

Publishing

Seminar „Android: Plattform für mobile Geräte“

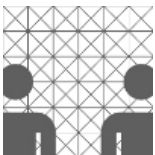
Von Artiom Wulis

SS2010



Universität Hamburg

MIN Faculty
Department of Informatics
Scientific Computing



Index

Publishing Voraussetzungen

- Publishing Checkliste
 - Versionierung
 - Signierung

Publishing auf:

- Android Market
- Endgerät

Index

Publishing Voraussetzungen

- Publishing Checkliste
 - Versionierung
 - Signierung

Publishing auf:

- Android Market
- Endgerät

Publishing Voraussetzungen

Publishing Checkliste:

1. Anwendung ausgiebig Testen
2. Endbenutzer-Lizenzvereinbarung hinzufügen
3. Icon und Label spezifizieren
4. Logger und Debugger ausschalten,
temp. Dateien löschen
5. Anwendung versionieren
6. Anwendung signieren

Publishing Checkliste

1. Anwendung ausgiebig Testen

- Auf physischen Geräten testen

Publishing Checkliste

1. Anwendung ausgiebig Testen

- Auf physischen Geräten testen
- Seltene Geräte können weitestgehend emuliert werden
(Emulator Optionen: -dpi, -scale, -netspeed, -cpu-delay...)

Publishing Checkliste

2. Endbenutzer-Lizenzvereinbarung hinzufügen

- In google nach: ***android eula.java*** suchen
- Oder *http://code.google.com/p/apps-for-android*

Source path:

*svn/trunk/Photostream/src/com/google/android/photo
stream/Eula.java*

Publishing Checkliste

3. Icon und Label spezifizieren

- *android:icon="@drawable/icon"*
- *android:label="@string/app_name"*

Publishing Checkliste

3. Icon und Label spezifizieren

- *android:icon="@drawable/icon"*
- *android:label="@string/app_name"*
- In Android Manifest Application (*AndroidManifest.xml*)

```
<application
    android:icon="@drawable/icon"
    android:label="@string/app_name" >
</application>
```

Publishing Checkliste

4. Logger und Debugger ausschalten, temp. Dateien löschen

- Evtl. Temp. Backup und Logdateien entfernen
- Alle Aufrufe zu den Log-Methoden löschen
- Debugger ausschalten: *android:debuggable="false"*
- In Android Manifest Application (AndroidManifest.xml)

```
<application
    android:debuggable="false" >
</application>
```

Index

Publishing Voraussetzungen

- Publishing Checkliste
 - Versionierung
 - Signierung

Publishing auf:

- Android Market
- Endgerät

Publishing Checkliste

5. Anwendung versionieren

Warum versionieren?

Publishing Checkliste

5. Anwendung versionieren

Warum versionieren?

- Benutzer müssen Programmversion sehen können

Publishing Checkliste

5. Anwendung versionieren

Warum versionieren?

- Benutzer müssen Programmversion sehen können
- Andere Anwendungen brauchen Versionsangaben

Publishing Checkliste

5. Anwendung versionieren

Warum versionieren?

- Benutzer müssen Programmversion sehen können
- Andere Anwendungen brauchen Versionsangaben
- Android Market, ähnliche Seiten und „Update Notifier“ benötigen auch Versionsangaben

Publishing Checkliste

5. Anwendung versionieren

Warum versionieren?

- Benutzer müssen Programmversion sehen können
- Andere Anwendungen brauchen Versionsangaben
- Android Market, ähnliche Seiten und „Update Notifier“ benötigen auch Versionsangaben

Achtung!

Android System prüft nicht die Versionsangaben.

