

Inhaltsverzeichnis

| | | |
|-----------|--|-----------|
| I | Einführung | 1 |
| 1 | Ein erstes Beispiel | 3 |
| 1.1 | Projekt anlegen | 4 |
| 1.2 | Die erste Activity | 8 |
| 1.3 | Layout definieren | 9 |
| 1.4 | Activities aufrufen | 13 |
| 1.5 | Das Android-Manifest | 15 |
| 1.6 | Fazit | 23 |
| 2 | Systemaufbau | 25 |
| 2.1 | Architekturübersicht | 26 |
| 2.2 | Die Android Runtime | 27 |
| 2.3 | Standardbibliotheken | 30 |
| 2.4 | Der Anwendungsrahmen | 30 |
| 2.5 | Android-Komponenten | 31 |
| 2.6 | Die Klasse Context | 33 |
| 3 | Sicherheit | 35 |
| 3.1 | Das Sandbox-Prinzip | 35 |
| 3.2 | Signieren von Anwendungen | 36 |
| 3.3 | Berechtigungen | 37 |
| 3.4 | Anwendungsübergreifende Berechtigungen | 39 |
| II | Android in der Praxis | 41 |
| 4 | Beispielanwendung »Amando« | 43 |
| 4.1 | Aus Sicht des Anwenders | 43 |
| 4.2 | Kapitelübersicht | 47 |
| 5 | Oberflächengestaltung | 49 |
| 5.1 | Ziel | 49 |
| 5.2 | Schnelleinstieg: Activities, Layouts und Views | 49 |

| | | |
|--------|--|-----|
| 5.2.1 | Grundbausteine der Oberflächengestaltung | 50 |
| 5.2.2 | Oberflächen implementieren | 51 |
| 5.3 | Ressourcen | 54 |
| 5.3.1 | Definition von Ressourcen | 55 |
| 5.3.2 | Zugriff auf Ressourcen | 59 |
| 5.3.3 | Textressourcen | 61 |
| 5.3.4 | Farbressourcen | 63 |
| 5.3.5 | Größendefinitionen | 63 |
| 5.3.6 | Bilder | 65 |
| 5.3.7 | Animationen | 66 |
| 5.3.8 | Multimediateien | 67 |
| 5.3.9 | Der raw-Ordner | 68 |
| 5.3.10 | XML-Dateien | 69 |
| 5.3.11 | Der Asset-Ordner | 69 |
| 5.4 | Layouts und Views | 70 |
| 5.4.1 | Definition von Layouts | 70 |
| 5.4.2 | Übersicht vorhandener Layouts | 73 |
| 5.4.3 | Übersicht vorhandener Views | 77 |
| 5.4.4 | Views verwenden | 78 |
| 5.5 | Schaltflächen und Menüs | 80 |
| 5.5.1 | Schaltflächen | 81 |
| 5.5.2 | Oberflächenereignisse | 82 |
| 5.5.3 | Menüs im Allgemeinen | 83 |
| 5.5.4 | Menüdefinition | 83 |
| 5.5.5 | Optionsmenüs | 85 |
| 5.5.6 | Kontextmenüs | 87 |
| 5.5.7 | Dynamische Menügestaltung | 89 |
| 5.6 | Formularverarbeitung | 90 |
| 5.6.1 | Zielsetzung | 91 |
| 5.6.2 | Arbeiten mit Views | 91 |
| 5.7 | Das Android-Manifest | 96 |
| 5.8 | Formatvorlagen: Styles und Themes | 97 |
| 5.8.1 | Styles | 98 |
| 5.8.2 | Themes | 99 |
| 5.9 | Implementierung einer Bildschirmseite | 102 |
| 5.9.1 | Checkliste: Bildschirmseite implementieren | 102 |
| 5.9.2 | Texte für Bildschirmseiten definieren | 103 |
| 5.9.3 | Styles und Themes definieren | 104 |
| 5.9.4 | Weitere Ressourcen definieren | 105 |
| 5.9.5 | Layouts definieren | 105 |
| 5.9.6 | Menüs definieren | 106 |
| 5.9.7 | Activity implementieren | 107 |
| 5.9.8 | Auf Schaltflächen-Ereignisse reagieren | 109 |

