LMU

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

FAKULTÄT FÜR MATHEMATIK, INFORMATIK UND STATISTIK
INSTITUT FÜR INFORMATIK
ARBEITSGRUPPEN MEDIENINFORMATIK UND
MENSCH-MASCHINE-INTERAKTION

Praktikum Entwicklung Mediensysteme (für Master)

An Introduction to Android

An Introduction to Android

- What is Android?
- Installation
- Getting Started
- Anatomy of an Android Application
- Life Cycle of an Android Application

What is Android?

- Released in Nov. 2007 – rumored to be some kind of GPhone
- Open, free mobile platform with a complete software stack
  - Operating system
  - Middleware
  - Key mobile applications
- Developed by the Open Handset Alliance
- Built on the open Linux kernel
- Custom Dalvik virtual machine for mobile environments
- Applications written in Java
- Open source; Apache v2 open source license
- Applications can access all core functionalities of a mobile device
- No differentiation between core and 3rd party applications
- Can be extended to incorporate new technologies

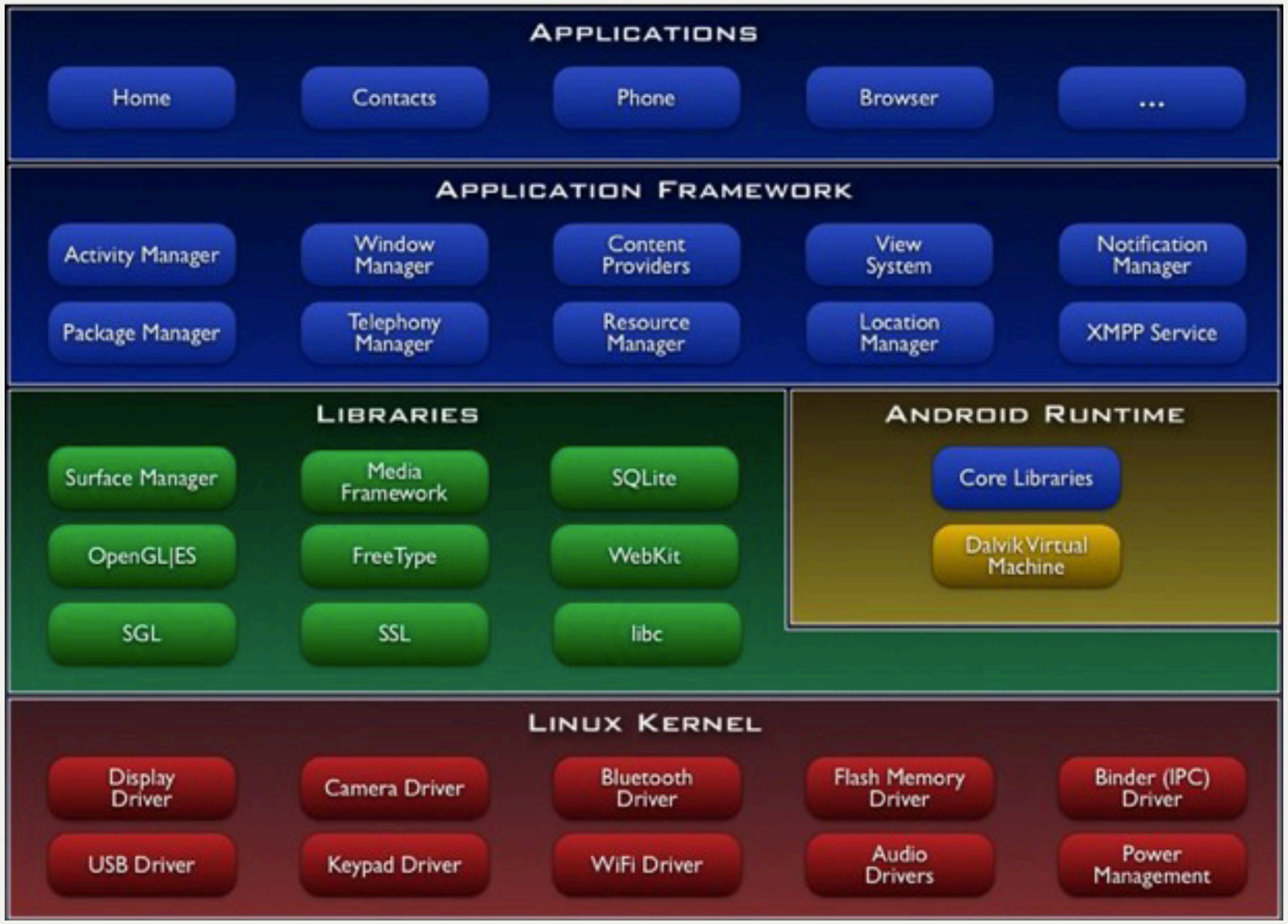
Open Handset Alliance

- Group of more than 30 technology and mobile companies led by Google
  - Mobile Operators, e.g. China Mobile, KDDI, NTT DoCoMo, TMobile,
  - Sprint Nextelk, Telefonica
  - Semiconductor Companies, e.g. Broadcom, Intel, Nvidia, Qualcomm, SiRF, Texas Instruments
  - Handset Manufactureres, e.g. HTC, LG, Motorola, Samsung
  - Software Companies, e.g. eBay, Google,
- Goal: „to accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience “
- Android as the first project towards an open and free mobile experience, but also commercial deployment
- URL: www.openhandsetalliance.com

Android Framework

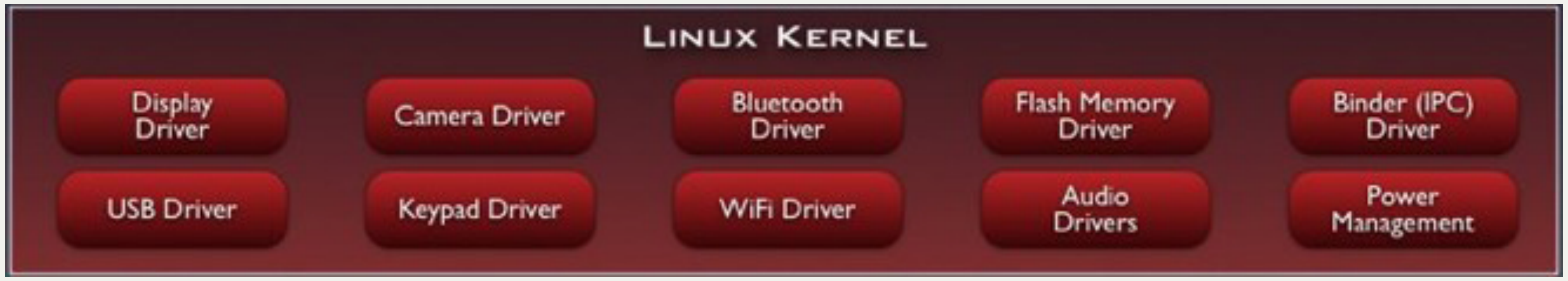
- **Application framework** enabling reuse and replacement of components
- **Dalvik virtual machine** optimized for mobile devices (register based)
- **Integrated browser** based on the open source WebKit engine
- **Optimized graphics** powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
- **SQLite** for structured data storage
- **Media support** for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- **GSM Telephony** (hardware dependent)
- **Bluetooth, EDGE, 3G, and WiFi** (hardware dependent)
- **Camera, GPS, compass, and accelerometer** (hardware dependent)
- **Rich development environment** including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE

