

Über Ant und Maven zu SBT und Gradle

Persönliche Build-Höllen für Jedermann

Andreas Hartmann & Dr. Halil-Cem Gürsoy





Andreas Hartmann [hartmann@adesso.de]

Principal Software Engineer

Tätigkeitsschwerpunkte:

- ▶ Leichtgewichtige Softwarearchitekturen und Frameworks auf Basis der JEE Plattform
- ▶ Serviceorientierte Architekturen und Portaltechnologien im Kontext der Versicherungs- und Banken-Branche



Dr. Halil-Cem Gürsoy

Senior Software Engineer

Tätigkeitsschwerpunkte:

- ▶ SOA und Integrationsprojekte auf Basis von JEE / Spring
- ▶ Build- & Konfigurationsmanagement

Buildmanagement Use Cases

Ant

Maven

Gradle

SBT

Conclusion

Buildmanagement Use Cases

Ant

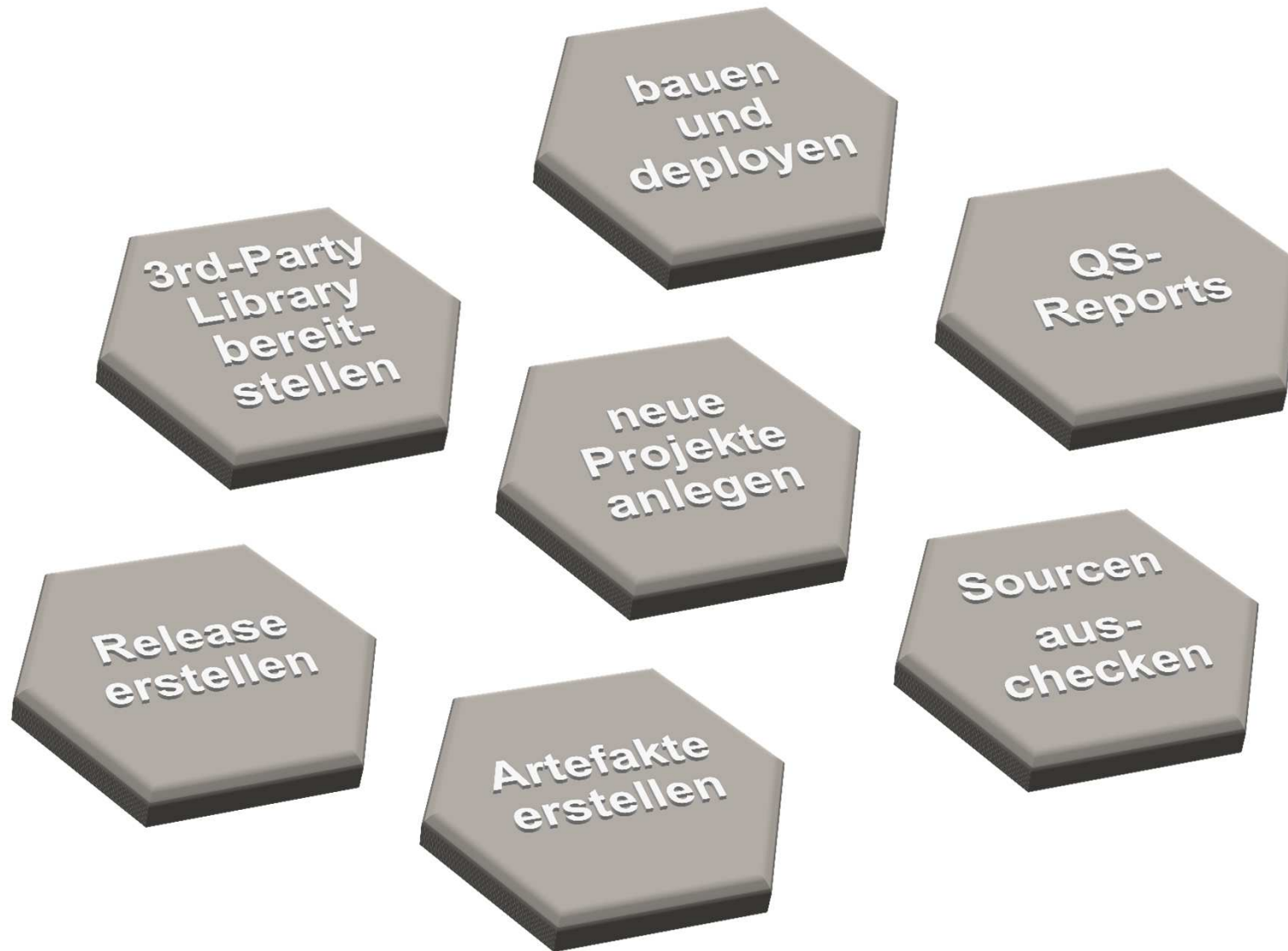
Maven

Gradle

SBT

Conclusion

Buildmanagement Use Cases



Buildmanagement Use Cases

Ant

Maven

Gradle

SBT

Conclusion

Imperative Ansatz

Target ↔ Funktionen

Tasks ↔ Aktionen

- ▶ javac
- ▶ delete
- ▶ mkdir
- ▶ junit
- ▶ . . .



- ▶ Programmieren in XML
- ▶ keine Vorgaben, wie die Ressourcen eines Ant-Scripts strukturiert sein sollen (src, dist, lib - Ordner)
- ▶ keine Standard für Target-Namen (Build, Run, Compile, usw.)
- ▶ kein Dependency-Management
- ▶



Ant – Repository Server

- ▶ Welche Bibliotheken werden in welcher Version wofür benötigt:
- ▶ Welche Abhängigkeiten habe ich zur Compile, Runtime und Test
- ▶ Wie kann ich meine Abhängigkeiten effizient Verwalten
- ▶ Wie kann ich Versionskonflikte zwischen den Bibliotheken einfacher identifizieren
- ▶ Wie kann ich leichter Reproduzierbarkeit von Builds sicherstellen
- ▶ Wie gestalte ich meine Buildskripte übersichtlich und wartungsfreundlich



