



Danke für eure Aufmerksamkeit

Build-Management-Tools



Prezi



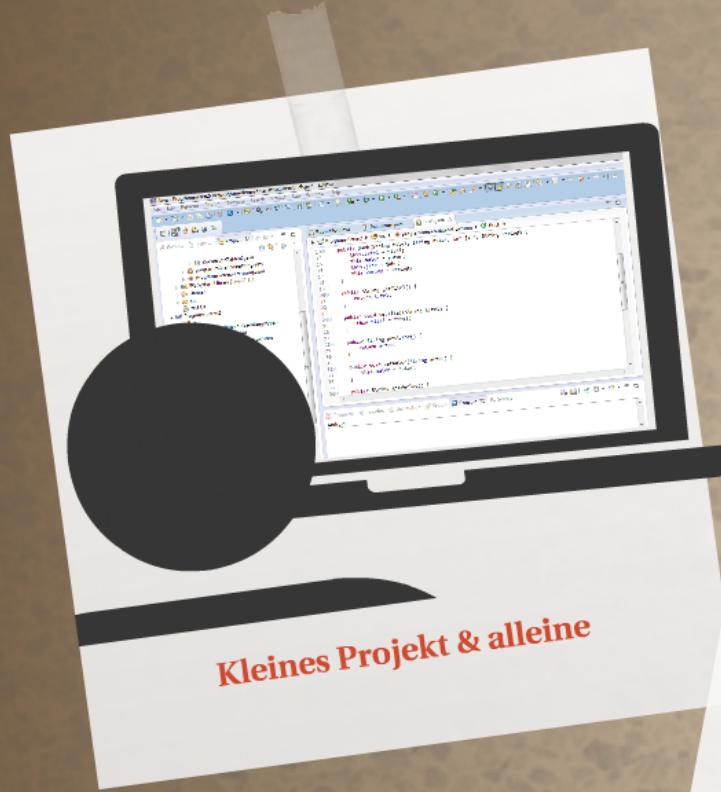
Danke für eure Aufmerksamkeit

Build-Management-Tools



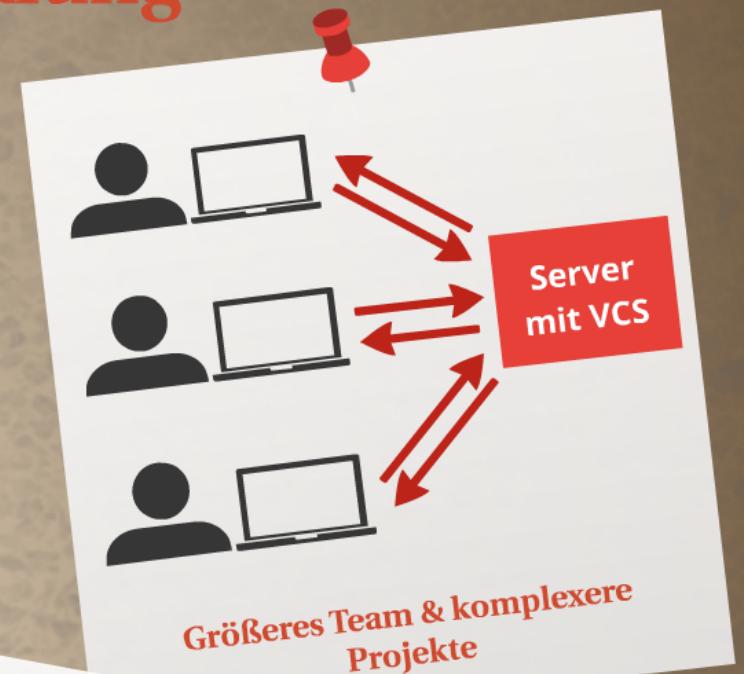
Prezi

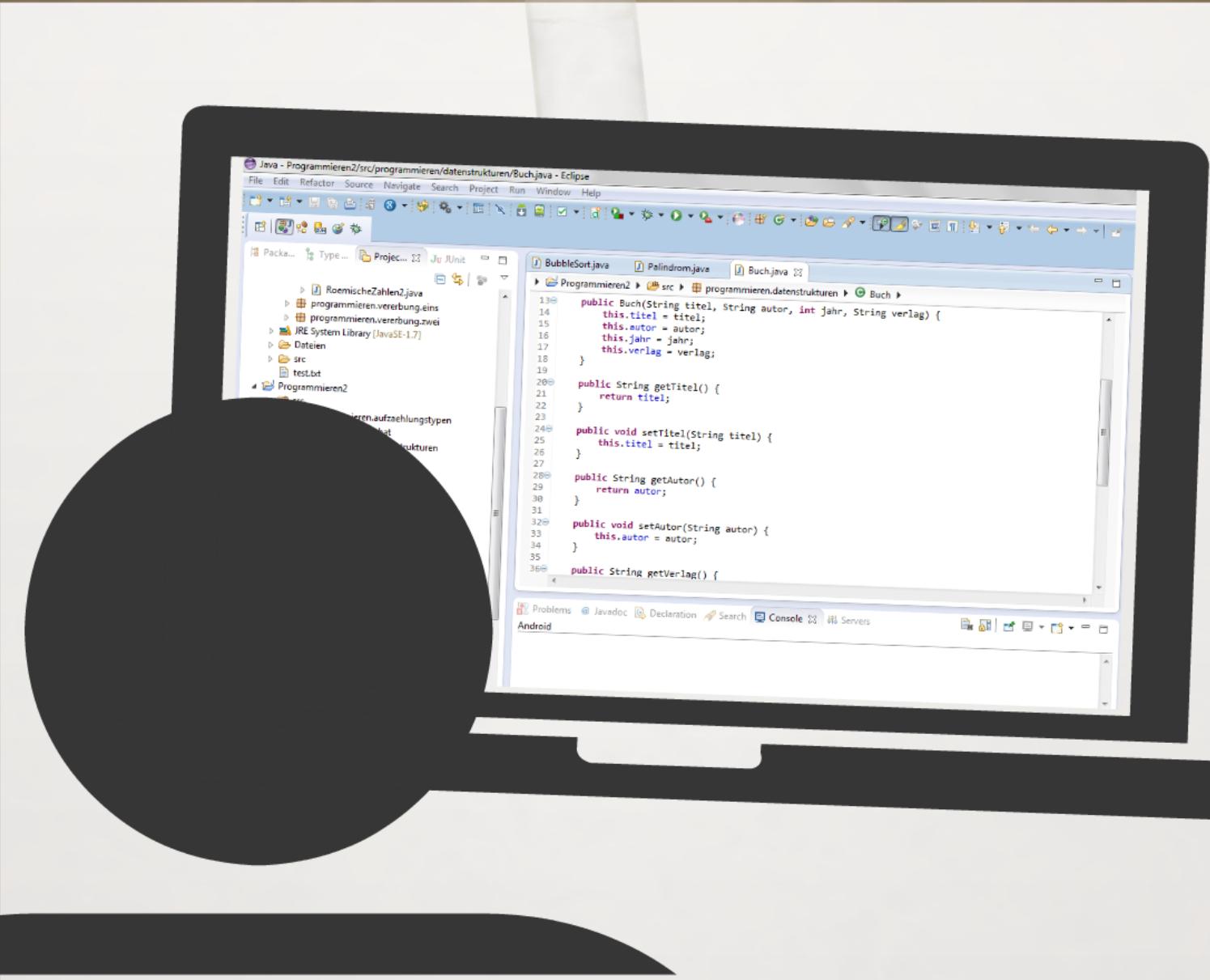
Softwareentwicklung



Kompilieren
Release erstellen
Dependencies auflösen
Dateien verschieben
fehleranfällig & zeitaufwendig
Manueller Build mit IDE

JAR-Datei erzeugen
Testen

A white sticky note with a red pushpin. It lists several steps of a manual build process. A red arrow points from the bottom-left towards the word "fehleranfällig".