Publishing Checkliste

5. Anwendung versionieren

In Android Manifest (*AndroidManifest.xml*)

- *android:versionCode*
- *android:versionName*

Publishing Checkliste

5. Anwendung versionieren

android:versionCode

- Integer
- Ist für andere Anwendungen / Systeme evaluierbar
- Keine Unterscheidung zwischen kleineren / größeren Updates
- Sollte für Anwender nicht sichtbar sein
- Sollte nach jedem Release erhöht werden

Publishing Checkliste

5. Anwendung versionieren

android:versionName

- String
- Ist für andere Anwendungen / Systeme nicht evaluierbar
- Unterscheidung zwischen kleineren / größeren Updates möglich (z.B. 1.21 / 3a usw.)
- Sollte für Anwender sichtbar sein
- Sollte nach jedem Release erhöht werden

Publishing Checklist

5. Anwendung versionieren

In Android Manifest (*AndroidManifest.xml*)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.package.name"
    android:versionCode="2"
    android:versionName="1.1">
    <application android:icon="@drawable/icon"
    android:label="@string/app_name">
        ...
    </application>
</manifest>
```

Publishing Checkliste

5. Anwendung versionieren

Android Sdk Version festlegen

- *android:minSdkVersion*
- *android:targetSdkVersion*
- *android:maxSdkVersion*

Publishing Checkliste

5. Anwendung versionieren

Android Sdk Version festlegen

- *android:minSdkVersion*
- *android:targetSdkVersion*
- *android:maxSdkVersion*

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android=http://schemas.android.com/apk/res/android
...
  <application ...</application>
  <uses-sdk android:minSdkVersion="3"
android:targetSdkVersion="7"/>>
</manifest>
```

Android Manifest Eclipse Gui

The screenshot shows the Eclipse IDE's AndroidManifest editor. At the top, there's a title bar with the Android logo and the text 'Android Manifest'. Below this, a section titled 'Manifest General Attributes' contains several input fields: 'Package' (com.example.helloandroid), 'Version code' (1), 'Version name' (1.0), 'Shared user id', and 'Shared user label'. Each field has a 'Browse...' button. Below this is the 'Manifest Extras' section, which contains a list of extras. One extra, 'Uses Sdk', is selected. To the right of this list are buttons for 'Add...', 'Remove...', 'Up', and 'Down'. Further right is the 'Attributes for Uses Sdk' section, which includes a description of the 'uses-sdk' tag and three input fields: 'Min SDK version' (3), 'Target SDK version' (7), and 'Max SDK version'. Below these sections are three expandable sections: 'Exporting', 'Links', and 'Exporting'. The 'Exporting' section provides instructions on how to export the application. The 'Links' section provides links to the Application, Permission, and Instrumentation sections of the manifest. At the bottom, there is a breadcrumb trail: 'Manifest | Application | Permissions | Instrumentation | AndroidManifest.xml'.

Android Manifest

Manifest General Attributes
Defines general information about the AndroidManifest.xml

Package: com.example.helloandroid [Browse...]
Version code: 1
Version name: 1.0 [Browse...]
Shared user id: [Browse...]
Shared user label: [Browse...]

Manifest Extras: [U] [S] [P] [U] [U] [Az]

Uses Sdk [U] [Add...] [Remove...] [Up] [Down]

Attributes for Uses Sdk
The uses-sdk tag describes the SDK features that the containing package must be running on to operate correctly.

Min SDK version: 3 [Browse...]
Target SDK version: 7 [Browse...]
Max SDK version: [Browse...]

Exporting
To export the application for distribution, you have the following options:

- Use the [Export Wizard](#) to export and sign an APK
- [Export an unsigned APK](#) and sign it manually

Links
The content of the Android Manifest is made up of three sections. You can also edit the XML directly.

- [Application](#): Activities, intent filters, providers, services and receivers.
- [Permission](#): Permissions defined and permissions used.
- [Instrumentation](#): Instrumentation defined.
- [XML Source](#): Directly edit the AndroidManifest.xml file.
- [Documentation](#): Documentation from the Android SDK for AndroidManifest.xml.

Manifest | Application | Permissions | Instrumentation | AndroidManifest.xml

Android Manifest Eclipse Gui

The screenshot shows the Eclipse IDE's Android Manifest editor. At the top, it says "Android Manifest Application". Below that is the "Application Toggle" section, which includes a checkbox to "Define an <application> tag in the AndroidManifest.xml" (checked). The "Application Attributes" section is expanded, showing a list of attributes with input fields and dropdown menus. The attributes include Name, Theme, Label, Icon, Description, Permission, Process, Task affinity, Allow task reparenting, and Has code. Each attribute has a "Browse..." button. The "Application Nodes" section at the bottom shows a list of nodes, currently containing ".HelloAndroid (Activity)". To the right of the list are buttons for "Add...", "Remove...", "Up", and "Down". At the bottom of the window, there are tabs for "Manifest", "Application", "Permissions", "Instrumentation", and "AndroidManifest.xml".