Agenda

Buildmanagement Use Cases

Ant

Maven

Gradle

SBT

Conclusion



Angry Bill

tech talk radio

HOME PUBLISHINGS WHY ANGRY BILL?

Bill Burke

JBoss old timer, Red Hatter, and successful open source entrepreneur. Co-wrote two books as well as a few other in print and online publications. Husband, father of two, and New England Patriots season ticket holder.

Blogs I m

JBoss Blog
Marc and Na
Mark Little
Sacha Labourey
Steve Vinoski
Russo and Telrod
Savio Rodrigues
Andy Oliver
Bob Lee
Fake Steve
Ryan McDonough
Mark Baker

« [Scannotation fix for /WEB-INF/classes](#)

[Resteasy Project: JAX-RS Restful Web Services implementation](#) »

Maven would be cool if...

Posted by [billburke](#) on February 22, 2008

Maven would be cool if the plugins weren't so god awful! I mean, are these plugin developers idiots? Do they even use their crap? I just spent a good day trying to get the maven-ear-plugin

I just can't believe people haven't cleaned up this shit. Are people really using Maven? I WANT to like Maven, I WANT to use Maven. Its too bad its so freakin painful.

I just can't believe people haven't cleaned up this shit. Are people really using Maven? I WANT to like Maven, I WANT to use Maven. Its too bad its so freakin painful.

This entry was posted on February 22, 2008 at 4:12 pm and is filed under [java](#), [maven](#). You can follow any responses to this entry through the [RSS 2.0](#) feed. You can [leave a response](#), or [trackback](#) from your own site.

- ▶ Repositories = Instabil
- ▶ Transitive Dependencys
- ▶ Lizenzen!
- ▶ Interne Plugins
 - > „*The latest and greatest*“
 - > Maven A != Maven B