| | | |
|----------|--------------------------------------|------------|
| 5.9.9 | Android-Manifest anpassen | 110 |
| 5.9.10 | Bildschirmseite im Emulator testen | 111 |
| 5.10 | Tipps und Tricks | 113 |
| 5.10.1 | Scrolling | 113 |
| 5.10.2 | Umgebungsabhängige Ressourcen | 115 |
| 5.10.3 | Hilfeseiten mit WebView darstellen | 118 |
| 5.10.4 | Der Hierarchy Viewer | 121 |
| 5.11 | Fazit | 124 |
| 6 | Oberflächen und Daten | 125 |
| 6.1 | Zielsetzung | 125 |
| 6.2 | AdapterViews und Ressourcen | 126 |
| 6.3 | AdapterViews und Adapter | 127 |
| 6.3.1 | ArrayAdapter | 128 |
| 6.3.2 | Auf Ereignisse reagieren | 130 |
| 6.4 | Anwendungseinstellungen | 134 |
| 6.4.1 | Begriffsdefinitionen | 135 |
| 6.4.2 | Einstellungen definieren | 136 |
| 6.4.3 | Einstellungsseite implementieren | 139 |
| 6.4.4 | Auf Einstellungen zugreifen | 140 |
| 6.4.5 | Einstellungen bearbeiten | 141 |
| 6.5 | Fortschrittsanzeige | 143 |
| 6.6 | Fazit | 144 |
| 7 | Intents | 145 |
| 7.1 | Warum gibt es Intents? | 145 |
| 7.2 | Explizite Intents | 146 |
| 7.3 | Implizite Intents | 147 |
| 7.3.1 | Intent-Filter für implizite Intents | 147 |
| 7.3.2 | Ein Beispiel | 151 |
| 7.3.3 | Intent-Resolution | 152 |
| 7.3.4 | Beispiele für implizite Intents | 152 |
| 7.3.5 | Fehlerbehandlung | 159 |
| 7.4 | Sub-Activities | 160 |
| 7.4.1 | Sub-Activities aufrufen | 161 |
| 7.4.2 | Sub-Activities verwenden | 162 |
| 7.5 | Fazit | 166 |
| 8 | Hintergrundoperationen | 167 |
| 8.1 | Ziel | 167 |
| 8.2 | Theorie: Prozesse, Threads, Services | 168 |
| 8.2.1 | Allgemeines | 168 |
| 8.2.2 | Main Thread | 168 |

| | | |
|-----------|--|------------|
| 8.2.3 | ANR | 169 |
| 8.2.4 | Prozesse vs. Threads | 171 |
| 8.2.5 | Services | 171 |
| 8.2.6 | Services vs. Threads | 173 |
| 8.2.7 | Service-Connections | 174 |
| 8.2.8 | Binder | 174 |
| 8.3 | Implementierung eines Local Service | 175 |
| 8.3.1 | Local Service | 175 |
| 8.3.2 | Mit einem Service verbinden | 178 |
| 8.3.3 | Services starten und stoppen | 181 |
| 8.4 | Callback-Mechanismen | 182 |
| 8.4.1 | Handler | 183 |
| 8.4.2 | Implementierung | 184 |
| 8.4.3 | Threads mit Schleifen | 189 |
| 8.5 | Single Action Service | 192 |
| 8.6 | Intent Service | 198 |
| 8.7 | Projekt Volta | 200 |
| 8.8 | Fazit | 203 |
| 9 | IPC – Inter Process Communication | 205 |
| 9.1 | AIDL | 205 |
| 9.2 | Implementierung | 207 |
| 9.3 | Eigene Datenobjekte per IPC übertragen | 210 |
| 9.3.1 | Eigene Datenobjekte erstellen | 211 |
| 9.3.2 | Parameterübergabe optimieren | 214 |
| 9.4 | Asynchrone Methodenaufrufe | 215 |
| 9.4.1 | Asynchrone Methodenaufrufe mit Rückgabewert | 216 |
| 9.4.2 | Asynchrone Methodenaufrufe verwenden | 220 |
| 9.5 | Fazit | 223 |
| 10 | Broadcast Receiver | 225 |
| 10.1 | Broadcast Intents | 225 |
| 10.2 | Broadcast Receiver | 228 |
| 10.2.1 | Dynamische Broadcast Receiver | 228 |
| 10.2.2 | Statische Broadcast Receiver | 230 |
| 10.3 | Notifications | 241 |
| 10.4 | Fazit | 248 |
| 11 | Datenbanken | 249 |
| 11.1 | Zielsetzung | 249 |
| 11.2 | Android: Wozu Datenbanken? | 249 |
| 11.3 | Das Datenbanksystem SQLite | 250 |

