Build-Management

von

Dominik Domonell, Felix Juhr, Albert Kaminski, Simon Núñez Aschenbrenner, Sahiram Ravikumar

Gliederung

Allgemein

Apache Ant

Apache Maven

Gradle

Xcode

Quellen

Allgemein

Build Management Essentials

Was ist Build-Management?

- Verwaltung von Abhängigkeiten
- Kompilierung des Quellcodes in Binärcode
- Verpacken des Binärcodes
- Automatisierte Tests
- Dokumentation

Build Management Essentials

- Funktionen sollten minimalen Aufwand erzeugen & korrekt, sowie automatisch durchgeführt werden

- Nach Start keine weiteren Benutzerinteraktion.

- Zur Erzeugung notwendigen Informationen zu Beginn des Builds zur Verfügung gestellt werden

Warum Build-Management?

- Nicht jedes Softwareprojekt braucht ein Build-Management Tool.

- Wenn ein Softwareprojekt, sehr komplex & groß wird, macht es Sinn Build-Tools zu benutzen.

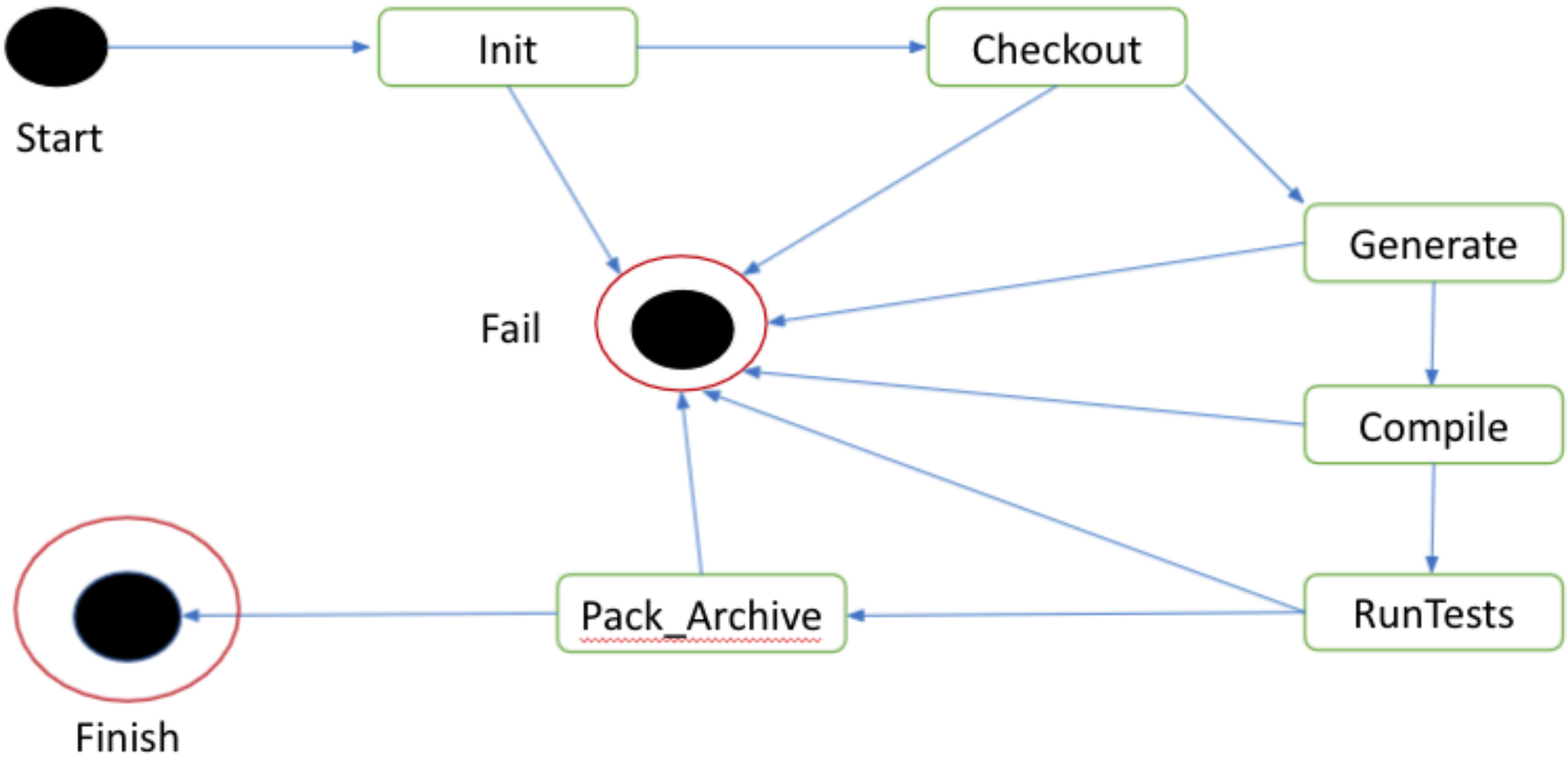
- Vorteile liegen auf der Hand:

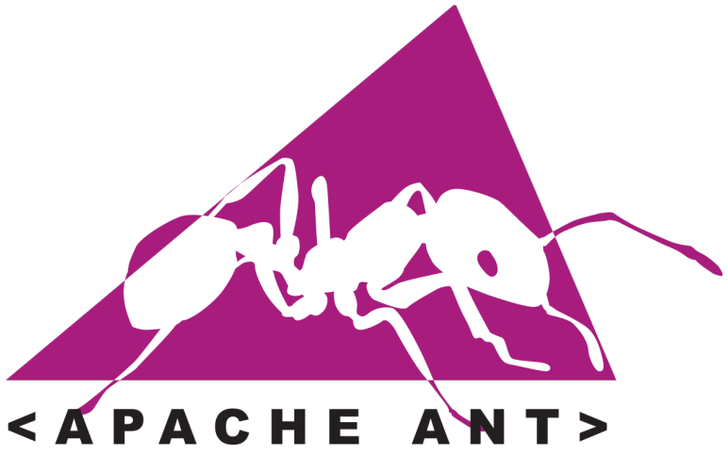
Beispiele

Beispiele für Ressourcen wären zum Beispiel:

- Quellcode aller Programmierer
- Dokumentation
- Zusatzprogramme, die mit ausgeliefert werden
- Hilfsdateien, die teilweise in alle unterstützten Sprachen übersetzt werden müssen
- Das Programm selbst

Ein typischer Build





Apache Ant

Einführung

Was ist Apache ANT?

- ANT (another Neat Tool = noch ein hübsches Werkzeug)

- plattformunabhängiges Werkzeug in Form einer Java-Bibliothek mit dem Quellcode zusammengestellt, bearbeitet, übersetzt & Programme ausgeführt werden können

- Unterstützt den Build-Prozess im Java-Umfeld

- Ant ist in vielen Entwicklungsumgebungen (Bsp. Eclipse) integriert, kann aber auch stand-alone genutzt werden

- Anlehnung an Make (Make ist ein vergleichbares Tool, für C-Programme entwickelt)

Geschichte

Wie ist ANT entstanden?

- ANT wurde 1999 von James Duncan Davidson entwickelt

- Ableitung von der Ameise: „Kleine Dinge können großes Bewegen / Leisten“

- Erste Version (1.1) Juli 2000

- Aktuelle Version (1.10.9) September 2020

Apache Ant Installation

- Aktuelle Apache ANT Version unter https://ant.apache.org/ herunterladen

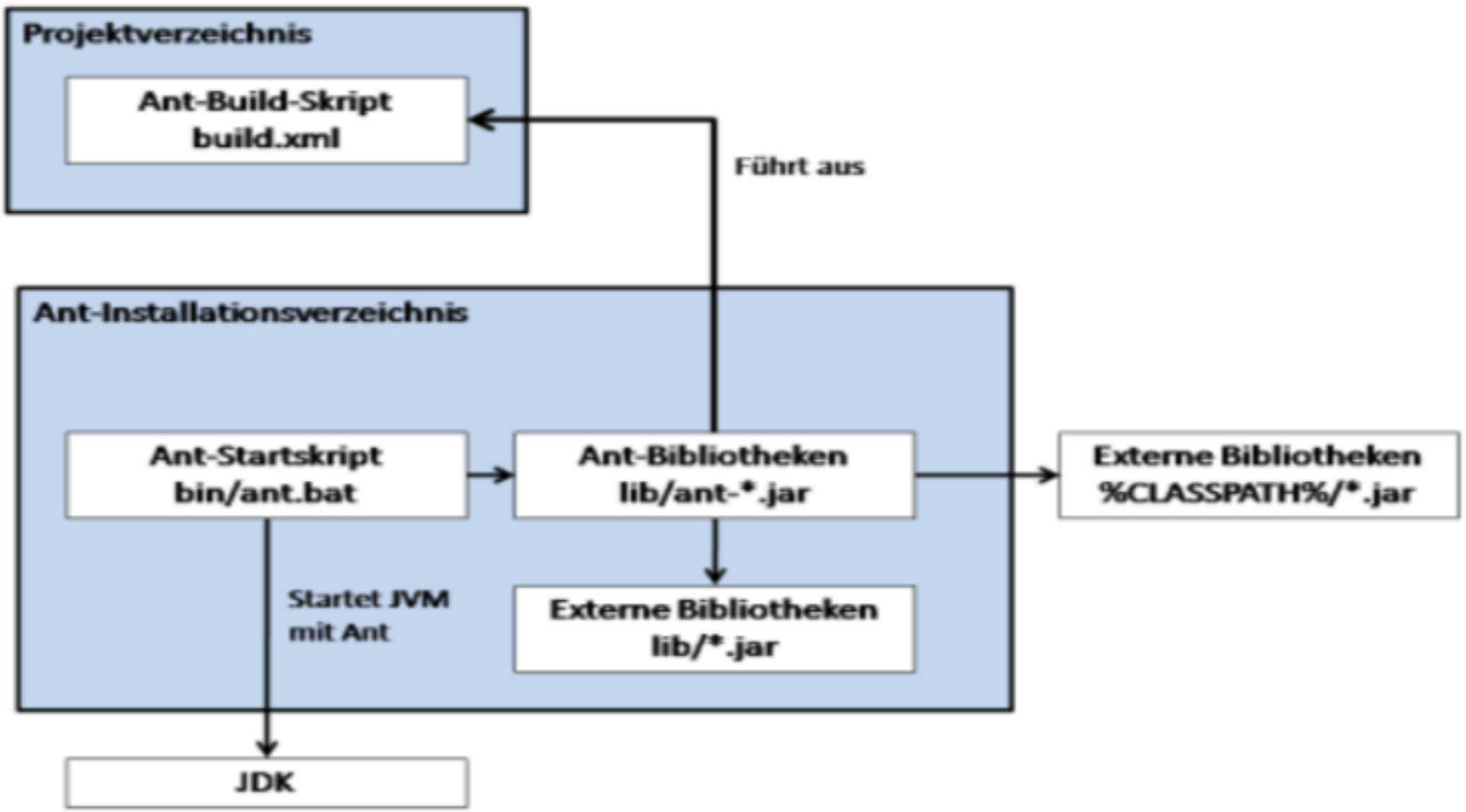
- Entpacken des Archivs in beliebigen Verzeichnis
  - C:\Programme

