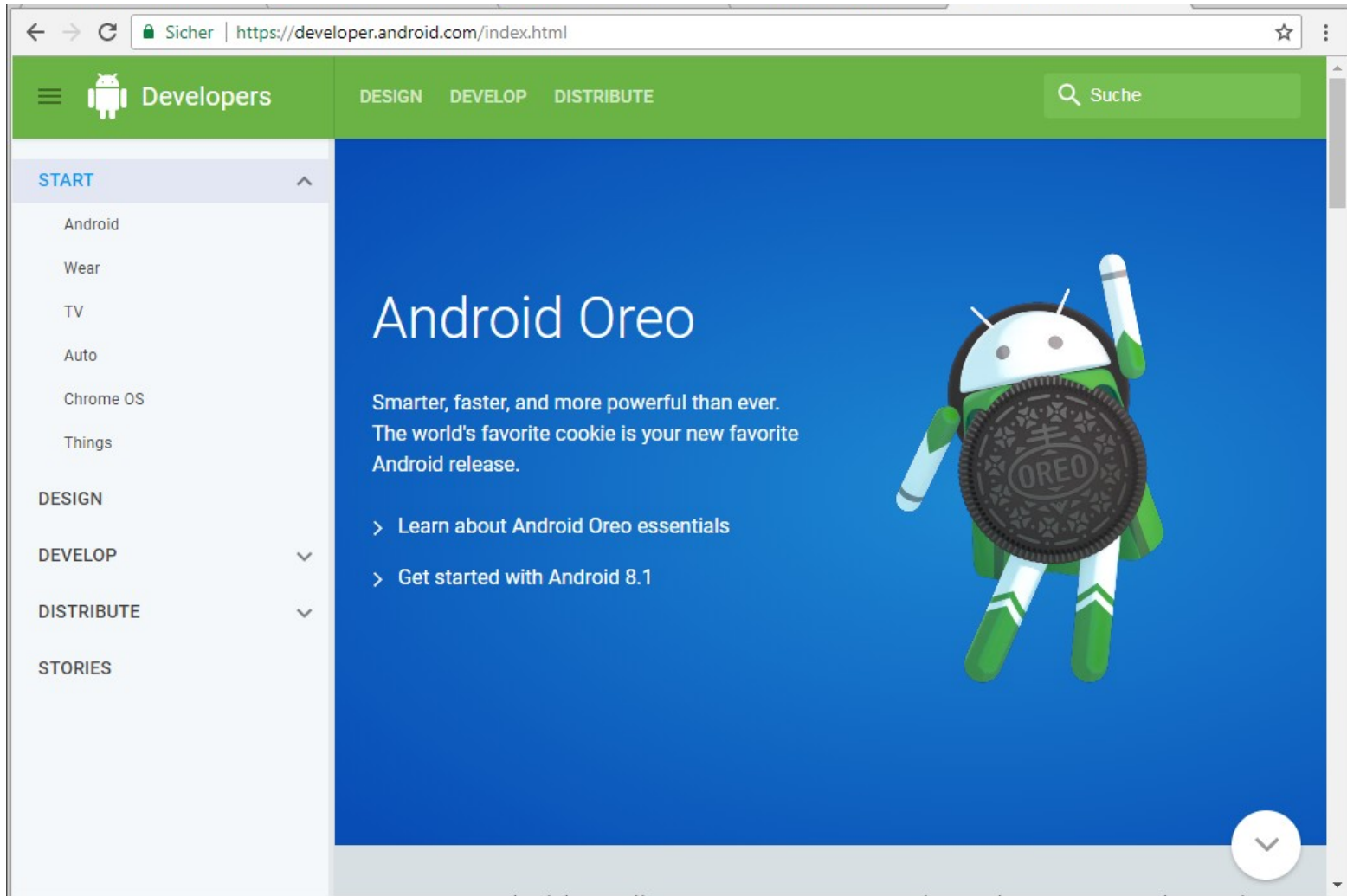




integrata
cegos

Android

Grundlagen der Programmierung



- Android und das Android SDK sind frei verfügbar
- Dies ist ein Programmier-Seminar
 - Damit werden die Inhalte durch Übungen vertieft und verinnerlicht
 - Musterbeispiele werden zur Verfügung gestellt
 - Diese können am Ende des Seminars als ZIP-Datei kopiert werden
 - USB-Stick oder ähnliches
- Dokumentation und Ressourcen stehen auch im Internet zur Verfügung
 - Insbesondere die API-Dokumentation

© Javacream

Javacream

Dr. Rainer Sawitzki

Alois-Gilg-Weg 6
81373 München

**Alle Rechte, einschließlich derjenigen des auszugsweisen Abdrucks,
der fotomechanischen und elektronischen Wiedergabe vorbehalten.**

Einführung	6
Strategien zur Mobilen Anwendungsentwicklung	12
Mobile Plattformen	32
Android	39
UI-Programmierung	60
Ressourcen-Zugriff	85
Fortgeschrittene Konzepte	95

1

EINFÜHRUNG

1.1

ÜBERSICHT

- „Software that is developed for small low power handheld devices such as personal digital assistants, enterprise digital assistants or mobile phones. “
 - Bestenfalls der Versuch einer eindeutigen Definition
 - Smartphones und insbesondere Tablets sind nicht unbedingt „klein“
- Andere Versuche einer Annäherung
 - Mobile Anwendungen
 - Installation und die Benutzung an jedem beliebigen Ort
 - Mobile Daten
 - Zentrale Datenhaltung auf Servern in der Cloud
 - Mobilität des Geräts
 - Potenzielle Lücken in der Netzwerk-Abdeckung

- Naive Definition
 - Anwendungen, auf die die eben angegebenen Begriffe zutreffen
 - Damit hat praktisch jede neu konzipierte Anwendungen einen mobilen Anteil
- Fachliche Definition
 - Anwendungen, die Informationen wie Geolokation benötigen
- Technische Definition
 - Anwendungen, die für den heutigen Markt von Mobilen Endgeräten entwickelt werden
 - Aktuell drei unterschiedliche Ansätze
 - Native Applikationen
 - Mobile Web Applikationen
 - Hybrid-Lösungen

- Native App
 - Neue Konzeption und Entwicklung
 - reichhaltige Benutzer-Interaktionen
 - Verwendung und Definition permanent laufender Hintergrund-Dienste
 - Zugriff auf alle Ressourcen und Sensoren des Geräts

- Mobile Web Anwendungen
 - können häufig durch Anpassungen bereits vorhandener Internet-Auftritte realisiert werden
 - Konzeptions- und Dokumentationsaufwand wird dadurch verringert.
 - Mobile Geräte haben einen relativ gut standardisierten Browser
 - gilt leider nur für moderne Smartphones und Tablets

- Web Anwendungen
 - Chat
 - Datenorientierte Anwendungen mit häufig wechselnden Daten
 - Spiele (Multiplayer)
 - Kataloge, Listen
 - Kalender und Aufgaben synchronisiert mit mehreren Benutzern
- SDK Anwendungen
 - Address-Bücher, Kontakte
 - Animierte Grafiken
 - Datenorientierte Anwendungen mit kritischen Informationen
 - Komplexe Spiele
 - Location-aware Anwendungen
 - Photo-/Video-Anwendungen

2

STRATEGIEN ZUR MOBILEN ANWENDUNGSENTWICKLUNG

2.1

NATIVE APPLIKATIONEN

- Installation über App Store
- Realisierung durch Programmiersprachen, die auf dem Mobile Device zur Verfügung gestellt werden
 - Damit muss die Anwendung in verschiedenen Zweigen mit völlig unterschiedlichen Quellcodes realisiert werden
- Hardware-nahe Programmierung und damit Optimierungen möglich
- Zugriff auf Ressourcen des Mobilgeräts möglich
 - Dateisystem
 - Threading und Hintergrund-Services
 - Embedded Datenbank
 - Client-Server-Zugriff durch http-Sockets
- GUI-Programmierung durch Verwendung von Betriebssystem-Widgets
 - Und damit (fast) automatisch ein einheitliches Look&Feel



Die drei Marktführer (2015)



ANDROID

- Jede native Anwendung ist eine komplett eigene Implementierung
 - Wiederverwendet werden kann „nur“ das Fachkonzept bzw. die Dokumentation
- Damit entsteht unvermeidbar pro Plattform beträchtlicher Aufwand
- Strategie: Konzentration auf den/die Marktführer
 - Andere Plattformen können bei Bedarf auch noch später unterstützt werden
 - Funktioniert innerhalb eines Unternehmens recht gut, da hier die Zielplattformen einigermaßen homogen vorgegeben werden können

2.2

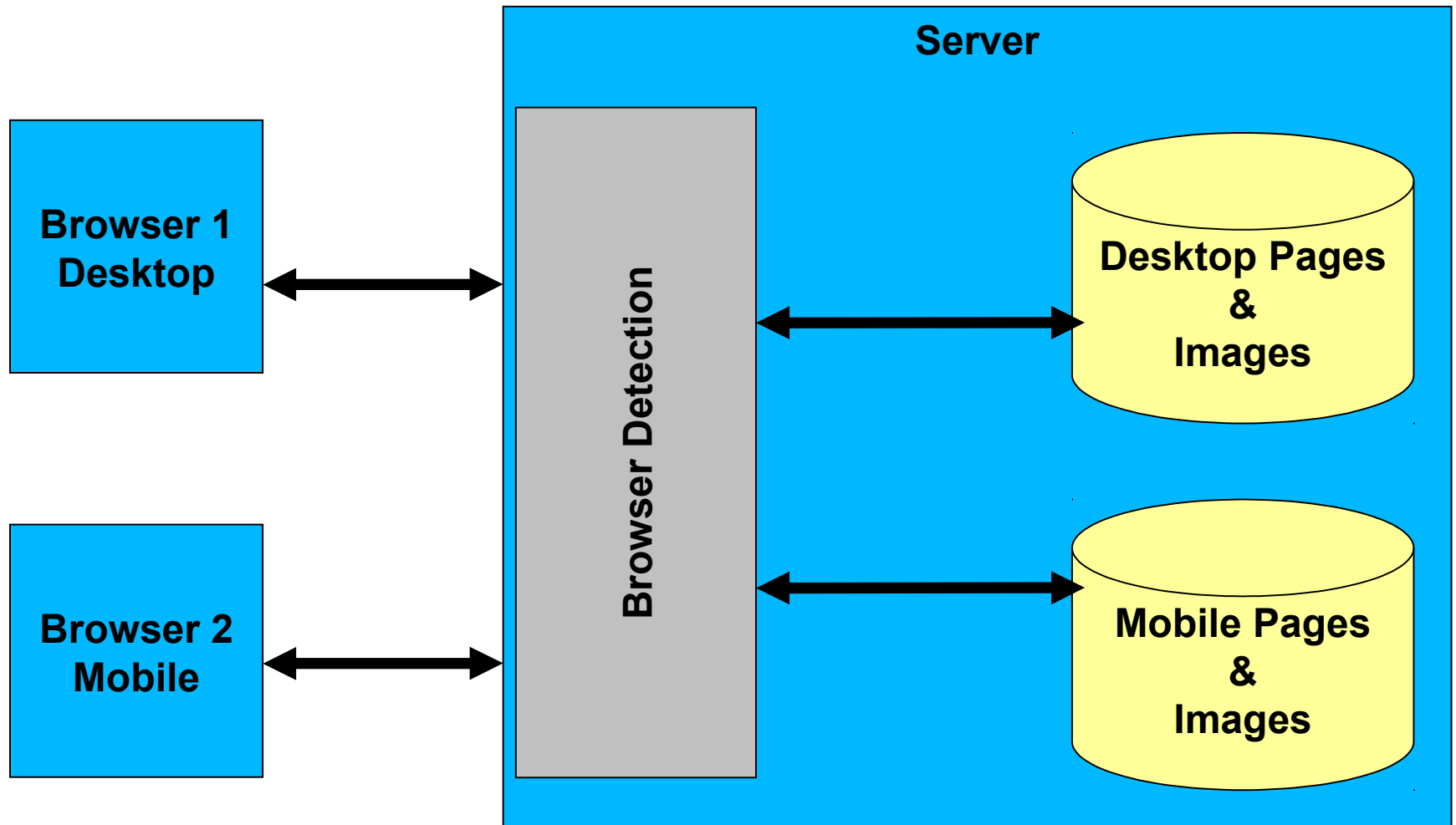
MOBILE WEB APPLICATIONS

- Bereitstellung der Seiten durch den Server
- Das Mobilgerät muss nur einen Browser zur Verfügung stellen
 - Damit können auch noch Feature Phones erreicht werden
- GUI-Programmierung durch Verwendung der vom Browser zur Verfügung gestellten Widgets
- Bei der Programmierung können nur die Technologien des mobilen Browsers benutzt werden
 - Caniuse.com bietet eine Übersicht