Android Architecture



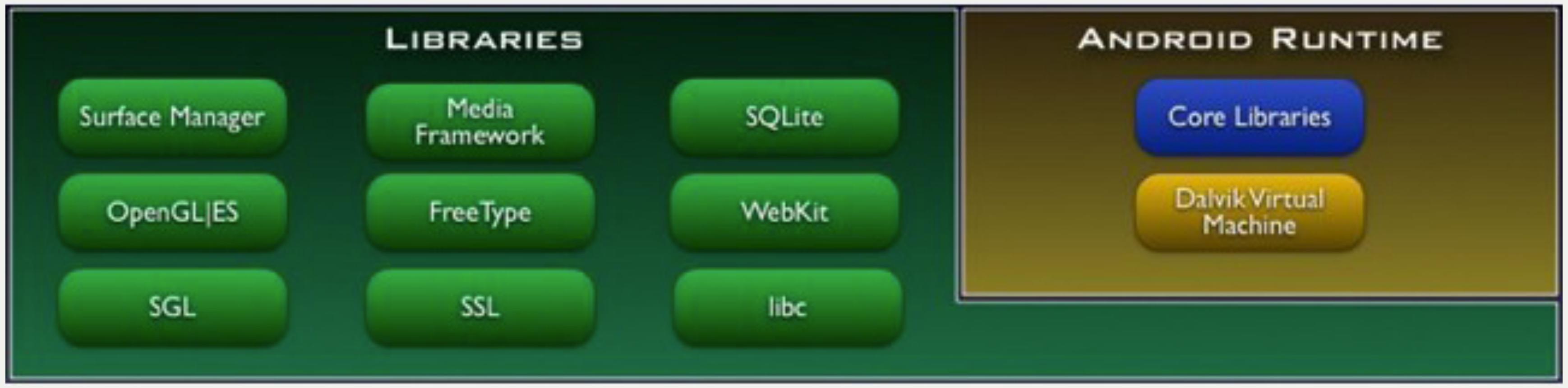
Linux Kernel

- Linux kernel version 2.6
- Abstraction layer between hardware and the software stack
- Core services
  - Security
  - Memory management
  - Process management
  - Network stack
  - Driver model



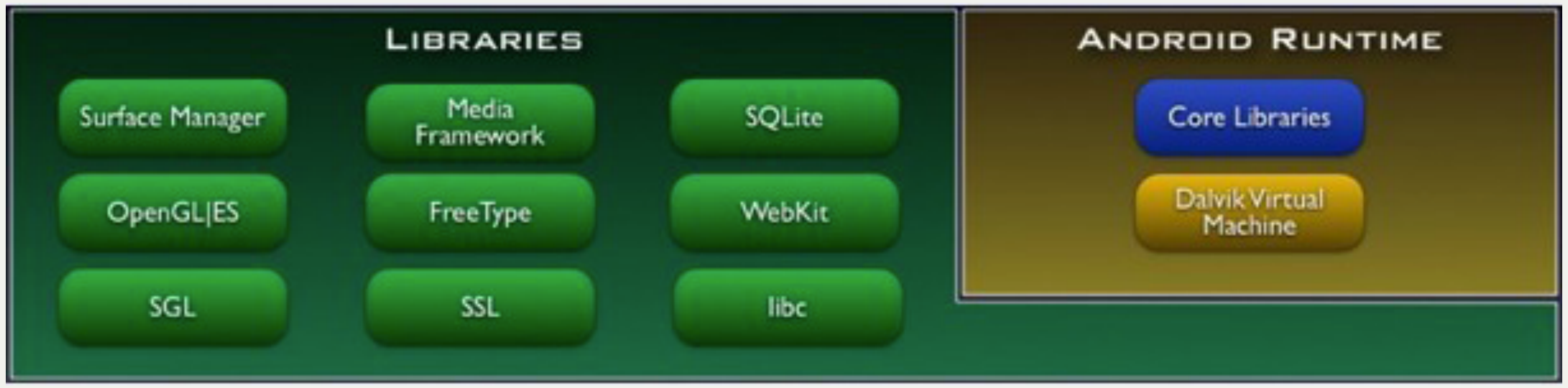
Libraries

- C/C++ libraries used by various Android components
- Developers can use their capabilities through the application framework
- Includes:
  - Media Libraries: includes MPEG4, H.264, MP3, JPG, PNG,
  - WebKit/LibWebCore: web browser engine
  - SQLite: relational database engine
  - Libraries/engines for 2D and 3D graphics



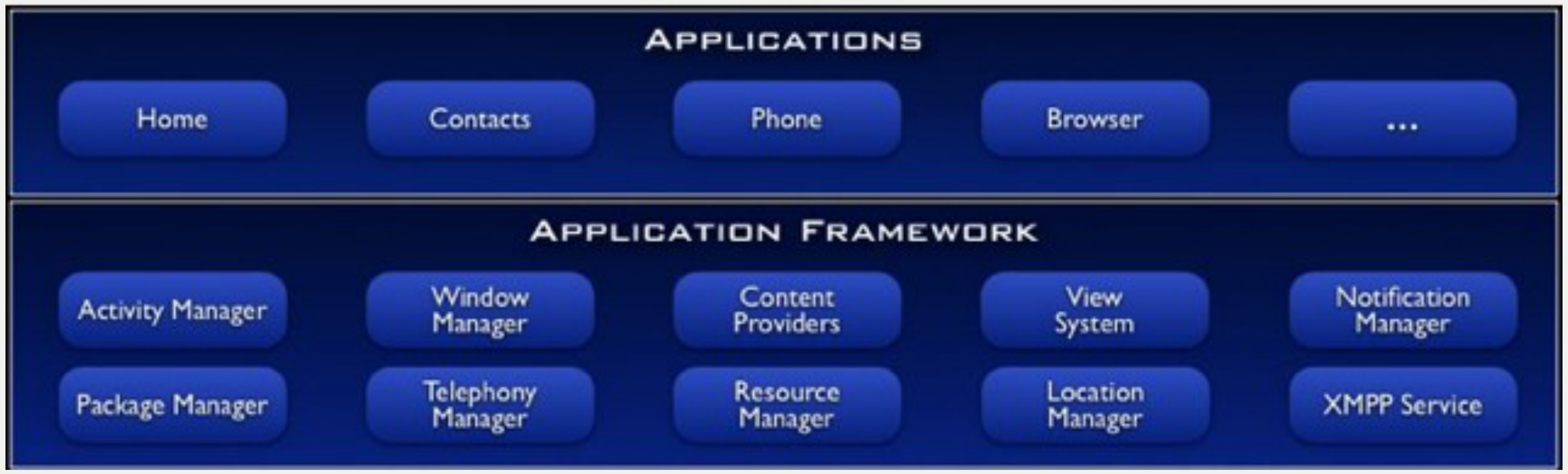
Android Runtime

- Core libraries provide Java functionalities
- Dalvik virtual machine relies on Linux kernel for e.g. threading or low-level memory management
- Devices can run multiple Dalvik VMs, every Android application runs with its own instance of Dalvik VM
- VM executes optimized Dalvik Executable files (.dex)
- Dx-tool transforms compiled Java-files into dex-files



