Das neue Android Build System

Besser Builden mit Gradle

JUGS Event, 18. Juni 2015

Kaspar von Gunten, Ergon Informatik AG

whois(kvg)

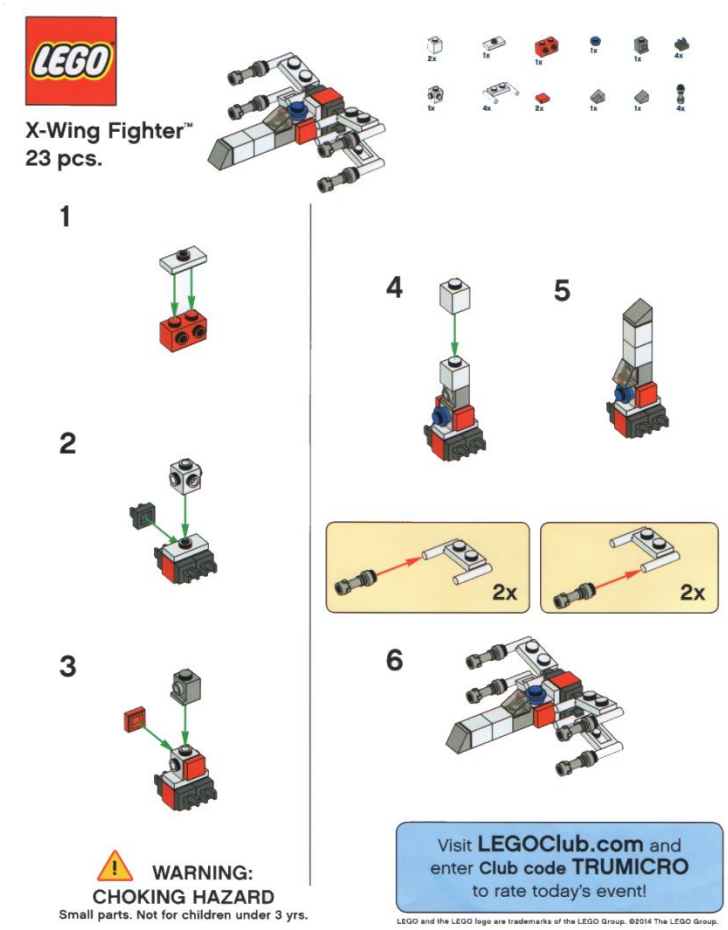
- Software-Engineering seit 2002
- Entwicklung in Java
  (Client/Server, Desktop, Android)
- Nebenberuflicher Dozent
  (Android, GUI-Programming, OOP, …)
- Angestellt bei Ergon Informatik AG, Zürich
  (Aktuell Cloud-Projekt im Bereich «Internet of Things»)

ergon

Inhaltsübersicht

- Was ist ein Build?
  - Bekannte Buildsysteme für Java
- Gradle
  - Gradle Basics
  - Android Plugins für Gradle
  - Android Build Files (build.gradle) Konfiguration
- Build Variants
  - Build Types, Product Flavors
- Demo

Build?



1. Check out code
2. Resolve dependencies
3. Compile
4. Static code analysis
5. Run tests
6. Check coverage
7. Obfuscate
8. Generate documentation
9. Create Artifacts
10. Deploy

Buildsysteme für Java

Ant

Deklarativ (XML)

Ant Tasks

Kein Projektmodell

Keine Konvention

Kein Lifecycle

Kein Dep.-Mgmt

Kein Skripting

Maven

Deklarativ (XML)

Projektmodell (POM)

Konventionen

- Projektstruktur
- Lifecycle-Phasen

Dependency-Mgmt

Plugins

Kein Skripting

Gradle

Deklarativ (DSL)