Kleines Projekt & alleine

Kom
Rele

Kompilieren

JAR-Datei
erzeugen

Release erstellen

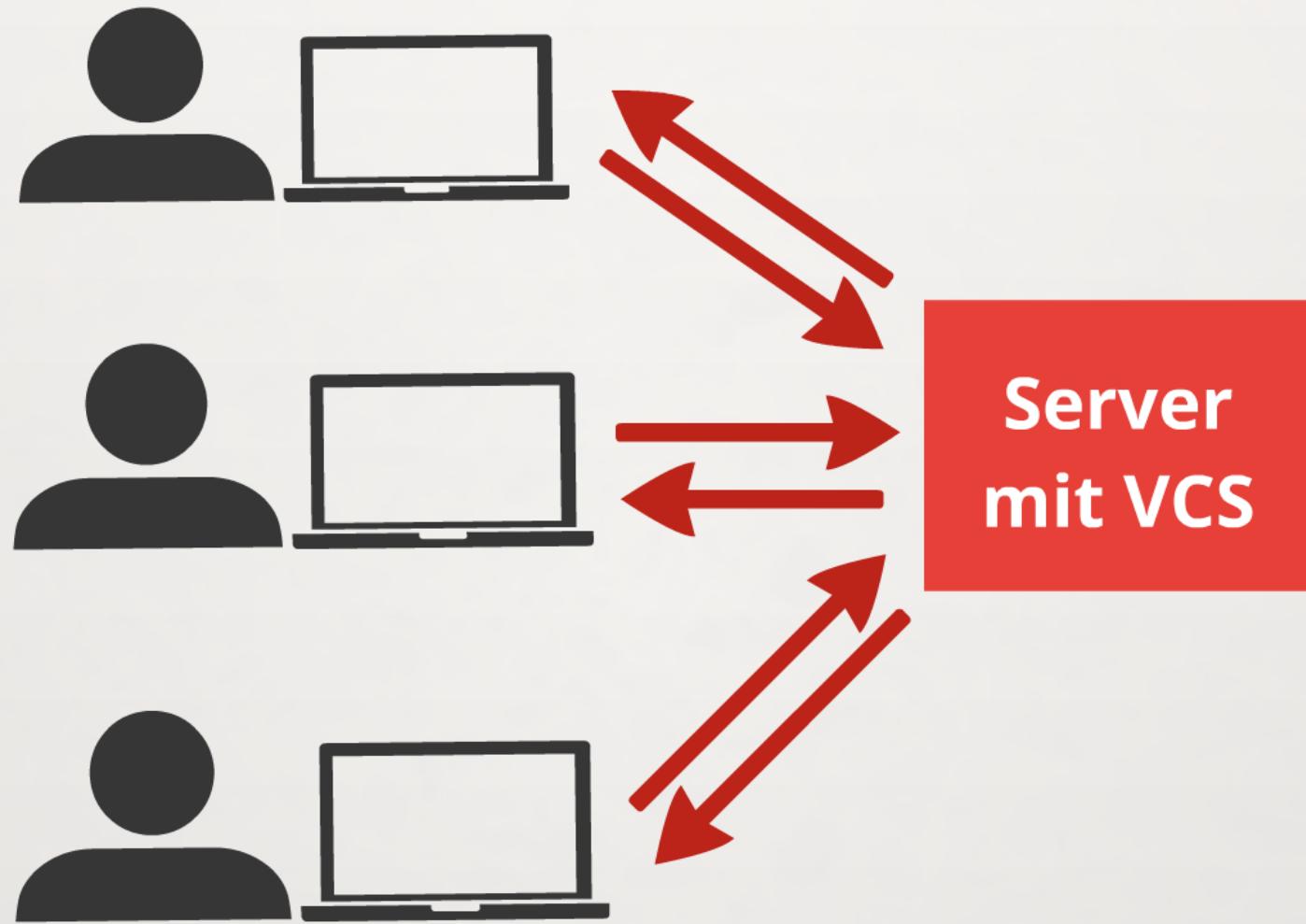
Testen

Dependencies auflösen

Dateien verschieben

→ fehleranfällig & zeitaufwendig

Manueller Build mit IDE



**Größeres Team & komplexere
Projekte**

Lösung: Projektautomatisierung

- immer dieselben Schritte beim Build
- vordefinierter "Bauplan" für Projekt
- Automatisierung

Welche Typen gibt es?