- Java Development Kit muss auf Rechner installiert sein

- Umgebungsvariable PATH erweitern, dass sie auf das bin/ Verzeichnis von ANT zeigt
  - PATH = %JAVA_HOME%\bin:%ANT_HOME%\bin:..

- Kommandozeile mit Ant ausführen
  - $ ant

ANT Architektur



Wie ist ANT aufgebaut?

- ANT wird durch XML-Datei, die Build-Ablauf-Prozess darstellt, gesteuert
  - build.xml

- Build Datei enthält ein Projekt
  - `<project name = “Apple Juice“ default =“build“ basedir =“.“/>`

- Property entspricht einer Variablen (Namen und Werte)

<property name = “build.classes“ value = “build“ />
<property name = “src“ value = “src“ />
<target name "compile" depends = "init">
<javac srcdir = "${src}" destdir= "${build}"/>
</target>

Wie ist ANT aufgebaut?

- Das Project enthält Targets („Ziele“), Jedes Target definiert ein bestimmten Teil des Build-Prozesses , er ist wie ein Container für Tasks und proberties($).
  - `<target name = “build“  description = “Baut die Anwendung“>`
  - `</target>`
  - `<target name =“prepare“>`
  - `<mkdir dir =“${build.classes}“/`
  - `</target>`

- Tasks werden im Target aufgerufen (über 150 Tasks auf Homepage) Aufgaben sind z.B. mkdir, copy, delete etc.
  - `<target name = “clean“ description=„“Aufräumen“>`
  - `<delete dir =“${build.classes}“/>`
  - `</target>`

Pro & Contra

+ Vollständig auf Java basiert	- Für wiederholt auftretenden Tasks, gibt es keine Hilfsmittel, um redundante Arbeitsschritte zu vermeiden
+ Plattformunabhängig	
+ Build-Files können auf verschiedene Plattformen verschoben werden	


Maven™

Einführung

Was ist Maven?

- Maven ist eine Projektmanagementssoftware, die den Bulidprozess standardisiert, verwaltet & durchführt
- Sie kann
  - Programme erstellen & verwalten
  - Ordnerstruktur automatisieren
  - Testen
  - Code verpacken und bereitstellen (Zip, Rar, Dmg, …)
  - JAR-Dateien in Repositories verwalten
  - Reduziert somit alle Schritte auf einen Build-Prozess

Geschichte

Wie ist Maven entstanden?

- Maven wurde als Nachfolger des Buildmanagement-Tools „Ant“ entwickelt
  - Erscheinungsjahr 2002 - Apache Software Foundation
  - Erste Version wurde im Jahr 2004 veröffentlicht
  - Zweite Version 2005 (Überarbeitung der Performanceprobleme)
  - Dritte und letzte Version 2008 (Augenmerk auf Kompatibilität zwischen Version 2 und 3)
  - Aktuelle Version 3.8.1 (03.04.2021)
  - Maven kommt aus dem Jiddischen - bedeutet „Sammler des Wissens“

Aufbau

Wie ist Maven aufgebaut?

- Aufbau der POM.xml

```
<project>
      <modelVersion>4.0.0</modelVersion> (POM-Modellversion)
      <groupId>Projektgruppenname</groupId> (ID der Projektgruppe)
      <artifactId>Projektname</artifactId> (ID des Projekts)
      <version>0.0.1</version> (Projektversion)
      <build>
        <sourceDirectory>src</sourceDirectory> (Verzeichnis)
        <plugins> (enthaltene Plugins)
</project>
```

Anwendung

Wie werden Dependecies in Maven angewendet?

```
<dependency>
    <groupID>Programmgruppenname</groupID>
    <artifactID>Projektname</artifactID>
    <version>0.0.1</version>
    <type>jar</type>
<dependency>
```

Vergleich

Vorteile zu Apache Ant

- Fördert Standardisierungen (Konvention vor Konfiguration)
- Fördert Wiederverwendung, einheitliche Verzeichnisstruktur & Organisation
- Automatische Verwaltung der Repositories
- Schneller, einfacher, ressourcenschonender
- Projekt kann als „Dependencies“ für andere Projekte aufgelöst werden

Gradle

Allgemein

- Gradle basiert auf gleichem Konzept wie Apache Ant und Maven
- Ant = Flexibilität
- Maven = Benutzerfreundlichkeit
- Aufgaben: Dependency Management, Kompilierung, automatisierte Tests, Installation, Dokumentation
- Gradle ist so erfolgreich, dass Google für Android verwendet

Funktionen

Gradle DSL

- Wird nicht XML genutzt, sondern eine andere Sprache
- Wegen DSL -> Skripte kürzer & klarer formuliert
- Deklarativ: Standardkonventionen (Maven)
- Imperativ: Alles im Detail definieren (Ant)
- Eigene Build-Änderungen schreiben oder vordefinierte Standards überschreiben & eigenen Belangen anpassen

Funktionen

Build Cache

- Wiederverwendung der Ausgaben von vorherigen Aufgaben oder von anderen Rechnern (shared build cache)

Build Scans

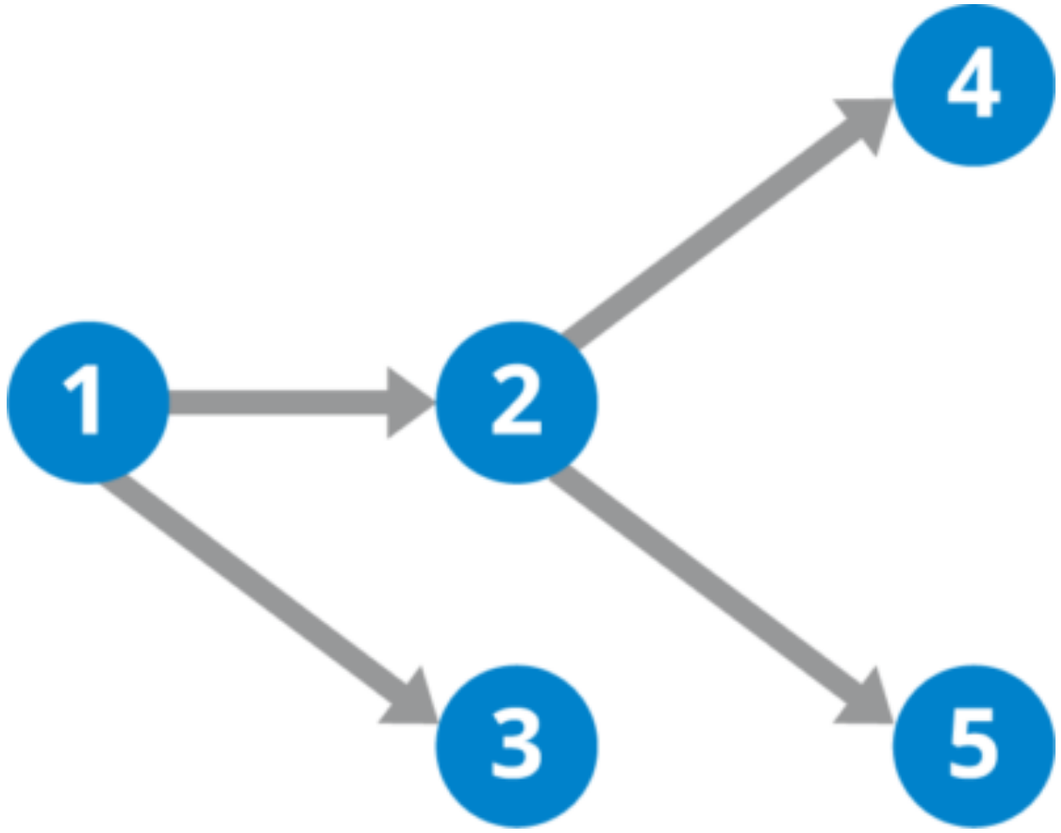
- Stellt umfangreiche Informationen über Build-Run / Build Ausführung = Identifizierung Build-Fehler
- Scans auch teilen, um Rat bei anderen zu suchen