Projektmodell

Scriptable Tasks + Lifecycle

Konventionen

- Projektstruktur

Dependency-Mgmt

Plugins

Skripting

Ant

```
<project name="MyProject" default="dist" basedir=".">
    <description>
        simple example build file
    </description>
  <!-- set global properties for this build -->
  <property name="src" location="src"/>
  <property name="build" location="build"/>
  <property name="dist"  location="dist"/>

  <target name="init">
    <!-- Create the time stamp -->
    <tstamp/>
    <!-- Create the build directory structure used by compile -->
    <mkdir dir="${build}"/>
  </target>

  <target name="compile" depends="init"
        description="compile the source " >
    <!-- Compile the java code from ${src} into ${build} -->
    <javac srcdir="${src}" destdir="${build}"/>
  </target>

  <target name="dist" depends="compile"
        description="generate the distribution" >
    <!-- Create the distribution directory -->
    <mkdir dir="${dist}/lib"/>

    <!-- Put everything in ${build} into the MyProject-${DSTAMP}.jar file -->
    <jar jarfile="${dist}/lib/MyProject-${DSTAMP}.jar" basedir="${build}"/>
  </target>

  <target name="clean"
        description="clean up" >
    <!-- Delete the ${build} and ${dist} directory trees -->
    <delete dir="${build}"/>
    <delete dir="${dist}"/>
  </target>
</project>
```

```
> ant clean dist
```

Maven

```
> mvn clean package
```

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.2</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Gradle

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:1.2.2'
    }
}

repositories {
    jcenter()
}

apply plugin: 'com.android.application'

android {
    compileSdkVersion 21
    buildToolsVersion "22.0.1"

    defaultConfig {
        applicationId "ch.example.hsludemo"
        minSdkVersion 16
        targetSdkVersion 21
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:22.1.1'
}
```

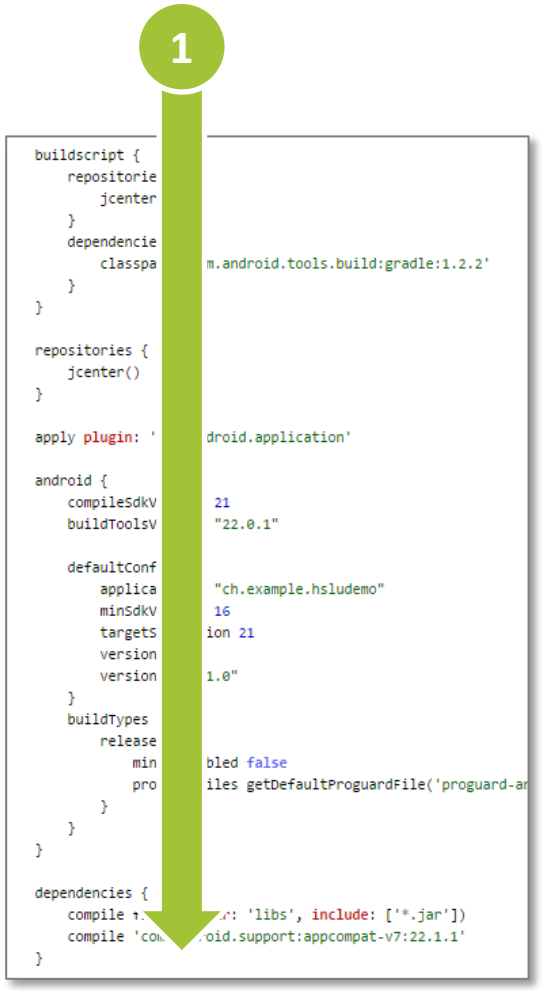
```
> gradlew build
```

Gradle Buildskripts

- Gradle Buildskripts enthalten
  - Task-Definitionen
  - Konfigurationen von Modellobjekten und Tasks
  - Ausführbaren Code

Objektnotation ähnlich JSON

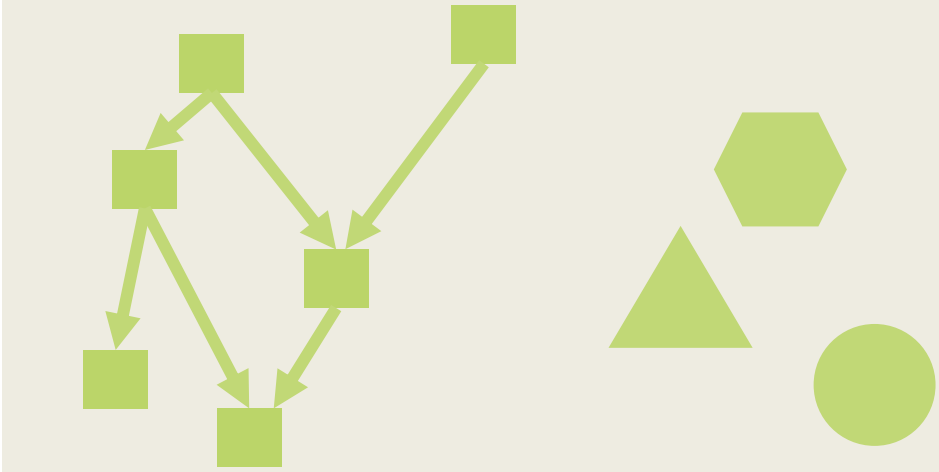
- Ein Gradle-Build läuft in 2 Phasen ab
  1. Konfiguration des Builds (Erstellung DAG)
  2. Ausführung des Builds (d.h. eines spezifischen Tasks)

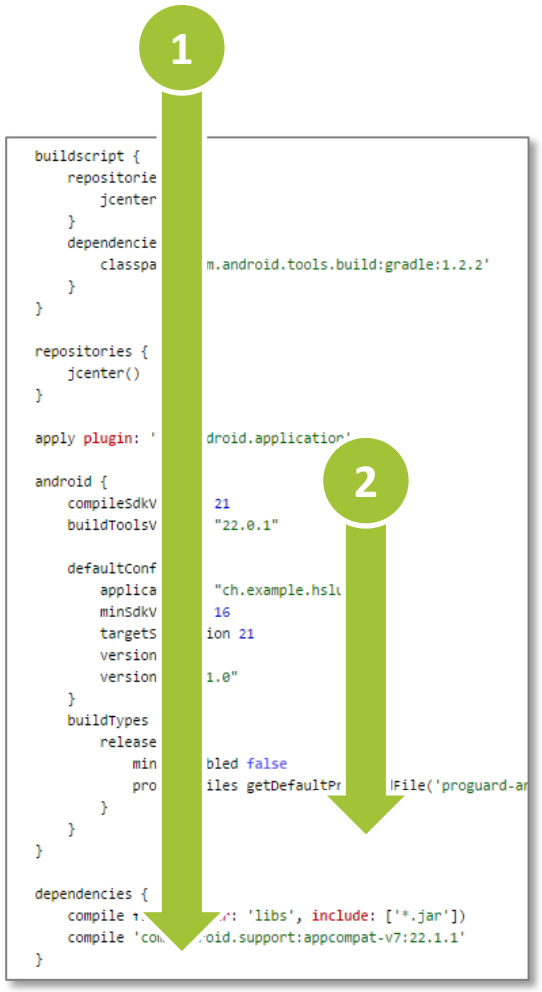


build.gradle