Android Manifest Application

▼ Application Toggle

The [application](#) tag describes application-level components contained in the package, as well as general application attributes.

Define an <application> tag in the AndroidManifest.xml

▼ Application Attributes

Defines the attributes specific to the application.

Name	<input type="text"/>	Browse...	Persistent	<input type="text"/>
Theme	<input type="text"/>	Browse...	Enabled	<input type="text"/>
Label	@string/app_name	Browse...	Debuggable	false
Icon	@drawable/icon	Browse...	Manage space activity	<input type="text"/>
Description	<input type="text"/>	Browse...	Allow clear user data	<input type="text"/>
Permission	<input type="text"/>		Test only	<input type="text"/>
Process	<input type="text"/>	Browse...	Backup agent	<input type="text"/>
Task affinity	<input type="text"/>	Browse...	Allow backup	<input type="text"/>
Allow task reparenting	<input type="text"/>		Kill after restore	<input type="text"/>
Has code	<input type="text"/>		Restore needs application	<input type="text"/>

Application Nodes

S P A A R M U Az

[A] .HelloAndroid (Activity)

Add...

Remove...

Up

Down

Manifest Application Permissions Instrumentation AndroidManifest.xml

Index

Publishing Voraussetzungen

- Publishing Checkliste
 - Versionierung
 - Signierung

Publishing auf:

- Android Market
- Endgerät

Publishing Checkliste

6. Anwendung signieren

Android Anwendungen müssen signiert werden!

Publishing Checkliste

6. Anwendung signieren

Android Anwendungen müssen signiert werden!

- Sicherheit

Publishing Checkliste

6. Anwendung signieren

Android Anwendungen müssen signiert werden!

- Sicherheit
- Eigenes Zertifikat kann verwendet werden

Publishing Checkliste

6. Anwendung signieren

Android Anwendungen müssen signiert werden!

- Sicherheit
- Eigenes Zertifikat kann verwendet werden
- Android System prüft Ablaufdatum des Zertifikats

Publishing Checkliste

6. Anwendung signieren

Android Anwendungen müssen signiert werden!

- Sicherheit
- Eigenes Zertifikat kann verwendet werden
- Android System prüft Ablaufdatum des Zertifikats
- Standardtools wie Keytool und Jarsigner können benutzt werden (Java SDK)

Publishing Checkliste

6. Anwendung signieren

Strategien

Empfehlung: Alle Anwendungen mit dem selben Zertifikat signieren.

- Update / Upgrade

Publishing Checkliste

6. Anwendung signieren

Strategien

Empfehlung: Alle Anwendungen mit dem selben Zertifikat signieren.

- Update / Upgrade
- Modularität (ein Prozess/Anwendung, Module)

Publishing Checkliste

6. Anwendung signieren

Strategien

Empfehlung: Alle Anwendungen mit dem selben Zertifikat signieren.

- Update / Upgrade
- Modularität (ein Prozess/Anwendung, Module)
- Code/data sharing

Publishing Checkliste

6. Anwendung signieren

Strategien

Empfehlung: Gültigkeitsdauer des Schlüssels vorausdenkend vergeben (>25Jahre).

- Zukünftige Updates?

Publishing Checkliste

6. Anwendung signieren

Strategien

Empfehlung: Gültigkeitsdauer des Schlüssels vorausdenkend vergeben (>25Jahre).

- Zukünftige Updates?
- Mehrere Anwendungen mit gleichem Schlüssel signiert?

Publishing Checkliste

6. Anwendung signieren

Strategien

Empfehlung: Gültigkeitsdauer des Schlüssels vorausdenkend vergeben (>25Jahre).

- Zukünftige Updates?
- Mehrere Anwendungen mit gleichem Schlüssel signiert?
- Android Market? => Ende nach 22.10.2033

Publishing Checkliste

6. Anwendung signieren

Es gibt zwei Arten zu Signieren.