IDE support

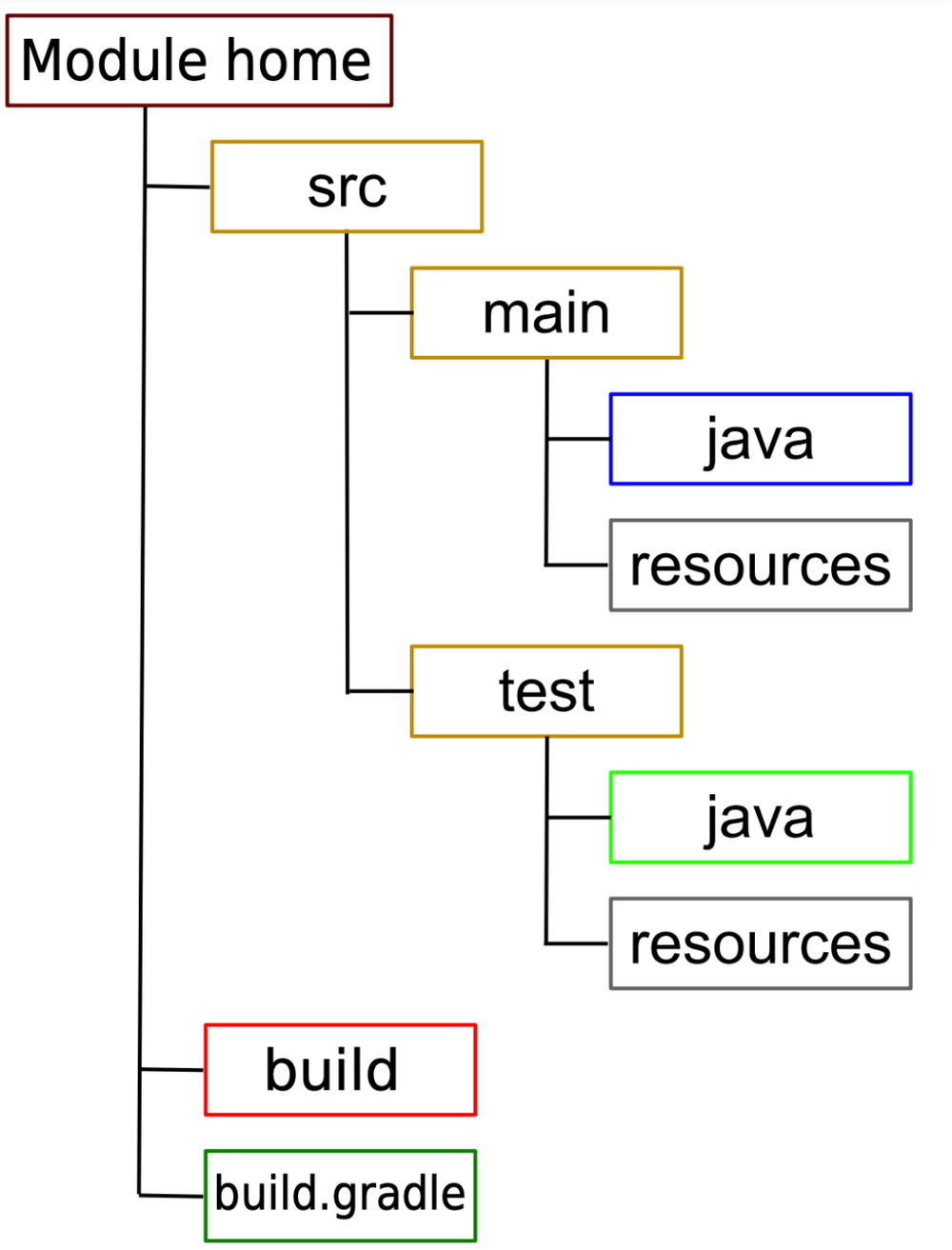
- Gibt einige wichtige IDEs mit denen man Gradle Builds importieren kann + benutzen:
- Android Studio, IntelliJ IDEA, Eclipse, NetBeans

Ablauf

- Inkrementell: Teile einer Software werden gebaut, die verändert wurden oder auf veränderten Teilen beruhen
- Parallel: bestimmte Tasks beim Build laufen parallel => wesentlich höhere Geschwindigkeit bei Erstellprozess
- Verwendet azyklischen gerichteten Graphen (DAG)
- Heißt: build konfiguriert eine Reihe an Aufgaben, basierend auf „dependencies“, verbindet sie zusammen = Erstellung des DAG

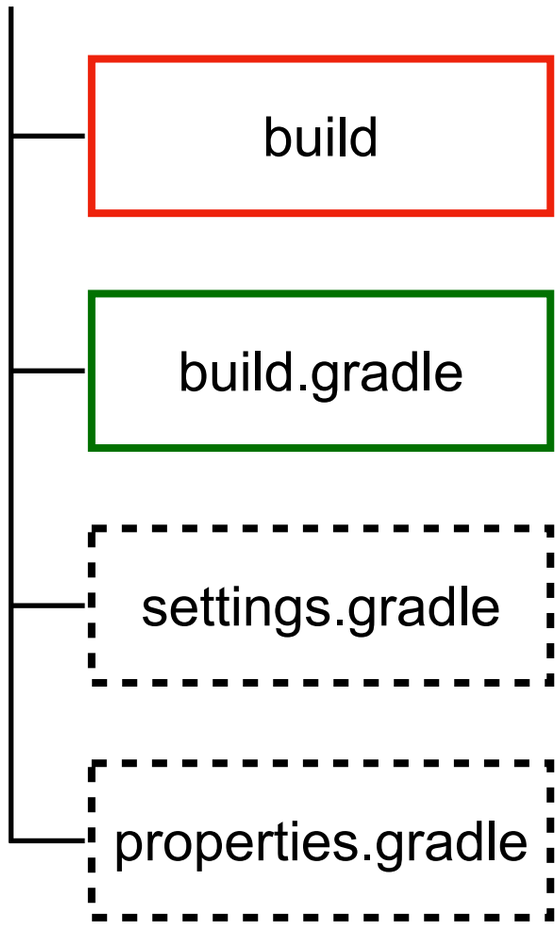


Konzeption



- Anfangs für Dependency-Management: Apache Ivy -> später eigene Engine
- „Convention over configuration“ für Verzeichnislayout & üblichen Phasen für Projektbau (von Maven)
- Abhängigkeiten eines Projekts von anderen Projekten oder Fremdbibliotheken (Maven)

Aufbau des Builds



build.gradle:

- Alle Tasks & Abhängigkeiten eines Projekts
- Vererbung der Eigenschaften von „Vater“-Buildskript

settings.gradle (optional):

- Multiprojekt: teilnehmenden Unterprojekte festgelegt

gradle.properties (optional):

- Liste von Properties, die für projektspezifische Gradle-Initialisierung gültig sind

```
apply plugin: 'java'
```

build.gradle


```
buildscript
{

    repositories
    {
        google()
            jcenter()
    }
    dependencies
    {
        classpath 'com.android.tools.build:gradle:3.0.1'

        // Do not place the application dependencies here;
        // they belong to individual module build.gradle files
    }
}

allprojects
{
    repositories
    {
        google()
            jcenter()
    }
}

task clean(type
            : Delete)
{
    delete rootProject.buildDir
}
```

Android Studio

Build Prozess für typische Android App:

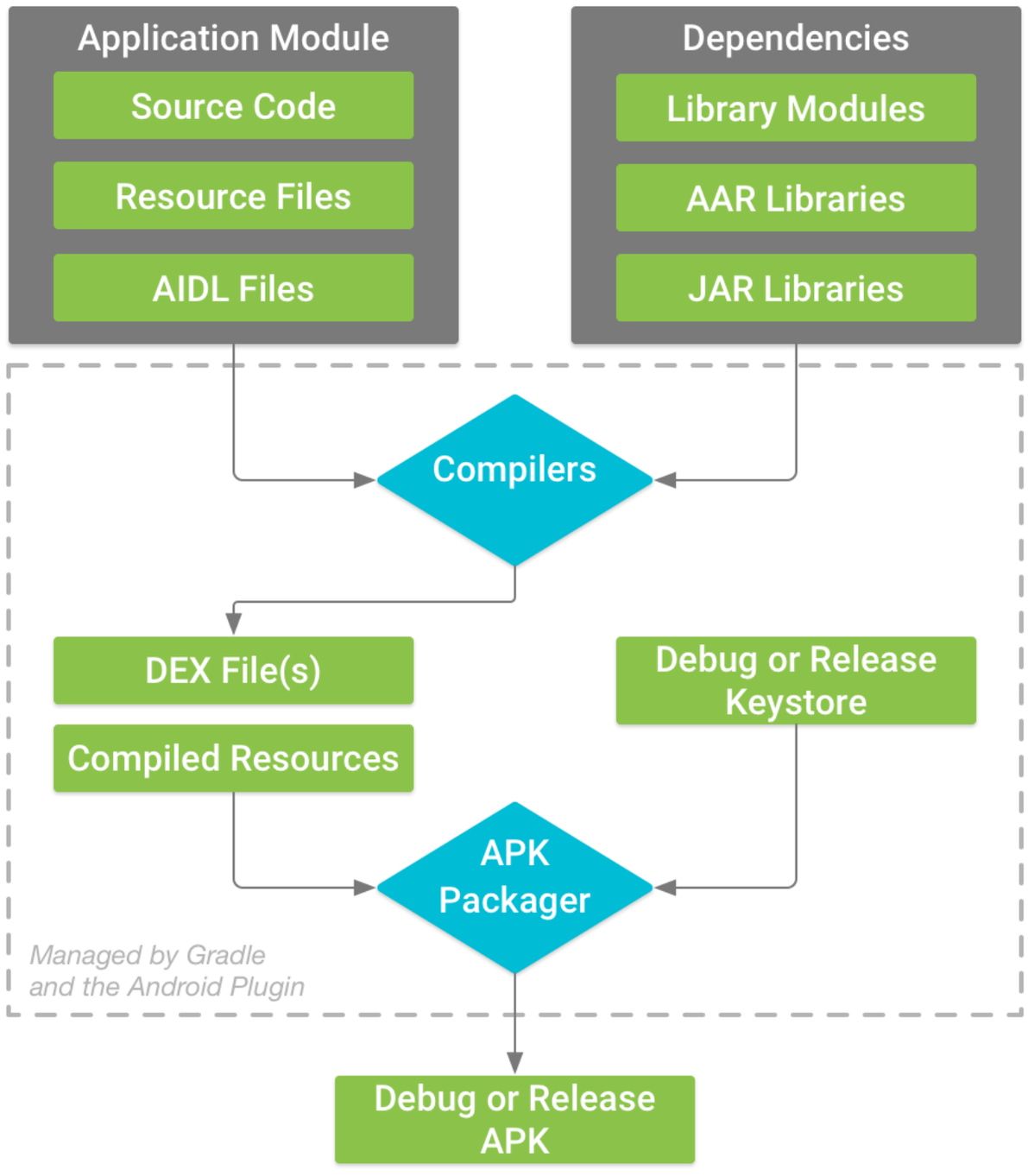
1. Compiler konvertiert Dateien in DEX files / kompilierte resources