Applications /Application

- Core applications, e.g. contacts, mail, phone, browser, calender, maps, ...
- Full access to all framework APIs for core applications
- Simplified reuse of components
- Applications written in Java



Core Android Packages

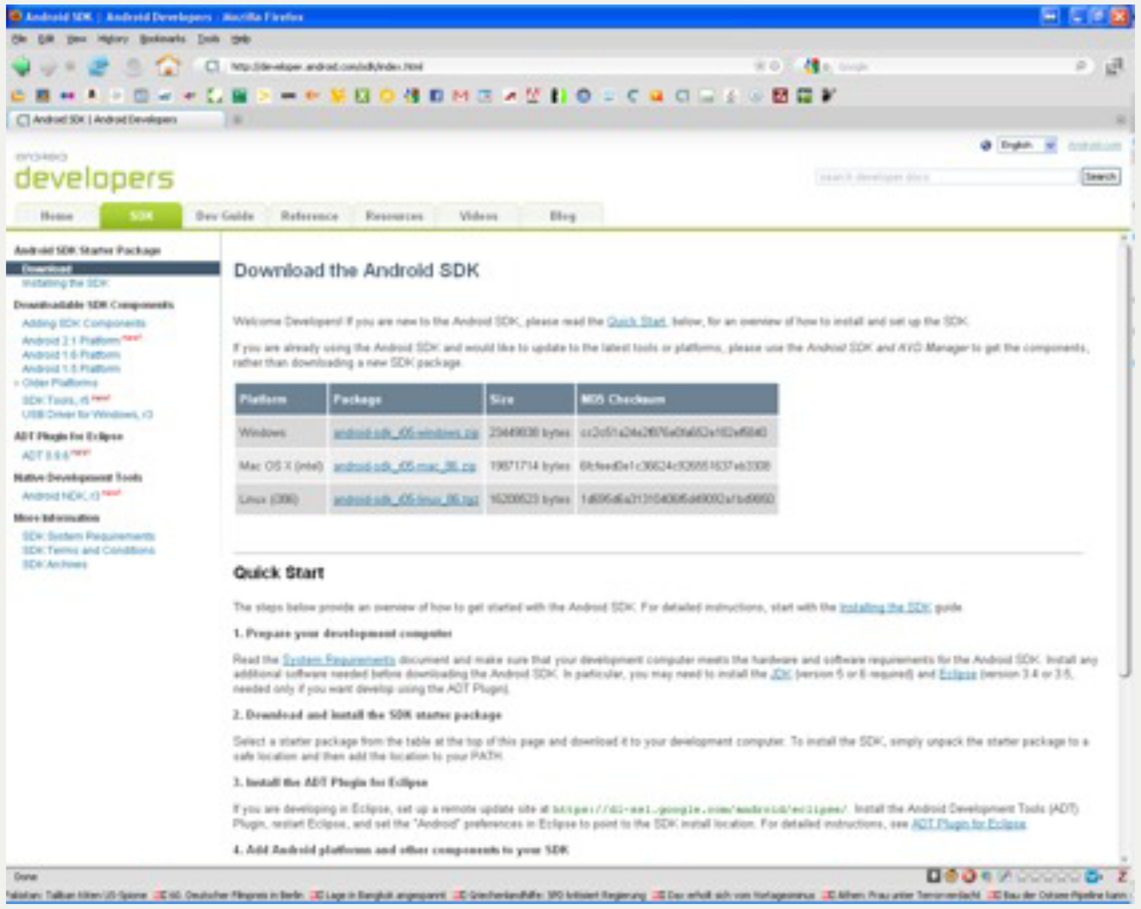
- **android.util:** contains various low-level utility classes, such as specialized container classes, XML utilities, etc.
- **android.os:** provides basic operating system services, message passing, and inter-process communication.
- **android.graphics:** is the core rendering package.
- **android.text, android.text.method, android.text.style, and android.text.util:** supply a rich set of text processing tools, supporting rich text, input methods, etc.
- **android.database:** contains low-level APIs for working with databases.
- **android.content:** provides various services for accessing data on the device: applications installed on the device and their associated resources, and content providers for persistent dynamic data.
- **android.view:** is the core user-interface framework.
- **android.widget:** supplies standard user interface elements (lists, buttons, layout managers, etc) built from the view package.
- **android.app:** provides the high-level application model, implemented using Activities.

Android Version History

Version	Features
1.5 Cupcake	**30.04.2009:** Onscreen-Keyboard with „Autocomplete“, Screen switch Animations, Videoupload
1.6 Donut	**15.09.2009:** Screenshots on the android market, Voice Search, WVGA resolutions
2.0/2.1 Eclair	**12.01.2010:** Speed improvements, More screen resolutions (dip), Camera flash support, Live wallpapers, Multitouch support
2.2. Froyo	**20.05.2010:** Speed and performance increase, Flash 10.1 support, Installing apps on SD-Card, Tethering
2.3 Gingerbread	**23.02.2011:** Dual-Core-Unterstützung, NFC, HTML5, bessere Garbage Collection
3.0 Honeycomb	Tablet Optimized
3.1 Ice Cream Sandwich	2.x und 3.x zu einer Version und Google TV

Installing SDK

- Please follow instructions from the Android doc
- Download and install the Android SDK
- SDK includes documentation, tools and examples
- Set up your IDE; Eclipse (Java EE) recommended
- Install Eclipse Android Development Tools
- (ADT) plugin, connect it with the Android
- SDK and Download your Platforms



Download the Android SDK

Welcome Developers! If you are new to the Android SDK, please read the Quick Start, below, for an overview of how to install and set up the SDK.

If you are already using the Android SDK and would like to update to the latest tools or platforms, please use the *Android SDK and AVD Manager* to get the components, rather than downloading a new SDK package.

Platform	Package	Size	MD5 Checksum
Windows	android-sdk_r05-windows.zip	23449838 bytes	[illegible]
Mac OS X (intel)	android-sdk_r05-mac_86.zip	19871714 bytes	[illegible]
Linux (i386)	android-sdk_r05-linux_86.tgz	16208523 bytes	[illegible]

Quick Start

The steps below provide an overview of how to get started with the Android SDK. For detailed instructions, start with the Installing the SDK guide.