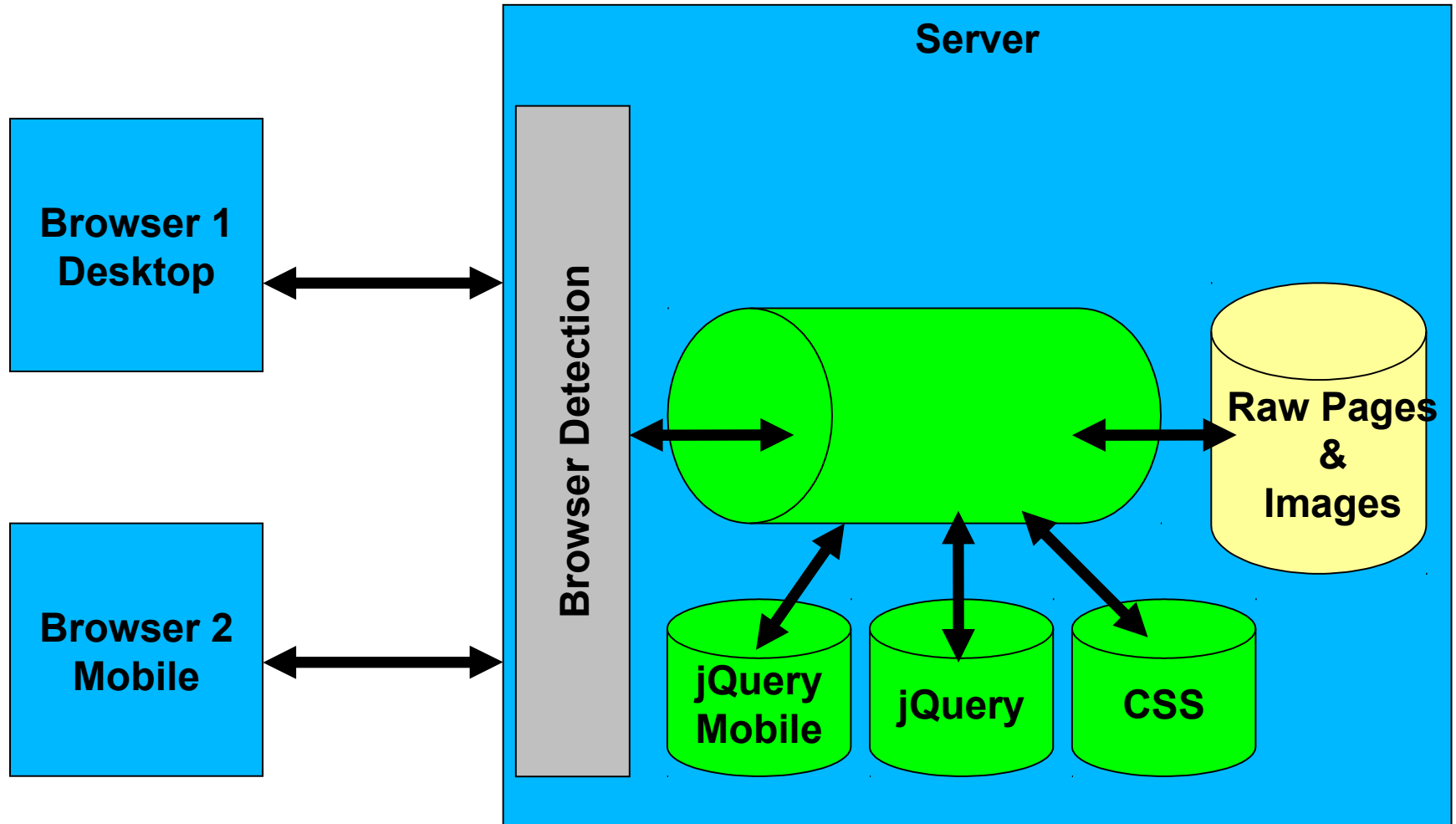
- Low-end Geräte
 - Basic (X)HTML, Bildschirmbreite maximal 176 Pixel, Basic CSS (text color, background color, font size), kein JavaScript
- Mid-range Geräte
 - Basic (X)HTML, Durchschnittliche Bildschirmbreite 240 Pixel, Medium CSS support (box model, images), Basic JavaScript
- High-end Geräte
 - (X)HTML oder HTML 4, Durchschnittliche Bildschirmbreite 320 Pixel, Advanced CSS (Desktop-Browser), Ajax und DOM
- Smartphones
 - HTML5, Relativ große Bildschirme mit High Resolution, Touch support, CSS3 (Animations, Effects), Ajax, Local Storage, Geolocation
- Web App für Smartphones
 - Zusätzlich Offline support, Full-Screen und Icon Installation, Native Integration, Device APIs

- Durch die ebenfalls vorhandene Fragmentierung des Browser-Markts müssen auch hier Strategien entwickelt werden, um
 - Wartbarkeit und
 - Wiederverwendbarkeit zu ermöglichen
- Zur Realisierung können sowohl Server- als auch Client-seitige Ansätze benutzt werden
 - Server
 - Server-side Adaption
 - Progressive Enhancement
 - Client
 - Graceful Degradation
 - Regressive Enhancement
 - Responsive Design

- Der Server stellt für jeden identifizierten Browser eine eigene Seite zur Verfügung
 - Desktop versus Mobile Version
- Die Unterscheidung übernimmt
 - der Server automatisch
 - beispielsweise mit einem Reverse Proxy
 - der Client durch Auswahl der Aufruf-URL
 - m.xyz.de



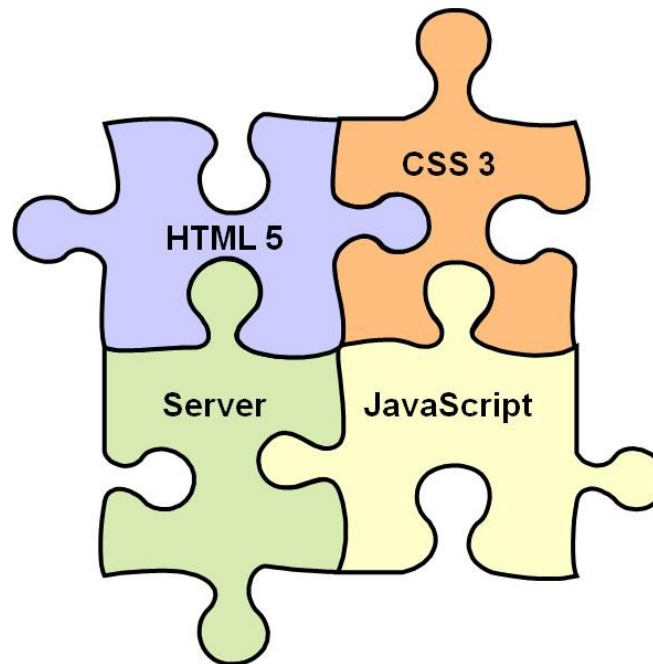
- Eine modulare Form der Server-seitigen Adaption
- Die Funktionalität der Anwendung wird durch verschiedene Schichten erreicht
 - Diese werden in dem Moment aktiv, in dem eine bestimmte Funktionalität vom Browser zur Verfügung gestellt wird
 - Der „raw“ Seite bleibt jedoch für alle Seiten identisch



- Der Browser entscheidet über das Design und Verhalten der Web-Seite
- Um dem Browser diese Entscheidung zu ermöglichen, müssen die Seiten jedoch bestimmten Vorgaben genügen
 - Einfaches Beispiel: Die HTML-Seite darf keine festen Größen- und Positionsangaben enthalten
- Die Browser muss Abfragelogik bereitstellen
 - Browsertyp
 - Bildschirm-Größe und Auflösung
 - Ändern der Darstellung bei einem Wechsel der Orientierung
- Gegebenenfalls muss die Seite in verschiedenen Umgebungen komplett anders dargestellt werden
 - Desktop: Feste Menüstruktur und „Tabbed Panes“
 - Kleingerät: Einblendbares Menü und Navigation zwischen Subseiten
- Beispielseiten
 - mediaqueri.es

- Graceful Degradation
 - Umkehrung des Progressive Enhancements
 - Features, die von alten Browsern nicht unterstützt werden, werden von diesem „fehlerfrei ignoriert“
 - Allerdings ist damit die Funktionalität nicht garantiert!
- Regressive Enhancement
 - Hier werden alte Browser durch Ergänzungen/Workarounds/Hacks in die Lage versetzt, moderne Features zu unterstützen
 - Polyfills sind meistens JavaScript-basierte Lösungen
 - Web Shims implementieren ein API neu
 - HTML5 Shivs stellen HTML5-Features zur Verfügung

- Responsive Web Design verlagert die Verantwortung für die Anpassung der Seitendarstellung auf den Browser
 - Dies alleine kann nicht optimal funktionieren
 - Beispiel: Eine große Bilddatei müsste komplett Übertragen werden, um vom Browser dann skaliert zu werden
- Eine Kombination der Strategien ist notwendig



2.3

HYBRID APPLIKATIONEN

- Eine Hybrid-Application benötigt ein Produkt, das auf allen unterstützten Plattformen installiert werden kann und intern eine homogene Programmier-Umgebung zur Verfügung stellt
- Die Programmierung der Seite erfolgt in einer Plattform-unabhängigen Sprache
 - Beispielsweise durch HTML5, CSS3, JavaScript
- Zugriff auf Ressourcen des Mobilgeräts möglich
 - Hierzu erweitert die Hybrid-Plattform beispielsweise das JavaScript-API um die entsprechenden Objekte
- Die Anwendung selbst ist native und wird über den Plattform-abhängigen Store verteilt
 - Damit ist pro zu unterstützendem Device ein eigener Build-Prozess notwendig

- Die Java-Laufzeitumgebung ist auf allen relevanten Betriebssystemen und Plattformen vorhanden
 - Insbesondere auf dem Server und dem Desktop
- Auch für Mobil-Geräte kann Java eingesetzt werden
 - Dafür existiert sogar eine eigene „Java Micro Edition“-Spezifikation (JME)
- Allerdings ist aktuell wenig Akzeptanz zu erkennen, Java-Standards von Oracle zu benutzen
 - Android
 - Eigene Java-Version ohne Bezug zu JME
 - iOS
 - Keinerlei Bereitschaft zur offiziellen Unterstützung von Java
 - Windows
 - Noch keine klare Linie zu erkennen
 - Android Apps für Windows möglich

3

MOBILE PLATTFORMEN

3.1

BETRIEBSSYSTEME

- Android
 - Linux-basiert
 - Prinzipiell Open Source
 - C-basierte Anwendungsprogrammierung möglich (Native Development Kit, NDK)
- iOS
 - Unix-basiert
 - Geschlossen mit streng reglementiertem Lizenzmodell
- Windows Phone
 - Windows
 - Geschlossen mit teilweiser Öffnung zu Providern

- Android
 - Programmiersprache Java
 - Ergänzt durch natives C
 - Entwicklungsumgebungen als Plugins für Standard-Java-Werkzeuge
 - Eclipse, Netbeans
 - Debugger, Profiler, Simulator
- iOS
 - Programmiersprache Objective C
 - Apple propagiert stark die neue Sprache „Swift“
 - XCode, frei erhältlich für MAC
- Windows Phone
 - Programmiersprachen C#, VB.NET
 - Visual Studio

3.2

INFRASTRUKTUR

- Android
 - Android-interner Browser
 - Chrome
 - Beliebige Browser-Apps
 - Firefox, Opera
- iOS
 - Safari
- Windows Phone
 - Internet Explorer
 - Beliebige Browser-Apps
 - Firefox, Opera

- Android
 - Prinzipiell kann das Android-Betriebssystem selbst compiliert und installiert werden
 - Standard-Funktionen
 - https
 - VPN
 - Installation von Zertifikaten durch Benutzer möglich
- iOS und Windows Phone
 - Geschlossene Systeme
 - Enterprise Umgebungen für Benutzer-Verwaltung, Installation von Zertifikaten etc.
 - https, VPN

4

ANDROID

4.1

EIN ÜBERBLICK

- Die Entwicklung mobiler Applikationen für Smartphones und Tablets ist der am stärksten expandierende Markt im ersten Jahrzehnt dieses Jahrtausends gewesen.
 - Und dieser Trend ist bisher ungebrochen bzw. scheint sich sogar noch weiter zu verstärken.
- Der Markt ist bzw. wird sehr wahrscheinlich in naher Zukunft zwischen drei großen IT-Unternehmen aufgeteilt sein:
 - Apple mit seiner iPhone bzw. iPad-Reihe
 - Microsoft versucht massiv, sein Windows Phone-Betriebssystem zu propagieren und bei der vorhandenen Marktmacht im Desktop-Bereich ist es wahrscheinlich, dass dies zumindest teilweise von Erfolg gekrönt sein wird.
 - Nach aktuellen Analysen wird sich jedoch das von Google entwickelte Android-System mittelfristig den größten Marktanteil sichern.

- Die Hardware-Ausstattung von Smartphones ist im Gegensatz zu den mittlerweile doch sehr stark vereinheitlichten Desktop-Systemen noch äußerst heterogen
 - Dies betrifft nicht nur eher triviale Unterschiede in der Bildschirmauflösung und im vorhandenen Speicher sondern beispielsweise auch Dienste wie Bluetooth, GPS oder die Nahfeld-Kommunikation
- Android selbst verlangt vom Hersteller eines Smartphones oder Tablets relativ wenig standardisierte Hardware und läuft deshalb auch auf einer sehr breiten Produktpalette
 - Weiterhin werden bei Updates auch ältere Plattformen in erstaunlichem Umfang unterstützt

- Diese potenziell heterogenen Hardware-Voraussetzungen werden am Besten durch ein Linux-basiertes Betriebssystem gekapselt
- Linux ist ein Open Source-Produkt und kann deshalb auf einen weiten Hardware-Bereich hin optimiert und kompiliert werden
- Google spezifiziert die notwendigen Features des Android-Kernels im Gegensatz zur darunter liegende Hardware-Plattform exakt

4.2

ANDROID SDK

- Die JVM selbst ist eine native C-basierte Anwendung, die für alle gängigen Betriebssysteme implementiert wurde
- Die JVM ist ein Interpreter für compilierte Java-Programme, dem so genannten Bytecode
 - Bytecode ist Plattform-unabhängig konzipiert und deshalb laufen Java-Anwendungen auf allen Systemen, für die eine JVM existiert
- Die JVM enthält als zentrales Konzept einen Garbage Collector
- Sämtliche Ressourcen-Zugriffe werden über einen Security Manager kontrolliert
- Die JVM stellt einen Fehler-toleranten Exception-Mechanismus zur Verfügung
- Die allermeisten Android-Applikationen laufen in einer speziell auf den Android-Kernel hin optimierten Virtual Machine, der Dalvik-VM

- Die Entwicklung von Android-Applikationen ist pures Java:
 - Syntax,
 - Sprachspezifikation,
 - Bytecode-Format
 - etc. sind komplett übernommen.
- Ebenso können etablierte Verfahren wie grundlegende Design-Umsetzungen, Programmierrichtlinien und Best Practices direkt übernommen werden
- Weiterhin sind alle Java-Werkzeuge wie IDEs, Debugger und Profiler sowie vorhandene Open Source-Bibliotheken zumindest teilweise für Android geeignet

Oracle JDK versus Android SDK

Archive Edit View Help

Open Extract

Back Location: /

Name	Size	Type	Date Modified
com	20.8 MB	Folder	
java	5.5 MB	Folder	
javax	7.4 MB	Folder	
META-INF	2.3 KB	Folder	
org	1.0 MB	Folder	
sun	12.2 MB	Folder	
sunw	644 bytes	Folder	

7 objects (46.9 MB)