```
> gradlew build
```

1 Pass 1: Configure Model

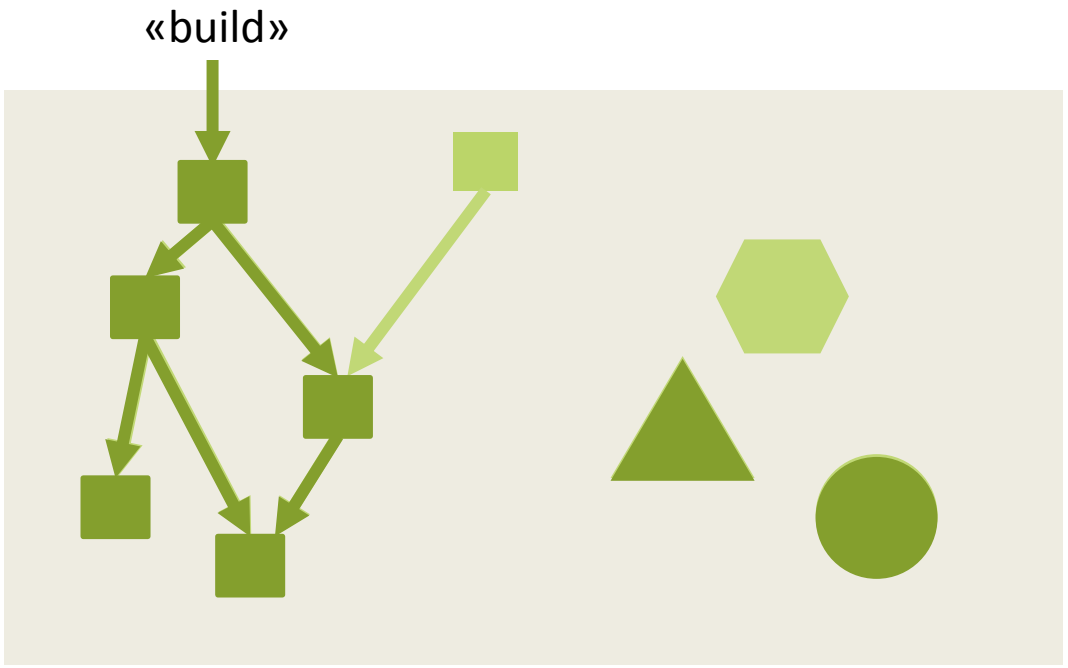




build.gradle