- On-demand Build: auf "Knopfdruck"
- Triggered Build: von bestimmtem Ereignis ausgelöst
- Scheduled Build: in regelmäßigen Zeitabständen oder zu bestimmten Zeitpunkten

Umsetzung: Build-Management-Tool

Build File (auch: Build Script): "Bauplan" für Projekt

Build Engine: Build File in Modell übersetzen

Dependency Manager (optional): Dependencies auflösen



maven

 gradle



Lösung: Projektautomatisierung

- **immer dieselben Schritte beim Build**
- **vordefinierter "Bauplan" für Projekt**
- **Automatisierung**

Welche Typen gibt es?

- **On-demand Build:** auf "Knopfdruck"
- **Triggered Build:** von bestimmtem Ereignis ausgelöst
- **Scheduled Build:** in regelmäßigen Zeitabständen oder zu bestimmten Zeitpunkten



Umsetzung: Build-Management-Tool

Build File (auch: Build Script): "Bauplan"
für Projekt

Build Engine: Build File in Modell
übersetzen

Dependency Manager (optional):
Dependencies auflösen



maven



t

?
uck"
em Ereignis
gen
mten

Maven

- deklarativ
- Konvention über Konfiguration
- einheitliche  Verzeichnisstruktur
- Abhängigkeiten
- Reports
- Artefakte

maven •

Konfigurationsmanagement

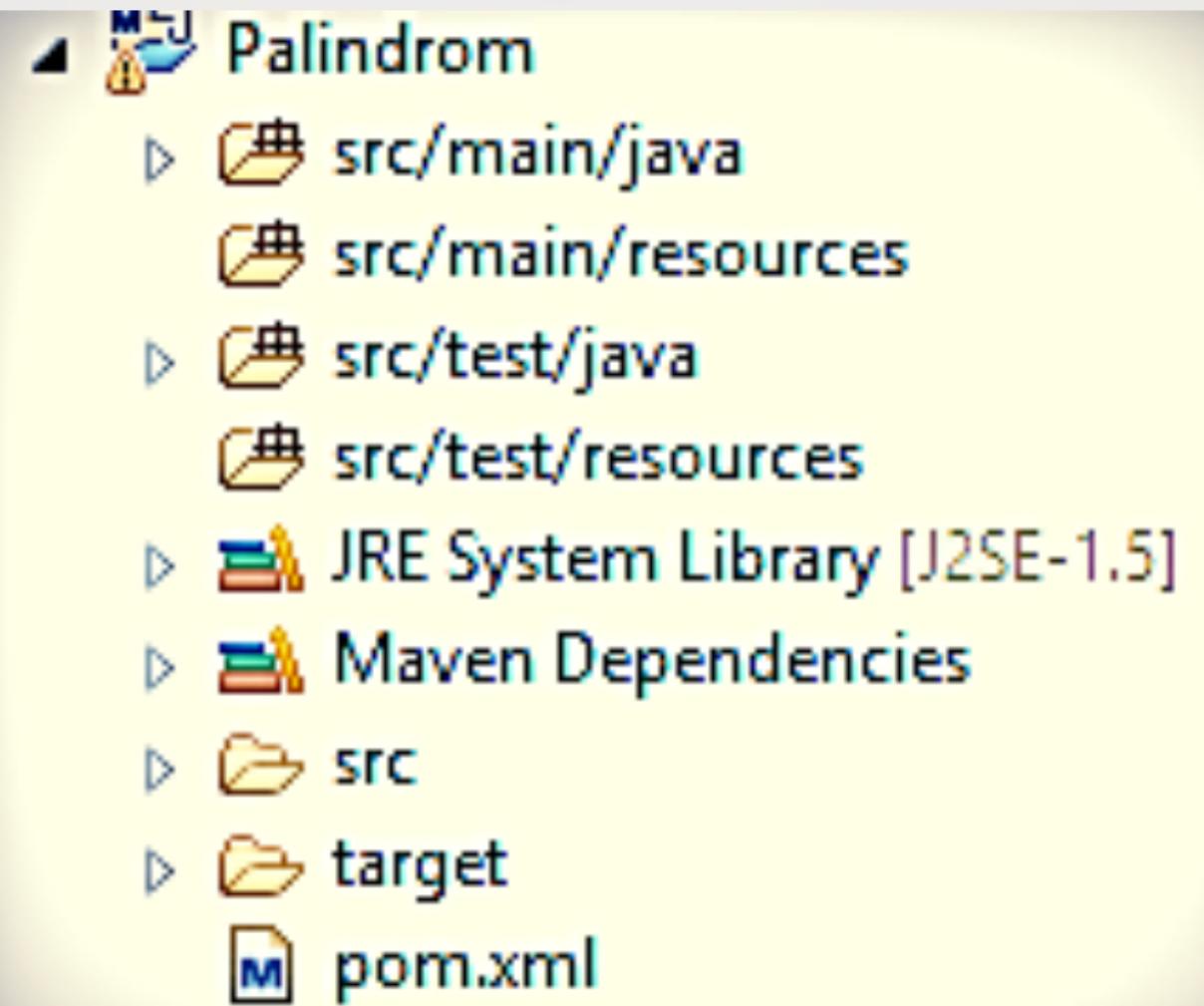
- Build-Management
- Versionsmanagement
- Release-Management
- Qualitätsmanagement
- Deployment-Management
- Reporting