| | | |
|-----------|--|------------|
| 11.4 | Eine Datenbank erstellen | 251 |
| 11.4.1 | Berechtigungen | 251 |
| 11.4.2 | Schemaverwaltung | 251 |
| 11.5 | Datenzugriffe programmieren | 255 |
| 11.5.1 | SQLiteDatabase – Verbindung zur Datenbank | 255 |
| 11.5.2 | Datenbankanfragen | 256 |
| 11.5.3 | Ergebnistyp Cursor | 261 |
| 11.5.4 | Änderungsoperationen | 264 |
| 11.6 | Datenzugriff per Kommandozeile | 267 |
| 11.7 | Implementierung | 268 |
| 11.7.1 | Ein Architekturvorschlag | 269 |
| 11.7.2 | Das Schema erstellen | 275 |
| 11.7.3 | Anfrageergebnisse an der Oberfläche darstellen | 276 |
| 12 | Dateisystem | 279 |
| 12.1 | Aufbau des Dateisystems | 279 |
| 12.1.1 | Das Anwendungsverzeichnis | 279 |
| 12.1.2 | SD-Karten | 279 |
| 12.2 | Verwaltung | 280 |
| 12.3 | Programmierung | 281 |
| 12.3.1 | Zugriff auf das Anwendungsverzeichnis | 283 |
| 12.3.2 | Zugriff auf den »öffentlichen« Speicher | 285 |
| 12.4 | Zusammenfassung | 289 |
| 13 | Content Provider | 291 |
| 13.1 | Zielsetzung | 291 |
| 13.2 | Übersicht/Grundbegriffe | 292 |
| 13.3 | Content-URIs | 293 |
| 13.4 | Content Provider | 295 |
| 13.4.1 | Stammdaten | 296 |
| 13.4.2 | Operationen | 296 |
| 13.4.3 | Lebenszyklus | 298 |
| 13.4.4 | Berechtigungen | 298 |
| 13.4.5 | Deployment | 298 |
| 13.5 | Content Consumer | 299 |
| 13.5.1 | Content Resolver | 300 |
| 13.5.2 | Zugriff auf Datenbankinhalte | 301 |
| 13.5.3 | Zugriff auf Dateien | 302 |
| 13.6 | Implementierung | 303 |
| 13.6.1 | Zugriff auf das Android-Adressbuch | 303 |
| 13.6.2 | Der FotoProvider | 305 |
| 13.6.3 | Der FotoProviderContract | 306 |

| | | |
|-----------|--|------------|
| 13.6.4 | Die Content-Provider-Klasse | 308 |
| 13.6.5 | FotoSpeicher als Content Consumer | 313 |
| 14 | Asynchrone Datenzugriffe, Loader | 317 |
| 14.1 | Einleitung | 317 |
| 14.2 | Datenzugriffe über Loader | 317 |
| 14.2.1 | Der CursorLoader | 318 |
| 14.2.2 | Loader für Datenbankabfragen | 321 |
| 14.3 | Änderungen am Datenbestand | 324 |
| 15 | Lebenszyklen | 325 |
| 15.1 | Prozessverwaltung | 326 |
| 15.2 | Lebenszyklus einer Activity | 328 |
| 15.3 | Lebenszyklus eines Service | 330 |
| 15.4 | Lebenszyklus eines Broadcast Receivers | 332 |
| 15.5 | Activities: Unterbrechungen und Ereignisse | 332 |
| 15.6 | onPause() vs. onSaveInstanceState(Bundle outState) | 334 |
| 15.7 | Beispiele aus der Praxis | 339 |
| 15.7.1 | Beispiel: Kalender-Activity | 339 |
| 15.7.2 | Beispiel: E-Mail-Programm | 341 |
| 15.7.3 | Beispiel: Quick-and-dirty-Alternative | 342 |
| 16 | Datenübertragung | 345 |
| 16.1 | Ziel | 345 |
| 16.2 | Theoretische Grundlagen | 346 |
| 16.2.1 | Das Emulator-Netzwerk | 346 |
| 16.2.2 | Die Internet-Einbahnstraße | 348 |
| 16.2.3 | Netzwerkunterstützung bei Android | 349 |
| 16.2.4 | Arten der Netzwerkübertragung | 351 |
| 16.3 | Netzwerken in der Praxis | 351 |
| 16.3.1 | Verfahren 1: Geoposition senden | 352 |
| 16.3.2 | Verfahren 2: dauerhafte Verbindung | 353 |
| 16.3.3 | Auf Funklöcher reagieren | 358 |
| 16.4 | Fazit | 362 |
| 17 | Standortbezogene Dienste | 363 |
| 17.1 | Ziel | 363 |
| 17.2 | Theoretische Grundlagen | 364 |
| 17.2.1 | Ortungsverfahren | 364 |
| 17.2.2 | GPS, KML und GPX | 365 |
| 17.2.3 | Entwickeln im Emulator | 366 |
| 17.2.4 | Debug Maps API Key erstellen | 368 |
| 17.2.5 | Testgerät einrichten | 372 |