**Instabile,
nicht reproduzierbare Builds!**

Buildmanagement Use Cases

Ant

Maven

Gradle

SBT

Conclusion

Buildsprache basiert auf Groovy

- ▶ Initiator: Hans Dockter
- ▶ Projektseite: <http://gradle.org/>

Gradle
a better way to build

niedrige Einstiegshürde



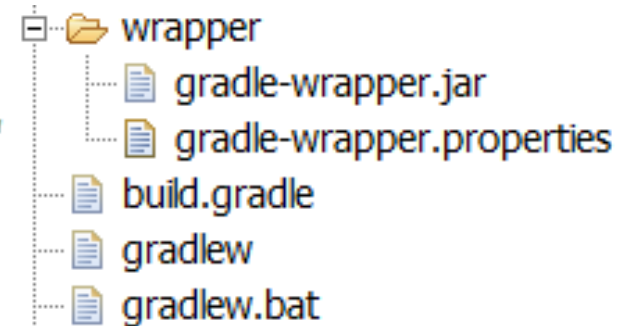
*apply plugin:
'java'*

- ▶ Convention over Configuration – Standardkonventionen basieren auf Maven
- ▶ Pluginkonzept – geeignet für die diversen Sprachen Java, Groovy, Scala
- ▶ Repository Enabled
 - > filebasiert oder Maven Repository
 - > automatisierte POM Erstellung
- ▶ taskbasiert und leicht erweiterbar – `doFirst/doLast`
- ▶ Konfiguration der Tasks (deklarativ)
- ▶ Tasktypen definieren das wie (imperativ)

- ▶ Abhängigkeitsstruktur der Tasks wird als DAG aufgebaut
 - > Hook Methoden im Buildlifecycle
- ▶ deterministische sequentielle Abarbeitung
- ▶ beliebig viele Artefakte pro Projekt
- ▶ Inkrementelle Builds
- ▶ Zugriff auf das Gradle Objektmodell
- ▶ Multi-Project Builds
- ▶ Ant Integration

▶ Gradle Wrapper

```
task wrapper(type: Wrapper) {  
    gradleVersion = '1.0-milestone-1'  
    jarPath = 'wrapper'  
}
```



▶ Testing enabled

- > Parallele Unit Tests
- > Seperate JVM für Unit Test
- > Neustart der JVM nach X Test konfigurierbar
- > Debug Modus startbar

```
test {  
    forkEvery = 42  
    maxParallelForks = 8  
    debug = true  
}
```

Gradle – build.gradle

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

// Maven Project configuration
version = '1.0-SNAPSHOT'
group = 'adesso'
artifactId = project.name.toLowerCase()

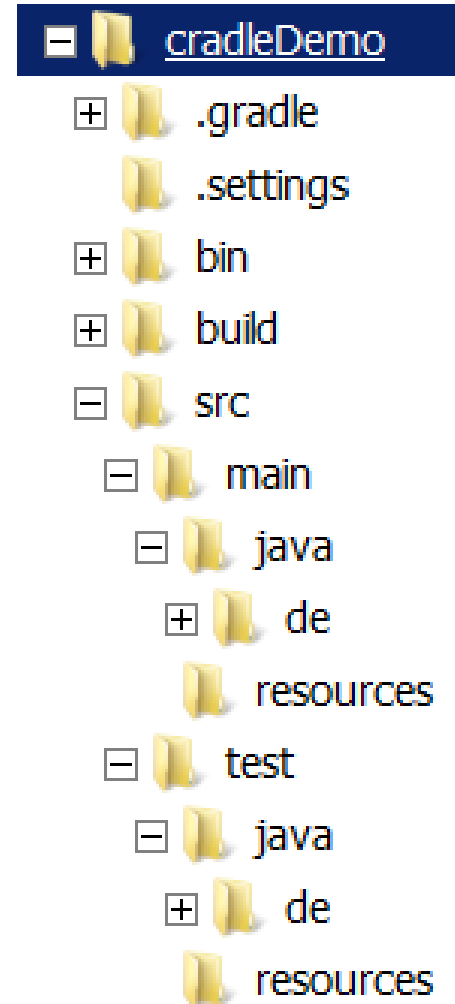
configurations {
    deployerJars
}

repositories {
    mavenRepo url: "http://127.0.0.1:8080/nexus/content/repositories/central/"
}

dependencies {
    testCompile group: 'junit', name: 'junit', version: '4.+'
    deployerJars "org.apache.maven.wagon:wagon-webdav-jackrabbit:1.0-beta-6"
}

sourceCompatibility = 1.6
jar {
    baseName=artifactId
    manifest {
        attributes 'Implementation-Title': 'Gradle Demo', 'Implementation-Version': version
    }
}