1. Prepare your development computer

Read the System Requirements document and make sure that your development computer meets the hardware and software requirements for the Android SDK. Install any additional software needed before downloading the Android SDK. In particular, you may need to install the JDK (version 5 or 6 required) and Eclipse (version 3.4 or 3.5, needed only if you want develop using the ADT Plugin).

2. Download and install the SDK starter package

Select a starter package from the table at the top of this page and download it to your development computer. To install the SDK, simply unpack the starter package to a safe location and then add the location to your PATH.

3. Install the ADT Plugin for Eclipse

If you are developing in Eclipse, set up a remote update site at `https://dl-ssl.google.com/android/eclipse/`. Install the Android Development Tools (ADT) Plugin, restart Eclipse, and set the "Android" preferences in Eclipse to point to the SDK install location. For detailed instructions, see ADT Plugin for Eclipse.

4. Add Android platforms and other components to your SDK

http://developer.android.com/sdk/index.html

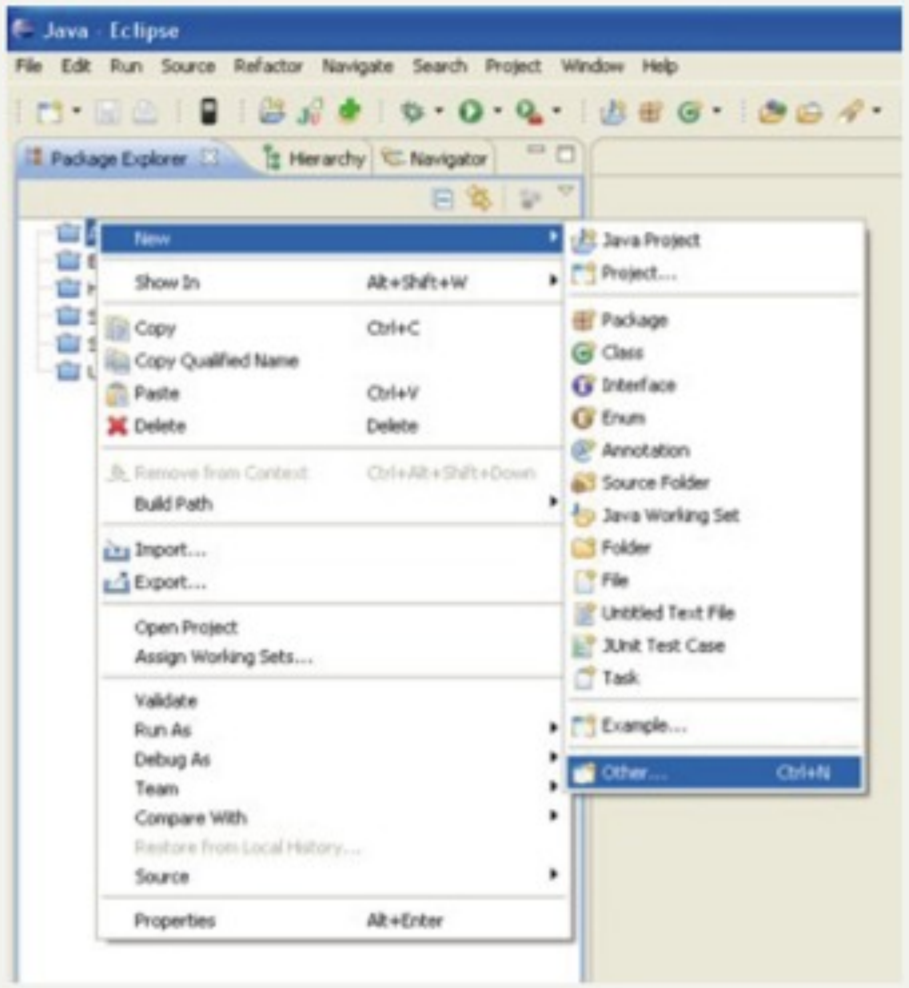
Installing SDK 2

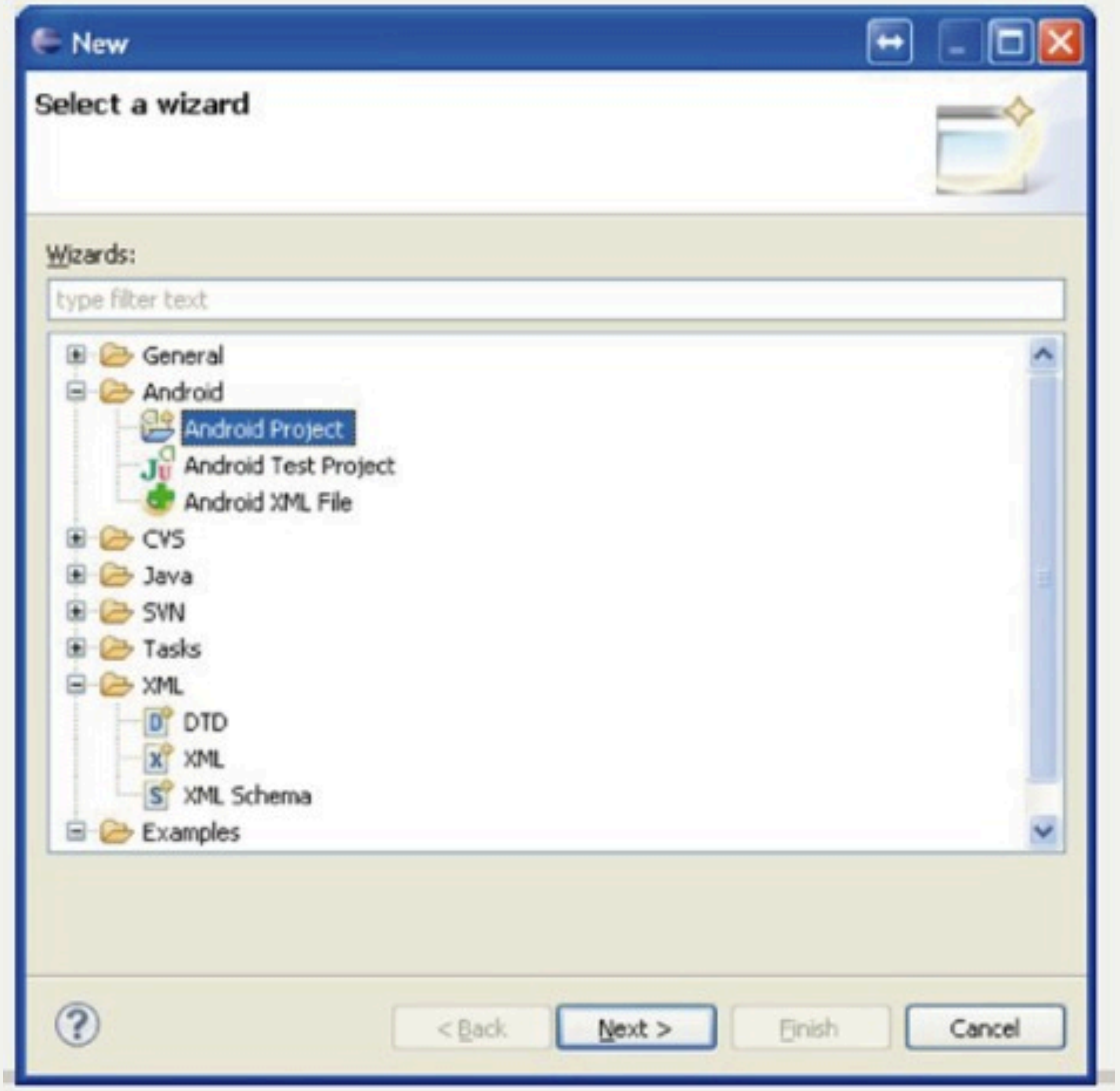
- Create an Android project
  - Standard Eclipse procedure
  - Automatically creates folders and a Manifest file
  - Can also be used to create a demo project
- Set up a launch configuration
  - Run application from menu or
  - Define settings for run configuration (project, activity, emulator options, …) from Run > Open Run Dialog >
- Run Android application in emulator
  - Be Patient! The emulator takes while to boot up.
  - Keep it open once it was started!