2. APK Packager beides in APK

3. Signiert APK entweder mit debug oder release keystore

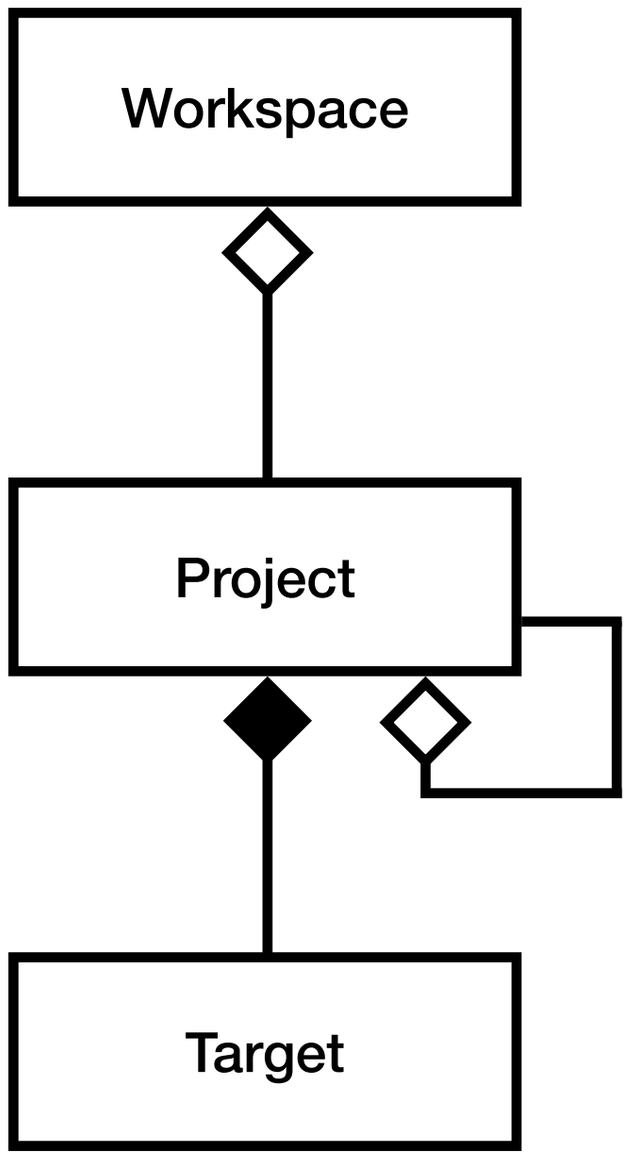
4. Zipalign tool optimiert App

-> Debug oder Release APK



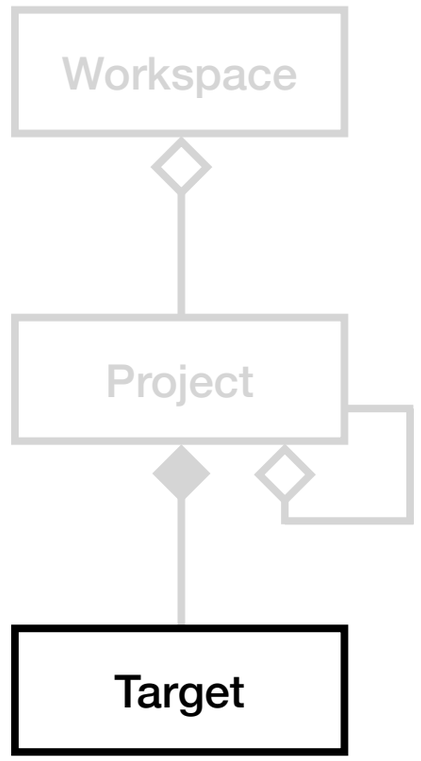
Xcode

Xcode Projektstruktur



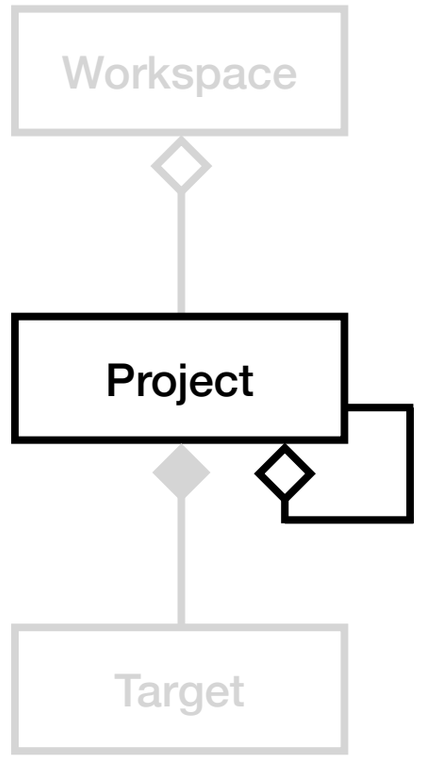
Target

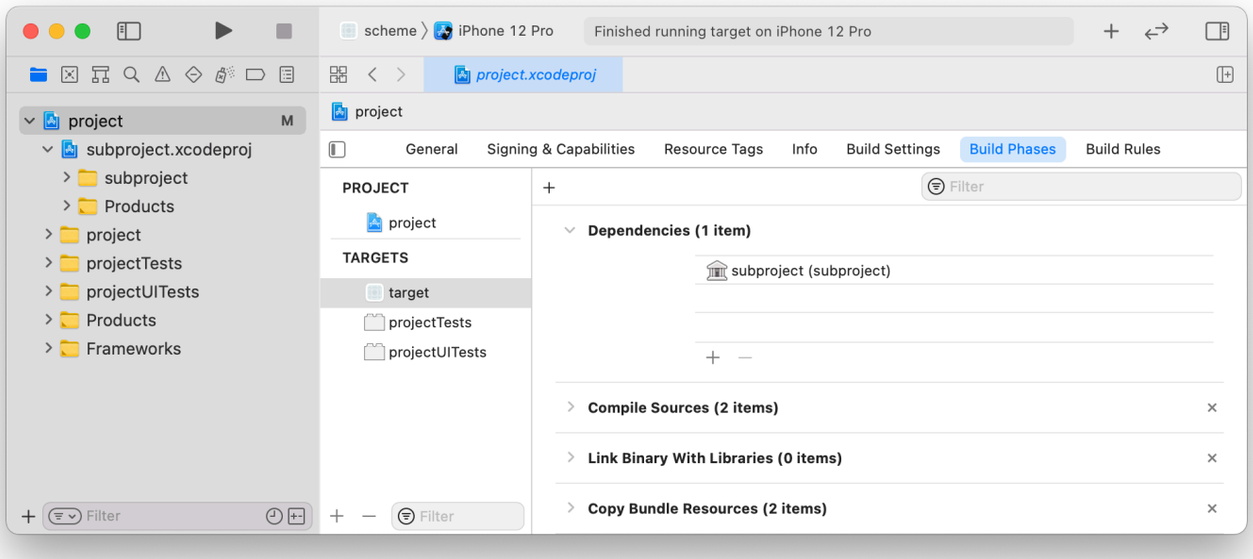
- Spezifiziert wie ein Produkt eines Projekts gebildet werden soll über **Build Phases** und **Build Settings**
  - Klassen, Ressourcen, Skripte
  - **User Defined Settings, Swift Flags**
  - Resource Tags, Capabilities, Service Information, etc.
- Jedes Projekt hat mindestens ein Target
- Ein Target kann von anderen Targets abhängen