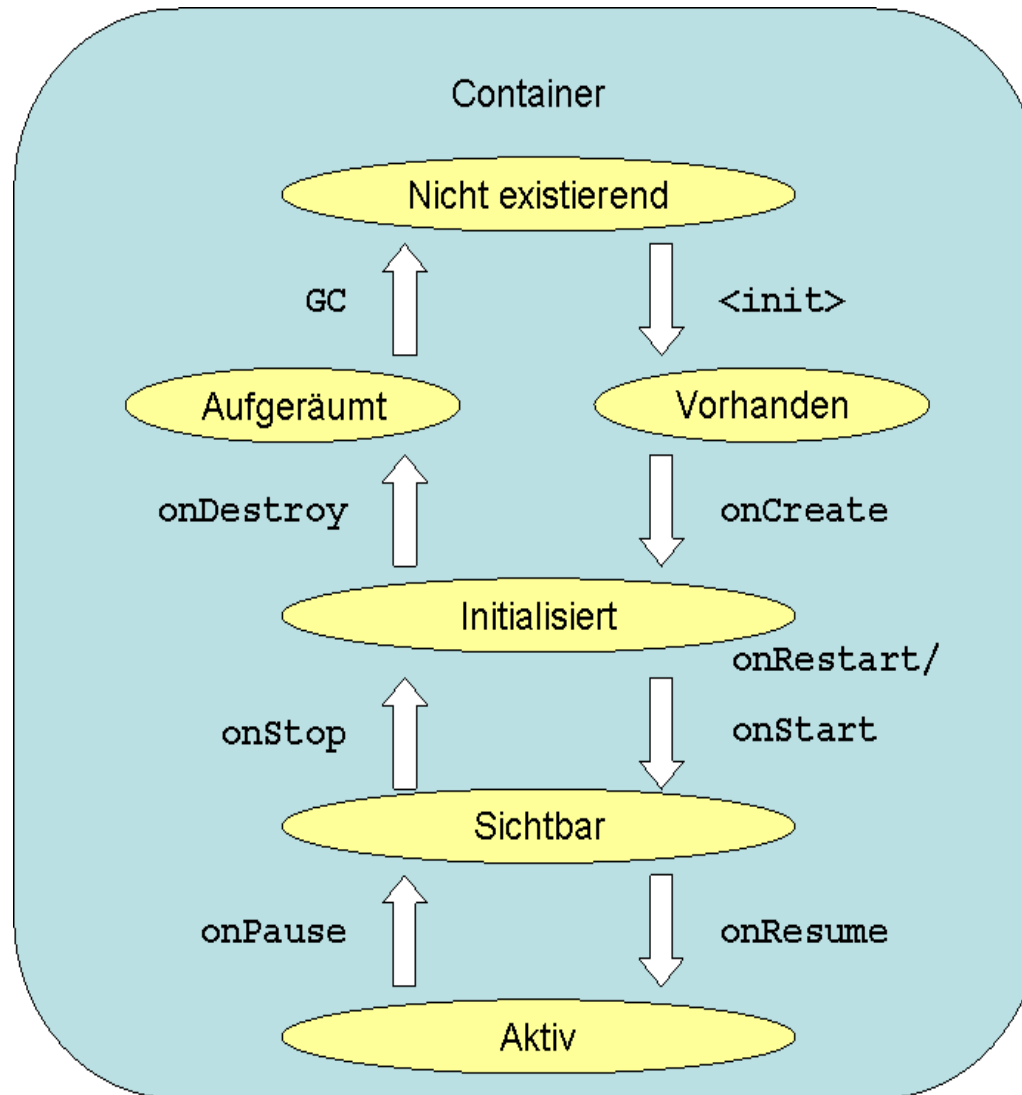
Archive Edit View Help

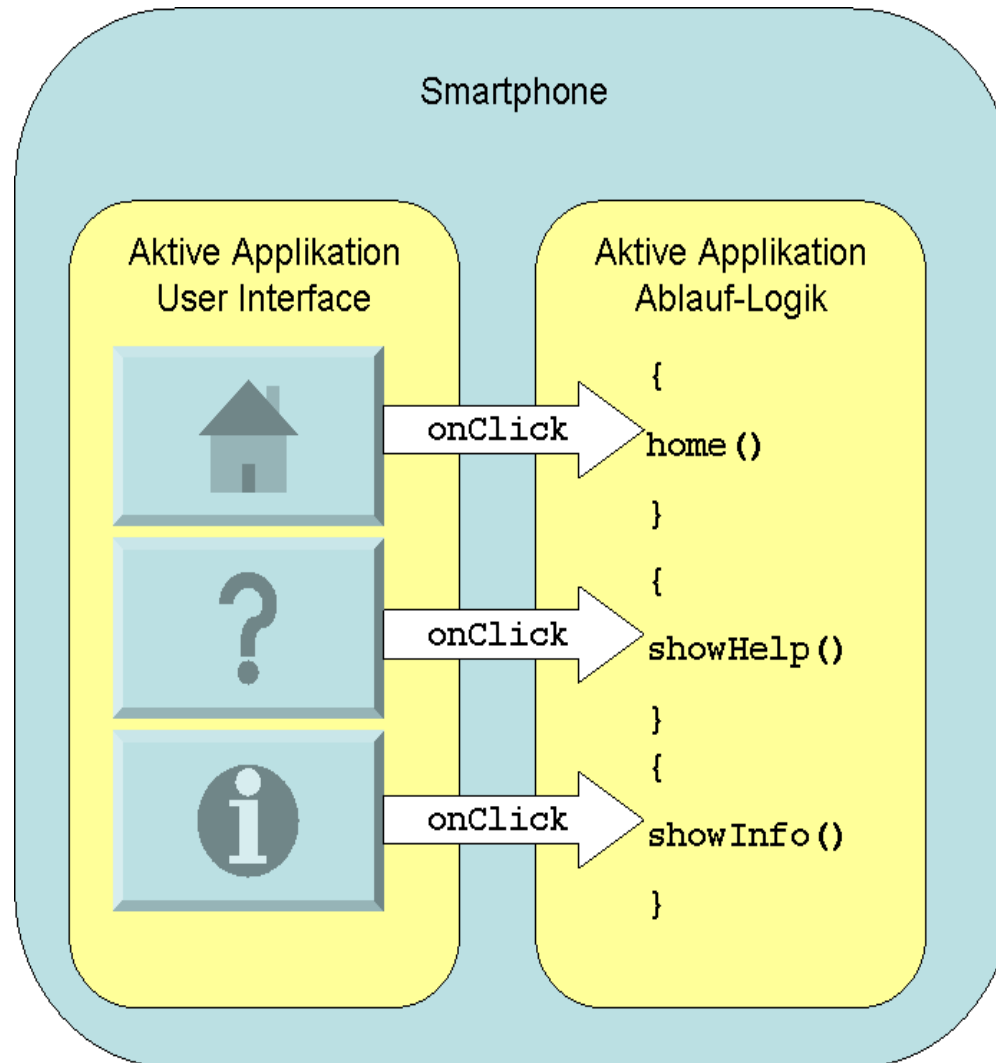
Open Extract

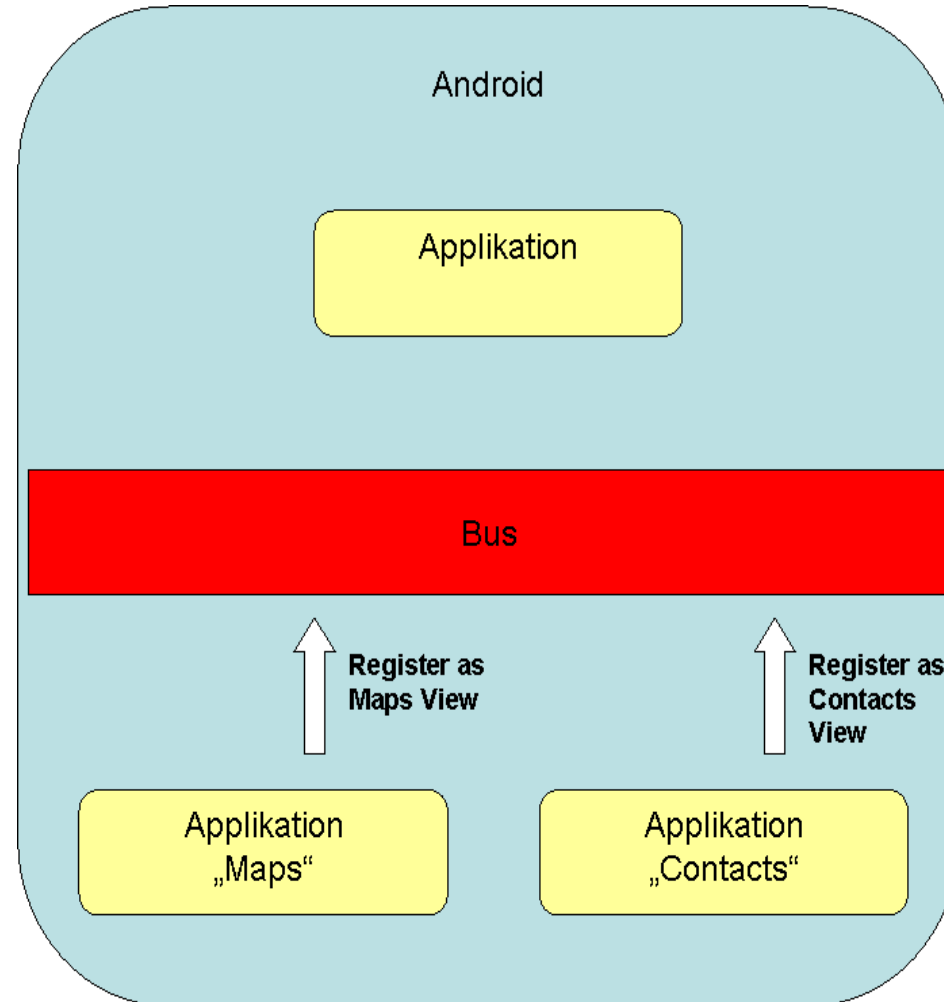
Back Location: /

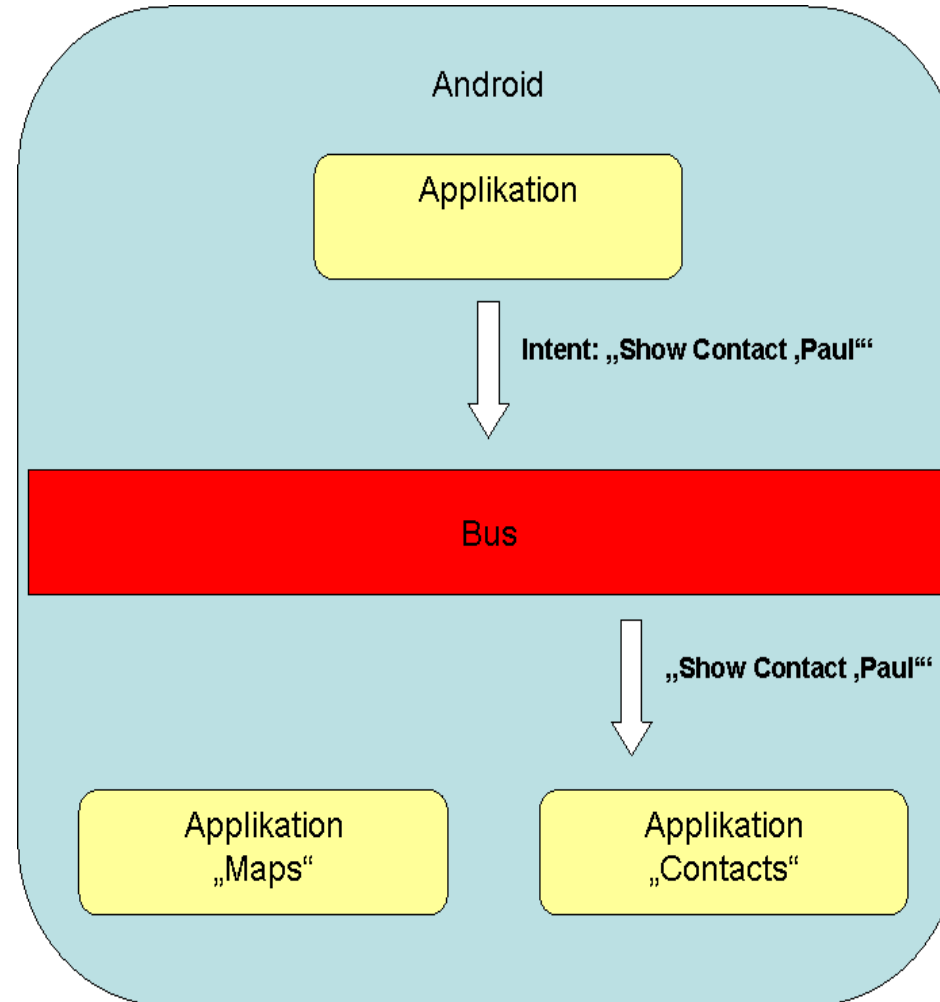
Name	Size	Type	Date Modified
android	2.2 MB	Folder	
assets	233.8 KB	Folder	
com	245 bytes	Folder	
dalvik	14.5 KB	Folder	
java	1.2 MB	Folder	
javax	213.3 KB	Folder	
junit	17.0 KB	Folder	
META-INF	71 bytes	Folder	
org	504.6 KB	Folder	
res	3.5 MB	Folder	
AndroidManifest.xml	49.9 KB	XML docu...	03 February 2011, 14:49
resources.arsc	3.7 MB	unknown	03 February 2011, 14:49

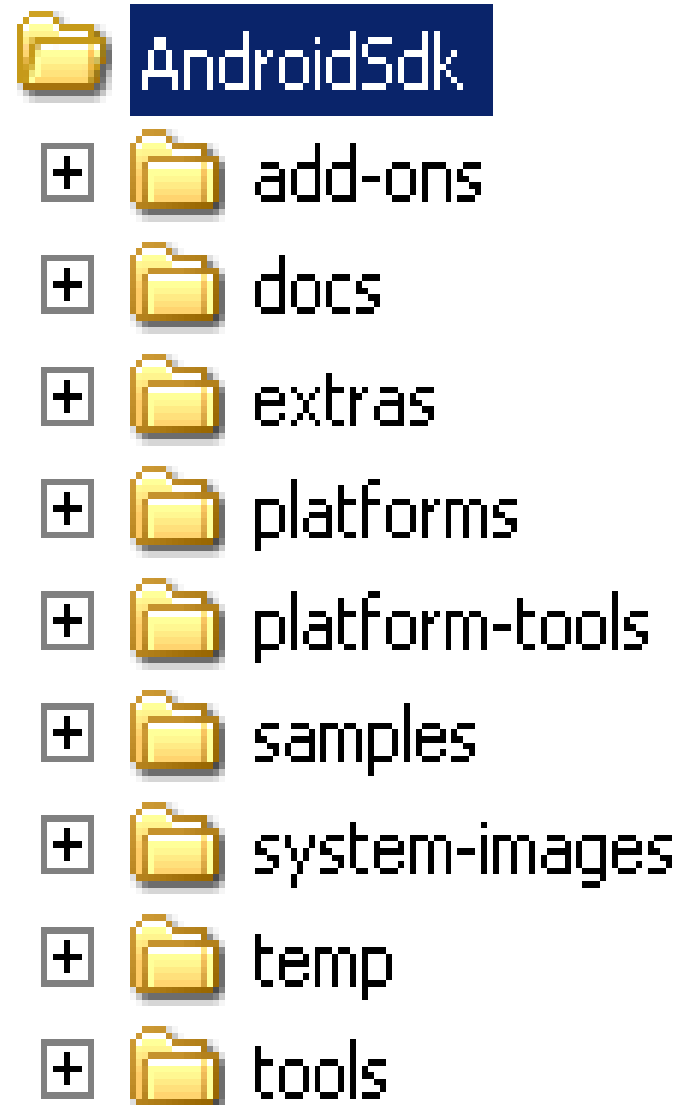
12 objects (11.7 MB)


