The Nexus One

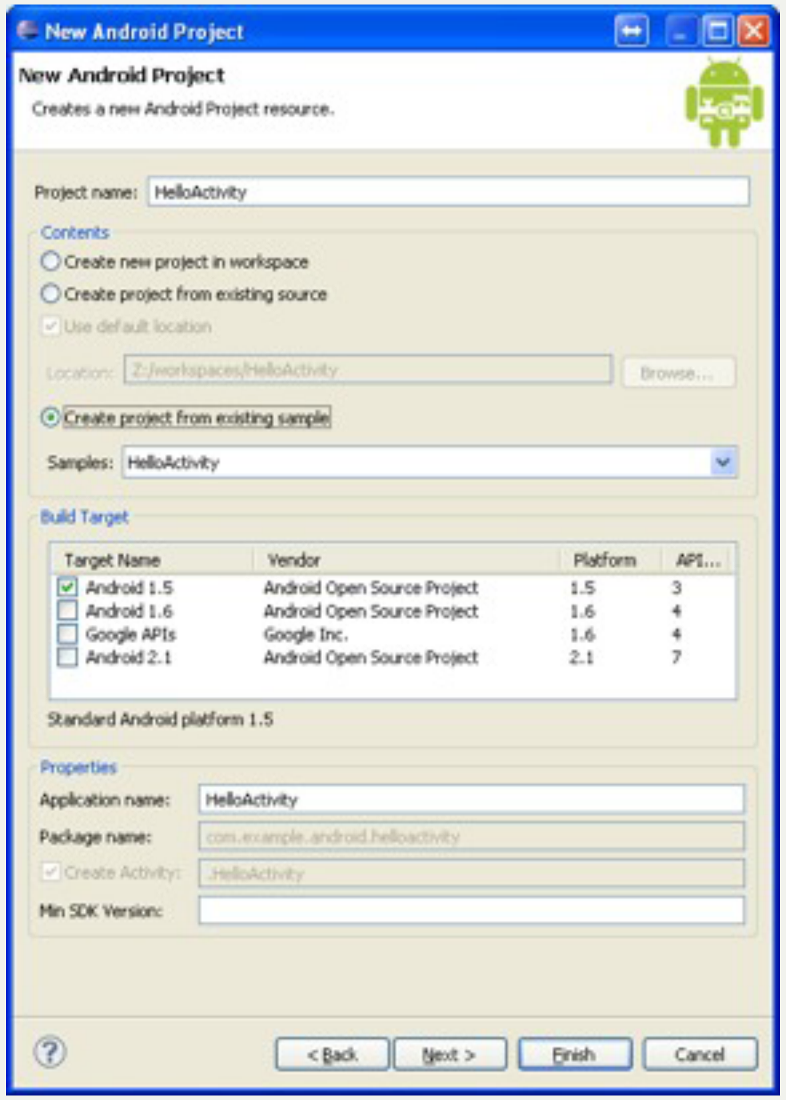
nexus one™

Hello Android





Hello Android



New Android Project

New Android Project

Creates a new Android Project resource.

Project name: HelloActivity

Contents

- ○ Create new project in workspace
- ○ Create project from existing source
- ☑ Use default location
- Location: Z:/workspaces/HelloActivity — Browse...
- ◉ Create project from existing sample
- Samples: HelloActivity

