5.– 8. September 2011
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Am Ziel angekommen?

Über Ant und Maven zu SBT und Gradle

Andreas Hartmann
Dr. Halil-Cem Gürsoy
adesso AG

adesso business. people. technology.

Über Ant und Maven zu SBT und Gradle

Persönliche Build-Höllen für Jedermann

Andreas Hartmann & Dr. Halil-Cem Gürsoy

Speaker

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

Agenda

- Buildmanagement Use Cases
- Ant
- Maven
- Gradle
- SBT
- Conclusion

Agenda

Buildmanagement Use Cases

Ant

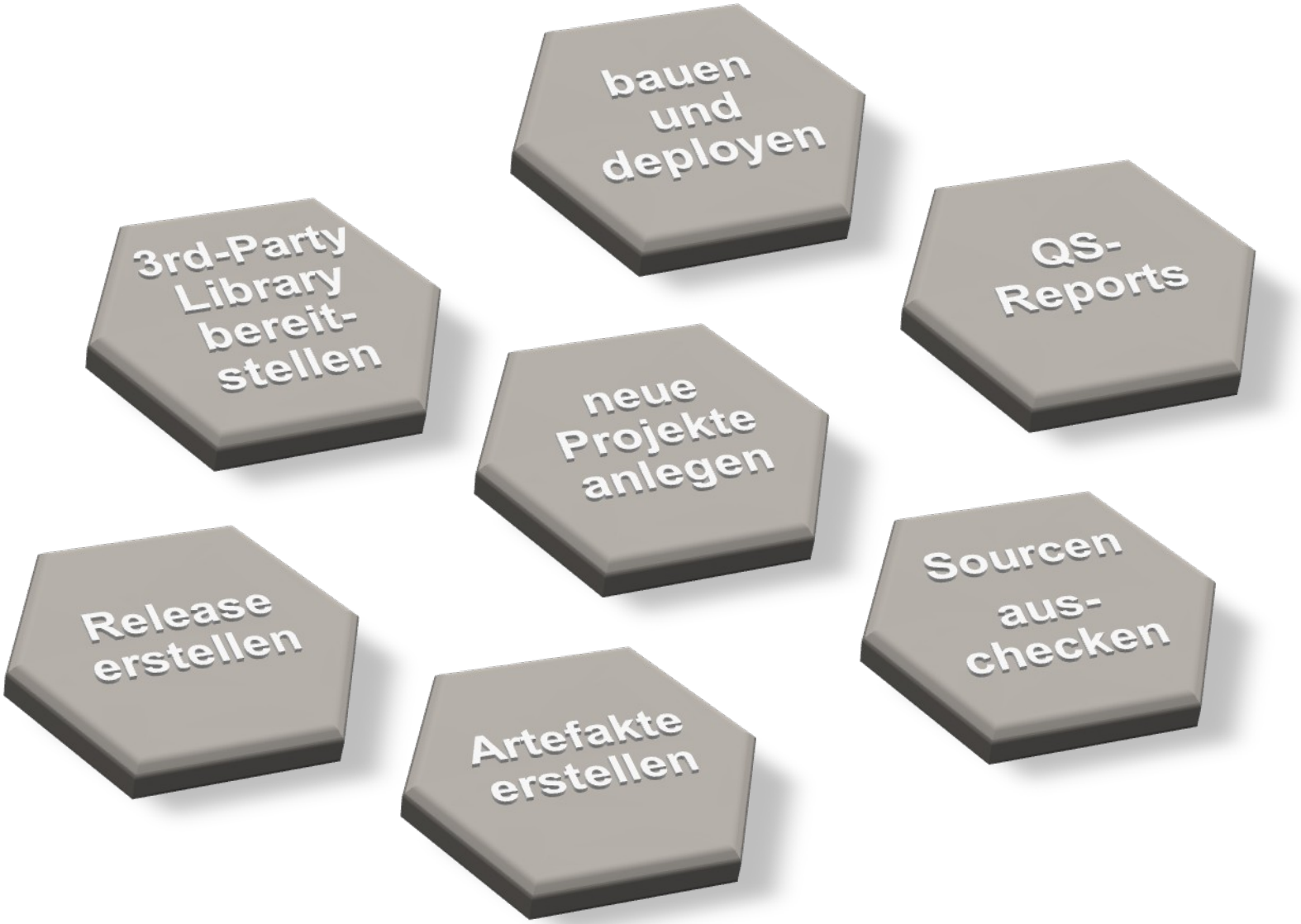
Maven

Gradle

SBT

Conclusion

Buildmanagement Use Cases



Agenda

- Buildmanagement Use Cases
- **Ant**
- Maven
- Gradle
- SBT
- Conclusion

Ant - Kernfeatures

Imperative Ansatz

Target ⇔ Funktionen

Tasks ⇔ Aktionen

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

Ant - Kritikpunkte

- 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 gestallte ich meine Buildskripte übersichtlich und wartungsfreundlich