Packages Tools

SDK Path: /home/user/Documents/tools/android-sdk-linux/

Packages

 Name	API	Rev.	Status
▶ <input type="checkbox"/> Android 3.1 (API 12)			
▶ <input type="checkbox"/> Android 3.0 (API 11)			
▼ <input type="checkbox"/> Android 2.3.3 (API 10)			
<input type="checkbox"/>  SDK Platform	10	2	 Installed
<input type="checkbox"/>  Samples for SDK	10	1	 Installed
<input type="checkbox"/>  DTS Add-On by KYOCERA Corporation	10	1	 Installed
<input type="checkbox"/>  DTS Add-On by KYOCERA Corporation	10	1	 Installed
<input type="checkbox"/>  EDK 1.1 by Sony Ericsson Mobile Comm	10	1	 Installed
<input type="checkbox"/>  DTS Add-On by KYOCERA Corporation	10	1	 Installed
<input type="checkbox"/>  Google APIs by Google Inc.	10	2	 Installed
▶ <input type="checkbox"/> Extras			

Show: ☒ Updates/New ☒ Installed ☐ Obsolete Select [New](#) or [Updates](#)

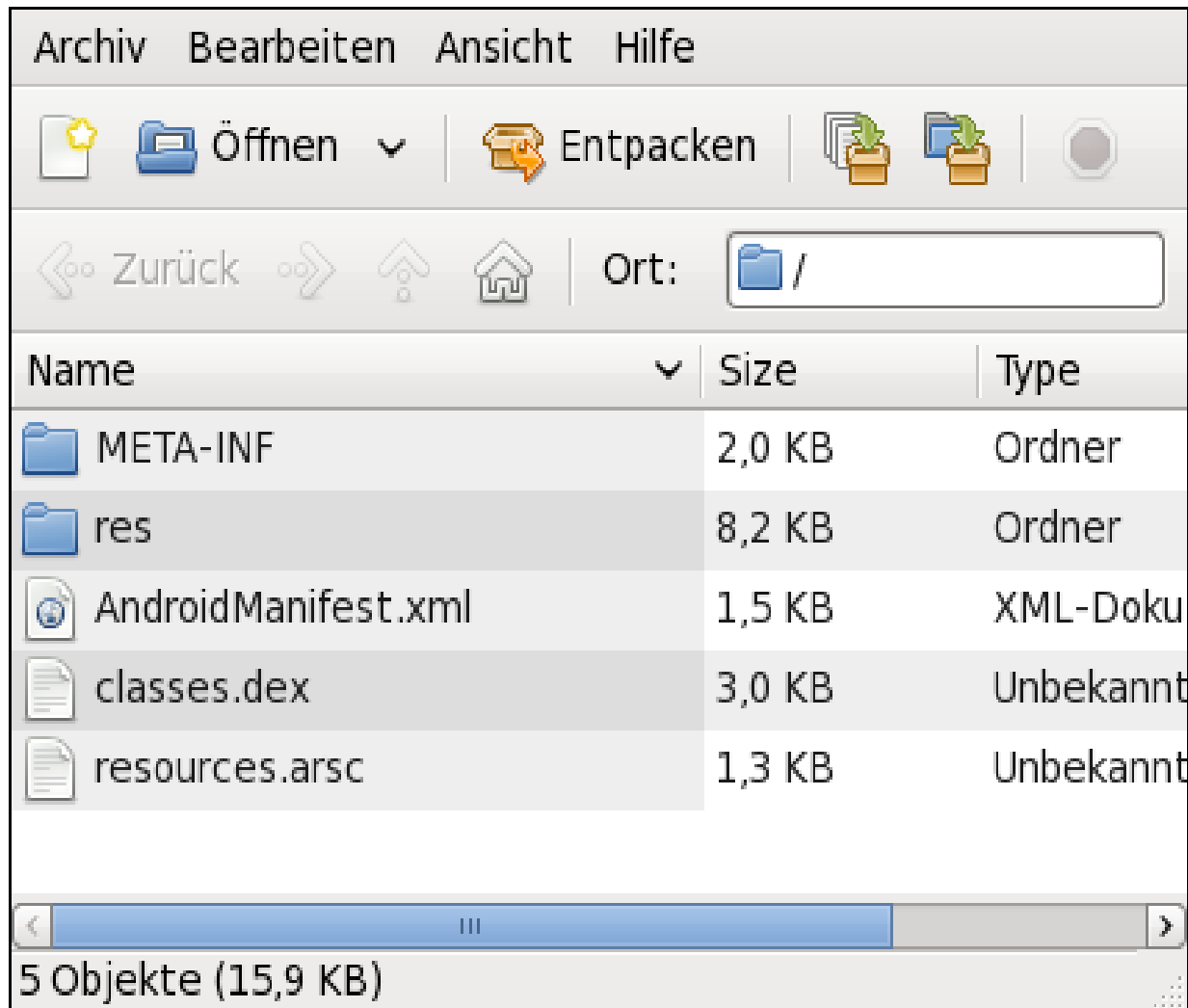
Sort by: ☒ API level ☐ Repository [Deselect All](#)

[Install packages...](#)

Selects all items that are either new or updates.

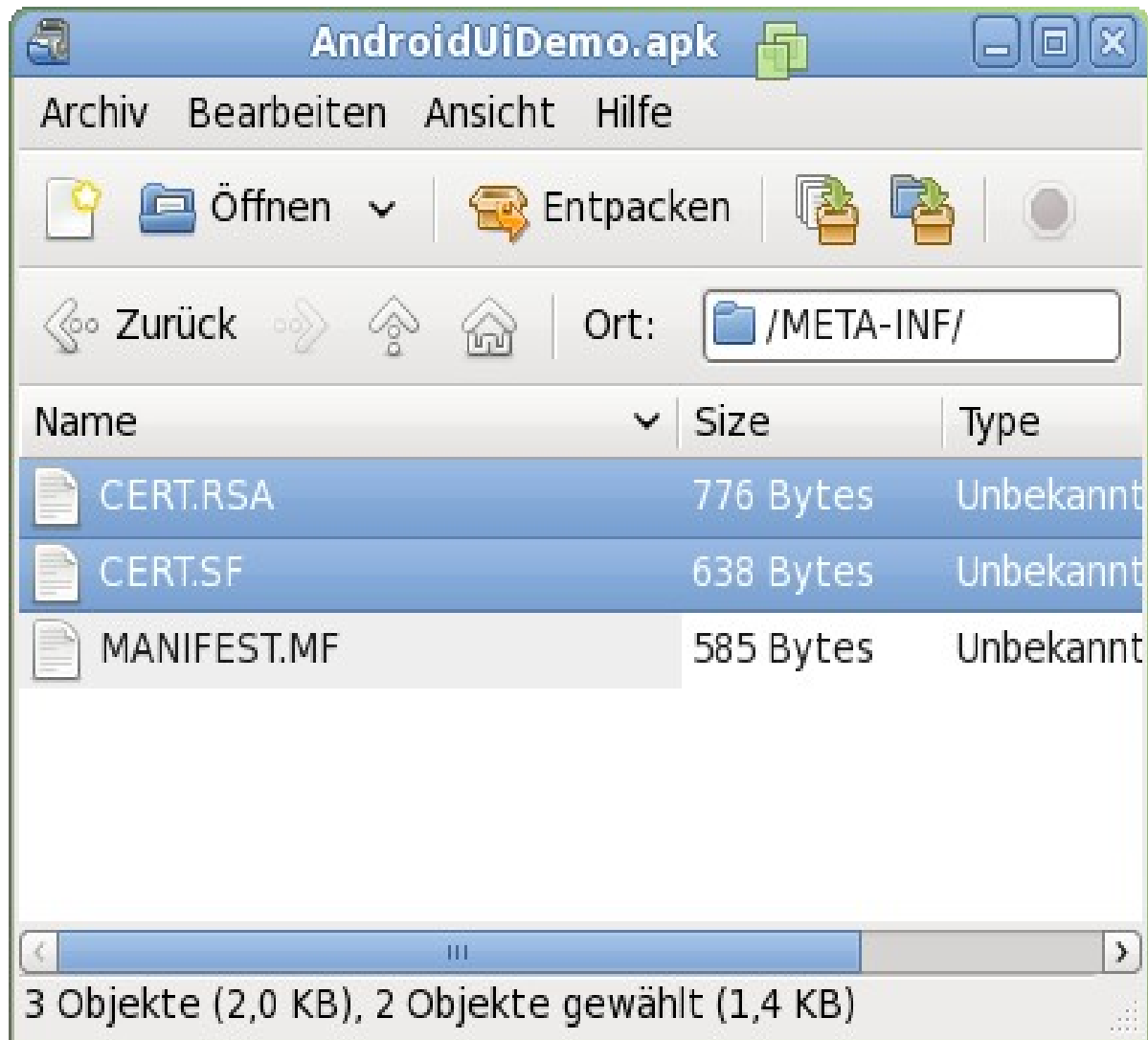
[Delete packages...](#)

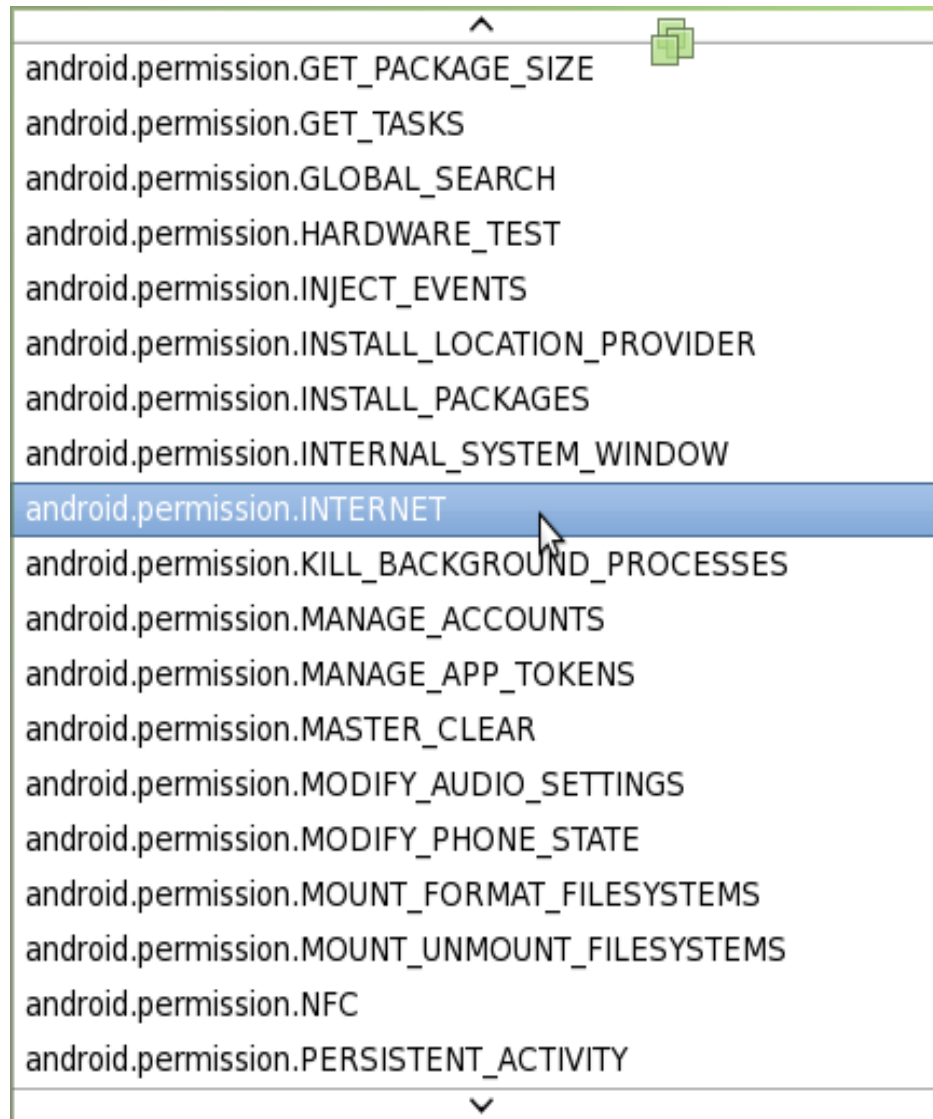
Done loading packages.



- Das Deployment der Anwendung kann auf mehrere Arten erfolgen:
 - Aus der Entwicklungsumgebung durch Starten eines Emulators
 - Auf ein über USB-Kabel verbundenes Device im Debug-Modus
 - Über den Google Play Store
 - Bei allen Verfahren wird effektiv das erzeugte Archiv transferiert. Allerdings verlangt der Google Market eine Registrierung sowie ein signiertes Archiv. Auf Security-Aspekte wird im Folgenden Abschnitt eingegangen.

- Signierte Anwendungen
 - Das Android-Archiv kann mit normalen Java-Werkzeugen signiert werden. Dies ist insbesondere deshalb wichtig, weil ein Android-Gerät ausschließlich signierte Anwendungen ausführen lässt.
 - Die Android Tools erzeugen und benutzen dafür ein Debug-Zertifikat.
 - Dieses ist jedoch für ein Verteilen der Anwendung über den Google Play-Markt nicht gültig und muss durch ein Developer-Zertifikat ersetzt werden, in dem der Hersteller eindeutig zugewiesen werden kann.
- Die Signatur verhindert nachträgliche Änderungen der Dateien:





- Als Programme mit Benutzer-Oberfläche sind Android-Anwendungen nicht trivial zu testen
- Android integriert in seiner Klassenbibliothek das etablierte Junit-Framework und stellt selbst Erweiterungen zur Verfügung
- Junit-Tests
 - Mit Hilfe der Basis-Klasse `junit.framework.TestCase` können sämtliche Klassen der Anwendung, die keinen UI-Bezug besitzen sowie keine Dienste des Devices benötigen getestet werden
 - Näheres zu diesen JUnit-Tests ist im Anhang zu finden.