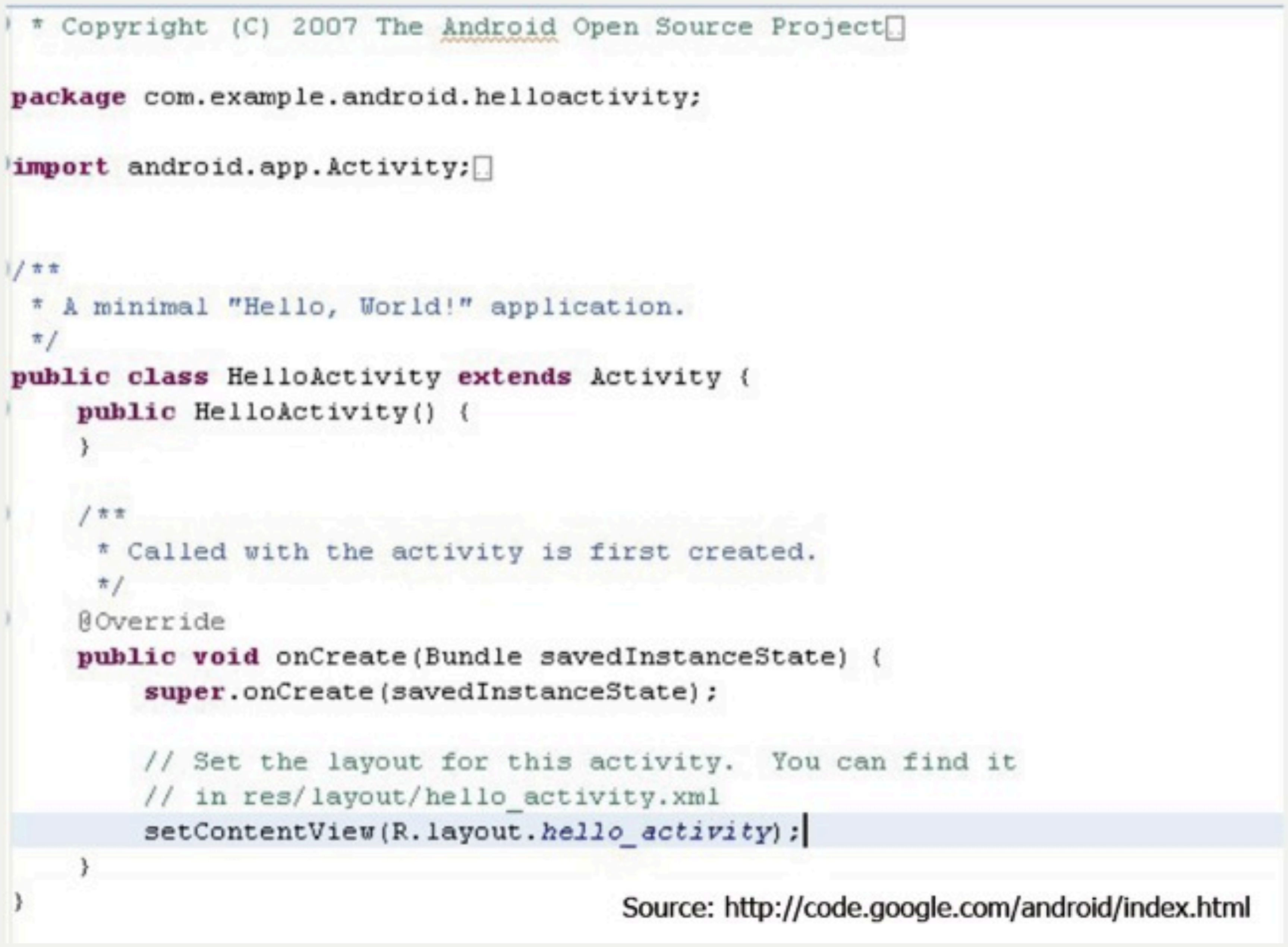
Build Target

	Target Name	Vendor	Platform	API...
☑	Android 1.5	Android Open Source Project	1.5	3
☐	Android 1.6	Android Open Source Project	1.6	4
☐	Google APIs	Google Inc.	1.6	4
☐	Android 2.1	Android Open Source Project	2.1	7

Standard Android platform 1.5

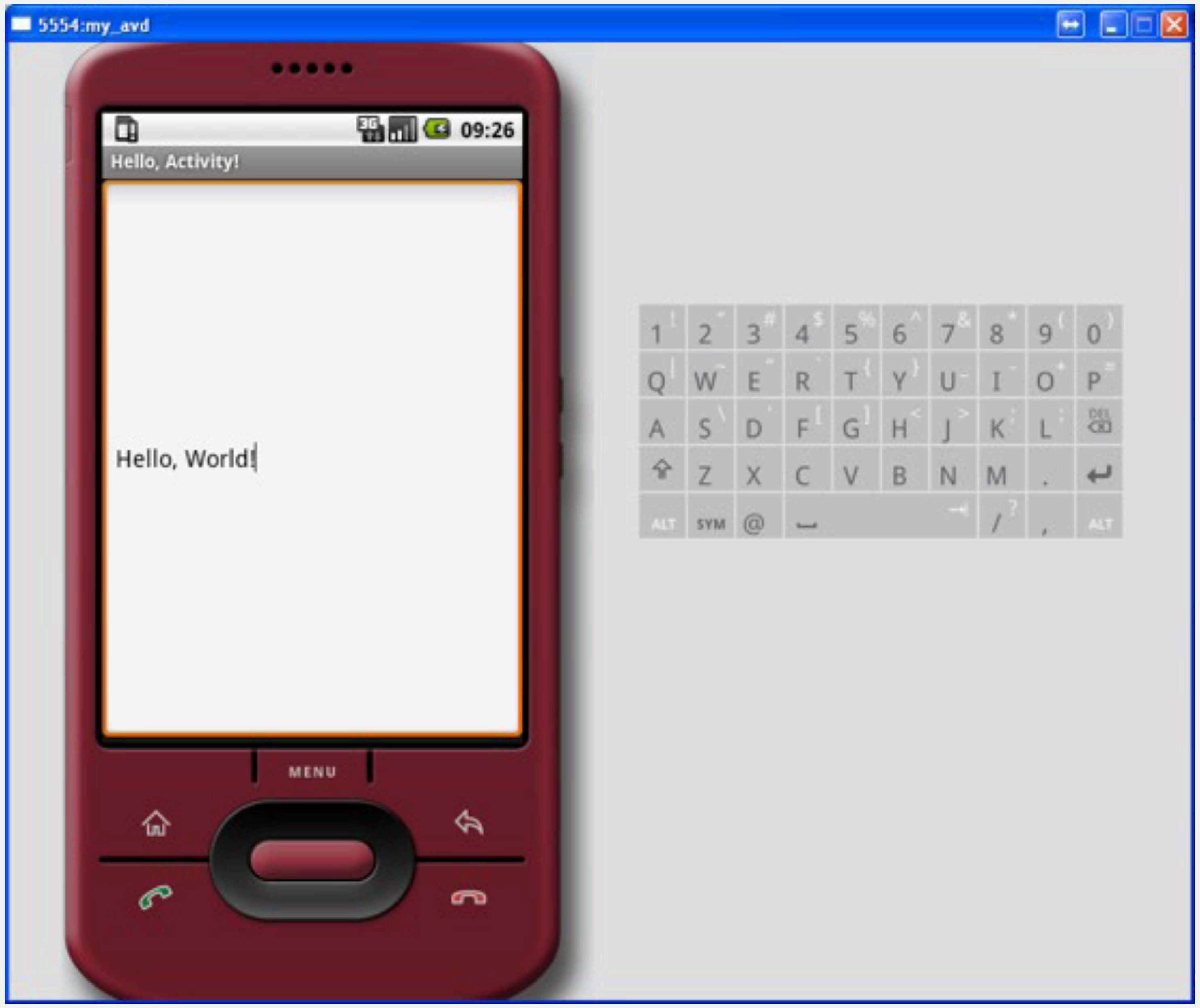
Properties

Application name:	HelloActivity
Package name:	com.example.android.helloactivity
☑ Create Activity:	.HelloActivity
Min SDK Version:	



```
 * Copyright (C) 2007 The Android Open Source Project

package com.example.android.helloactivity;

import android.app.Activity;


/**
 * A minimal "Hello, World!" application.
 */
public class HelloActivity extends Activity {
    public HelloActivity() {
    }

    /**
     * Called with the activity is first created.
     */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Set the layout for this activity.  You can find it
        // in res/layout/hello_activity.xml
        setContentView(R.layout.hello_activity);
    }
}
```