uploadArchives {
    repositories {
        deployer = mavenDeployer {
            configureAuth = {
                authentication(userName: 'admin', password: 'admin123')
            }
            configuration = configurations.deployerJars
            snapshotRepository(url: "http://127.0.0.1:8080/nexus/content/repositories/snapshots/", configureAuth)
            repository(url: "http://127.0.0.1:8080/nexus/content/repositories/releases/", configureAuth)
        }
    }
}
```



Gradle – User Interface

The image shows two windows illustrating the Gradle user interface. On the left is a Windows command prompt window titled 'C:\Windows\system32\cmd.exe - gradle --gui'. It shows the command 'D:\Technologie_Evaluierung\cradleDemo>gradle -?' and the resulting help text for the 'gradle' command. On the right is the 'Gradle' GUI application window. The 'Task Tree' tab is active, displaying a tree view of tasks for the 'cradleDemo' project. Below the task tree, a status bar indicates 'Completed successfully at 8:23:37 PM'. The main content area shows the output of the 'install' task, including a list of rules and a 'BUILD SUCCESSFUL' message.

```
C:\Windows\system32\cmd.exe - gradle --gui
D:\Technologie_Evaluierung\cradleDemo>gradle -?
USAGE: gradle [option...] [task...]

-?, -h, --help           Shows this help message
-a, --no-rebuild         Do not rebuild project dependencies
-b, --build-file         Specifies the build file (Gradle build script)
-C, --cache              Specifies how compiled build artifacts are used
-c, --settings-file     Specifies the settings file (Gradle settings script)
-D, --system-prop       Set system property of the JVM
-d, --debug             Log in debug mode (includes stack traces)
--daemon               Uses the Gradle daemon to speed up execution
-e, --embedded          Specify an embedded build
--foreground          Starts the Gradle daemon in the foreground
-g, --gradle-user-home  Specifies the gradle user home directory
--gui                 Launches a GUI application
-I, --init-script        Specifies an initialization script
-i, --info              Set log level to info.
-m, --dry-run           Runs the builds with all tasks up to date
-n, --dependencies      Show list of all project dependencies
--no-color             Do not use color in the output
--no-daemon           Do not use the Gradle daemon
--no-opt              Ignore any task optimizations
-P, --project-prop      Set project property for the build
-p, --project-dir       Specifies the start directory
--profile             Profiles build execution
-q, --quiet            Log errors only.
-r, --properties        Show list of all available properties
-S, --full-stacktrace  Print out the full (very verbose) stack trace
-s, --stacktrace       Print out the stack trace
--stop               Stops the Gradle daemon
-t, --tasks            Show list of available tasks
-u, --no-search-upward Don't search in parent folders
-U, --version          Print version info.
-x, --exclude-task     Specify a task to be excluded
```

Gradle Task Tree Favorites Command Line Setup

Description

- cradleDemo
 - assemble Assembles all Jar, War, Zip, and Tar archives.
 - build Assembles and tests this project.
 - buildDependencies Assembles and tests this project and all projects that depend on it.
 - buildNeeded Assembles and tests this project and all projects it depends on.
 - check Runs all checks.
 - classes Assembles the main classes.
 - clean Deletes the build directory.
 - compileJava Compiles the main Java source.
 - compileTestJava Compiles the test Java source.
 - install Does a maven install of the archives artifacts into the local .m2 cache.
 - jar Assembles a jar archive containing the main classes.
 - javadoc Generates Javadoc API documentation for the main source code.
 - processResources Processes the main resources.
 - processTestResources Processes the test resources.
 - test Runs the unit tests.
 - testClasses Assembles the test classes.
 - uploadArchives Uploads all artifacts belonging to configuration ':archives'.

Refresh ×

Completed successfully at 8:23:37 PM

```
install - Does a maven install of the archives artifacts into the local .m2 cache.
Rules
-----
Pattern: build<ConfigurationName>: Assembles the artifacts of a configuration.
Pattern: upload<ConfigurationName>: Assembles and uploads the artifacts belonging to
Pattern: clean<TaskName>: Cleans the output files of a task.

To see all tasks and more detail, run with --all.

