
gradle

Build Management Tool?

Funktionen

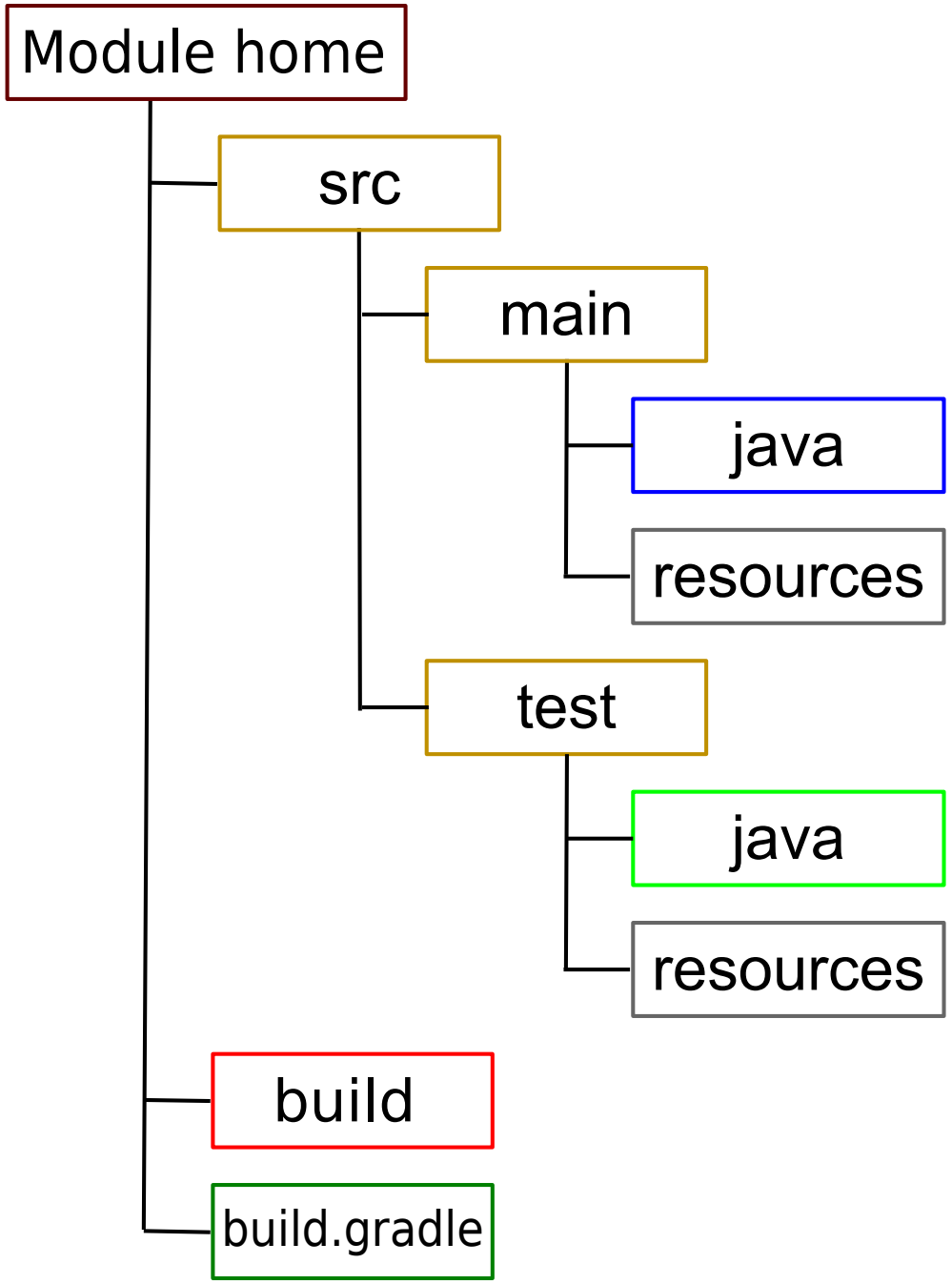
- Dependency Management
- Kompiliert den Sourcecode in Binaries
- Führt automatisierte Tests aus
- Installiert Software oder Lädt sie in Repositories
- Erstellt eine Dokumentation
- Erstellt Software Pakete (Releases/Distributionen)

Vertreter

- Make
- Ant
- Maven
- **Gradle**

Gradle Project Structure

- Im wesentlichen von Maven entliehen
- **src**: source files
  - **main**: java Klassen mit resources
  - **test**: test java Klassen mit resources
- **build**: compilierte files
  - NICHT in die Versionskontrolle
  - Wird automatisch erstellt wenn benötigt



Java Modules vs Packages

Package:

- Dient der Einordnung von Java Klassen in Namespaces um Rechte/Sichtbarkeit zu organisieren

Module:

- Einheit die als eigenständiges **JAR** zur Verfuegung stehen soll
- Einteilung in use cases
- Welche teile meiner software/library moechte ich einzeln bereitstellen
  - Bsp.: API ↔ konkrete Implementierungen
- Der interessante Organisationsmechanismus fuer das Build-System

Begriffe:

Repository (Versionskontrolle)

- Zentraler Server der eure Versionskontrolle bereitstellt

Repository / Artifactory

- Zentraler Server der libraries (jars) öffentlich bereitstellt

Artifact (debloy, publish)

- Einheit/Archiv/Library/Jar in einem Repository

Distribution (debloy, publish)

- Paket das an User ausgeliefert wird
- Starter, Manual, Libs, Installer, Lizenzen

Gradle

- Build System / Build Management Tool
- DSL (Domain-specific language) die auf Groovy Basiert
- Man schreibt scripte und **keine** Config files (Maven → xml)

Gradle

- **build.gradle** (Module Root)
  - Sind die eigentlichen build scripte
  - Beinhalten tasks und deren configuration
  - Erben tasks und configs von obermodulen
  - Hier kann gradle ausgefuehrt werde

- **setting.gradle** (Project Root)
  - Definiert module die von gradle gebaut werden sollen
  - Einige globale optionen

Minimales Projekt

build.gradle

```
group 'de.unijena.bioinf.teaching.20[year].grp[number]'
version '1.0-SNAPSHOT'
```

settings.gradle (Optional: sonst wird Verzeichnisname verwendet)

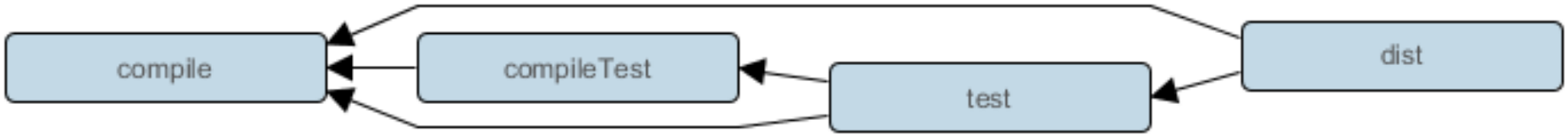
```
rootProject.name = 'projectName'
```

Tasks

Tasks `(gradle -q tasks)`

- Aufgaben die Gradle erledigen kann
- Verschiedene Typen von Tasks
- Koennen Abhängigkeiten haben
- Gradle bringt grundlegende tasks (**Sprachenunabhängig**) mit
- Weitere Tasks
  - Ueber Plugins (zB. java, maven)
  - Selbst definieren

Beispiel: `gradle test`



Java Plugin – Java spezifische tasks

Was muss ich tun?