```
> gradlew build
```

1. Pass 1: Configure Model
2. Pass 2: Execute Task «build»



Build Script (build.gradle)

```
println 'Begin...'

String getUpTime = "7:00h"
int coffeeCount = 2
def workToDo = [ "Fix bugs" ]

println 'Initialized properties'

task getUp {
    println "Set alarm clock to $getUpTime"
    doFirst {
        println "Ring! Ring! Ring!"
    }
    doLast {
        println "Getting out of bed at $getUpTime"
    }
}

task drinkCoffee(dependsOn: getUp) {
    doLast {
        println "Drinking $coffeeCount coffees"
    }
}

task doWork(dependsOn: drinkCoffee ) {
    doLast {
        workToDo.each { workTask -> println workTask }
    }
}

task prepareDaysWork {
    doFirst {
        println "Setting up enough work for 1 day"
        workToDo << "Implement Feature" …    }
}

task workAllDay(dependsOn:[ prepareDaysWork, doWork ]) {
    doWork.shouldRunAfter prepareDaysWork
}
```

Output

```
> gradlew doWork

Begin...
Initialized properties
Set alarm clock to 7:00h
:getUp
Ring! Ring! Ring!
Getting out of bed at 7:00h
:drinkCoffee
Drinking 2 coffees
:doWork
Fix bugs
```

Config Phase

Execution Phase