Agenda

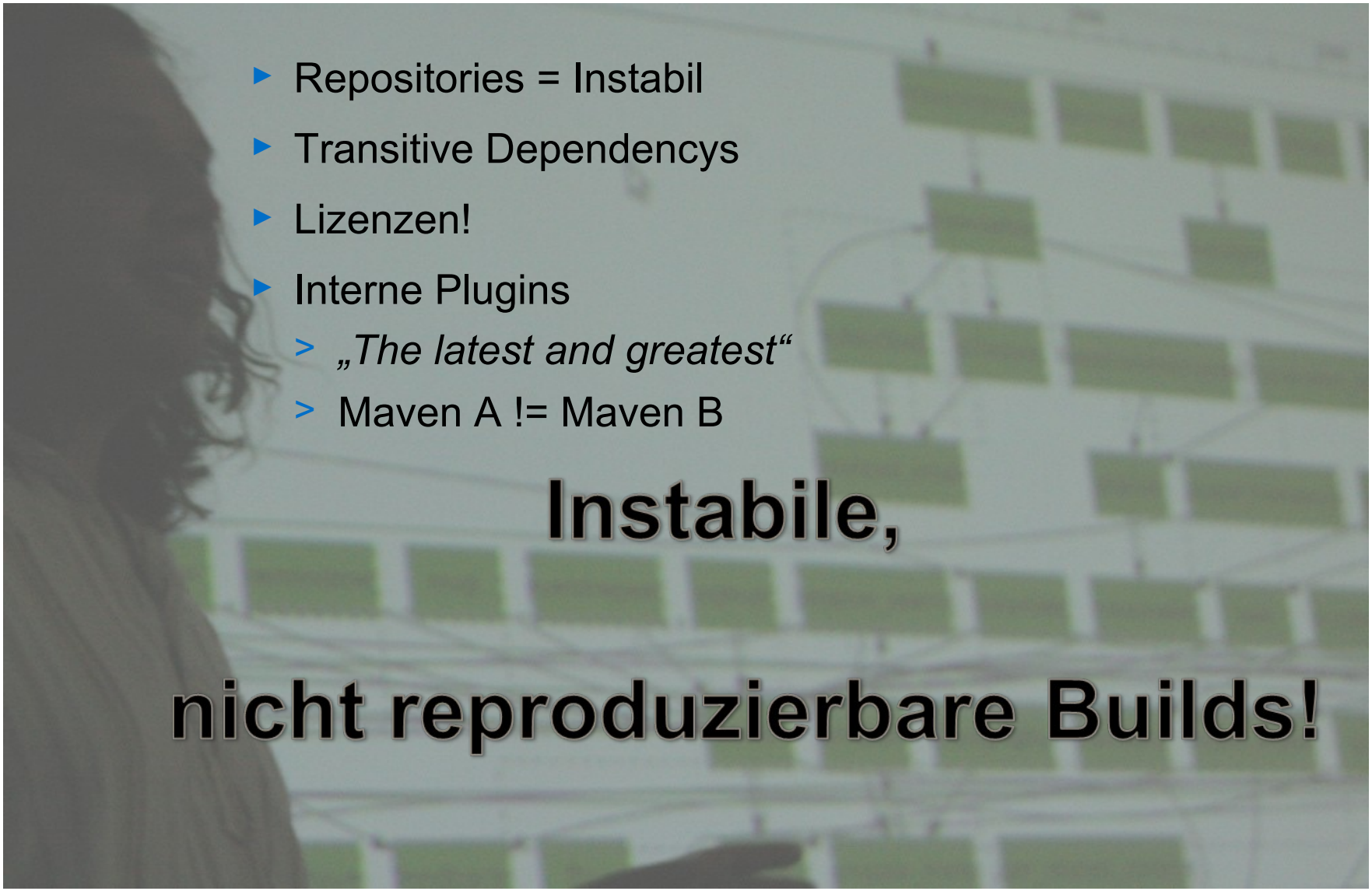
- Buildmanagement Use Cases
- Ant
- **Maven**
- Gradle
- SBT
- Conclusion

Maven

Maven


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.