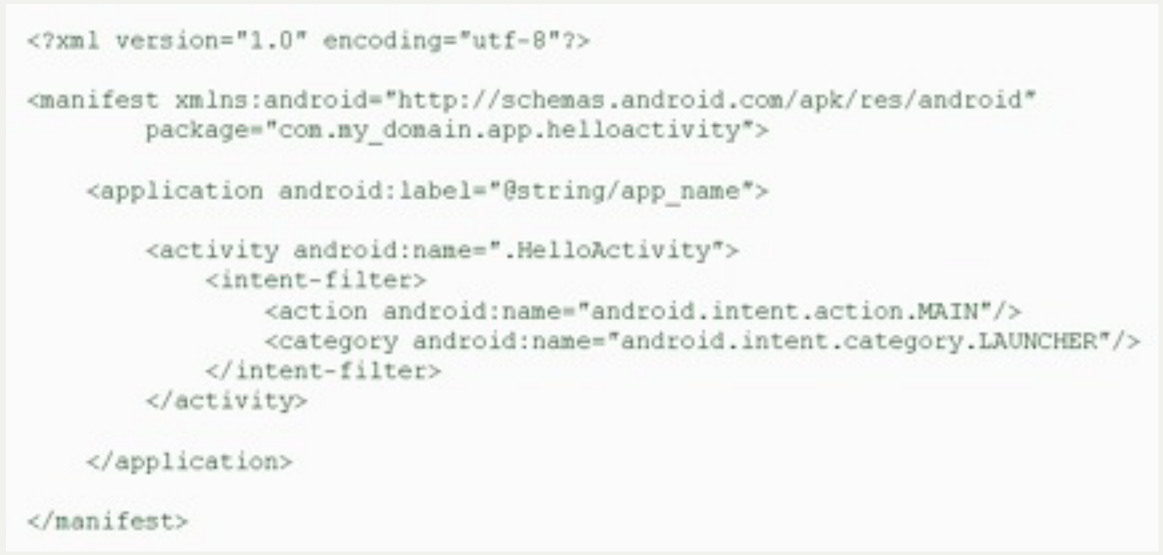
Source: http://code.google.com/android/index.html



Hello, World!

Anatomy of an Android Application

- 4 main building blocks for Android applications
  - Activity
  - Intent Receiver
  - Service
  - Content Provider

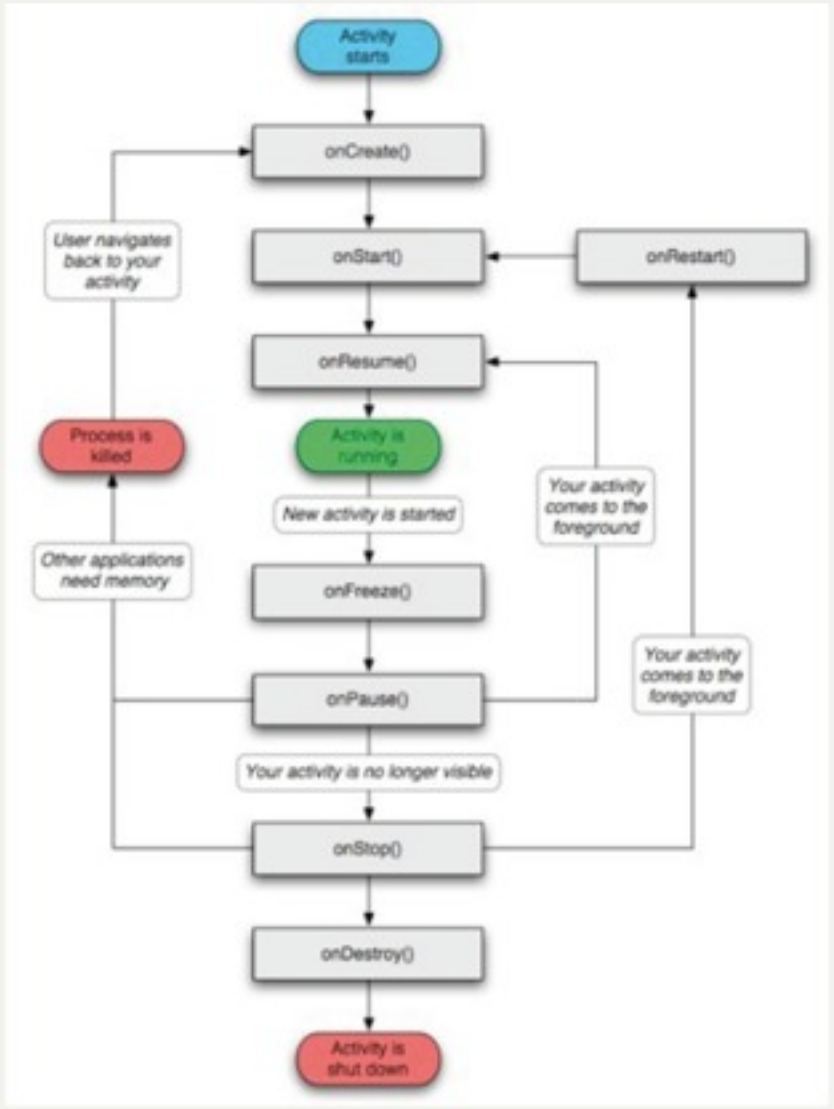


```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.my_domain.app.helloactivity">

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

        <activity android:name=".HelloActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>

    </application>

</manifest>
```

- AndroidManifest.xml lists all components of an application, their capabilities and requirements

Activity

- Single, focused thing or task
- Extends the Activity base class
- Refers to a single screen in a (multi-screen) application
- Displays a UI, interacts with user, responds to events
- Different main methods:
  - onCreate(Bundle): initialization of activity, set UI, …
  - onPause(): leaving an activity
- Moving through screens by starting other activities
- Activities managed by activity stack
- New activity put on top of the stack
- 4 states: active/running, paused, stopped, killed/ shutdown


Activity starts
onCreate()
User navigates back to your activity
onStart()
onRestart()
onResume()
Process is killed
Activity is running
Your activity comes to the foreground
New activity is started
Other applications need memory
onFreeze()
Your activity comes to the foreground
onPause()
Your activity is no longer visible
onStop()
onDestroy()
Activity is shut down

Intents and Intent Filters

- Intent
  - Abstract description of an operation/action to be performed
  - Mostly used for launching activities; "glue between activities"
  - Action: general action to be performed, e.g. VIEW_ACTION, EDIT_ACTION, MAIN_ACTION, ...
  - Data: data to operate on, expressed as a URI
  - Example: VIEW_ACTION content://contacts/1
- Intent Filter
  - Describes what Intents an activity can handle
  - Activities publish Intent Filters describing their capabilities/ how they can handle certain Intents and their actions
  - Navigating between screens is accomplished by resolving Intents => system matches Intents and Intent Filters
  - Activity calls method startActivity(myIntent)

Intent Receiver, Service, Content Provider

- Intent Receiver
  - Used to execute code upon an external event, e.g. phone rings
  - Usually no UI; may use the NotificationManager
- Service
  - Application component running in the background
  - Runs indefinitely, no UI, no interaction with user
  - E.g. media player
- Content Provider
  - Used to share data with other applications