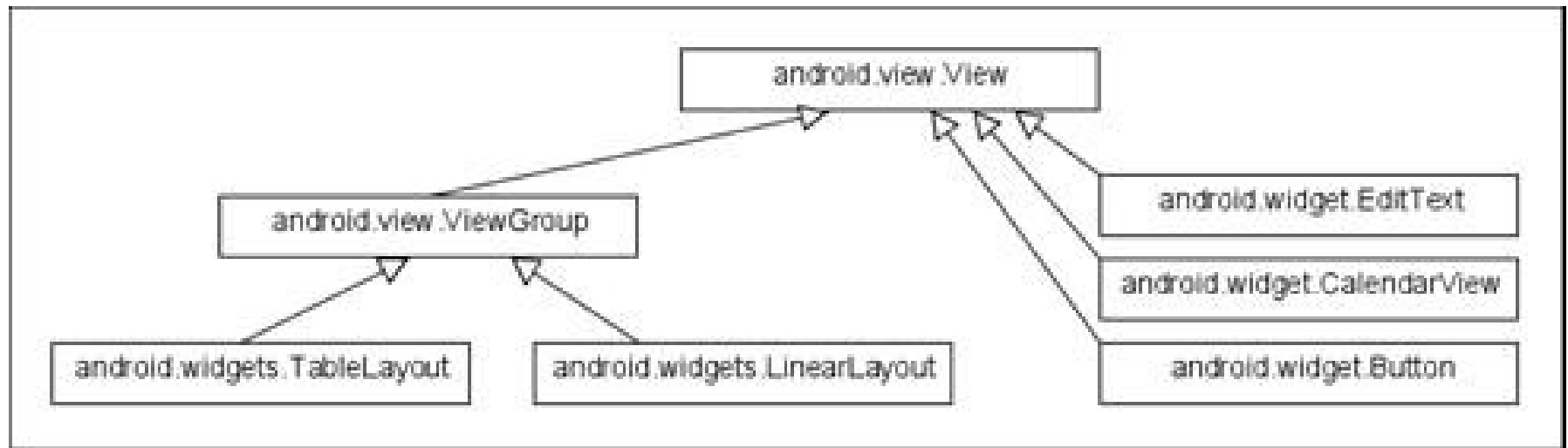
5

UI-PROGRAMMIERUNG

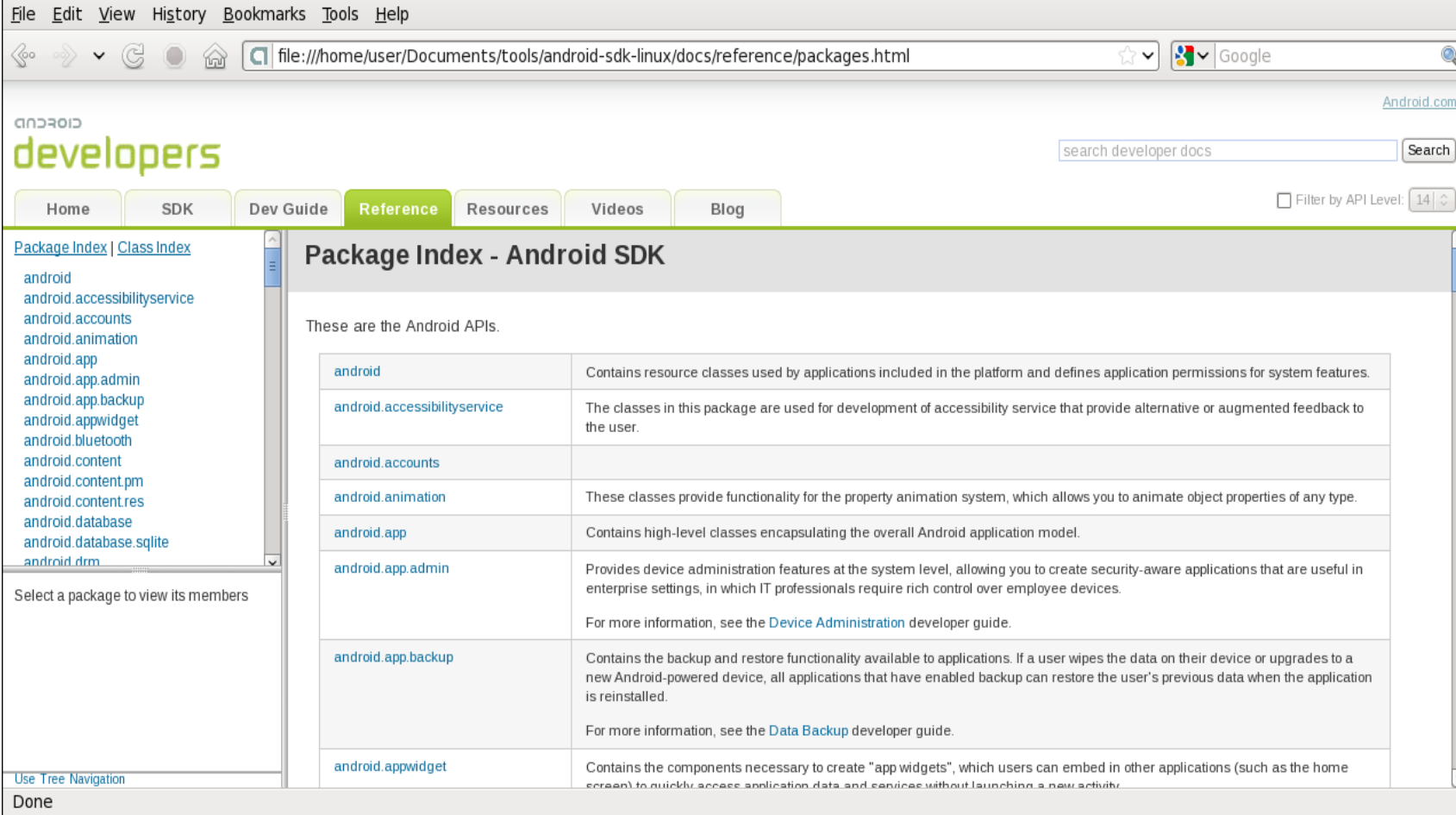
5.1

OBERFLÄCHEN-KOMPONENTEN

- Android enthält in seiner Java-Bibliothek einen kompletten Satz von Oberflächen-Komponenten, den sogenannten Widgets, die zur Definition eines User Interfaces benutzt werden können
- Neben den Widget-Komponenten, die eine direkte visuelle Repräsentation besitzen, existieren die Layouts
 - Diese sind Container für Widgets
- Layouts und Widgets befinden sich im Paket `android.widget`. die gemeinsame Basisklasse ist `android.view.View`
- Die Widget-Klassen selbst stehen in einer flachen Vererbungshierarchie zueinander und enthalten Methoden und Eigenschaften, die eine Anpassung der Oberfläche an die geforderten Vorgaben ermöglichen.



- Zur Definition einer UI gibt es prinzipiell zwei Möglichkeiten:
 - Direkte Programmierung des Objektbaums
 - Dabei werden die einzelnen Widgets direkt im Programmcode erzeugt und in die jeweils passenden Layouts eingefügt
 - Definition der Oberfläche in einem XML-Dokument



The screenshot shows a web browser window displaying the Android SDK reference documentation. The address bar shows the file path: `file:///home/user/Documents/tools/android-sdk-linux/docs/reference/packages.html`. The page title is "Package Index - Android SDK". The left sidebar contains a list of package names under the heading "Package Index | Class Index". The main content area lists several packages with their descriptions.

Package Index - Android SDK

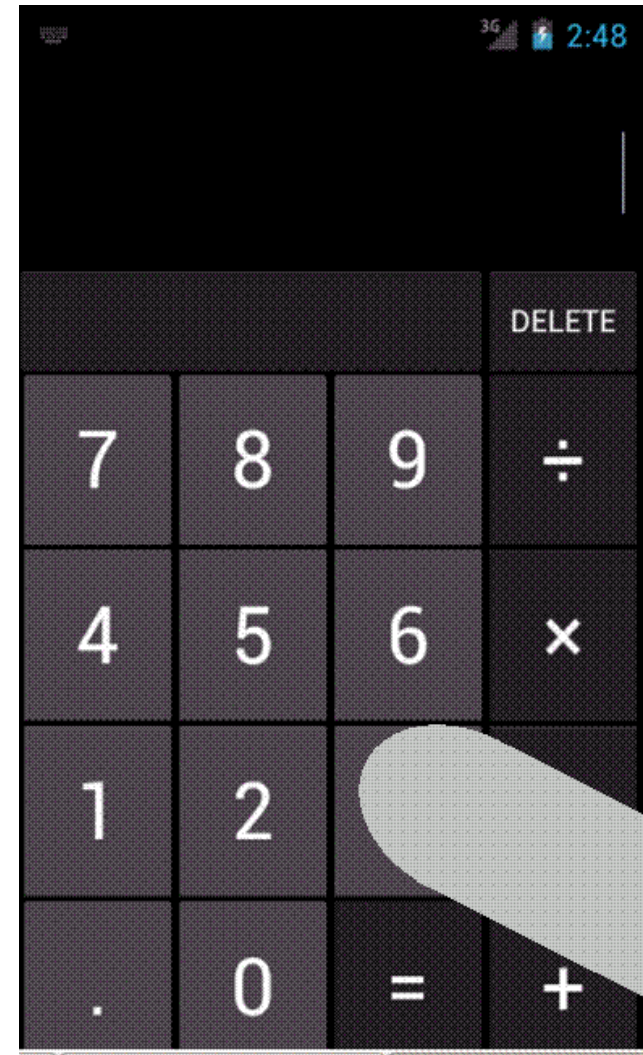
These are the Android APIs.

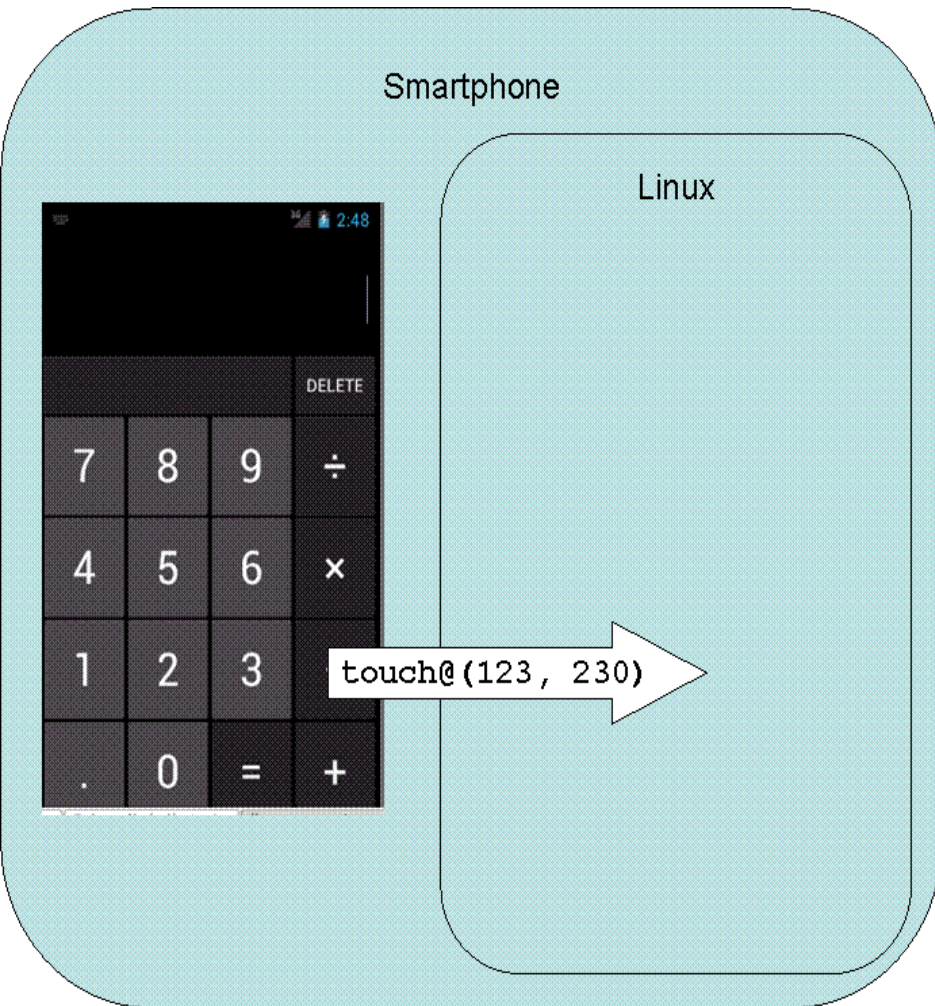
android	Contains resource classes used by applications included in the platform and defines application permissions for system features.
android.accessibilityservice	The classes in this package are used for development of accessibility service that provide alternative or augmented feedback to the user.
android.accounts	
android.animation	These classes provide functionality for the property animation system, which allows you to animate object properties of any type.
android.app	Contains high-level classes encapsulating the overall Android application model.
android.app.admin	Provides device administration features at the system level, allowing you to create security-aware applications that are useful in enterprise settings, in which IT professionals require rich control over employee devices. For more information, see the Device Administration developer guide.
android.app.backup	Contains the backup and restore functionality available to applications. If a user wipes the data on their device or upgrades to a new Android-powered device, all applications that have enabled backup can restore the user's previous data when the application is reinstalled. For more information, see the Data Backup developer guide.
android.appwidget	Contains the components necessary to create "app widgets", which users can embed in other applications (such as the home screen) to quickly access application data and services without launching a new activity.