Maven


Repositories = Instabil
Transitive Dependencys
Lizenzen!
Interne Plugins
„The latest and greatest“
Maven A != Maven B
Instabile,
nicht reproduzierbare Builds!

Agenda

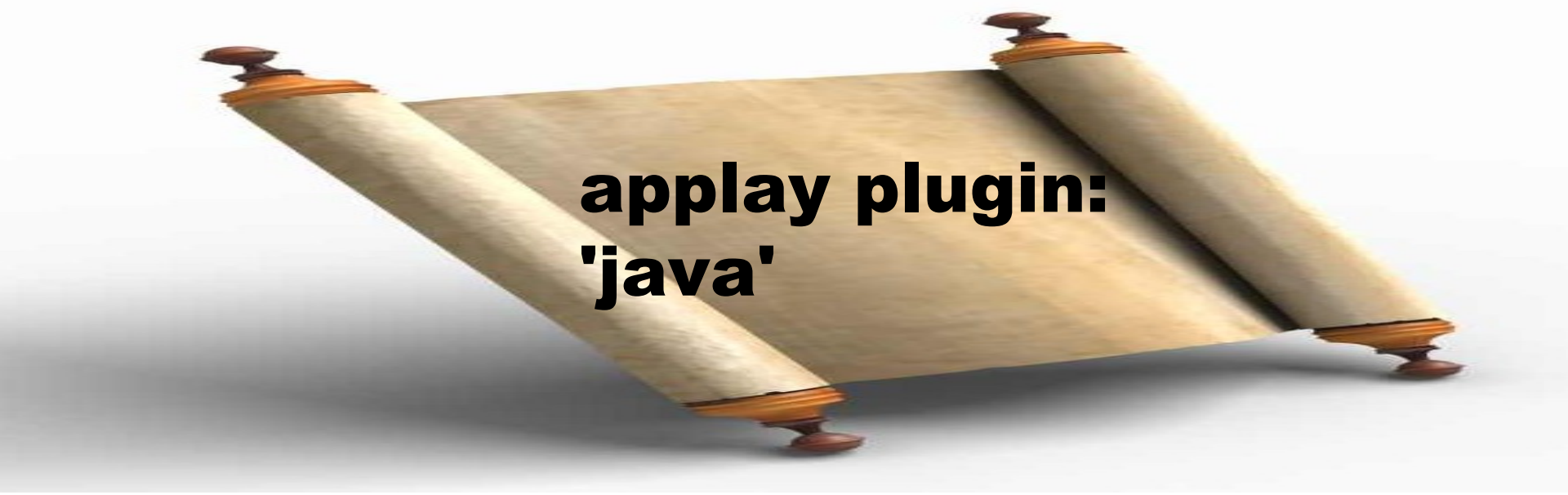
- Buildmanagement Use Cases
- Ant
- Maven
- **Gradle**
- SBT
- Conclusion

