University of Basel

Gradle

Silvan Heller <silvan.heller@unibas.ch>

Department Mathematik & Informatik, Universität Basel

HS17– Software Engineering

Warum brauchen wir ein Build System?

- `javac -cp libs/junit-4.5.jar:libs/slf4j-2.1.jar *.java && jar cmf my_manifest project *.class`
- ’Manually add our 50 dependencies by hand via the user Interface of your IDE’
- ’Before executing the tests, you need to manually run scripts/setup-test.sh’

Manuelle Builds: Schwächen

- Unübersichtlich
- Systempfadabhängig
- `javac` hat eine unübersichtliche Anzahl an Optionen
- Wer ruft den Build Befehl auf?
- Manuelles Verwalten von verwendeten Libraries

Die Zeit des Entwicklers ist eine teure Ressource

Warum nicht einfach Eclipse oder IntelliJ?

Eclipse kann

- Automatisch jars erstellen

IDEs können meistens nicht (automatisch)

- Libraries verwalten
- Optionen von `javac` abstrahieren
- Von überall aufgerufen werden (ssh auf remote server)
- Dependencies über mehrere Projekte cachen

Die Lösung

Gradle ist relevant!

Adobe

Gradle: Aufbau

- Gradle besteht aus zwei Teilen: **Projekten** und **Tasks**

  Projekt Sammlung von Aufgaben

  Task Eine spezifische Aufgabe. Etwa das kompilieren einer Java Klasse

- Gradle speichert alle Informationen in einer *build.gradle* Datei

Gradle: Ordnerstruktur

Vorgeschlagene Ordnerstruktur

Gradle: Dependencies

- Größere (Java) Projekte benötigen Libraries
- Beispiele: HTTP, JSON, Logging, Testing, Math
- → Zentrales Repository mit verschiedenen Versionen der Library

```
repositories {
    mavenCentral()
}
dependencies {
    compile "joda-time:joda-time:2.2"
    testCompile "junit:junit:4.12"
    compile fileTree(dir: 'lib', include: ['*.jar'])
}
```

- `compile` und `testCompile` ermöglichen testspezifische Abhängigkeiten
- Es können auch jars manuell eingebunden werden

Gradle: Wichtige Befehle (Standard)

- Projekt bauen
  - `./gradlew build`
- Tests ausführen
  - `./gradlew test`
- Projekt bauen und ausführen
  - `./gradlew run`

Gradle in Ganttproject

- ~~Vorgeschlagene Ordnerstruktur~~
- ~~Dependency Management~~
- Gradle Wrapper
- Mehrere Subprojekte, jedes hat ein eigenes build.gradle file
- Gradle wird zum builden verwendet und nicht / nur halb fürs dependency management
- ’./gradlew runApp’ zum starten