Life Cycle of an Android Application

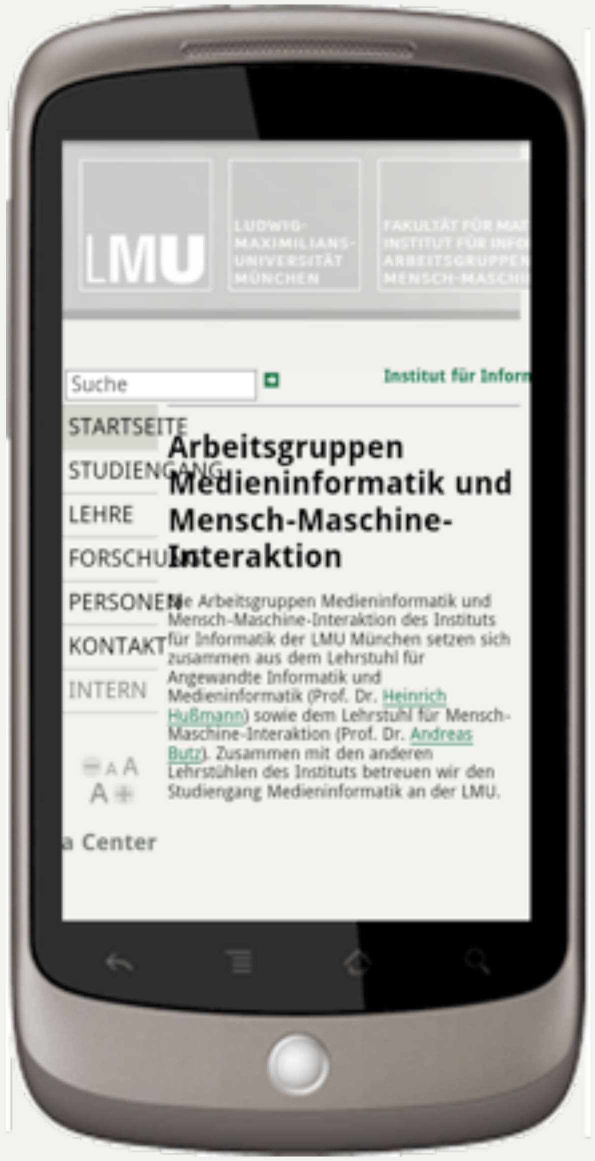
- Each Android application runs in its own Linux process
  - Process's lifetime not directly controlled by application
  - Determined by the system, depending on running applications, their importance, available memory
- Components (Activity, Service, Intent Receiver) impact the lifetime of the application's process
- Importance hierarchy for killing processes based on
  - Components running in them
  - The state of these components

Android's Importance Hierarchy

- Foreground Process
  - Required for current user activities
  - E.g. running an Activity at the top of the screen
- Visible Process
  - Activity is visible but not in the foreground (onPause())
  - E.g. previous activity displayed behind a foreground dialog
- Service Process
  - Holds a Service, not directly visibleE.g. media player, network up/download
- Background Process
  - Holds an Activity that is currently not visible (onStop())
  - Can be killed at any time to reclaim memory
- Empty Process
  - Holds no active application components

Exercise 1

- Create a first Andoird Activity
  - It should incorporate a mini web browser
  - Always load URL: www.medien.ifi.lmu.de
  - Make it fullscreen!
  - Projectname: „exercise01_[CIP-Kennung]“
- Create your personal folder „nachname“ in the SVN-repository of your group
- Create a folder for each exercise named „exerciseX“ and put all necessary source files there
- Solutions are always due to following Thursday, 12 noon!

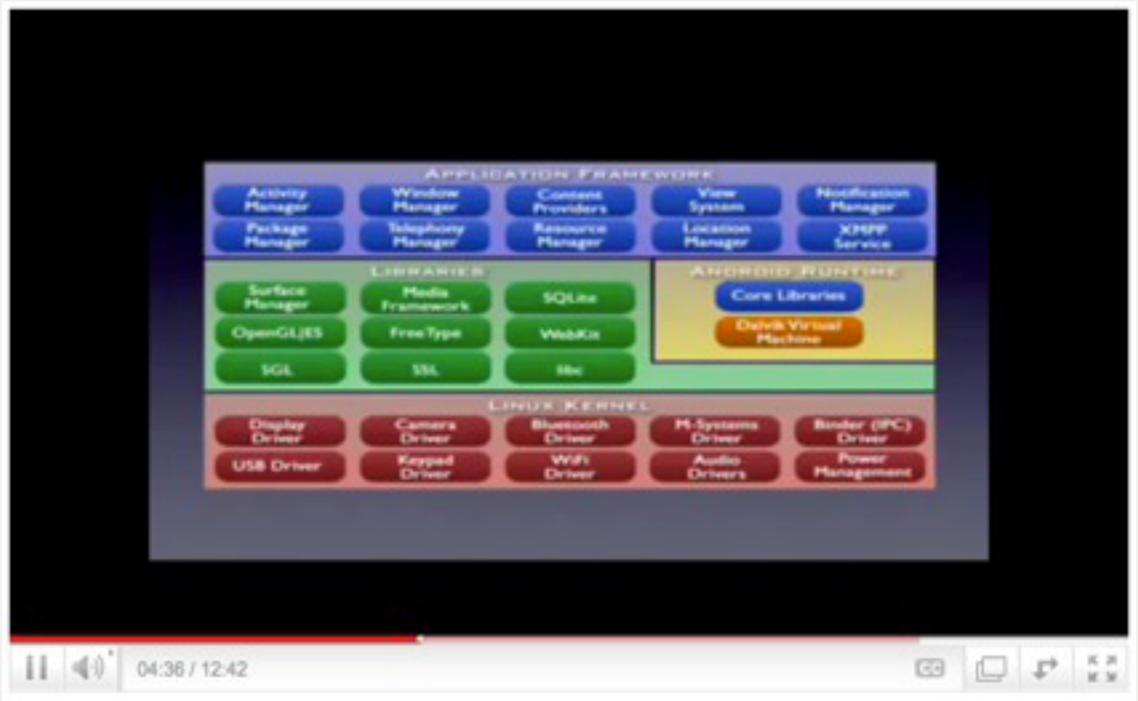


Arbeitsgruppen Medieninformatik und Mensch-Maschine-Interaktion

Die Arbeitsgruppen Medieninformatik und Mensch-Maschine-Interaktion des Instituts für Informatik der LMU München setzen sich zusammen aus dem Lehrstuhl für Angewandte Informatik und Medieninformatik (Prof. Dr. Heinrich Hußmann) sowie dem Lehrstuhl für Mensch-Maschine-Interaktion (Prof. Dr. Andreas Butz). Zusammen mit den anderen Lehrstühlen des Instituts betreuen wir den Studiengang Medieninformatik an der LMU.

Links

- Android website: http://code.google.com/android/
- YouTube: Androidology



http://www.youtube.com/watch?v=QBGfUs9mQYY