Project

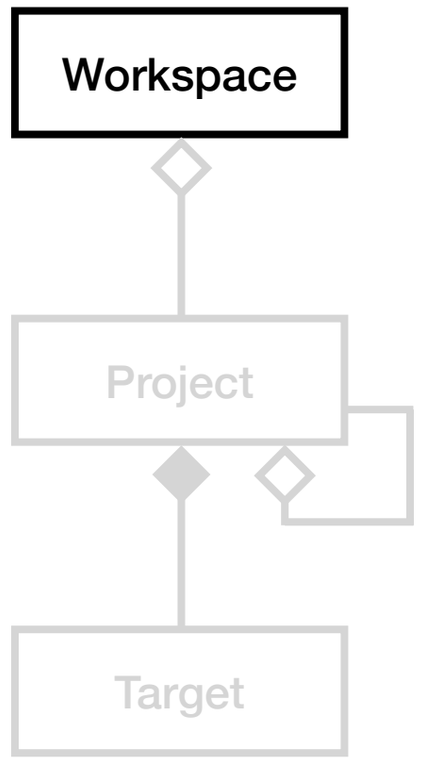
- Behälter für alle Dateien, die für Builds benötigt werden
- **Default Build Settings,** die ganz oder teilweise von den Targets überschrieben werden können
- Spezifiziert **Build Configurations** und **Build Schemes**
- Enthält **Gruppen** (Ordner)
- Kann **Subprojects** beinhalten
  - Subproject Target kann als **Explicit Dependency** zum übergeordneten Projekt hinzugefügt werden



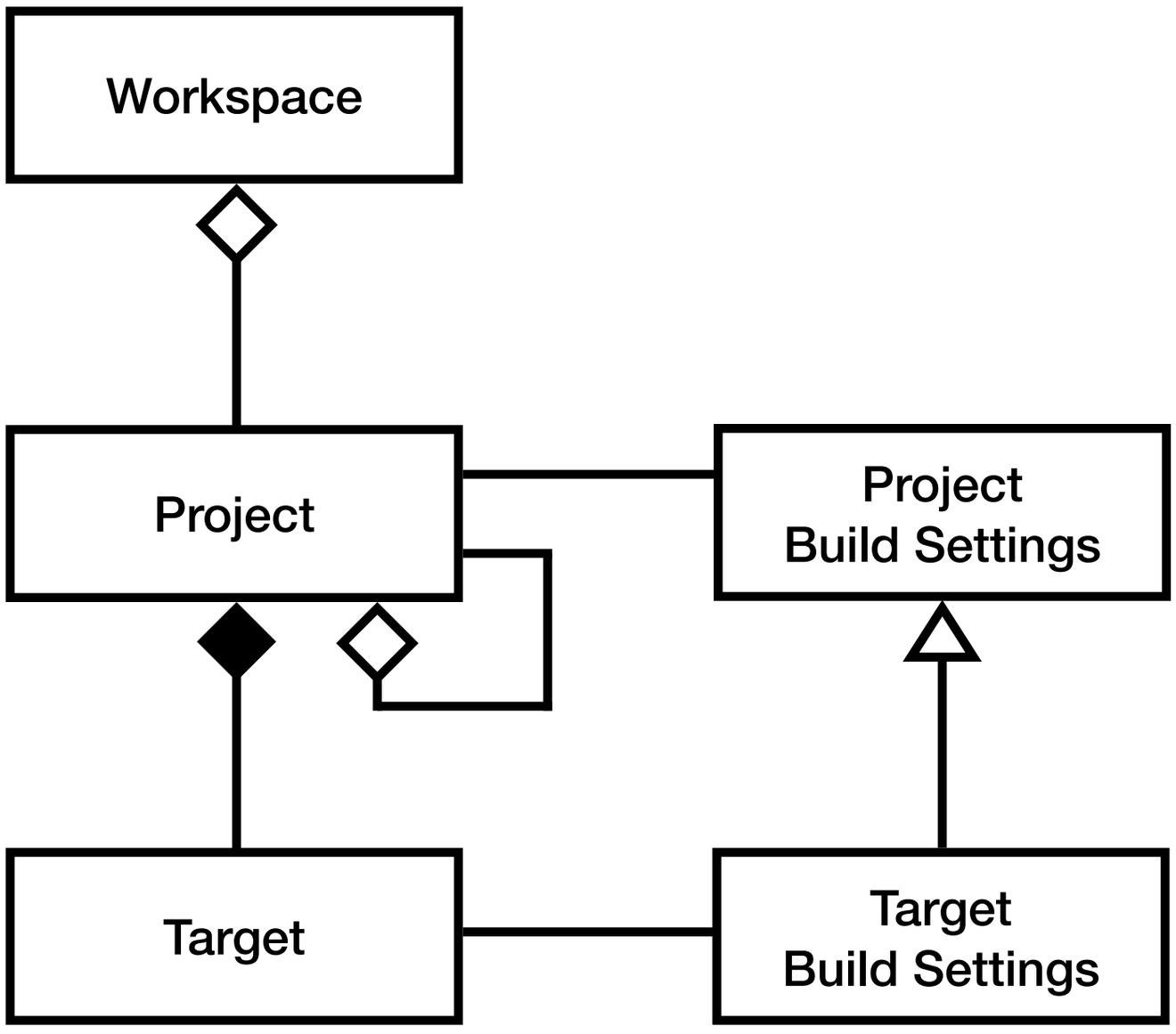


Workspace

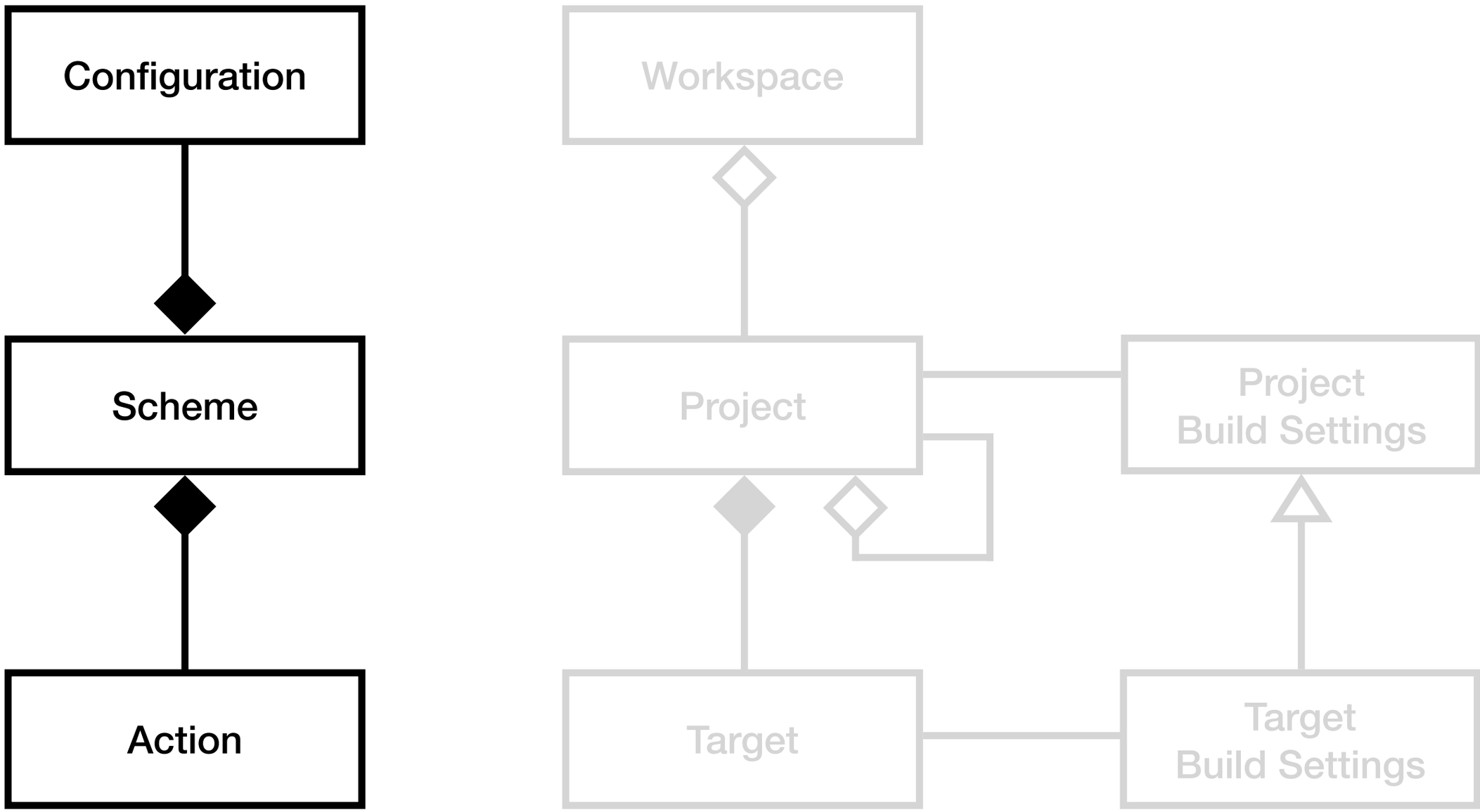
- Behälter für beliebig viele Xcode Projekte
- **Implicit Dependencies:** Projekte haben automatisch Zugriff auf alle Dateien
- Projekte sind unabhängig, Workspaces werden in separater Datei *.xcworkspace verwaltet
- Per Default landen alle Builds im **workspace build directory:** *~/Library/Developer/Xcode/DerivedData*