BUILD SUCCESSFUL

Total time: 2.293 secs

Completed Successfully
```

Buildmanagement Use Cases

Ant

Maven

Gradle

SBT

Conclusion

„Simple Build Tool“

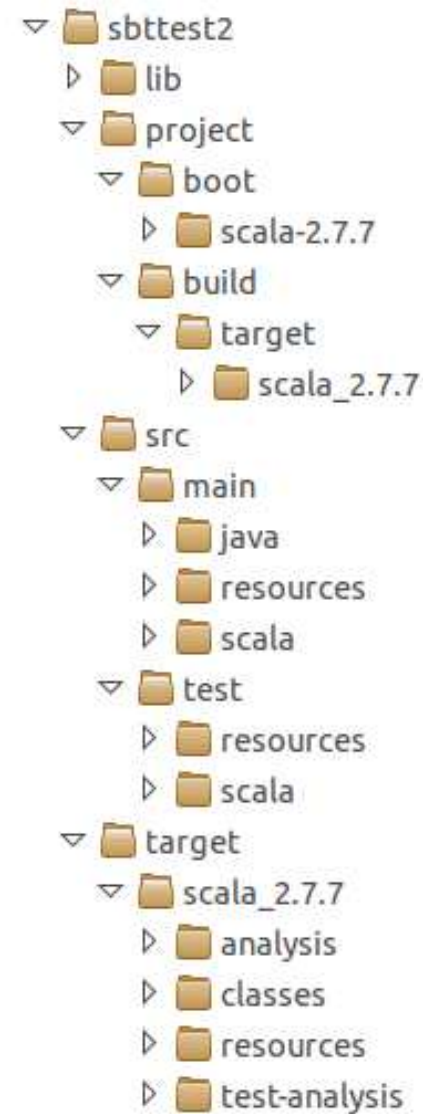
- ▶ In Scala implementiert
- ▶ Für Scala-Projekte ... aber auch Java!
- ▶ Inzwischen Teil des Typesafe-Stacks



- ▶ Projekt sehr einfach zu initialisieren:
 - > JAR herunterladen
 - Home: <https://github.com/harrah/xsbt/wiki>
 - > Starten... warten...



- ▶ Konfiguration
 - > In Scala-Klassen
- ▶ Convention over Configuration
 - > Dokumentation ?
- ▶ Erweiterungen
 - > In Scala
 - > Plugins
 - > Processors
 - > Actions



- ▶ Simple Konfiguration durch Konfigurationsdatei mit eigener DSL
- ▶ Ab 0.10.x, bis 0.7.x nur über Scala-Klassen

```
name := "ScalaTrain",  
version := "0.1",  
organization := "de. adesso.hgu",  
javacOptions += Seq("-source", "1.6", "-target", "1.6")  
scalacOptions += "-deprecation",  
scalaVersion := "2.8.1",  
libraryDependencies += "org.mockito" % "mockito-all" % "1.8.5" % "test",  
libraryDependencies += "org.scala-tools.testing" %% "specs" % "1.6.7",  
libraryDependencies += "org.scala-tools.testing" %% "scalacheck" % "1.8"
```

- ▶ Einfache eigene Tasks innerhalb der Projektkonfiguration
 - > Klasse unter project/build/

```
import sbt._

object MyBuild extends Build {
  lazy val root = Project("root", file(".")) aggregate(sub1)

  configuration.
    lazy val sub1: Project = Project("sub1", file("a"))
      dependsOn(sub2 % "test")

  project.
    lazy val sub2 = Project("sub2", file("b"), delegates = root ::
      Nil)
}
```

- ▶ Dependency Management
 - > Manuell möglich (lib-Verzeichnis)
 - > POM, Ivy
 - > Konfigurationen

```
libraryDependencies += groupId % artifactID % revision
```

- ▶ Transitive Dependencys ausklammern

