getScaleFactor());
      }
    });
```

Der ScaleGestureDetector landet in der neuen Klasse ExtensionsFroyo, die nur dann instanziiert wird, falls mindestens Android 2.2 läuft:

```
ExtensionsFroyo extensionsFroyo = null;
if (Build.VERSION.SDK_INT >=
    Build.VERSION_CODES.FROYO)
  extensionsFroyo = new ExtensionsFroyo(context);
```

Wenn extensionsFroyo gesetzt ist, steht die Zoom-Geste zur Verfügung, sonst läuft die App mit einer älteren Android-Version und Zoom-Gesten werden ignoriert:

```
if (extensionsFroyo != null) {
  // Android 2.2 oder neuer
  extensionsFroyo.onTouchEvent(event);
}
```

Leider bietet der Android-Emulator keine Möglichkeit, Gesten mit mehreren Fingern zu simulieren. Zum Testen der Zoom-Geste ist man daher auf ein Gerät mit mindestens Android 2.2 angewiesen.

Will man auf die beschriebene Weise kompatibel programmieren, ist es neben Tests des Programms auf älteren Android-Versionen wichtig, regelmäßig zu prüfen, dass sich nicht irgendwo Code eingeschlichen hat, der nur in neueren Android-Versionen verfügbar ist. Das API Analysis Plug-In für Eclipse 3.6 (siehe c’t-Link) hilft bei der Analyse des Codes und markiert alle nicht in der „Min SDK Version“ vorhandenen Klassen und Funktionsaufrufe mit Warnings. (ola)

Literatur

[1] Andreas Linke, Appfelmännchen, Grafiken, Threads und C-Programme auf der Android-Plattform, c’t 5/11, S. 188

www.ct.de/1112187

Problems | Javadoc | Declaration

0 errors, 10 warnings, 0 others

Description	Resource	Path	Location
Warnings (10 items)			
API Level 4 compatibily problem (FROYO cannot be resolved or is not a field)	ApfelView.java	/Apfel4/src/co...	line 79
API Level 4 compatibily problem (ScaleGestureDetector cannot be resolved to a type)	ApfelView.java	/Apfel4/src/co...	line 198
API Level 4 compatibily problem (ScaleGestureDetector cannot be resolved to a type)	ApfelView.java	/Apfel4/src/co...	line 201
API Level 4 compatibily problem (ScaleGestureDetector cannot be resolved to a type)	ApfelView.java	/Apfel4/src/co...	line 201
API Level 4 compatibily problem (ScaleGestureDetector cannot be resolved to a type)	ApfelView.java	/Apfel4/src/co...	line 202
API Level 4 compatibily problem (ScaleGestureDetector cannot be resolved to a type)	ApfelView.java	/Apfel4/src/co...	line 204
API Level 4 compatibily problem (ScaleGestureDetector cannot be resolved to a type)	ApfelView.java	/Apfel4/src/co...	line 215
API Level 4 compatibily problem (The import android.view.ScaleGestureDetector cannot be resolved)	ApfelView.java	/Apfel4/src/co...	line 20
API Level 4 compatibily problem (The method getPointerCount() is undefined for the type MotionEvent)	ApfelView.java	/Apfel4/src/co...	line 219

Das API Analysis Plug-In für Eclipse 3.6 hilft beim Aufspüren von Funktionsaufrufen, die nicht mit der unter Min SDK Version eingestellten Android-Version kompatibel sind. c't