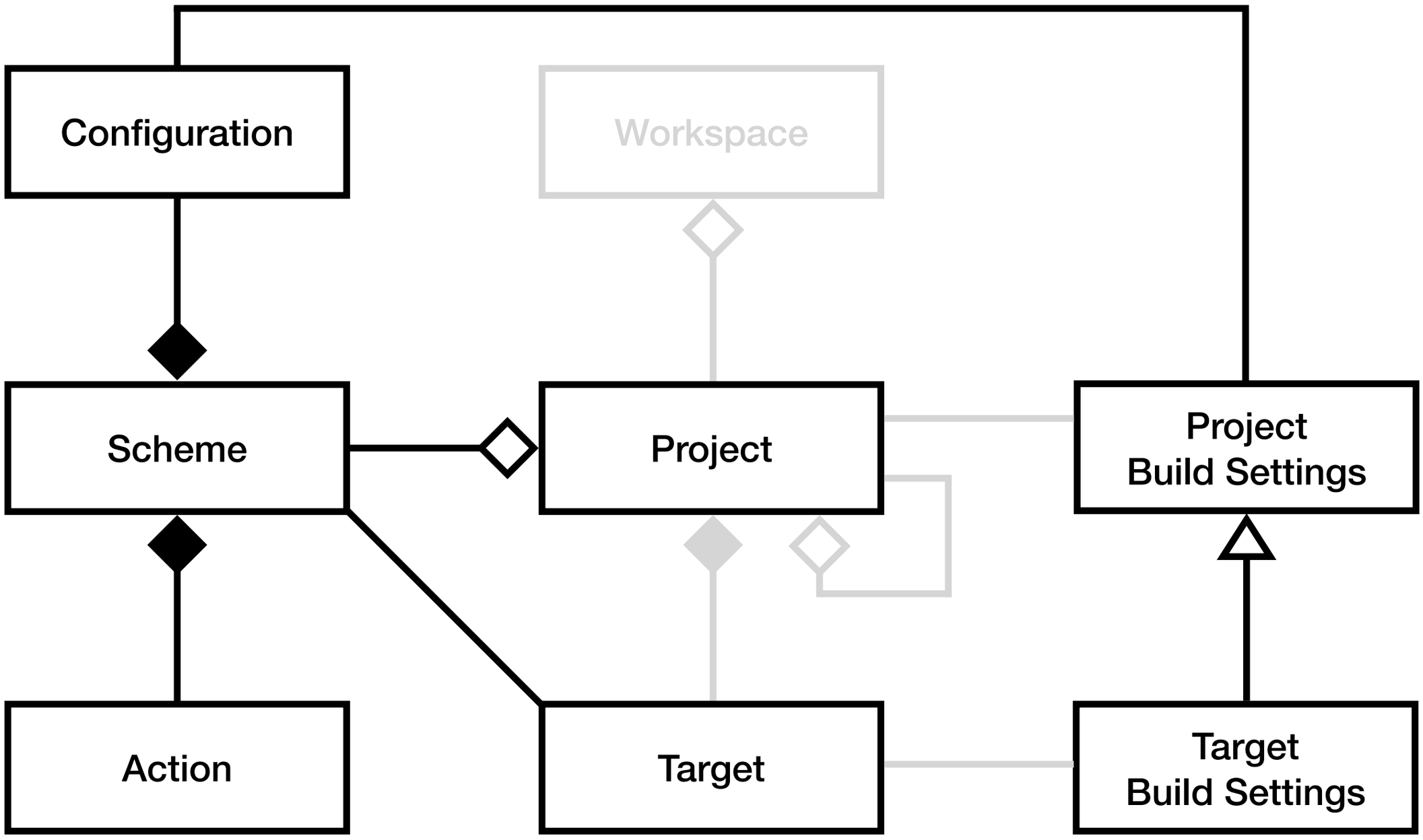
Xcode Projektstruktur



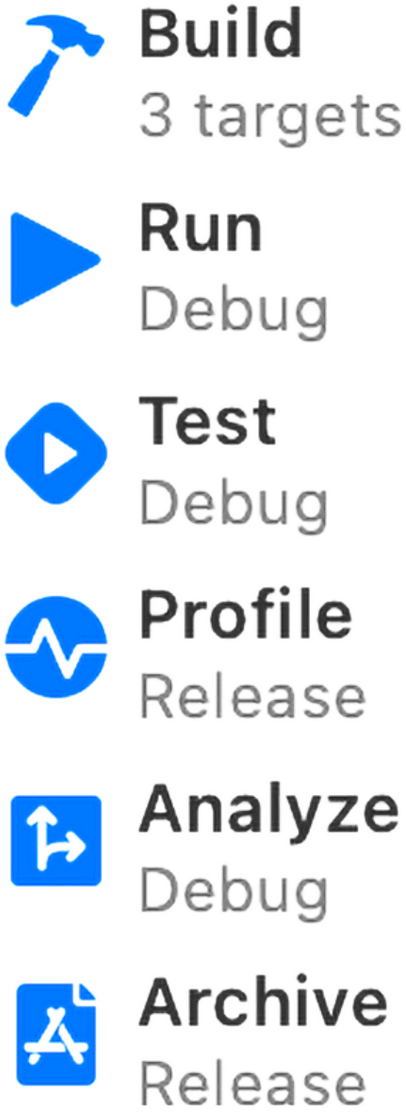
Xcode Buildprozess



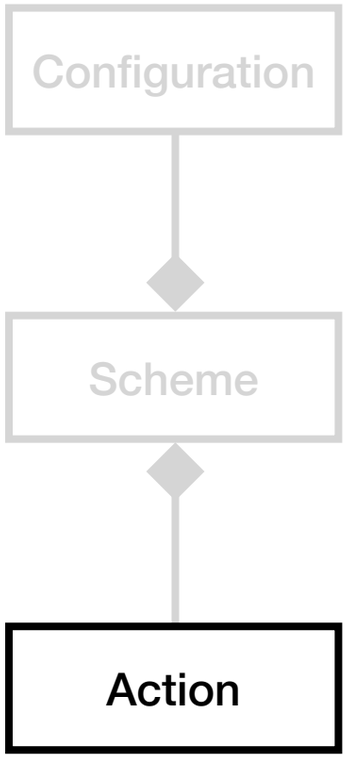
Xcode Buildprozess



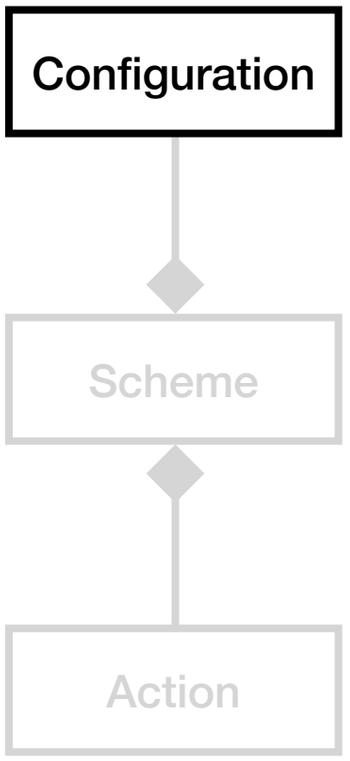
Action



- Unterschiedliche Verwendungen eines Builds
- **Scheme** definiert für jede Action ein **Target** und eine **Build Configuration**
- "Test" startet Unit Tests
- "Profile" zur Performancemessung
- "Analyze" für statische Codeanalyse
- "Archive" verpackt die Applikation für die Distribution (z.B. App Store)