Buildsprache basiert auf Groovy

- Initiator: Hans Dockter
- Projektseite: http://gradle.org/

Gradle
a better way to build

niedrige Einstiegshürde



Gradle - Essentials

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

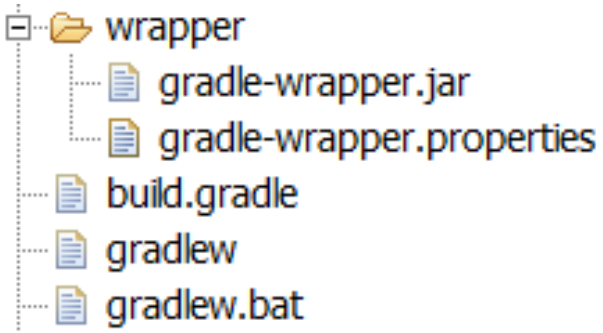
Gradle - Essentials

- 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 - Essentials

- Gradle Wrapper

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



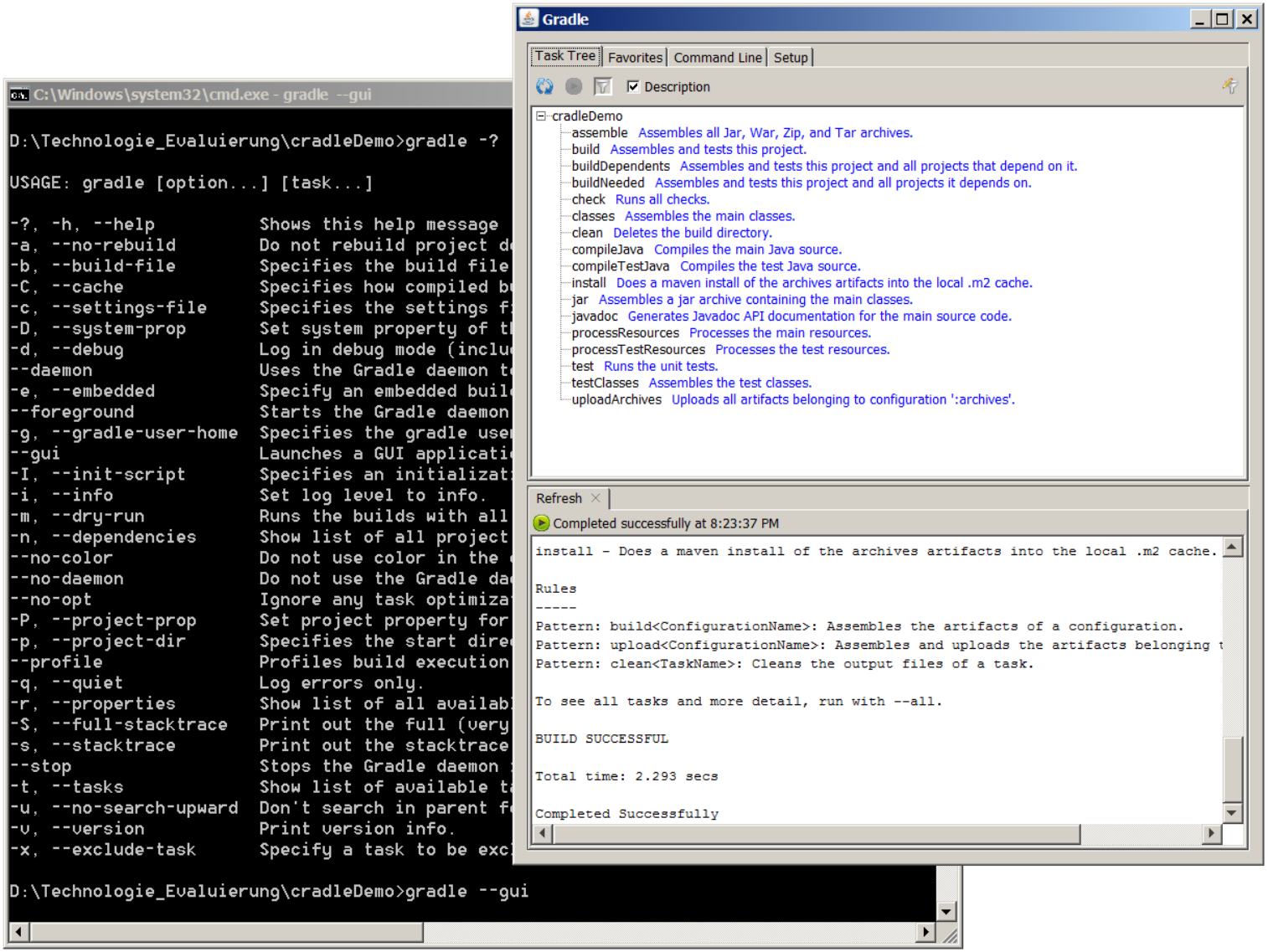
- Testing enabled
  - Parallele Unit Tests
  - Seperate JVM für Unit Test
  - Neustart der JVM nach X Test konfigurierbar

```
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 urls: "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



Agenda

- 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

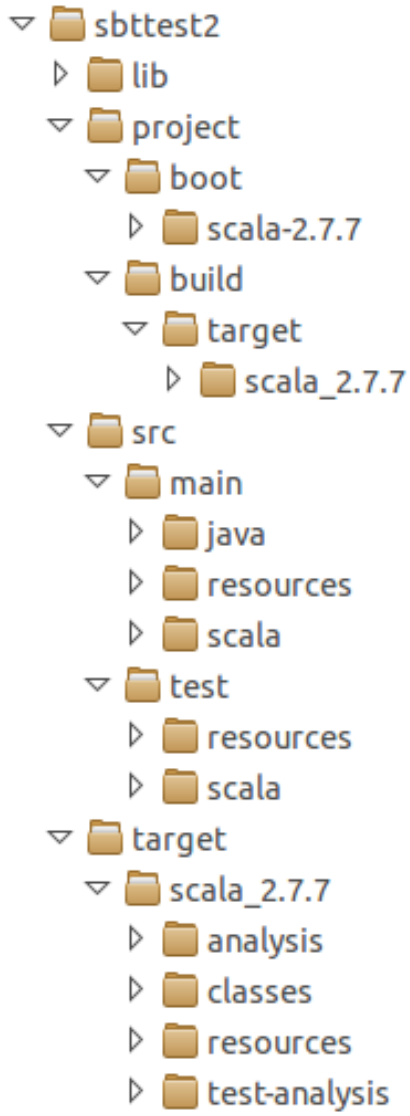
SBT - Essentials

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

http://www.flickr.com/photos/carbonnyc/64581364/

SBT - Essentials

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



Basic Configuration

- Simple Konfiguration durch Konfigrationsdatei 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"
```