```
> Gradlew workAllDay

Begin...
Initialized properties
Set alarm clock to 7:00h
:prepareDaysWork
Setting up enough work for 1 day
:getUp
Ring! Ring! Ring!
Getting out of bed at 7:00h
:drinkCoffee
Drinking 2 coffees
:doWork
Fix bugs
Implement Feature
Review code
Do other stuff
:workAllDay
```

Configuration during Execution

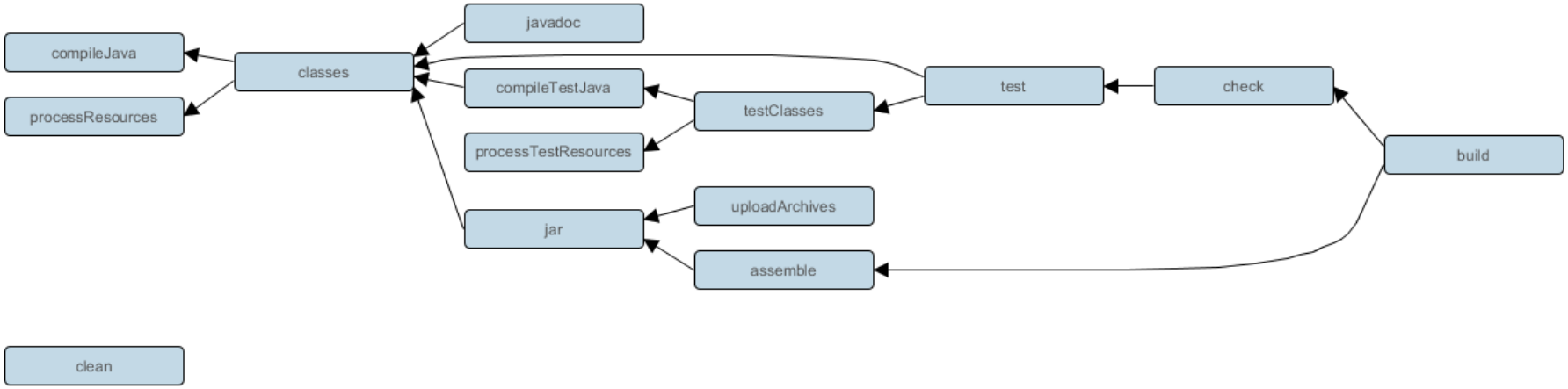
Gradle Plugins

- Ein Gradle-Skript kann Plugins anwenden
  - Plugin als Abhängigkeit hinzufügen (im *buildscript*-Block)
  - Plugin mit ‘apply’ aktivieren

- Plugins
  - Definieren neue Modellobjekte
  - Fügen einem Projekt weitere Tasks hinzu
  - Konfigurieren Buildparameter gemäss Konventionen

Java Plugin

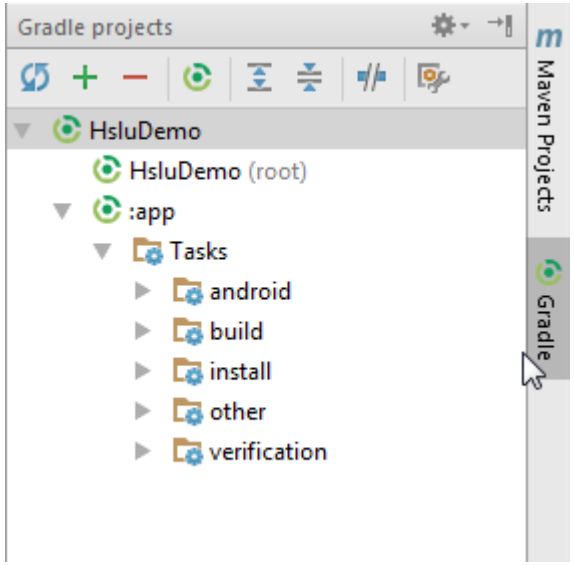
- Definiert u.a. die folgenden Tasks



- Definiert:
  - sourceSets (main, test), compile + runtime classpath
  - Diverse Properties (dirs, compatibiltiy, etc.)

Android Plugin

- Definiert u.a. die folgenden Tasks
  - assemble → compile, copy res
  - check → lint
  - test
  - build → assemble, check
  - install
  - clean
- Definiert das Modellobjekt
  - android




```
> Gradlew tasks
```

Konventionen?

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:1.2.2'
    }
}
```

1

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 21
    buildToolsVersion "22.0.1"

    defaultConfig {
        applicationId "ch.example.hsludemo"
        minSdkVersion 16
        targetSdkVersion 21
        versionCode 1
        versionName "1.0"
    }
}
```

2

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:22.1.1'
}
```

3

Module- und Project-Buildfile

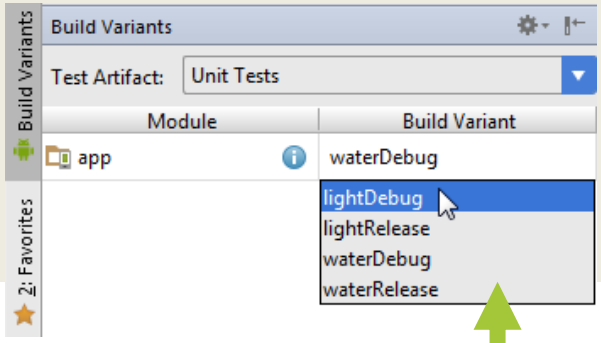
- Android-Applikationen können grundsätzlich aus mehreren Modulen bestehen
- Im einfachsten Fall gibt es ein einzelnes «app» Modul mit einem eigenen Build-File
- Auf Projekt-Ebene gibt es noch ein «root» Build-File, welches die gemeinsame Build-Konfiguration definiert und die Subprojekt-Builds aufruft

Build Konfiguration

- Default Configuration
  - applicationId, minSdkVersion, targetSdkVersion, versionCode, versionName
- Dependencies für tasks
  - compile
  - testCompile
- *applicationId* vs. *package* (in Manifest)
  - ID für Store und App
  - Base-Package für Javaklassen (und R-Klasse)

Kann auch dynamisch durch Skripting berechnet werden !