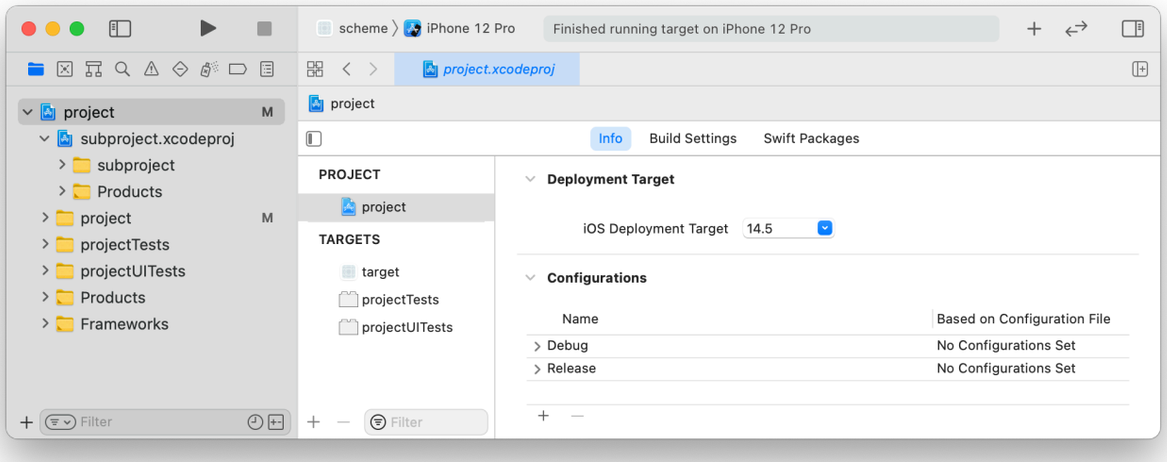


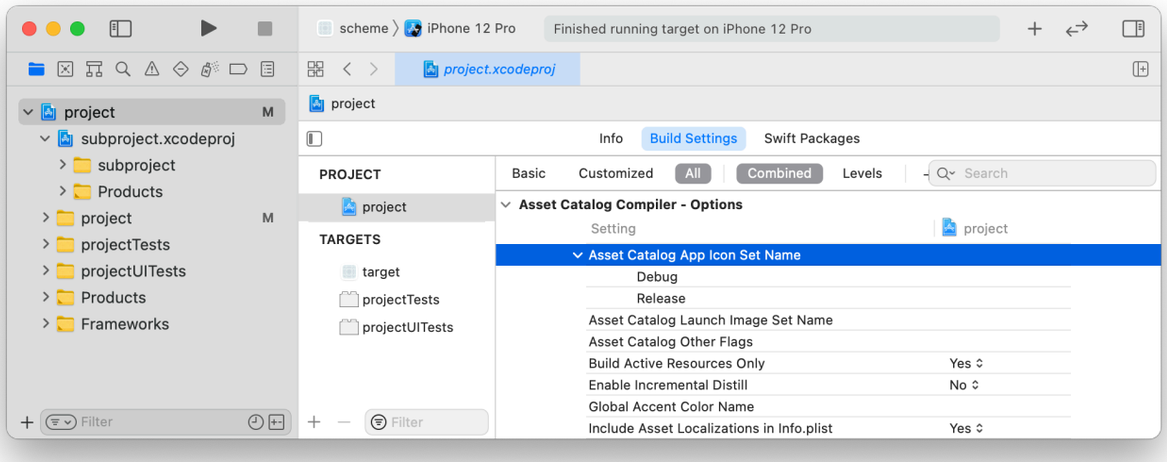
Configuration



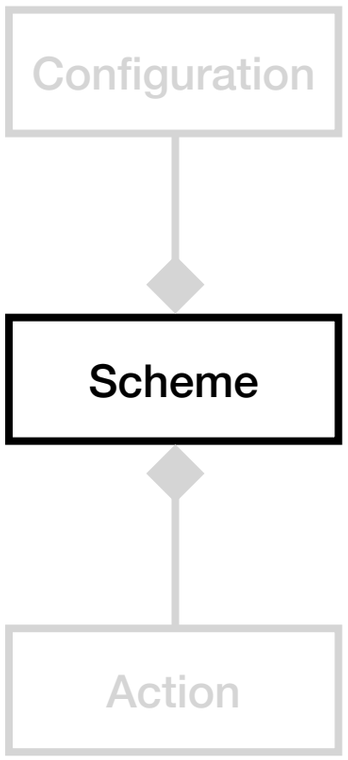
- Spezifische Build Settings für verschiedene **Environments,** z.B. Debug, Release, QA, …
- Neue Configuration über Info Tab im Project Editor
- *.xconfig **Configuration Files:**

https://help.apple.com/xcode/mac/current/#/dev745c5c974

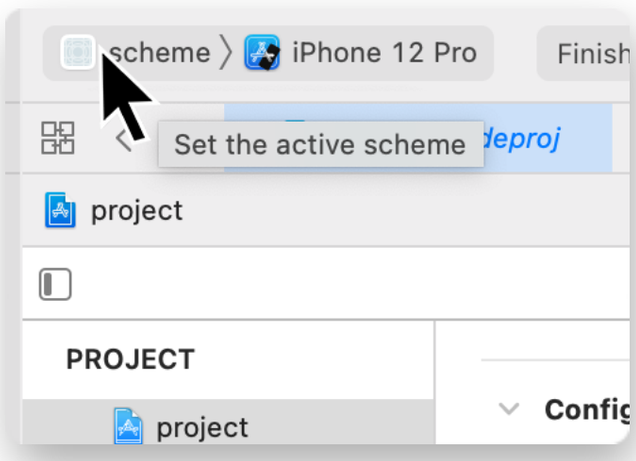


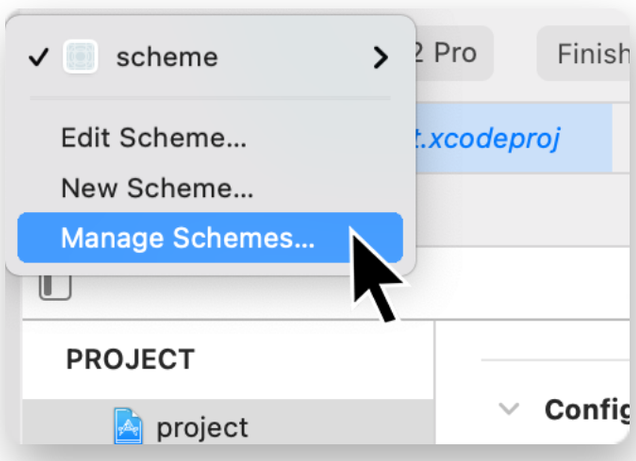


Scheme



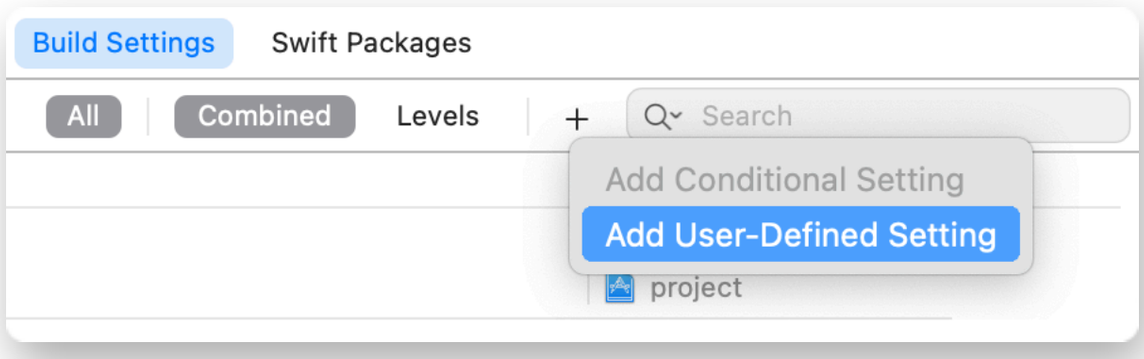
- Verknüpft Actions mit Targets und Configurations
- Setzen von **Argumenten** und **Umgebungsvariablen**
- Blaupause für den gesamten Buildprozess
- Best Practice:
  Ein Scheme pro Configuration/Environment
- Schema links neben **Destination,**
  aber unabhängig davon