```
libraryDependencies += "org.apache.hadoop" % "hadoop-core"  
    % "0.20.2" intransitive()
```

- ▶ Repositorys definieren

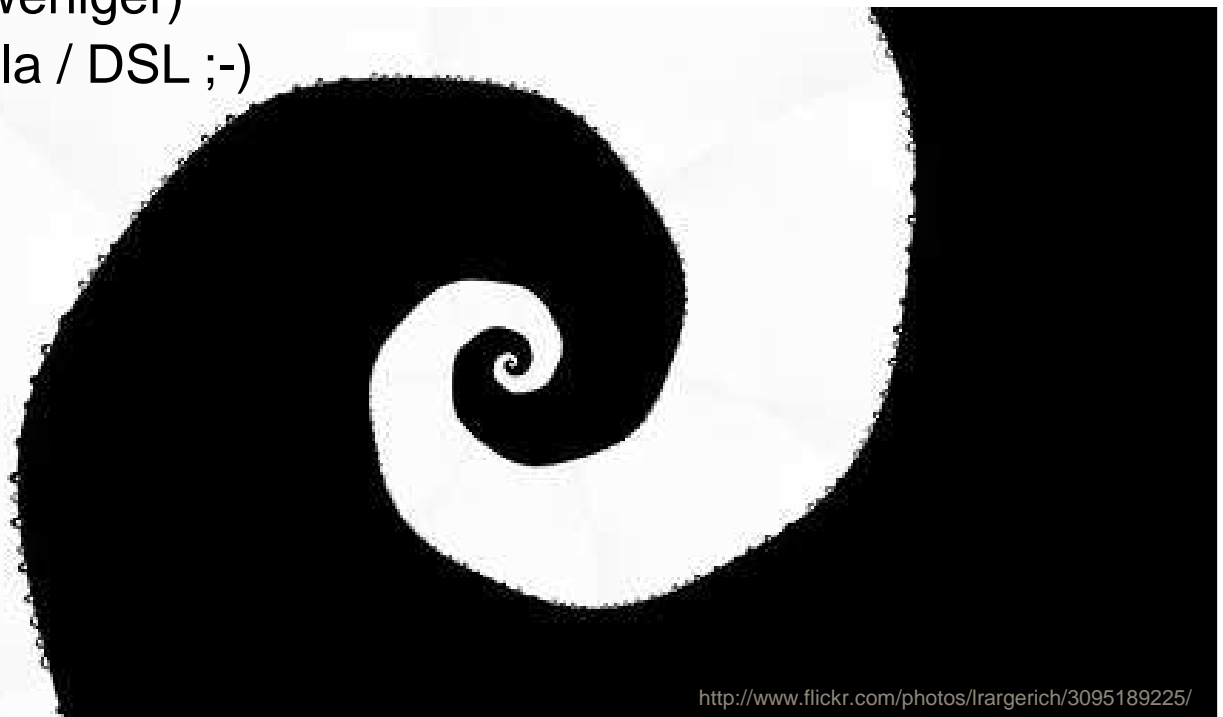
```
scala resolvers += "Scala-Tools Maven2 Snapshots Repository" at  
    "http://scala-tools.org/repo-snapshots"
```

- ▶ Publishing

- > Abhängig von Ivy
- > Viele Randbedingungen zu beachten
 - wann zieht welche Konfiguration?

- ▶ Tiefe Ivy-Kenntnisse nötig!

- ▶ Vorteile
 - > Kein XML, Konfiguration in eigener DSL bzw. Scala
 - > Programmieren
 - > Verzeichnisstrukturen
- ▶ Nachteile (mehr oder weniger)
 - > Konfiguration in Scala / DSL ;-)
 - > Lernkurve
 - > Dokumentation
 - > Ivy-Wissen
 - > Schwache IDE Integration



<http://www.flickr.com/photos/lrargerich/3095189225/>

Agenda

Buildmanagement Use Cases

Ant

Maven

Gradle

SBT

Conclusion

Pest oder Cholera?

- ▶ Ant und Maven haben ihre Schwächen
- ▶ **SBT** ist aussichtsreich
 - > Dokumentation der Defaults stark verteilt
 - > Erweiterbarkeit gut
 - > Aktuell nur in der Scala-Welt „sichtbar“
 - > Schlechte IDE-Integration
- ▶ **Gradle**, der Anwärter
 - > Erweiterbarkeit recht einfach
 - > Gut lesbare Konfiguration
 - > Gute Unterstützung von Multi-Modul Projekten
 - > **Aussichtsreichster Kandidat**



Wir suchen Sie als

- Software-Architekt (m/w)
- Projektleiter (m/w)
- Senior Software Engineer (m/w)

jobs@adesso.de
www.AAAjobs.de