Build Variants



Auswahl in Studio

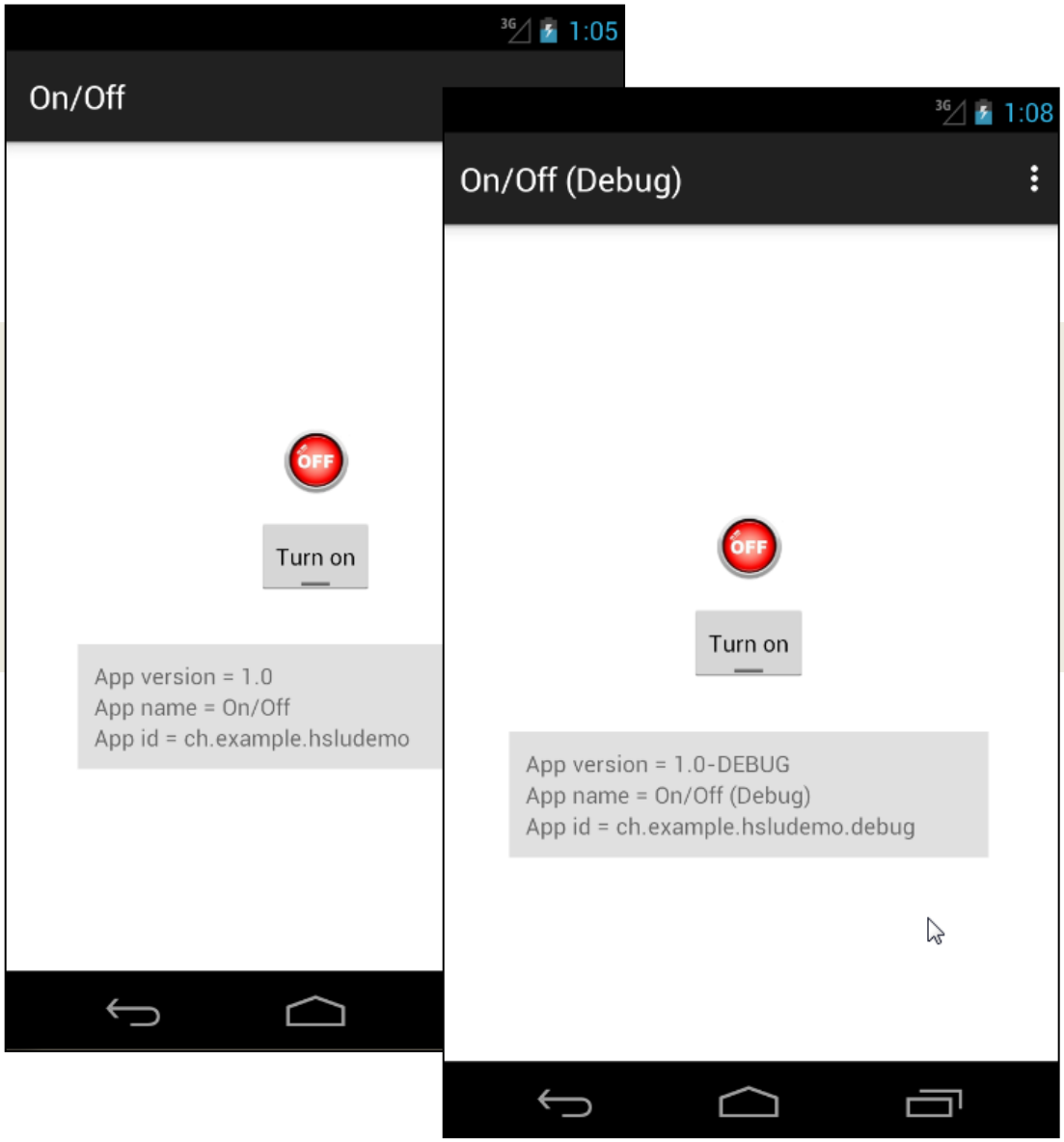
- Der Android Build kann von derselben Applikation mehrere Varianten erstellen
  - Build Types: Debug, Release
  - Product Flavors: Free, Paid, Company1, Company2
- Ergibt Task Matrix für Build + Priorität