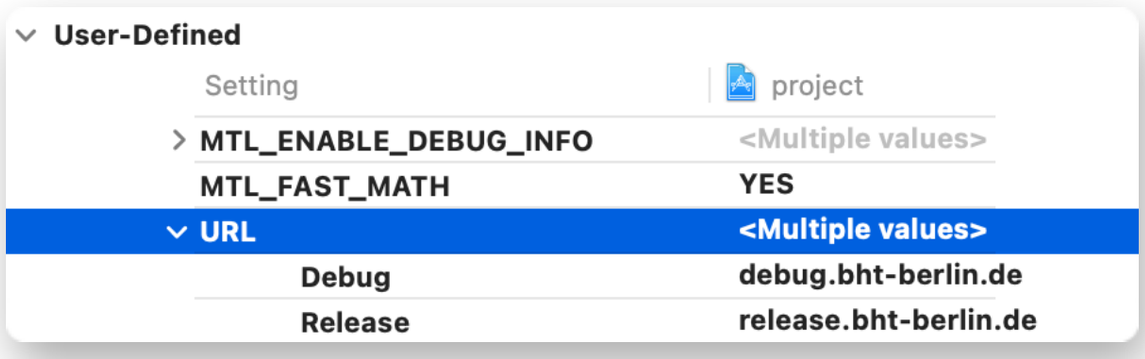


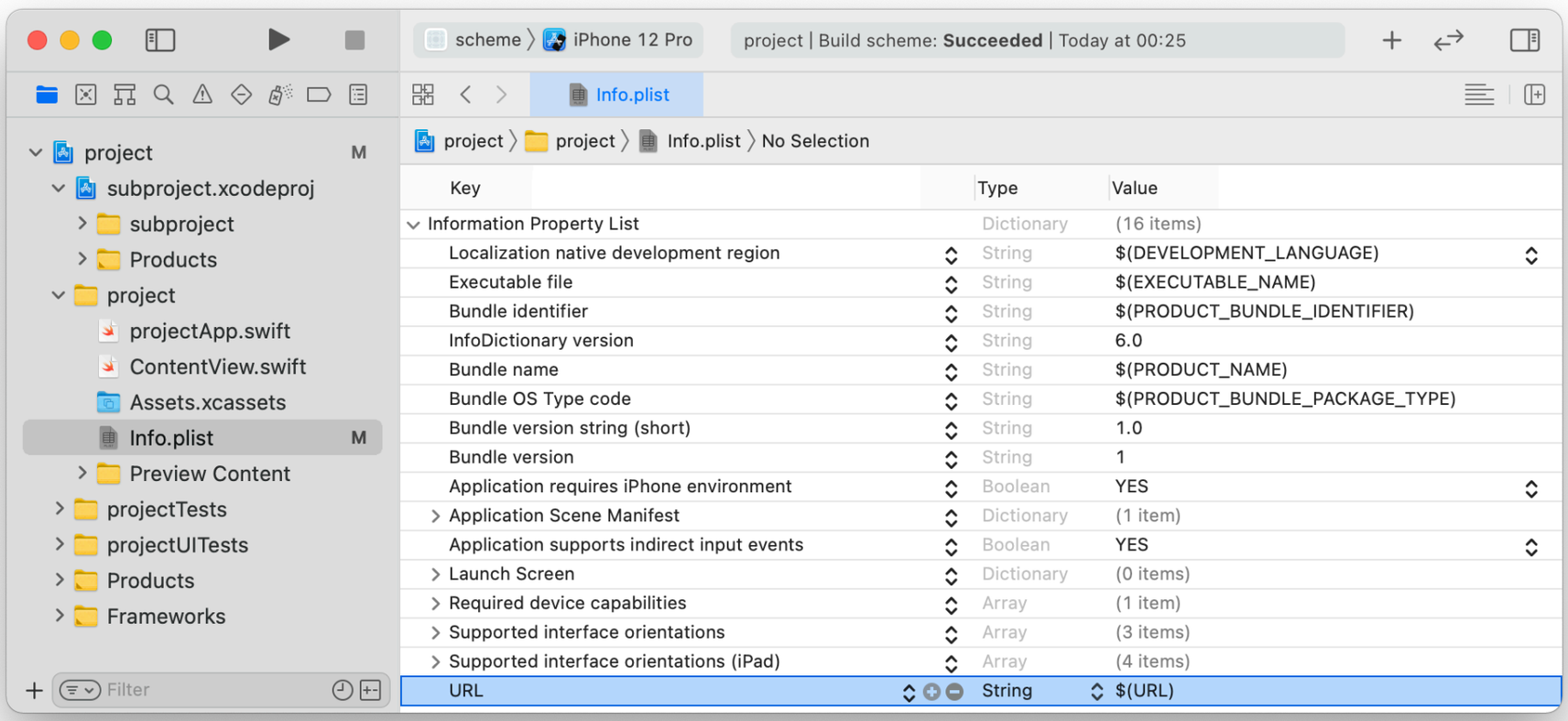


Beispiel

User Defined Setting



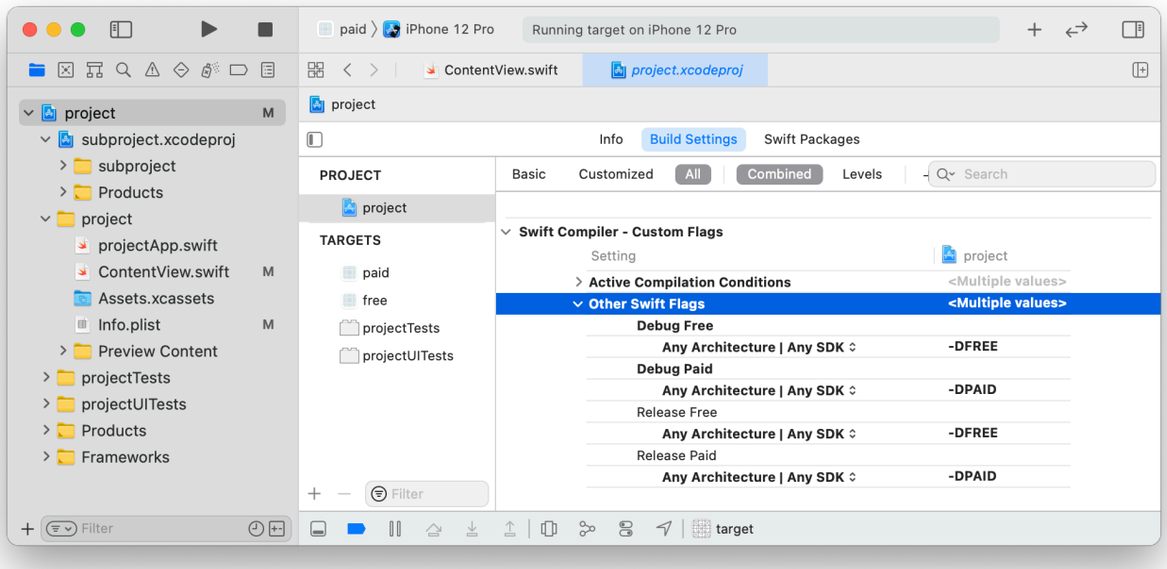


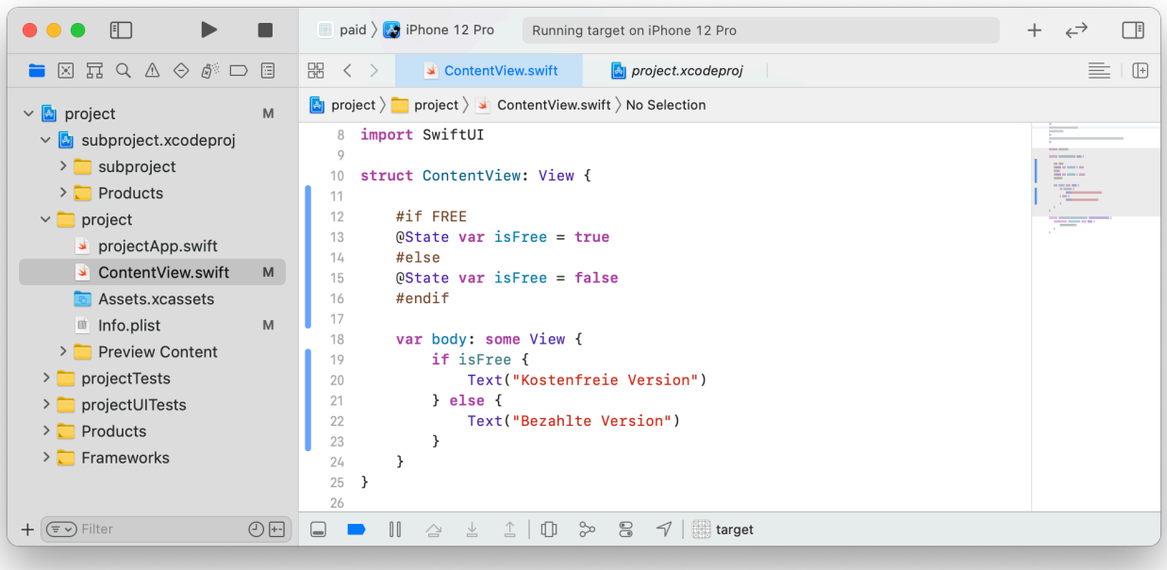


- Compiler setzt Wert in **Info.plist** je nach Umgebung
- Praktisch, aber Daten **nicht sicher.** Besser: *https://cocoacasts.com/tips-and-tricks-managing-build-configurations-in-xocde*

Beispiel

Swift Flag





- Zwei Versionen einer App: "paid" und "free"
- Flags mit **Präfix "-D"**
- Wert der Variable "isFree" wird je nach Umgebung zur Compilezeit bestimmt
- https://medium.com/practical-ios-development/some-practical-uses-for-xcode-build-schemes-and-build-configurations-swift-e50d15a1304f

Quellen

Quellen - Apache ANT

https://de.wikipedia.org/wiki/Apache_Ant

https://www.dev-insider.de/was-ist-ant-a-724722/

https://ant.apache.org/

http://home.edvsz.fh-osnabrueck.de/skleuker/CSI/Werkzeuge/Ant/index.html

https://link.springer.com/content/pdf/bbm%3A978-3-8348-9724-4%2F1.pdf

https://upload.wikimedia.org/wikipedia/commons/thumb/2/2f/Apache-Ant-logo.svg/1200px-Apache-Ant-logo.svg.png

https://openbook.rheinwerk-verlag.de/javainsel9/javainsel_26_005.htm

Quellen - Apache Maven

https://www.torsten-horn.de/techdocs/maven2.htm#Vergleich-Ant

http://home.edvsz.fh-osnabrueck.de/skleuker/CSI/Werkzeuge/Maven/index.html

https://de.wikipedia.org/wiki/Apache_Maven

https://geekflare.com/de/apache-maven-for-beginners/

http://www.lab4inf.fh-muenster.de/lab4inf/docs/HPK/02-HPK-Buildmanagement.pdf

https://upload.wikimedia.org/wikipedia/commons/thumb/5/52/Apache_Maven_logo.svg/1280px-Apache_Maven_logo.svg.png

Quellen - Gradle

https://jaxenter.de/8-build-tools-im-vergleich-ant-buildr-maven-bazel-buck-gradle-pants-sbt-41627

https://de.wikipedia.org/wiki/Gradle

https://docs.gradle.org/current/userguide/what_is_gradle.html

https://de.wikipedia.org/wiki/Dom%C3%A4nenspezifische_Sprache

https://de.wikipedia.org/wiki/Apache_Ivy

https://bio.informatik.uni-jena.de/wp/wp-content/uploads/2018/10/gradle.pdf

https://stackshare.io/gradle

https://www.trustradius.com/products/gradle/reviews?qs=pros-and-cons

https://stackoverflow.com/questions/16754643/what-is-gradle-in-android-studio

https://developer.android.com/studio/build

https://www.geeksforgeeks.org/android-build-gradle/

https://gradle.org/images/gradle-knowledge-graph-logo.png?20170228

https://hazelcast.com/wp-content/uploads/2019/08/diagram-DirectedAcrylicGraph-400x314.png

Quellen - Xcode

https://developer.apple.com/videos/play/wwdc2016/413/?time=2715

https://stackoverflow.com/questions/21631313/xcode-project-vs-xcode-workspace-differences

https://stackoverflow.com/questions/20637435/xcode-what-is-a-target-and-scheme-in-plain-language

https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/Xcode_Overview/CreatingProjects.html

https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/Xcode_Overview/WorkingwithTargets.html

https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/Xcode_Overview/ManagingSchemes.html

https://developer.apple.com/library/archive/documentation/ToolsLanguages/Conceptual/Xcode_Overview/BuildingYourApp.html

Quellen - Xcode

https://www.dev2qa.com/how-to-configure-xcode-default-workspace-project-build-directory-location/

https://help.apple.com/xcode/mac/current/#/dev745c5c974

https://medium.com/flawless-app-stories/managing-different-environments-using-xcode-build-schemes-and-configurations-af7c43f5be19

https://cocoacasts.com/tips-and-tricks-managing-build-configurations-in-xocde

https://medium.com/practical-ios-development/some-practical-uses-for-xcode-build-schemes-and-build-configurations-swift-e50d15a1304f

https://www.clinkitsolutions.com/introduction-to-xcode-build-configuration/

https://www.ralfebert.de/ios/xcode-aufbau-build/