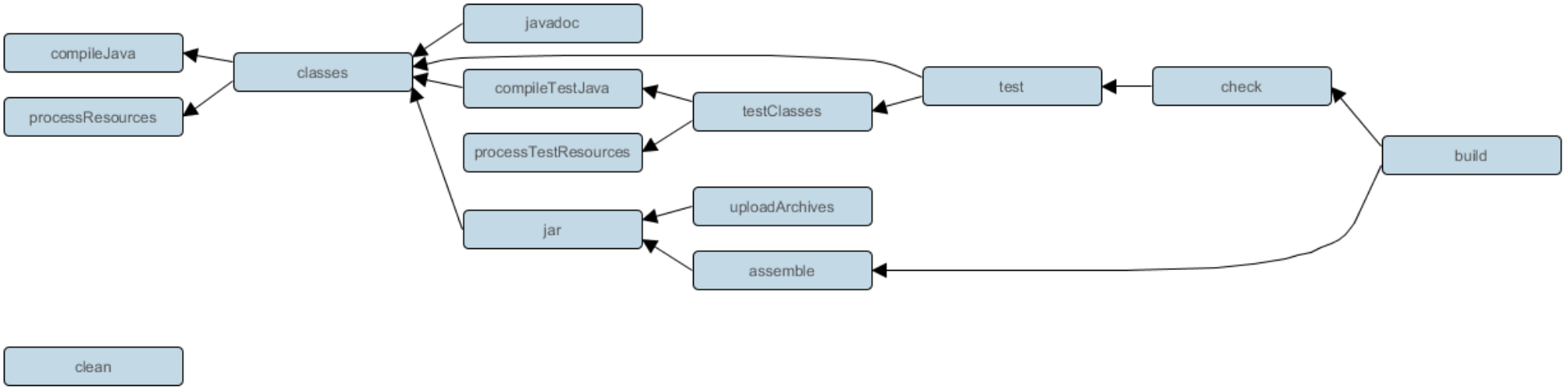
build.gradle

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

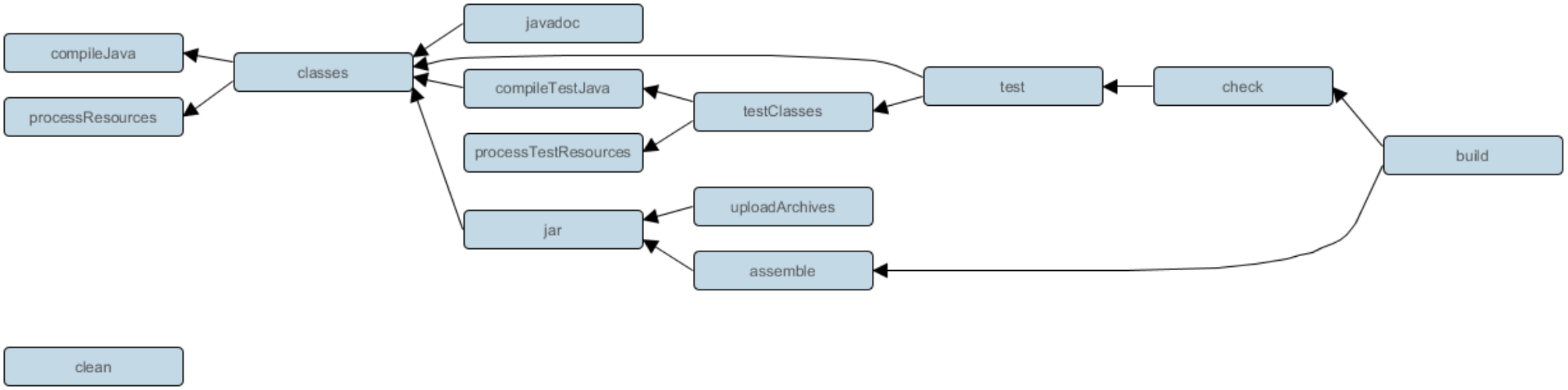
Konsole

```
gradle build
```

Was bekomme ich?



Java Plugin – Java spezifische tasks



- Das Projekt wird kompiliert
- Javadoc wird erzeugt
- Die Test werden ausgeführt
- Jar wird erzeugt und im Ordner moduleRoot/`build` gespeichert

Javadoc

Konsole

```
gradle javadoc
```

Clean

Konsole

```
gradle clean
```

Welche Taks gibt es?

Konsole

```
gradle tasks
```

Tasks Konfigurieren

```
dependencies {
    compile group: 'org.slf4j', name: 'slf4j-api', version:'1.21'
    testCompile group: 'junit', name: 'junit', version: '4.12'
}
jar {
    baseName = 'fancy-App'
    manifest {
        attributes 'Main-Class': 'package.path.to.your.MainClass'
    }
}
```

- Konfiguriere dependencies
- erzeugt das Jar mit dem Namen **fancy-App-0.2.1.jar**
- erzeugt ausführbares Jar

Repositories

build.gradle

```
repositories{
   mavenCentral()
   mavenLocal()
   maven {
      url “http://repo.xyz.de/maven2“
   }
}
```

Eine Anwendung starten

build.gradle

```
apply plugin: 'application'
mainClassName = „de.fsu.example.Test“
applicationDefaultJvmArgs = [„-Xmx4G“]
```

/src/main/java/de/fsu/Example.java

```
package de.fsu.example;
public class Test {
    public static void main(String[] args){
      System.out.println("Programm gestartet");}
}
```

Konsole

```
> gradle run
:compileJava UP-TO-DATE
:processResources UP-TO-DATE
:classes UP-TO-DATE
:run
Programm gestartet

BUILD SUCCESSFUL

Total time: 4.55 secs
```

Projekt ins lokale Repository packen

build.gradle

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

Konsole

```
> gradle  install
```

Installationen

Was brauchen wir:

- Java Installation (JDK)
  - Test: java -version
- Groovy installation
  - Test: groovy -v
- Gradle installation
  - Test: gradle -v

Installation von Groovy

- Siehe auch http://groovy-lang.org/install.html (5.Install binary)
- Groovy 2.4.X herunterladen und in geeignetes Verzeichnis entpacken
- Anpassungen in `~/.bashrc oder ~/.profile:`

```
…
export GROOVY_HOME="/path/to/groovy-2.4.X/"
…
export PATH="${PATH}:${GROOVY_HOME}/bin"
…
```

Installation von Gradle

- Siehe auch http://www.gradle.org/installation (Install manually)
- Gradle 3.x.x herunterladen und in geeignetes Verzeichnis entpacken
- Anpassungen in `~/.bashrc oder ~/.profile:`

```
…
export GRADLE_HOME="/path/to/gradle-3.4/"
…
export PATH="${PATH}:${GRADLE_HOME}/bin"
…
```