SBT – Full Configuration

- 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)
}
```

SBT - Dependencys

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

```
libraryDependencies += groupID % artifactID % revision
```

SBT - Dependencys

- Transitive Dependencys ausklammern

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

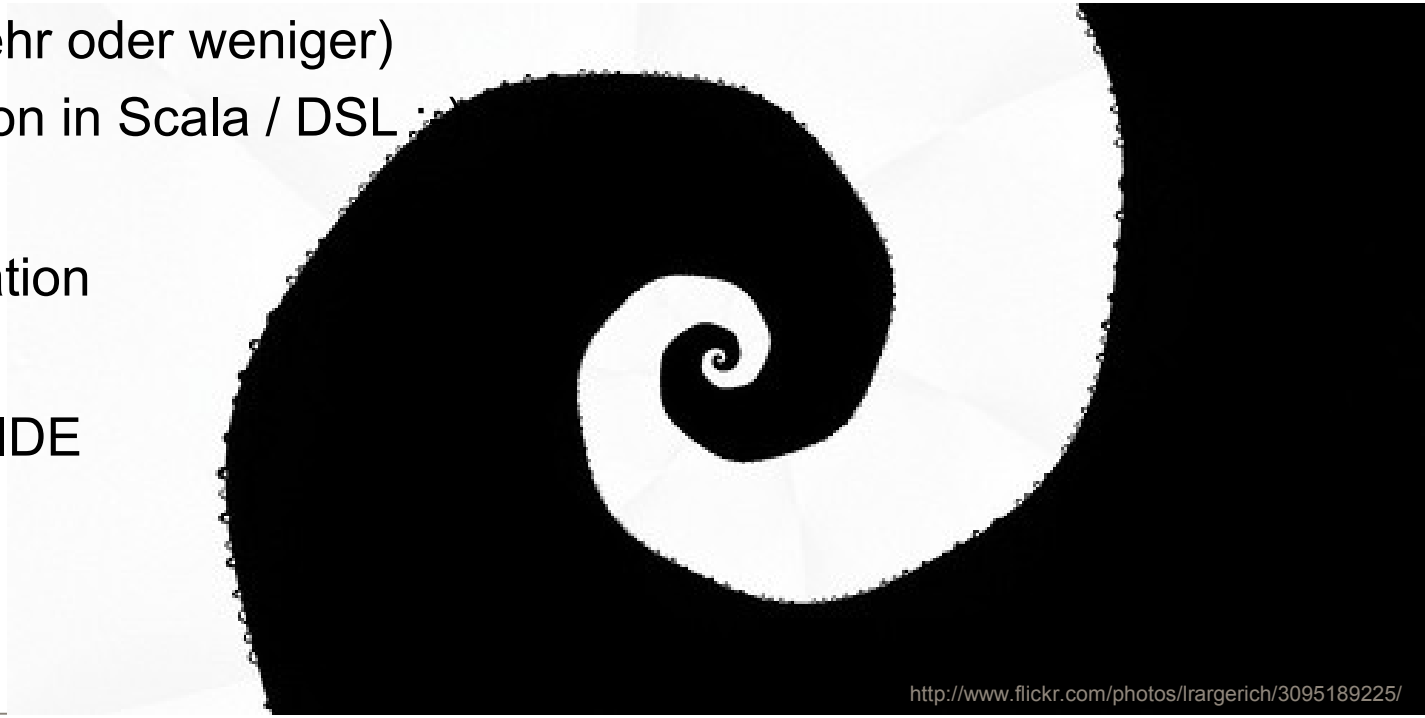
- Repositorys definieren

```
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!

SBT

- 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**

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**

adesso business. people. technology.

Wir suchen Sie als

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

jobs@adesso.de
www.AAAjobs.de