→ Plugins

Maven

- deklarativ
- Konvention über Konfiguration
- einheitliche Verzeichnisstruktur
- Abhängigkeiten
- Reports
- Artefakte





Maven

- deklarativ
- Konvention über Konfiguration
- einheitliche Verzeichnisstruktur
- Abhängigkeiten
- Reports
- Artefakte





Konfigurationsmanagement

- Build-Management
- Versionsmanagement
- Release-Management
- Qualitätsmanagement
- Deployment-Management
- Reporting

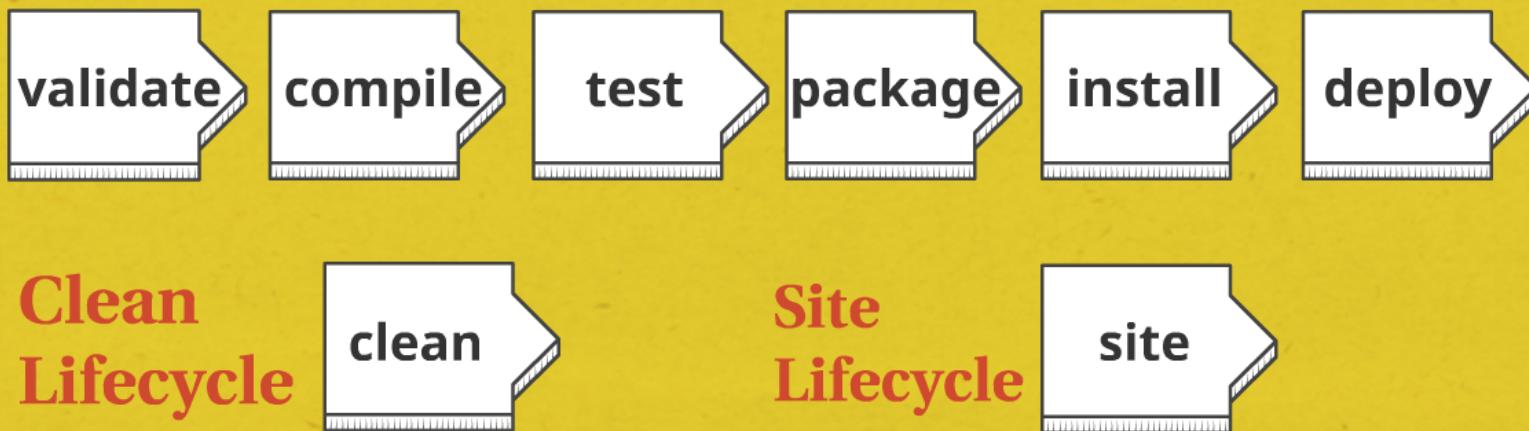


Plugins

Project Object Model (POM)