- „Release mode“
- „Debug mode“

Publishing Checkliste

6. Anwendung signieren

„Release mode“

- Passenden privaten Schlüssel generieren
- Anwendung in release mode compilieren
- Fertige .apk signieren

Publishing Checkliste

6. Anwendung signieren

„Release mode“

- Passenden privaten Schlüssel generieren
- Anwendung in release mode compilieren
- Fertige .apk signieren

Publishing Checkliste

6. Anwendung signieren

Passenden privaten Schlüssel generieren

Keytool.exe liegt in JavaSDK/bin Ordner

Shell:

```
keytool.exe -genkey -v -keystore my-release-  
key.keystore  
-alias alias_name -keyalg RSA -validity 10000
```


Publishing Checkliste

6. Anwendung signieren

Passenden privaten Schlüssel generieren

Keytool.exe liegt in JavaSDK/bin Ordner

Shell:

```
keytool.exe -genkey -v -keystore my-release-  
key.keystore  
-alias alias_name -keyalg RSA -validity 10000
```

generiert ein Schlüsselpaar

Publishing Checkliste

6. Anwendung signieren

Passenden privaten Schlüssel generieren

Keytool.exe liegt in JavaSDK/bin Ordner

Shell:

```
keytool.exe -genkey -v -keystore my-release-  
key.keystore  
-alias alias_name -keyalg RSA -validity 10000
```

detaillierte Ausgabe

Publishing Checkliste

6. Anwendung signieren

Passenden privaten Schlüssel generieren

Keytool.exe liegt in JavaSDK/bin Ordner

Shell:

```
keytool.exe -genkey -v -keystore my-release-  
key.keystore  
-alias alias_name -keyalg RSA -validity 10000
```

generiert keystore (Schlüssel DB)

Publishing Checkliste

6. Anwendung signieren

Passenden privaten Schlüssel generieren

Keytool.exe liegt in JavaSDK/bin Ordner

Shell:

```
keytool.exe -genkey -v -keystore my-release-  
key.keystore  
-alias alias_name -keyalg RSA -validity 10000
```

Name der keystore (Schlüssel DB) Datei

Publishing Checkliste

6. Anwendung signieren

Passenden privaten Schlüssel generieren

Keytool.exe liegt in JavaSDK/bin Ordner

Shell:

```
keytool.exe -genkey -v -keystore my-release-  
key.keystore  
-alias alias_name -keyalg RSA -validity 10000
```

vergibt ein Alias für den Schlüssel

Publishing Checkliste

6. Anwendung signieren

Passenden privaten Schlüssel generieren

Keytool.exe liegt in JavaSDK/bin Ordner

Shell:

```
keytool.exe -genkey -v -keystore my-release-  
key.keystore  
-alias alias_name -keyalg RSA -validity 10000
```

Schlüsselalias

Publishing Checkliste

6. Anwendung signieren

Passenden privaten Schlüssel generieren

Keytool.exe liegt in JavaSDK/bin Ordner

Shell:

```
keytool.exe -genkey -v -keystore my-release-  
key.keystore  
-alias alias_name -keyalg RSA -validity 10000
```

Verschlüsselungsalgorithmus

Publishing Checkliste

6. Anwendung signieren

Passenden privaten Schlüssel generieren

Keytool.exe liegt in JavaSDK/bin Ordner

Shell:

```
keytool.exe -genkey -v -keystore my-release-  
key.keystore  
-alias alias_name -keyalg RSA -validity 10000
```

Gültigkeitsdauer in Tagen

Shell Ausgabe

```
PS C:\Program Files\Java\jdk1.6.0_18\bin> keytool.exe -genkey -v -keystore my-release-key.keystore  
-alias alias_name -keyalg RSA -validity 10000
```

Geben Sie das Keystore-Passwort ein:

Geben Sie das Passwort erneut ein:

Wie lautet Ihr Vor- und Nachname? *[Unknown]: Heinz Mustermann*

Wie lautet der Name Ihrer organisatorischen Einheit? *[Unknown]: Android developer*