5.2

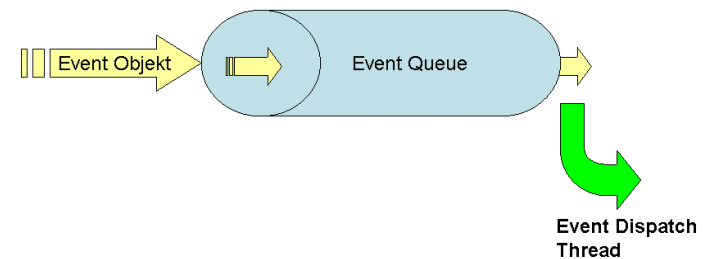
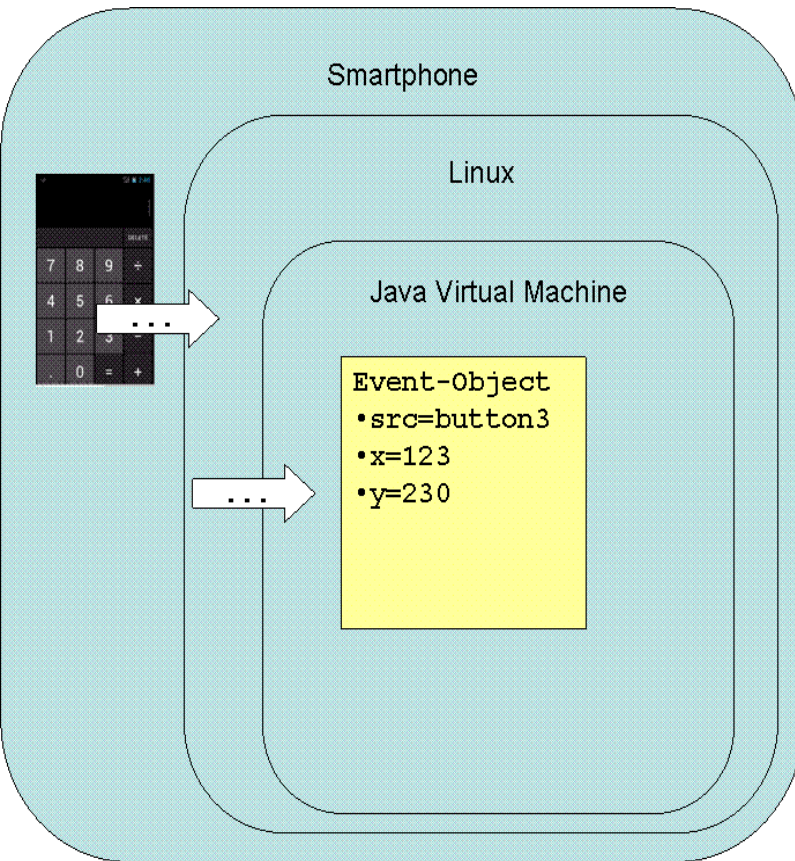
EVENT-MODELL

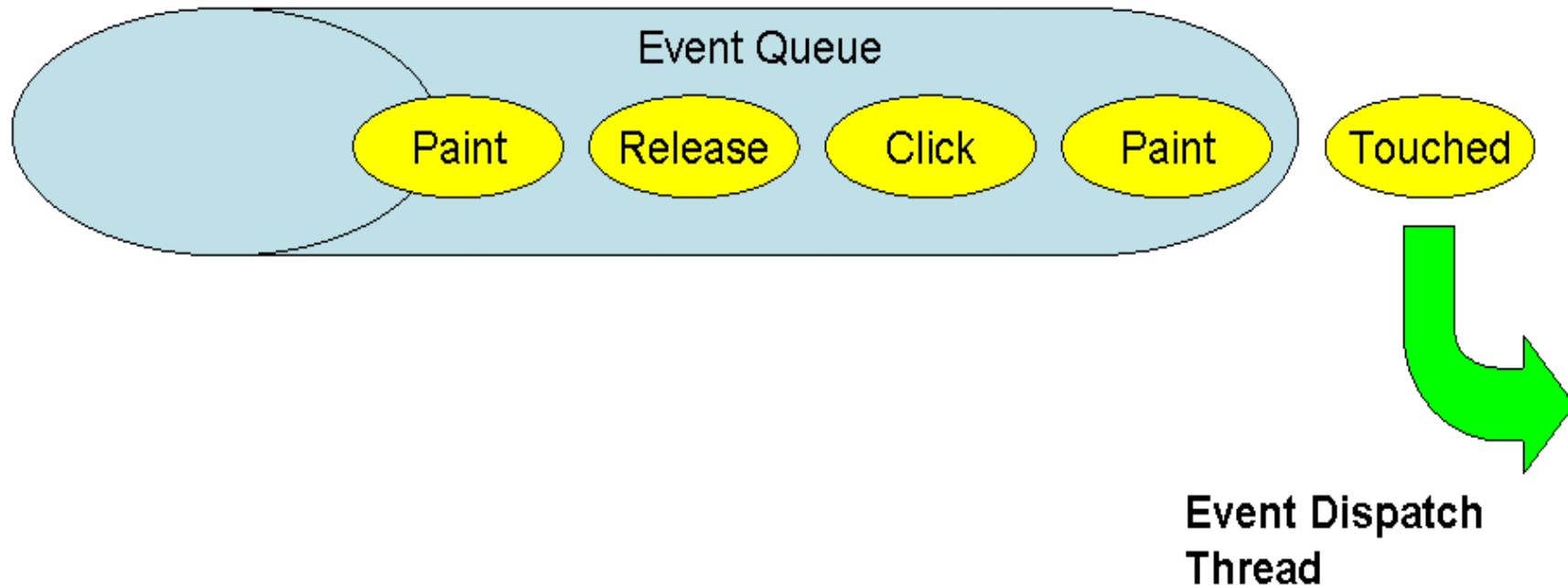
Ausgangspunkt: Eine Benutzer-Aktion



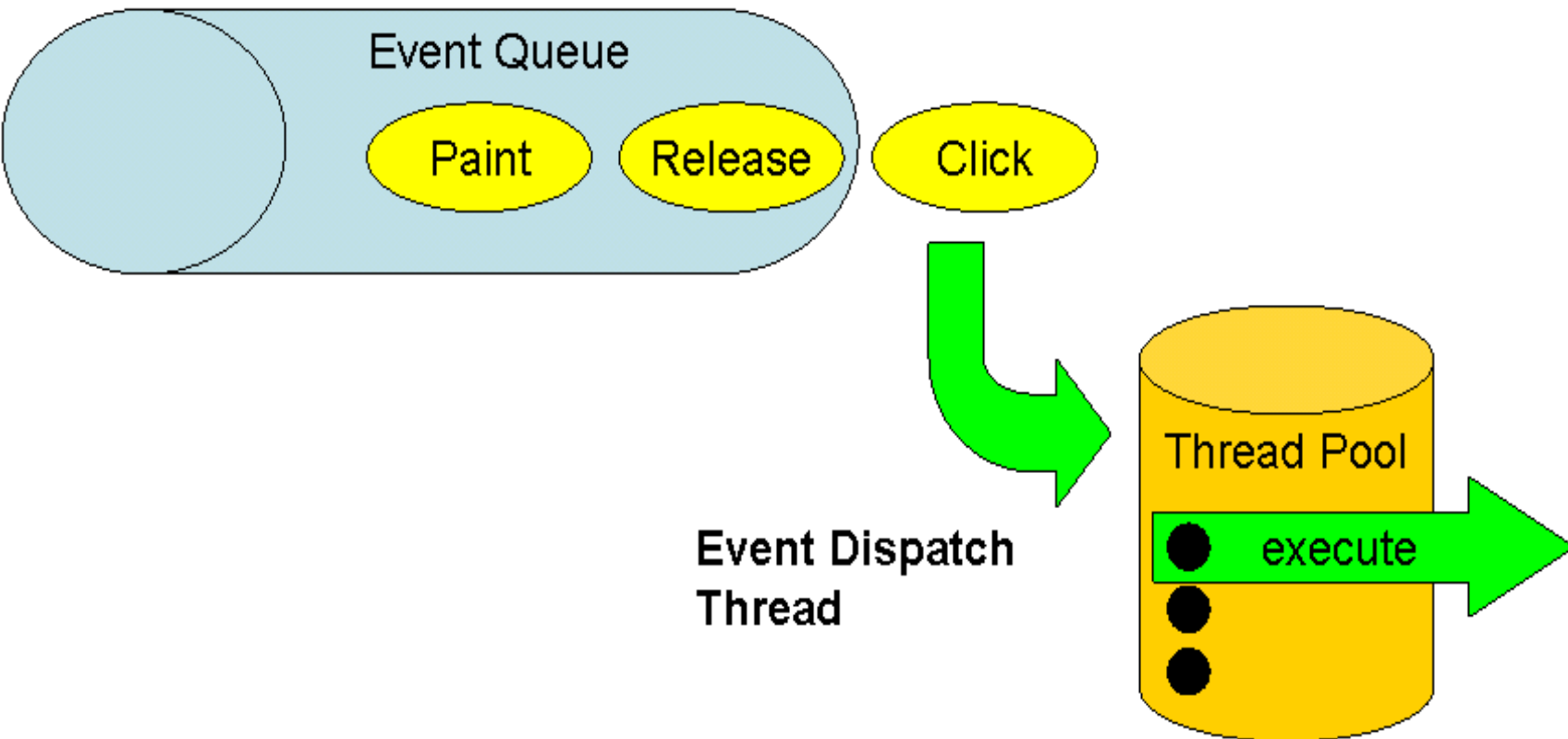


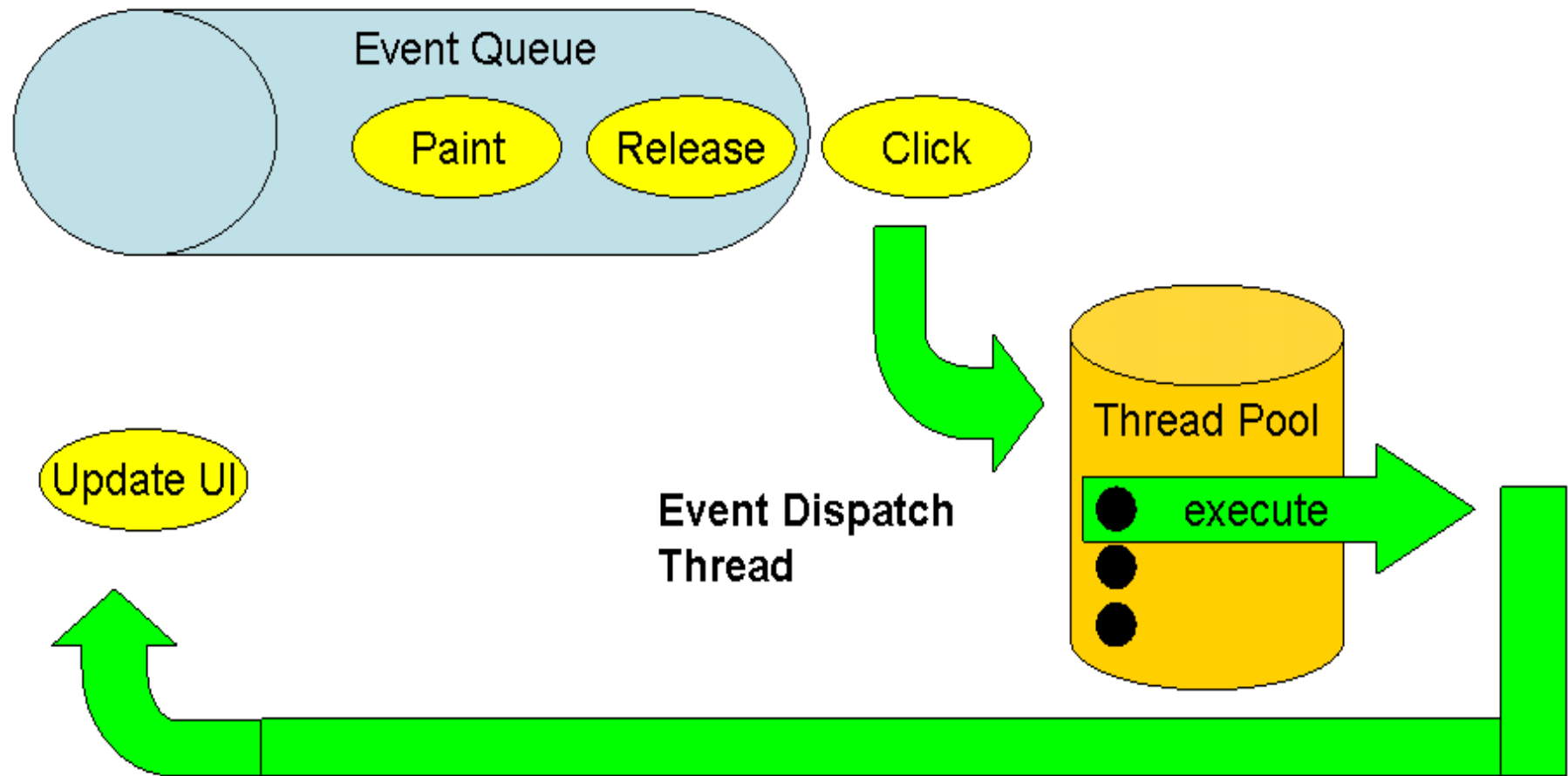
Dispatching an die Virtuelle Maschine





View.OnClickListener	Klick auf die View
View.OnFocusChangeListener	Fokus-Wechsel
View.OnHoverListener	Bewegung über die View
View.OnKeyListener	Drücken einer Taste
View.OnLongClickListener	Drücken und Halten der View
View.OnTouchListener	Berühren der View