	Debug	**Release**
Free	buildFreeDebug	buildFreeRelease
Paid	buildPaidDebug	buildPaidRelease

1.Build Type
2.Flavor
3.Default Config

Build Types

- Default Build Typen: debug, release
  - Unterschiedliche Signing-Konfigurationen (Keys)
  - ProGuard (Obfuscator/Shrinker) für Release
  - Setzen des «debuggable»-Flags
  - Unterschiedliche *applicationId* und *versionName* (Parallel-Installation möglich)
- Subconfig im Build-File: **android.buildTypes**
  - Pro Build-Typ gibt ein neues *src/<buildType>* Directory im Modul
  - Definition von alternativen Ressourcen und Klassen

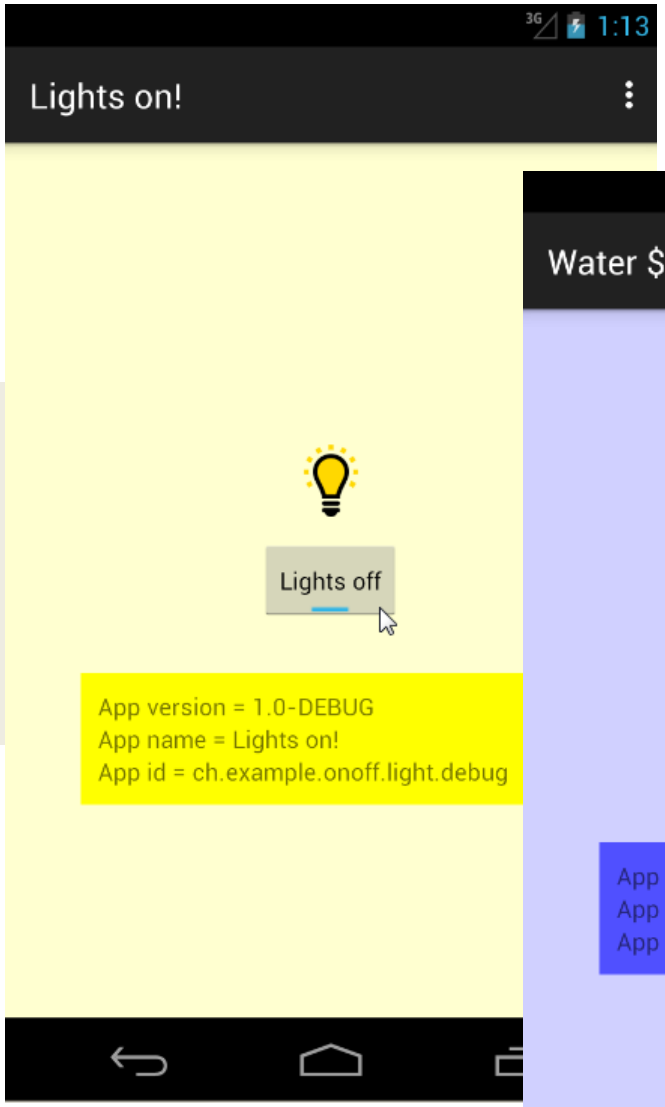

1:05
On/Off
OFF
Turn on
App version = 1.0
App name = On/Off
App id = ch.example.hsludemo
1:08
On/Off (Debug)
OFF
Turn on
App version = 1.0-DEBUG
App name = On/Off (Debug)
App id = ch.example.hsludemo.debug