- xml-Darstellung
- Projektumgebung,
Projektbeziehungen, Abhängigkeiten
(Dependencies)

Build-Lifecycle



Project Object Model (POM)

- **xml-Darstellung**
- **Projektumgebung,**
- Projektbeziehungen, Abhängigkeiten
(Dependencies)**



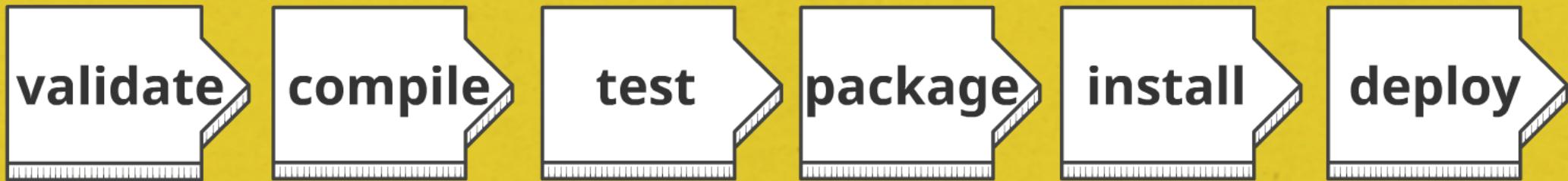
```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>de.mavenbuch.beispiele</groupId>
  <artifactId>beispiel-projekt</artifactId>
  <version>1.0-SNAPSHOT</version>
</project>
```

Project Object Model (POM)

- **xml-Darstellung**
- **Projektumgebung,**
Projektbeziehungen, Abhängigkeiten
(Dependencies)

- xml-Darstellung
- Projektumgebung,
Projektbeziehungen, Abhängigkeiten
(Dependencies)