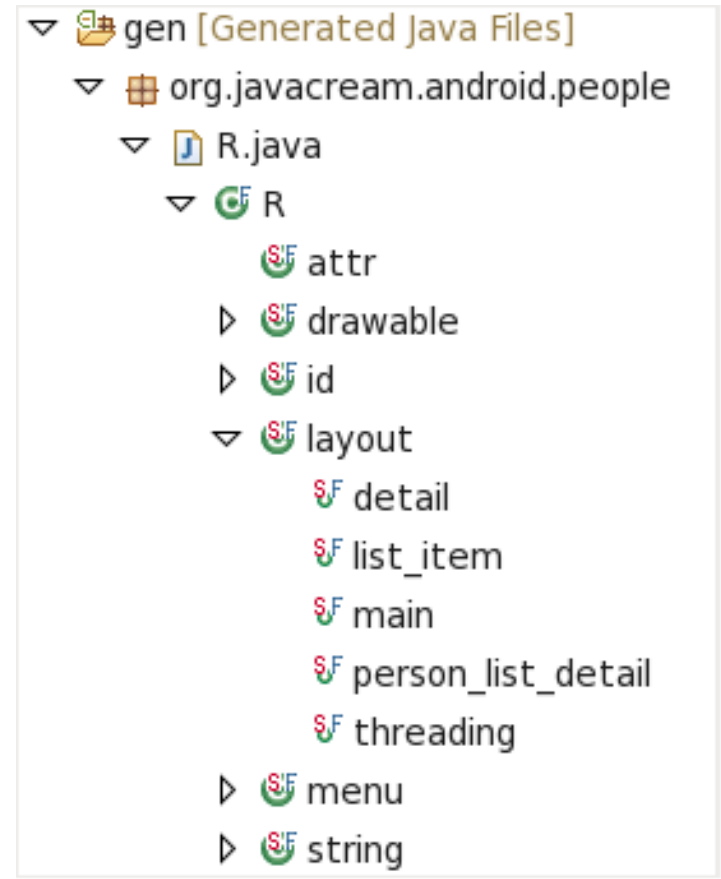
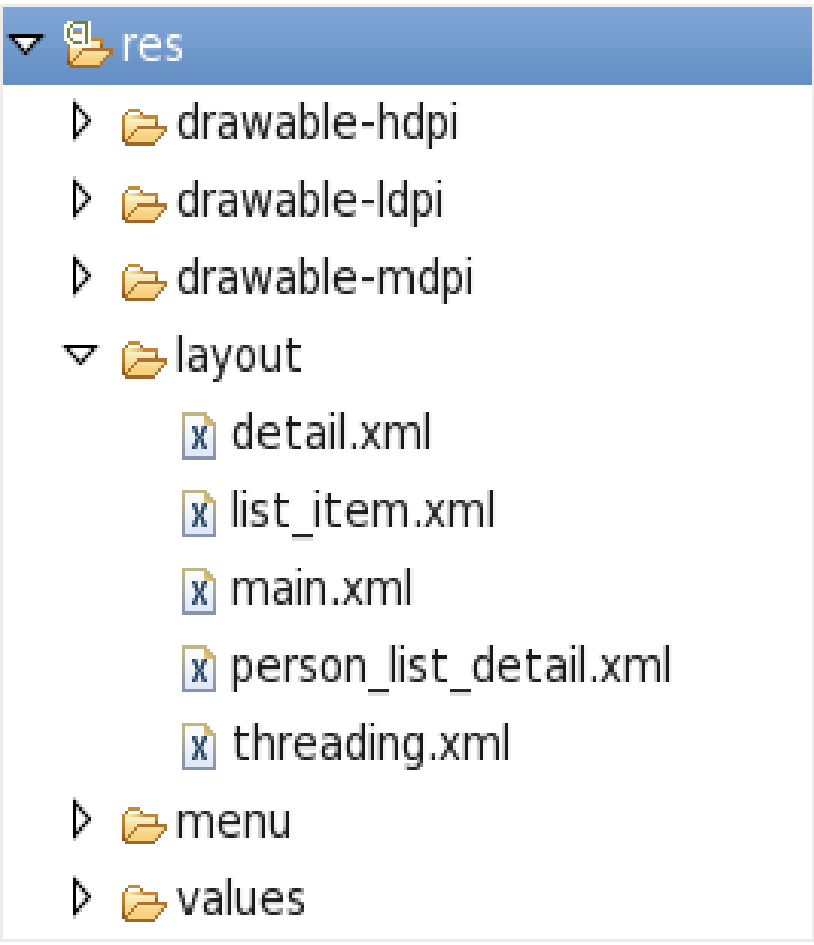


- `android.os.AsyncTask<Param, Process, Result>`
 - Durch ein mehr oder weniger simples Erben von dieser Klasse können Aktionen in den jeweils richtigen Threads ausgeführt werden:
 - `doInBackground` bekommt Parameter des angegebenen generischen Param-Typs und läuft in einem externen Thread
 - `onPostExecute` bekommt als Parameter den Result-Typ und läuft im Event Dispatch Thread

5.3

UI-RESSOURCEN

Der res-Ordner und die Klasse R



- Dafür stellt die Klasse Activity die Methode findViewById zur Verfügung
- Das Prefix @+id bewirkt, dass die generierte Klasse eine Enumeration Namens inputLastname bekommt, die zum Zugriff auf das Eingabeelement benutzt werden kann
 - Values
 - Auch Konstanten der Anwendung werden in XML-Dateien externalisiert und durch Enumerationen angesprochen
 - Diese Werte können sowohl innerhalb der Java-Anwendung als auch innerhalb einer Layout-Definition benutzt werden
 - In letzterem Falle wird wiederum ein spezielles Prefix, nämlich @string, benutzt

5.4

WEITERE THEMEN

- Komplexere Widgets wie Listen oder Auswahlfelder benötigen für ihre Darstellung komplexere Datenstrukturen
 - So könnte beispielsweise die Aufgabe darin bestehen, eine Personen-Informationen in einer Liste darzustellen.
 - Die Klasse Person selbst ist eine einfache JavaBean ohne Bezug zum Android-API
- Layout
 - Eine Liste benötigt eine eigene Layout-Definition
 - Diese wird wie üblich im Layout-Ordner gespeichert
- Adapter
 - Layout und Daten werden von einem Adapter gekoppelt
 - Dieser erbt vom Android-BaseAdapter und implementiert dessen abstrakte Methoden
- Zur Darstellung der Liste genügt nun eine einfache Instanziierung der obigen Implementierung und Verwendung innerhalb einer ListView

- Toast
 - Mit Hilfe der Toast-Klasse kann ein simpler Dialog eingeblendet werden, der auch ohne Benutzer-Interaktion nach Ablauf einer gewissen Zeit verschwindet
 - Die Länge der Darstellung wird dabei durch eine der statischen Konstanten LENGTH_SHORT und LENGTH_LONG bestimmt
- Statuszeile
 - Auch in die Statuszeile des Mobilgeräts können Meldungen integriert werden

- Für Standard-Dialoge stellt Android eine Reihe von Klassen zur Verfügung
 - AlertDialog
 - DatePickerDialog
 - ProgressDialog
 - TimePickerDialog

- Belegung der Menu-Taste
 - Die Belegung der Menu-Taste des Mobilgeräts erfolgt auf die mittlerweile gewohnte Weise
 - Ein Layout definiert die Menü-Befehle
 - R.java enthält die Identifier der Menü-Befehle
 - In der Activity wird in der Lifecycle-Methode onCreateOptionsMenu das Standard-Menü gesetzt
- Die Activity ist der zentrale Listener für alle Menu-Events
 - Dazu dient die Methode onOptionsItemSelected
 - Darin wird geprüft, welches Menü-Item gewählt wurde und welche Aktion daraufhin ausgeführt werden soll

- Widgets können Context-Menüs anbieten
 - Diese werden durch ein längeres Berühren aktiviert.
- Programmatisch müssen die Widgets, die ein Context-Menü erhalten sollen, bei der Activity registriert werden
 - Dazu dient die Methode `registerForContextMenu`, die als Parameter eine View erwartet
- Das eigentliche Menü wird wie üblich in einer Callback-Methode erzeugt, wahrscheinlich mit einem Layout, das über einen Inflater gelesen wird

- Ab Android API Level 11 steht auch eine Symbolleiste zur Verfügung
 - Diese stellt eine Auswahl der Menübefehle in einer Titelleiste zur Verfügung

6

RESSOURCEN-ZUGRIFF

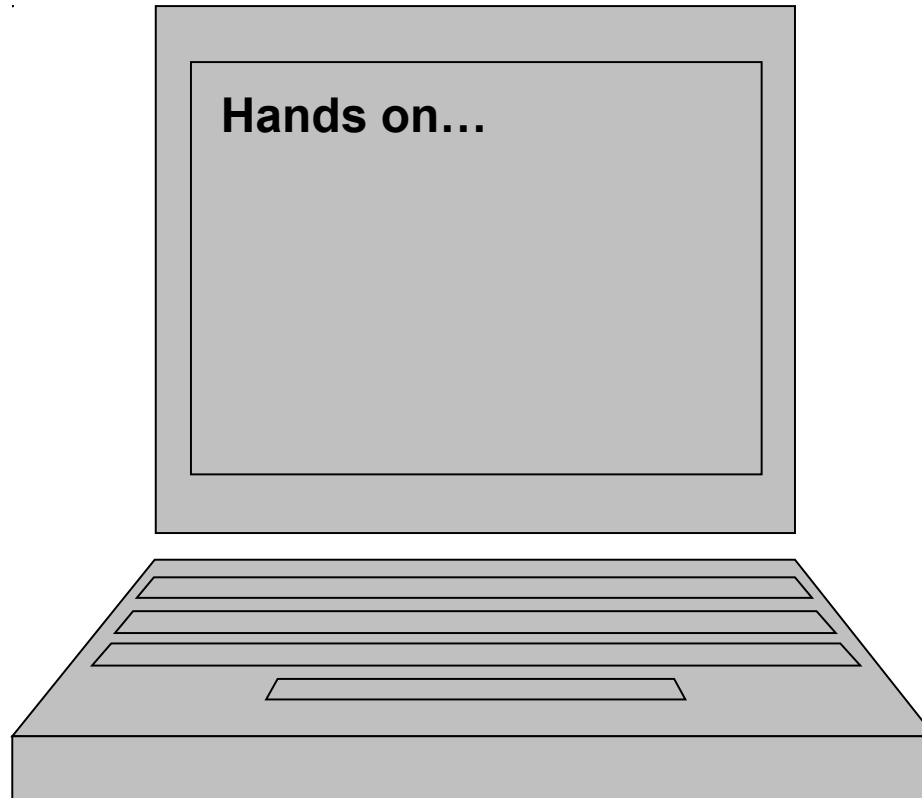
6.1

DATEISYSTEM

- Eine Ressource
 - Wird vom Betriebssystem und der Hardware verwaltet.
 - Der Applikation über einen „Handle“ zur Verfügung gestellt. Dieser wird von der Java Virtual Machine als Objekt zur Verfügung gestellt.
 - Ressourcen sind relativ teuer und beschränkt und müssen deshalb im Rahmen der Anwendung gesondert angefordert und, noch viel wichtiger: unbedingt auch wieder geschlossen werden.
- Ressourcen werden im Folgenden zum Senden und Empfangen von Daten benutzt
 - Shared Preferences. Dies sind flache typisierte Properties, die intern in Dateien abgelegt werden. Android übernimmt hierbei die gesamte Ressourcen-Verwaltung der File-Handles.
 - Ein internes Dateisystem. Dies ist Bestandteil des Mobilgeräts und kann zum Lesen und Schreiben beliebiger Dateien benutzt werden.
 - Falls installiert: Ein externes Dateisystem, beispielsweise eine Flash-Karte.
 - Eine komplette SQLite Datenbank.
 - http-basierte Client-Server-Programmierung.

- Beim Erstellen von Dateien können verschiedene Modi angegeben werden, die grob die Lese- und Schreibberechtigung kontrollieren können:
 - `MODE_PRIVATE`
 - der default, beschränkt den Zugriff auf die Applikation, die die Datei erzeugt hat
 - `MODE_WORLD_READABLE`
 - erlaubt den lesenden Zugriff für alle Applikationen
 - `MODE_WORLD_WRITABLE`
 - analog für den schreibenden Zugriff

- Die Android-Implementierung einer Java Virtual Machine stellt einen hervorragenden Debug- und Monitor Service zur Verfügung. Dieser wird über die DDMS-Perspektive in Eclipse visualisiert



- Das Mobilgerät stellt ein Dateisystem zur Verfügung
 - Mit welcher Technologie die „Festplatte“ realisiert wird ist für die Entwicklung von Android-Applikationen fast gleichgültig
 - Das Betriebssystem kapselt den Zugriff und stellt Java-Streams zur Verfügung
- Um eine Datei zu lesen oder zu schreiben werden Methoden der Activity-Klasse benutzt, um die Streams zu erhalten
- Das Dateisystem des Mobilgerätes wird als Bestandteil der DDMS-Perspektive dargestellt

6.2

DATENBANK

- Android beinhaltet eine SQLite-Datenbank
- Der Zugriff hierauf erfolgt durch die Methode `openOrCreateDatabase` der Activity
- Dabei werden wieder die Access-Modi berücksichtigt
- Mit Hilfe der Datenbank können SQL-Statements abgesetzt werden
- Dies erfolgt entweder
 - gekapselt über `execSQL`
 - oder mit einem Raw-Statement unter Verwendung eines Cursors

6.3

CLIENT-SERVER

- Zum Absenden eines http-Requests benutzt Android eine Open Source-Bibliothek der Apache-Group
- Die Erzeugung der Verbindung, das Setzen der Header und die Fehler-Behandlung werden dabei sehr schön gekapselt