| | | |
|-----------|---|------------|
| 17.3 | Praxisteil | 373 |
| 17.3.1 | Google APIs installieren | 373 |
| 17.3.2 | Google-Play-Services-Bibliothek einbinden | 374 |
| 17.3.3 | Android-Manifest anpassen | 374 |
| 17.3.4 | Der Location Manager | 376 |
| 17.3.5 | Location API v1 | 377 |
| 17.3.6 | Location API v2 – Fuse Location Provider | 383 |
| 17.3.7 | Google Maps API v2 | 388 |
| 17.3.8 | Fortgeschrittene Techniken | 406 |
| 17.4 | Fazit | 413 |
| 18 | Fragments | 415 |
| 18.1 | Ziel | 416 |
| 18.2 | Fragments | 416 |
| 18.2.1 | Lebenszyklus von Fragments | 418 |
| 18.3 | Praxisbeispiel | 419 |
| 18.3.1 | Projektaufbau | 420 |
| 18.3.2 | Activity-Layout | 422 |
| 18.3.3 | Action Bar, Tabs und Optionsmenü | 423 |
| 18.3.4 | Fragments für Listendarstellung | 428 |
| 18.3.5 | Drag and Drop | 431 |
| 18.3.6 | Artikel anzeigen | 431 |
| 18.3.7 | Auswahlliste mit Drop-Funktion | 433 |
| 18.3.8 | Animationen | 436 |
| 18.4 | Fazit | 440 |

III Weiterführende Themen 441

| | | |
|-----------|---|------------|
| 19 | Debugging und DDMS | 443 |
| 19.1 | Anschluss eines Android-Geräts | 443 |
| 19.2 | Systemausgaben mit der LogCat | 444 |
| 19.3 | DDMS: Dalvik Debug Monitor Service | 447 |
| 19.3.1 | Emulator Control | 448 |
| 19.3.2 | Debugging | 449 |
| 19.4 | Traceview | 450 |
| 20 | Sicherheit und Verschlüsselung | 453 |
| 20.1 | Motivation | 453 |
| 20.2 | Grundbegriffe der Verschlüsselung | 454 |
| 20.2.1 | Verschlüsselte Datenübertragung | 455 |
| 20.2.2 | Daten oder Objekte verschlüsseln | 470 |
| 20.2.3 | Verschlüsselung anwenden | 473 |

| | | |
|-----------------------------|--|------------|
| 21 | Automatisiertes Testen | 475 |
| 21.1 | Was testen wir? | 475 |
| 21.2 | Oberflächentests | 477 |
| 21.2.1 | Instrumentierung und Robotium | 478 |
| 21.2.2 | Ein Beispiel | 479 |
| 21.2.3 | Den Test starten | 482 |
| 21.2.4 | Konflikte vermeiden | 483 |
| 21.3 | Modultests | 483 |
| 21.3.1 | Androidfreie Klassen | 483 |
| 21.3.2 | Androidabhängige Klassen | 484 |
| 21.3.3 | Beispiel für einen Modultest | 486 |
| 21.4 | Eine Teststrategie | 488 |
| 22 | Anwendungen marktreif machen | 491 |
| 22.1 | Hintergrundwissen | 491 |
| 22.2 | Zertifikate selbst erstellen | 493 |
| 22.3 | Anwendungen mittels Kommandozeile marktreif machen | 498 |
| 22.3.1 | Ein eigenes Zertifikat erstellen | 498 |
| 22.3.2 | Eine Android-Anwendung signieren | 499 |
| 23 | Storage Access Framework | 501 |
| 23.1 | Überblick | 501 |
| 23.2 | Beispiel 1: Bilder-Client | 502 |
| 23.3 | Beispiel 2: Dokumenten-Client | 506 |
| 23.4 | Fazit | 509 |
| 24 | Optimierung und Performance | 511 |
| 24.1 | Erste Optimierungsregeln | 511 |
| 24.2 | Datenobjekte | 512 |
| 24.3 | Cursor oder Liste? | 512 |
| 24.4 | Time is Akku! | 513 |
| 25 | Buildsystem Gradle | 515 |
| 25.1 | Warum Gradle? | 515 |
| 25.2 | Gradle in der Praxis | 516 |
| 25.2.1 | Import von Eclipse-Projekten | 516 |
| 25.2.2 | Der Gradle Wrapper | 517 |
| Anhang | | 519 |
| Literaturverzeichnis | | 545 |
| Index | | 549 |