Wie lautet der Name Ihrer Organisation? *[Unknown]: Uni HH*

Wie lautet der Name Ihrer Stadt oder Gemeinde? *[Unknown]: Hamburg*

Wie lautet der Name Ihres Bundeslandes oder Ihrer Provinz? *[Unknown]: Hamburg*

Wie lautet der Landescode (zwei Buchstaben) für diese Einheit? *[Unknown]: DE*

Ist CN=Heinz Mustermann, OU=Android developer, O=Uni HH, L=Hamburg, ST=Hamburg, C=DE
richtig? *[Nein]: ja*

**Erstellen von Schlüsselpaar (Typ RSA, 1.024 Bit) und selbstunterzeichnetem Zertifikat
(SHA1withRSA) mit einer Gültigkeit von 10.000 Tagen**

**für: CN=Heinz Mustermann, OU=Android developer, O=Uni HH, L=Hamburg, ST=Hamburg,
C=DE**

Geben Sie das Passwort für <alias_name> ein.

(EINGABETASTE, wenn Passwort dasselbe wie für Keystore):

Geben Sie das Passwort erneut ein:

[my-release-key.keystore wird gesichert.]

Publishing Checkliste

6. Anwendung signieren

„Release mode“

- Passenden privaten Schlüssel generieren
- Anwendung in release mode compilieren
- Fertige .apk signieren