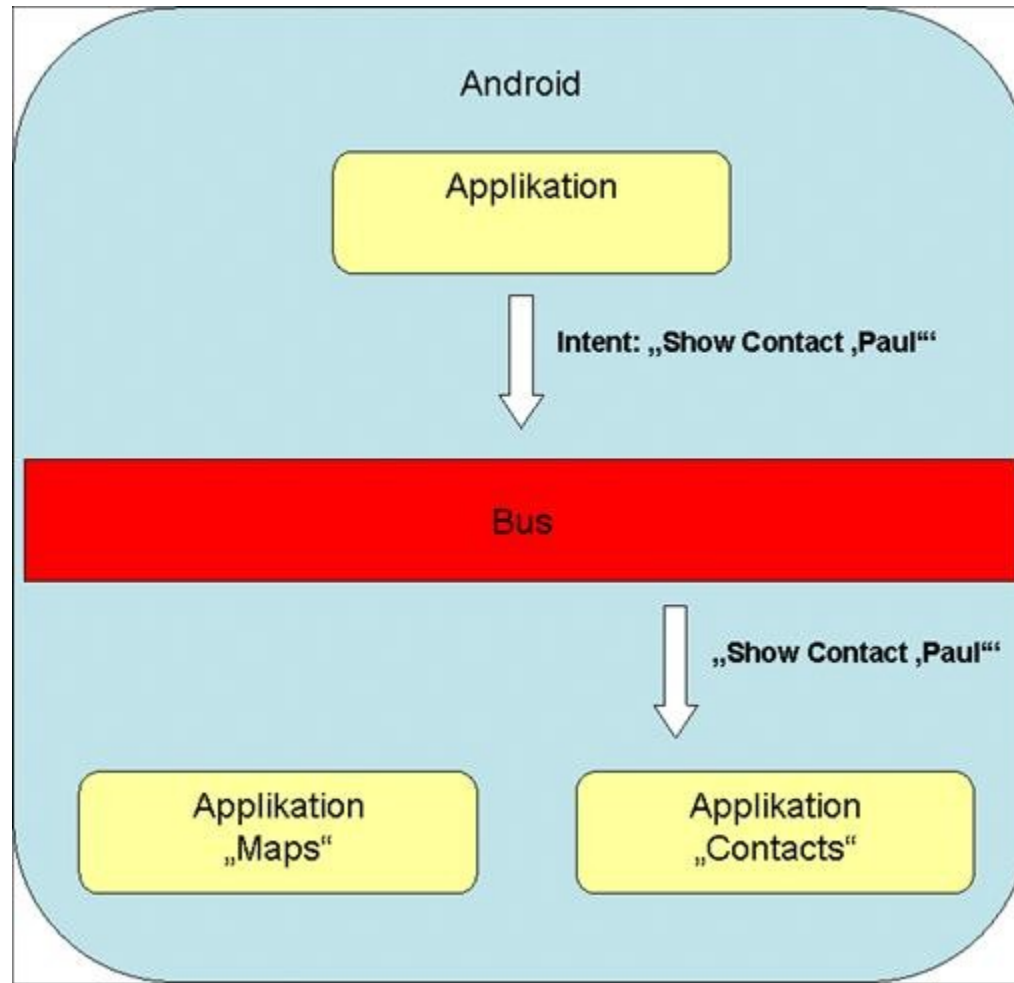
7

FORTGESCHRITTENE KONZEPTE

7.1

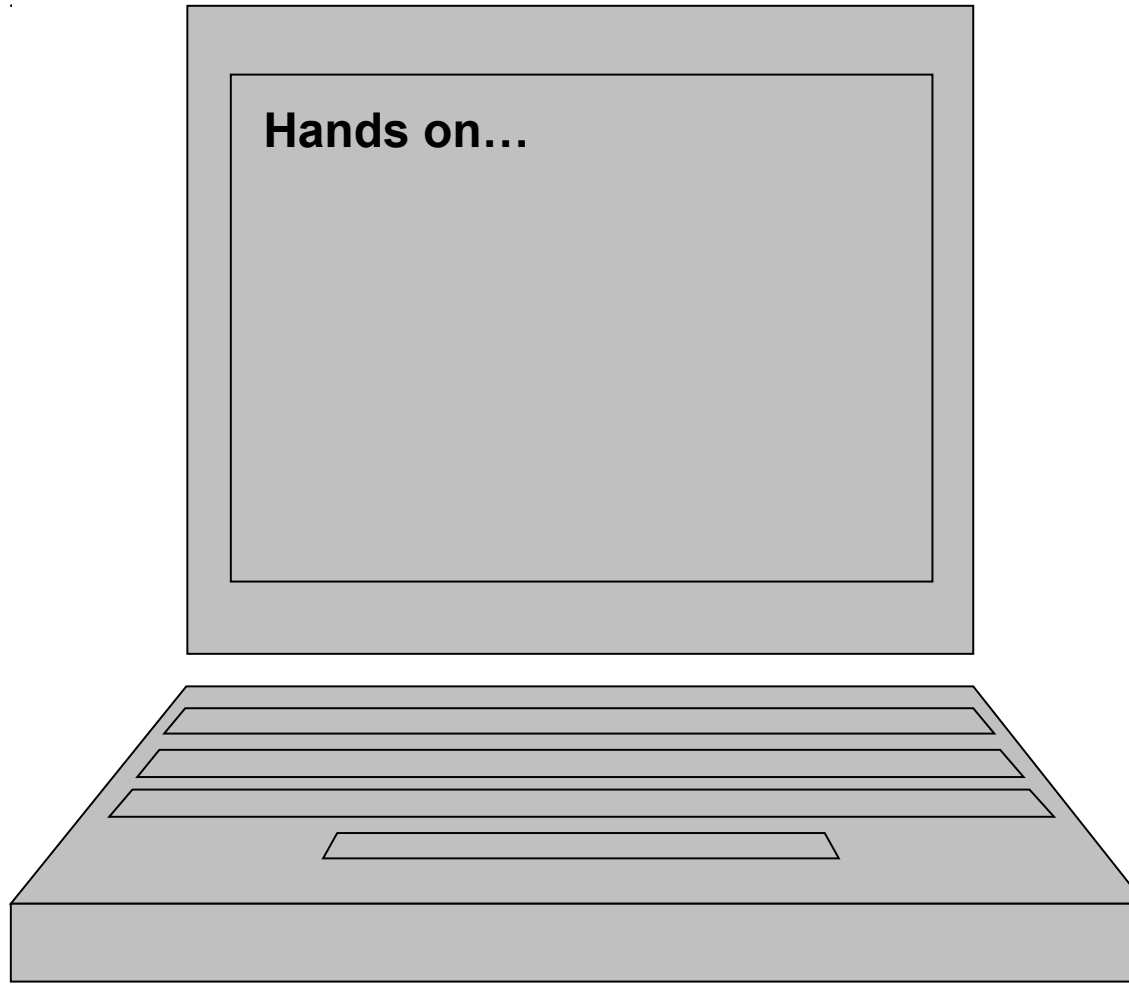
INTENTS

- Mit Hilfe von Intents wird eine Interaktion innerhalb einer Applikation aber auch zwischen Applikationen erreicht
- Jede Anwendung kann aktiv einen Intent signalisieren bzw. sich als Listener für Intents registrieren



- Ein Intent wird ist nichts anderes als eine Instanz vom Typ `android.content.Intent`
- Ein Intent benötigt in jedem Falle eine Ziel-Aktivität.
- Jede Aktivität kann einen so genannten Intent Filter hinterlegen
 - Dieser beruht
 - Auf einer beliebigen Zeichenkette bzw. einer statischen Konstante
 - Auf dem MIME-Type der durch das Intent transportierten Daten
- Das Intent kann Daten transportieren
 - Diese werden bei der Konstruktion oder durch den Aufruf der Methode `setData(android.net.Uri data)` gesetzt.

- Der eigentliche Aufruf eines Intents erfolgt über Methoden der Activity-Klasse.
 - `startActivity`
 - Eine andere Aktivität wird aufgerufen. Es wird kein Rückgabewert gemeldet.
 - `startActivityForResult`
 - Eine andere Aktivität wird aufgerufen. Neben dem Intent wird eine Request-Code mit übergeben
 - Beim Beenden der Ziel-Aktivität wird die Callback-Methode `protected void onActivityResult(int requestCode, int resultCode, Intent data)` mit dem Request-Code übergeben



7.2

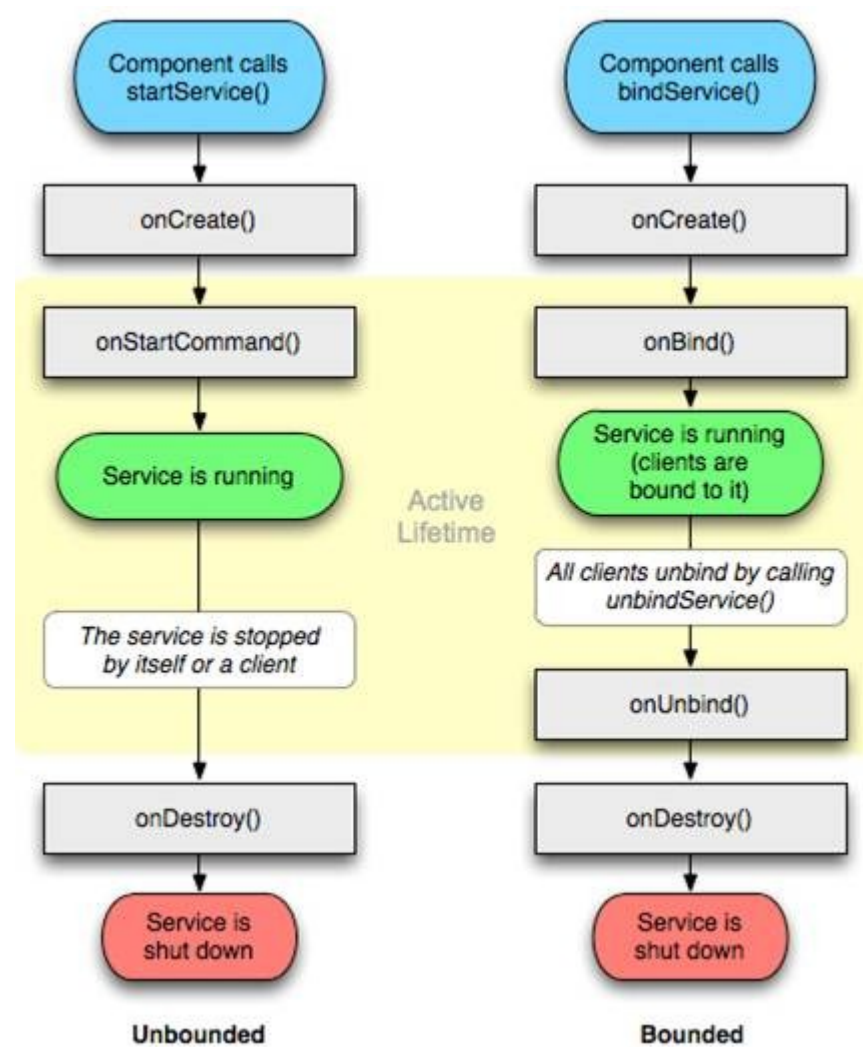
CONTENT PROVIDER

- Content Provider stellen Informationen Anwendungsübergreifend zur Verfügung
- Grob skizziert ergibt sich folgender Ablauf:
 - Daten werden vom Content Provider in Form einer tabellarischen Datenstruktur zur Verfügung gestellt.
 - Der Zugriff erfolgt über einen Cursor
 - Ein Content Provider erweitert die abstrakte Basisklasse `android.content.ContentResolver`.
 - Darin existiert die Methode `query`, die einen `android.database.Cursor` liefert
 - Jeder Content Provider kennt eine URI, die den eben bereits besprochenen Intent-URIs entspricht.

7.3

SERVICES

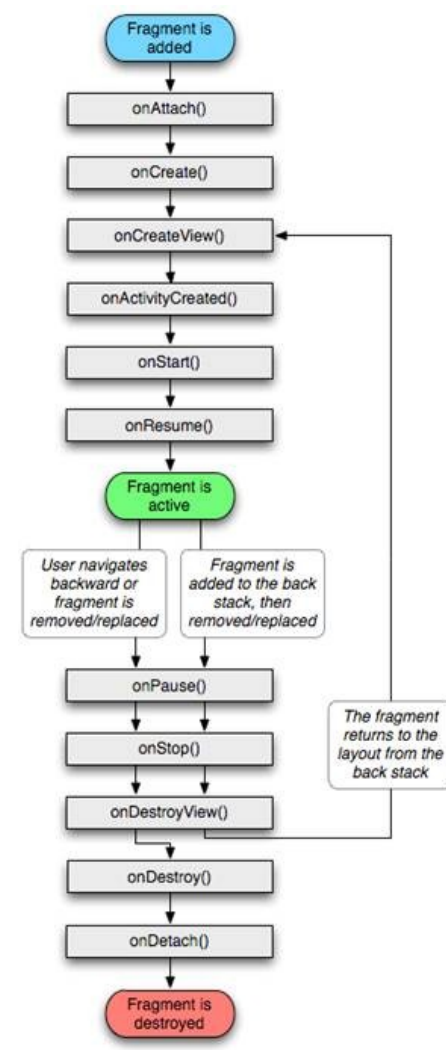
- Services sind, knapp gesprochen, Aktivitäten ohne visuelle Oberflächen. Sie werden durch einen Eintrag im Android-Manifest definiert und konfiguriert.
- Im Rahmen einer Anwendung übernehmen Services häufig die Aufgabe, lang-laufende Hintergrund-Prozesse abzubilden.
- Der Service wird, beispielsweise auf Grund einer Benutzer-Aktion in einer Aktivität, gestartet.
- Hat der Service seine Aufgabe erledigt wird dieser automatisch beendet.
- Eine andere Art von Services sind die gebundenen („bound“) Services.
 - Diese werden in der Service-Registry angemeldet
 - Auf diese Art und Weise sind auch System-Services realisiert

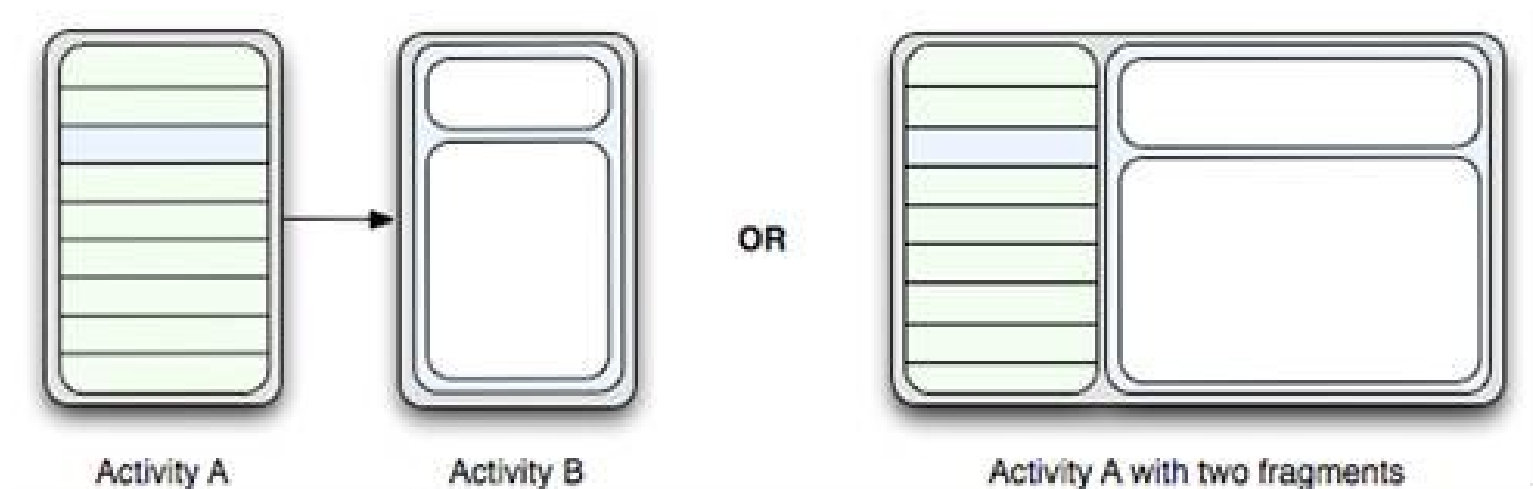


7.4

FRAGMENTE

- Ein relativ neues Konzept von Android, das jedoch auch für ältere Versionen portiert wurde, sind die Fragmente. Ein Fragment ist sehr ähnlich einer Activity
- Es hat einen eigenen Lifecycle, dieser ist an die Activity gekoppelt, an die das Fragment gebunden ist





7.5

WEITERE THEMEN

- Die Manipulation von Fragmenten wird über “Transaktionen” gesteuert
 - Damit kann die Anwendung relativ einfach einen mehrstufige Ablaufreihenfolge implementieren.

- Bei den folgenden Themen wird vorwiegend auf die Online-Android-Dokumentation verwiesen, da diese Themen erfahrungsgemäß noch recht viele Änderungen und Erweiterungen erfahren
 - <http://developer.android.com/guide/index.html>
- Übersicht
 - Location und GPS
 - Android Interface Definition Language
 - Web Applications
 - Search
 - Grafik und Animation