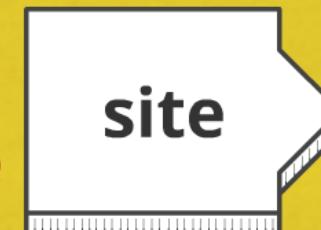
Build-Lifecycle

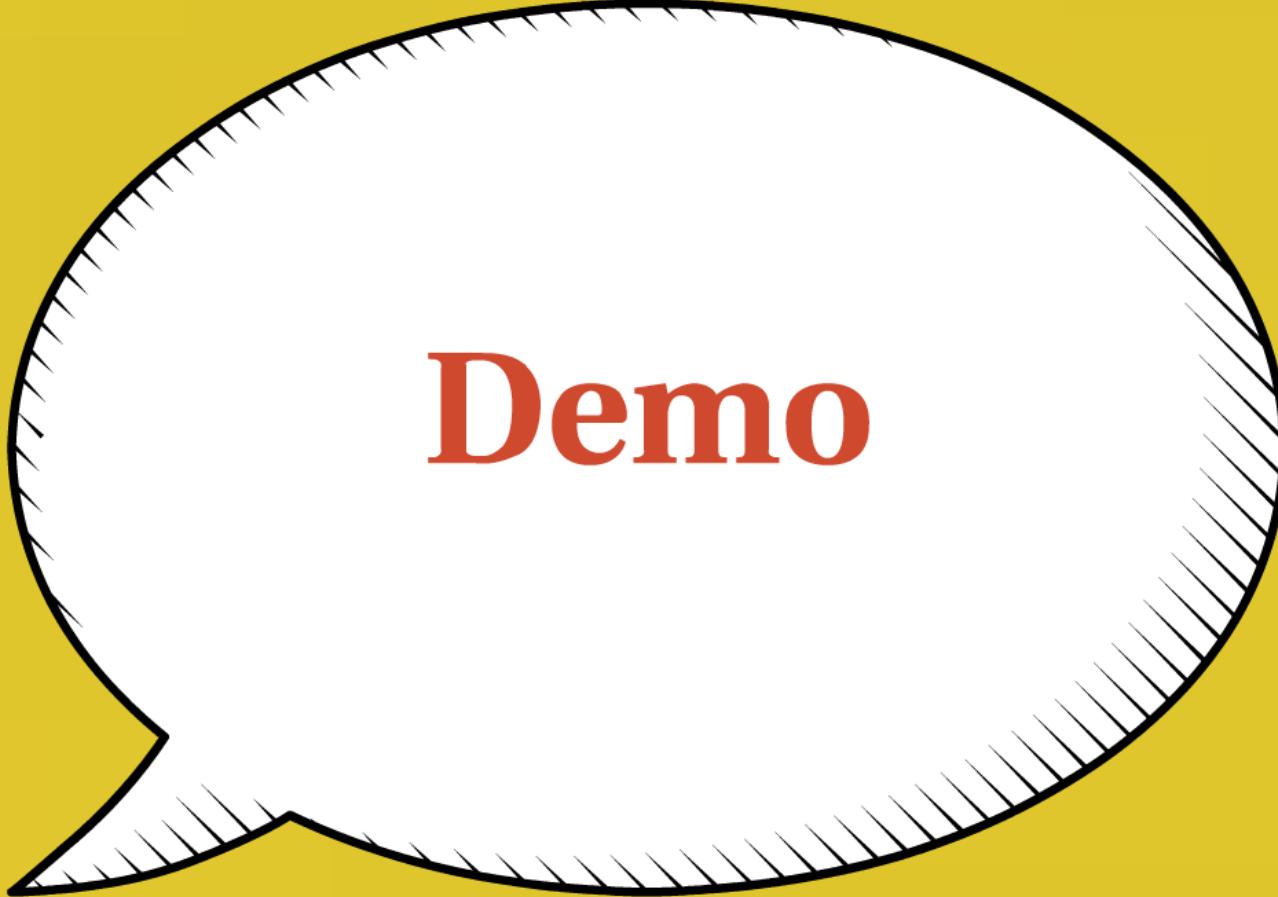


Clean Lifecycle



Site Lifecycle



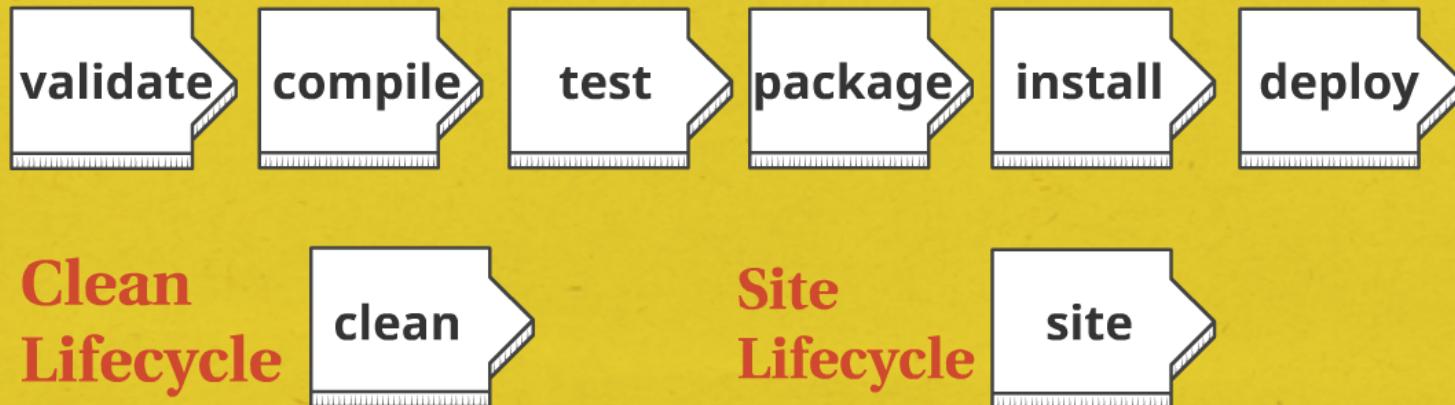


Demo

Project Object Model (POM)

- xml-Darstellung
- Projektumgebung, Projektbeziehungen, Abhängigkeiten (Dependencies)

Build-Lifecycle



Probleme bei etablierten Build-Management-Tools

Ant

- XML
- keinerlei Vorgaben
- sehr lange Build Scripts
- unübersichtlich
- Jedes Build Script eigene Struktur
- immer neu einarbeiten
- selbst Standard festlegen
- kein Dependency Management (erst mit Ivy)