Publishing Checkliste

6. Anwendung signieren

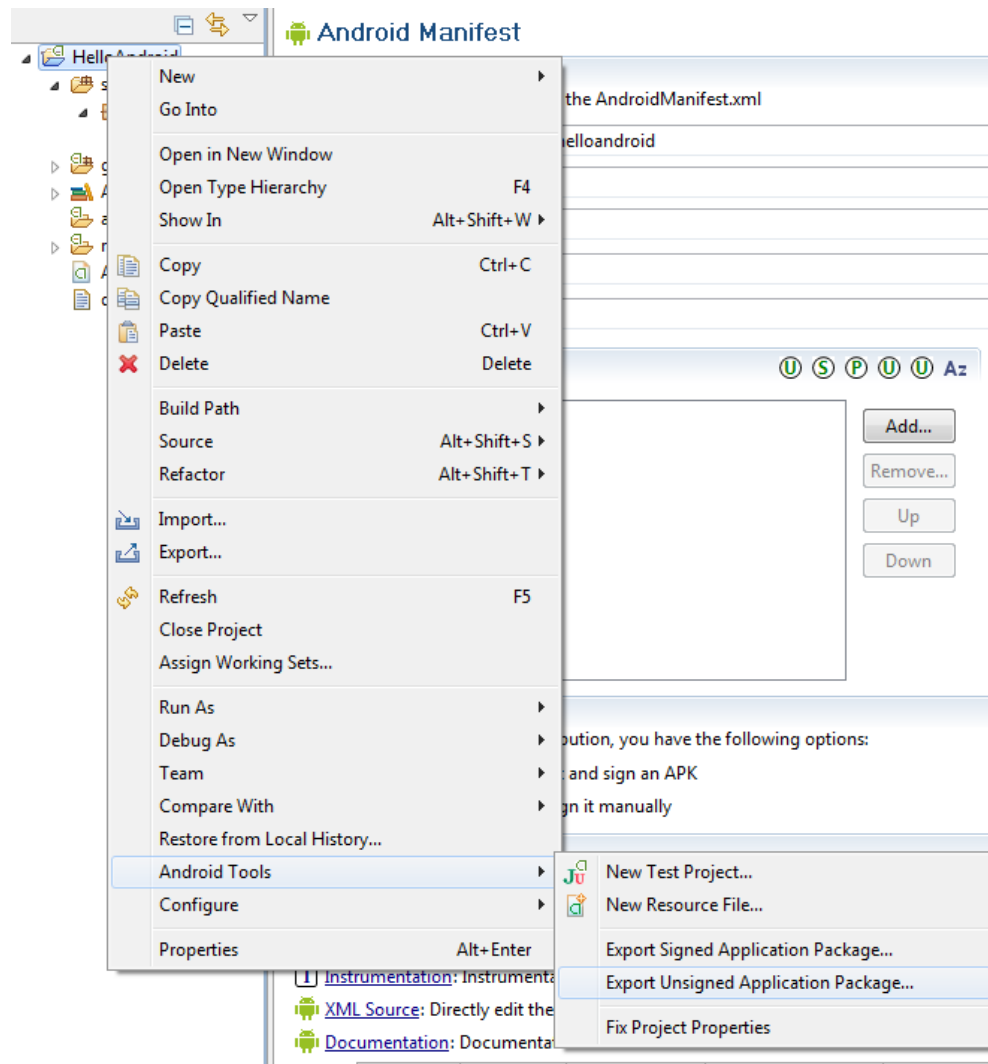
Anwendung in release mode compilieren

In Eclipse:

Rechtsklick auf das Projekt in *Package Explorer* >

Android Tools > *Export Unsigned Application Package*

Anwendung in release mode compilieren



Publishing Checkliste


6. Anwendung signieren

Anwendung in release mode compilieren

Alternative

*In Android Manifest > Exporting > Export an unsigned
APK*

Anwendung in release mode kompilieren

 **Android Manifest**

Manifest General Attributes
Defines general information about the AndroidManifest.xml

Package

Version code

Version name

Shared user id

Shared user label

Manifest Extras U S P U U Az

U Uses Sdk

Exporting
To export the application for distribution, you have the following options:


- [Use the Export Wizard](#) to export and sign an APK
- [Export an unsigned APK](#) and sign it manually


Links
The content of the Android Manifest is made up of three sections. You can also edit the >

A [Application](#): Activities, intent filters, providers, services and receivers.

P [Permission](#): Permissions defined and permissions used.

I [Instrumentation](#): Instrumentation defined.

 [XML Source](#): Directly edit the AndroidManifest.xml file.

 [Documentation](#): Documentation from the Android SDK for AndroidManifest.xml.

Manifest

Publishing Checkliste

6. Anwendung signieren

„Release mode“

- Passenden privaten Schlüssel generieren
- Anwendung in release mode compilieren
- Fertige .apk signieren

Publishing Checkliste

6. Anwendung signieren

Fertige .apk signieren

jarsigner.exe liegt in JavaSDK/bin Ordner

Shell:

```
jarsigner.exe -verbose -keystore my-release-  
key.keystore my_application.apk alias_name
```


Shell Ausgabe

```
PS C:\Program Files\Java\jdk1.6.0_18\bin> jarsigner.exe -verbose -keystore my-release-key.keystore
C:\HelloAndroid.apk alias_name
Enter Passphrase for keystore:
Enter key password for alias_name:
  adding: META-INF/ALIAS_NA.SF
  adding: META-INF/ALIAS_NA.RSA
  signing: res/layout/main.xml
  signing: AndroidManifest.xml
  signing: resources.arsc
  signing: res/drawable-hdpi/icon.png
  signing: res/drawable-ldpi/icon.png
  signing: res/drawable-mdpi/icon.png
  signing: classes.dex
```

Publishing Checkliste

6. Anwendung signieren

Signierte .APK optimieren

zipalign.exe liegt in AndroidSDK/tools Ordner

Shell:

```
zipalign -v 4 your_project_name-unaligned.apk  
your_project_name.apk
```

Publishing Checkliste

6. Anwendung signieren

Es geht auch alles einfacher mit Eclipse ADT plugin 😊

Projekt in *Package Explorer* auswählen > *Export* > *Android* > *Export Android Application*.

Keystore kann erstellt oder ausgewählt werden

Publishing Checkliste

6. Anwendung signieren

„Debug mode“

- Hilfreich während der Entwicklungsphase

Publishing Checkliste

6. Anwendung signieren

„*Debug mode*“

- Hilfreich während der Entwicklungsphase
- 365 Tage gültig, danach neu erstellen

Publishing Checkliste

6. Anwendung signieren

„Debug mode“

- Hilfreich während der Entwicklungsphase
- 365 Tage gültig, danach neu erstellen
- Automatisches signieren in Debug Modus

Publishing Checkliste

6. Anwendung signieren

„Debug mode“

- Hilfreich während der Entwicklungsphase
- 365 Tage gültig, danach neu erstellen
- Automatisches Signieren in Debug Modus
- standardisiert