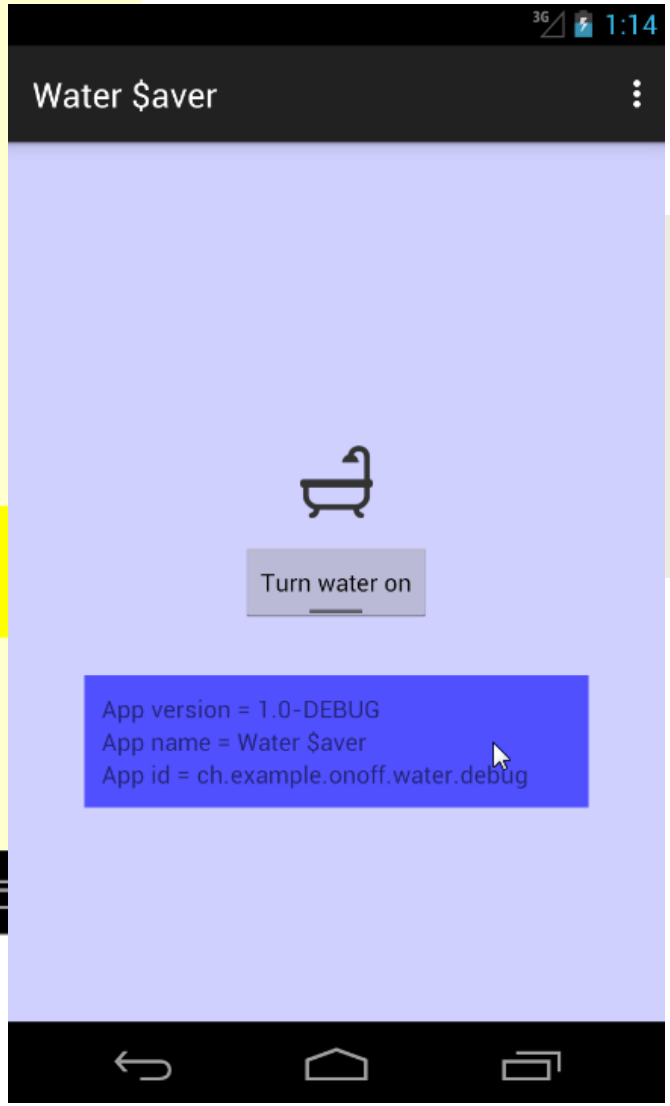
Product Flavors

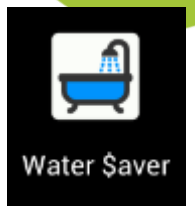
- Erlaubt mehrere Varianten derselben App
  - Branding: <Firma 1>, <Firma 2>
  - Funktionalitätsunterschied: <free>, <paid>

- Subconfig im Build-File: **android.productFlavors**
  - Pro Build-Typ gibt ein neues src/<buildType> Directory im Modul
  - Hier können alternative Ressourcen und Klassen definiert werden


Lights on!


1:13
Lights on!
Lights off
App version = 1.0-DEBUG
App name = Lights on!
App id = ch.example.onoff.light.debug


1:14
Water $aver
Turn water on
App version = 1.0-DEBUG
App name = Water $aver
App id = ch.example.onoff.water.debug


Water $aver

Tipp: Gradle Daemon

- Gradle kann mit einem Daemon laufen
  - Modell wird In-Memory gecached
  - Muss nicht immer wieder alles von neuem Parsen
  - Speed-Up für Build!

- Wie aktivieren?
  - Im File *<project_root>/gradle.properties* die Zeile **org.gradle.daemon=true** eintragen

Tipp: App von Konsole starten

Root *build.gradle* ergänzen:

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:1.2.2'
        classpath 'com.novoda:gradle-android-command-plugin:1.4.0'
    }
}
```

App *build.gradle* ergänzen (oder *allprojects* in Root):

```
apply plugin: 'com.android.application'
apply plugin: 'android-command'
```

```
> gradlew runDebug
> gradlew runFreeRelease
```

Referenzen

Gradle Home
http://gradle.org

Android Developer: Build System Overview
https://developer.android.com/sdk/installing/studio-build.html

Android Developer: Building and Running
https://developer.android.com/tools/building/index.html

Android Plugin Documentation
http://tools.android.com/tech-docs/new-build-system/user-guide

Android Command Plugin
https://github.com/novoda/gradle-android-command-plugin

Kaspar von Gunten
kaspar.vongunten@ergon.ch

Ergon Informatik AG
www.ergon.ch
twitter.com/ErgonAG