Maven

- XML
- Default-Werte nur in begrenztem Maß selbst zu konfigurieren
- Standards in Beispielprojekten sinnvoll, aber seltener in realen Projekten
- Projekt muss häufig an Build Tool angepasst werden
- Erweiterungen durch eigene Plugins mühsam & aufwendig
- starr & unflexibel



Ant

- XML
 - keinerlei Vorgaben
- sehr lange Build Scripts
- unübersichtlich
- Jedes Build Script eigene Struktur
- immer neu einarbeiten
- selbst Standard festlegen
- kein Dependency Management (erst mit Ivy)



Maven

- XML
 - Default-Werte nur in begrenztem Maß selbst zu konfigurieren
- Standards in Beispielprojekten sinnvoll, aber seltener in realen Projekten
- Projekt muss häufig an Build Tool angepasst werden
- Erweiterungen durch eigene Plugins mühsam & aufwendig
- starr & unflexibel



Gradle

Vorteile aus Ant & Maven

Convention over Configuration

Dependency Management

Groovy

Vorgegebene Struktur, aber beliebig
anpassbar

Don't repeat yourself

Demo



- Wichtige Befehle für Konsole
- Kleines Gradle-Projekt
- Struktur von Projekten
- Eigene Tasks definieren
- Dependencies definieren
- Dependencies auflösen
- Flexibilität von Gradle



Gradle

Der

- Vorteile aus Ant & Maven

- Convention over Configuration

- Dependency Management

- Groovy

- Vorgegebene Struktur, aber beliebig anpassbar

- Don't repeat yourself

auf Java basierend



einfacher

Statt `System.out.println("Hello world");`
nur noch `println 'Hello world'`



Gradle

Der

Vorteile aus Ant & Maven

• V

• K

Convention over Configuration

• C

Dependency Management

• S

Groovy 

• S

Vorgegebene Struktur, aber beliebig
anpassbar

• S

Don't repeat yourself

• S

Demo



- Wichtige Befehle für Konsole
- Kleines Gradle-Projekt
 - Struktur von Projekten
 - Eigene Tasks definieren
 - Dependencies definieren
 - Dependencies auflösen
 - Flexibilität von Gradle

n
ction
t
beliebig

Vergleich

Maven vs. Gradle

Convention over Configuration: beide

Don't repeat yourself (Flexibilität & Erweiterbarkeit):
Maven: sehr umständlich, Gradle: einfach

Aus Ant- oder Mavenprojekten Gradleprojekte machen?

- () Ant Scripts einfach importierbar, normale Weiterverwendung mit Vorteilen von Gradle
- Dennoch Kosten- und Zeitaufwand abzuschätzen!
- () Integration von Maven Scripts noch nicht so einfach

Toolempfehlung

Bei Verwendung von Maven:

- Wenn keine Probleme → Maven weiterbenutzen
- Wenn Probleme erträglich sind → Maven weiterbenutzen
- Bei neuem Projekt → Gradle ausprobieren



Maven vs. Gradle

Convention over Configuration: beide

Dont't repeat yourself (Flexibilität & Erweiterbarkeit):
Maven: sehr umständlich, Gradle: einfach

Aus Ant- oder Mavenprojekten Gradleprojekte machen?



Ant Scripts einfach importierbar, normale
Weiterverwendung mit Vorteilen von Gradle



Dennoch Kosten- und Zeitaufwand abzuschätzen!



Integration von Maven Scripts noch nicht so einfach

Weiterverwendung mit Vorteilen von Gradle

→ Dennoch Kosten- und Zeitaufwand abzuschätzen!

(:() Integration von Maven Scripts noch nicht so einfach

Toolempfehlung

Bei Verwendung von Maven:

Wenn keine Probleme



Maven weiterbenutzen

Wenn Probleme erträglich sind



Bei neuem Projekt



Gradle ausprobieren



Build-Management-Tools

Wenn Probleme erträglich sind

Maven weiterbenutzen

Bei neuem Projekt

Gradle ausprobieren

Danke für eure Aufmerksamkeit