Keystore name:	"debug.keystore"
Keystore password:	"android"
Key alias:	"androiddebugkey"
Key password:	"android"
CN:	"CN=Android Debug,O=Android,C=US"

Index

Publishing Voraussetzungen

- Publishing Checkliste
 - Versionierung
 - Signierung

Publishing auf:

- Android Market
- Endgerät

Publishing auf Android Market

Voraussetzungen vom Android Market Server:

- Anwendung muss signiert sein und Ablaufdatum nach 22.10.2033
- *android:versionCode* und *android:versionName* müssen definiert sein
- *android:icon* und *android:label* müssen definiert sein

Publishing auf Android Market

Weitere Voraussetzungen:

- Google Konto
- Registrierung als Entwickler auf Android Market (25\$)
- Eigene Webseite?

Index

Publishing Voraussetzungen

- Publishing Checkliste
 - Versionierung
 - Signierung

Publishing auf:

- Android Market
- Endgeräten

Publishing auf Endgeräten

Es gibt mehrere Möglichkeiten

- Adb.exe (Android SDK/tools)
- DDMS tool in Eclipse
- Kopieren der .apk Anwendung auf SD Karte

Publishing auf Endgeräten

Adb.exe (Android SDK/tools)

Shell:

adb.exe -e install Myapp.apk (Emulator)

adb.exe -d install Myapp.apk (USB Gerät)

Evtl.

im Android Gerät > *Einstellungen* > *Anwendungen*>
Unbekannte Quellen Aktivieren

Publishing auf Endgeräten

DDMS tool in Eclipse

DDMS Ansicht öffnen > Unter *Devices* Gerät auswählen
> *File Explorer* > *Data/app* >
Knopf („*push a file onto the device*“) >
Fertige .apk Datei auswählen

Publishing auf Endgeräten

The screenshot shows the Android Studio interface. On the left, the 'Devices' panel lists several virtual devices, including 'emulator-5554' which is online. Below it, the 'Emulator Control' panel shows telephony status and actions. On the right, the 'File Explorer' panel displays the file system of the selected device. The 'app' directory is selected, showing two APK files: 'com.example.helloandroid.apk' (13499 bytes) and 'com.ringdroid.apk' (248728 bytes). Other directories like 'data', 'sdcard', and 'system' are also visible.

Name	Size	Date	Time	Permissions	Info
data		2010-03-11	21:31	drwxrwx--x	
anr		2010-03-11	21:31	drwxrwx--x	
app		2010-03-23	21:25	drwxrwx--x	
com.example.helloandroid.apk	13499	2010-03-23	21:17	-rw-r--r--	com.exam...
com.ringdroid.apk	248728	2010-03-21	21:45	-rw-r--r--	com.ringdr...
app-private		2010-03-11	21:31	drwxrwx--x	
backup		2010-03-11	21:31	drwx-----	
dalvik-cache		2010-03-23	21:17	drwxrwx--x	
data		2010-03-21	23:32	drwxrwx--x	
dontpanic		2010-03-11	21:31	drwxr-x---	
local		2010-03-11	21:31	drwxrwx--x	
lost+found		2010-03-11	21:31	drwxrwx---	
misc		2010-03-11	21:31	drwxrwx-t	
property		2010-03-21	21:36	drwx-----	
system		2010-03-23	21:44	drwxrwxr-x	
sdcard		1970-01-01	00:00	d---rwxr-x	
system		2009-12-19	00:45	drwxr-xr-x	

Zusammenfassung

Publishing Voraussetzungen

- Versionierung
 - Anwendungsversionierung
 - SDK Versionen
- Signierung
 - Strategien
 - Möglichkeiten

Publishing

- Android Market
- Endgerät

Quellen

- <http://www.android.com>
- <http://www.devx.com/wireless/Article/39972/1954>
- <http://www.androiddevelopment.org/2009/01/19/signing-an-android-application-for-real-life-mobile-device-usage-installation/>

Ende

```
package com.example.helloandroid;

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

public class HelloAndroid extends Activity {
    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);

        tv.setText("Fragen?");
        setContentView(tv);
    }
}
```