Pengenalan Build Automation

Pengenalan Build Automation

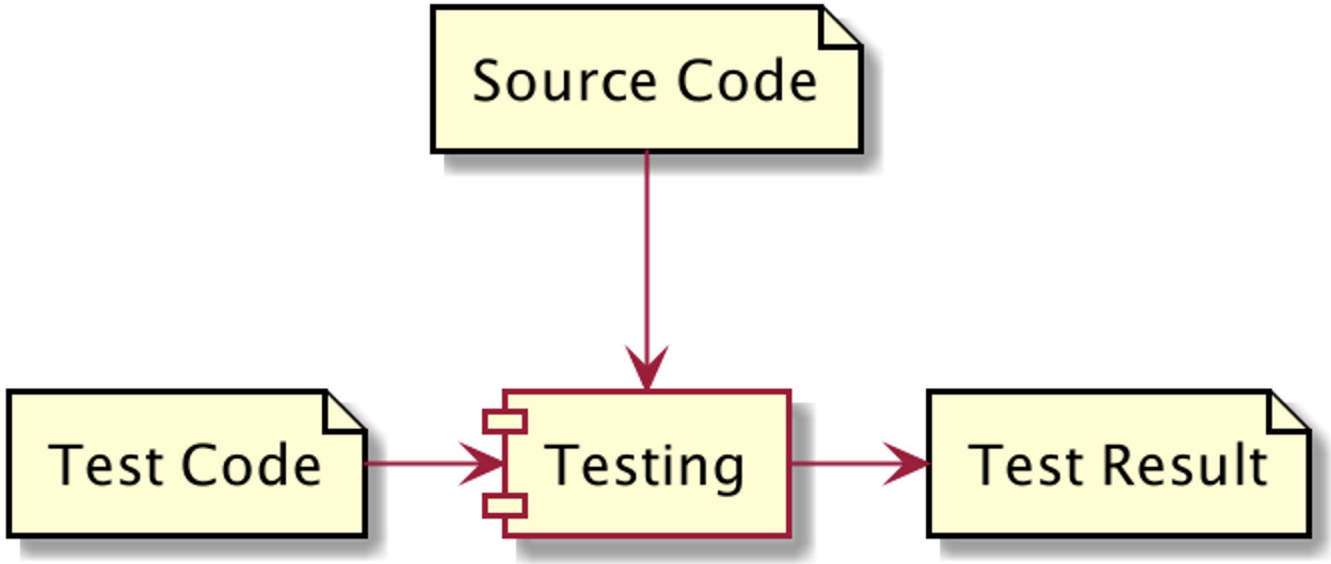
- Build Automation adalah proses meng-otomatisasi tahapan pembuatan software dan hal-hal yang berhubungan dengannya, seperti: kompilasi source code menjadi binary code, mem-package binary code menjadi distribution file, membuat dokumentasi, menjalankan automated test sampai manajemen dependency.

Kompilasi Source Code

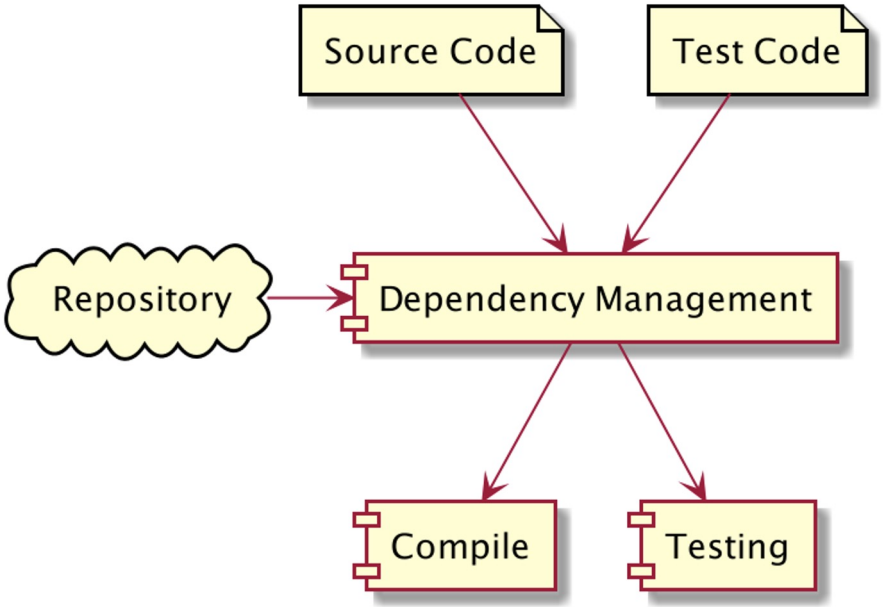
Mem-Package Binary File

Membuat Dokumentasi

Menjalankan Automated Test



Management Dependency



Contoh Build Automation Tool

- Apache Maven
- Apache Ivy
- Gradle

Pengenalan Apache Maven

- Apache Maven adalah salah satu build automation yang free dan open source
- Apache menggunakan XML untuk mendefinisikan build script nya
- Apache Maven saat ini sangat populer di kalangan Developer Java
- Apache Maven menggunakan JVM sebagai fondasi dasar
- https://maven.apache.org/

Teknologi yang Didukung

Apache Maven mendukung build automation untuk banyak teknologi, seperti :

- Java
- Kotlin
- Groovy
- Scala
- dan lain-lain

Menginstall Apache Maven

Download Maven

- https://maven.apache.org/download.cgi

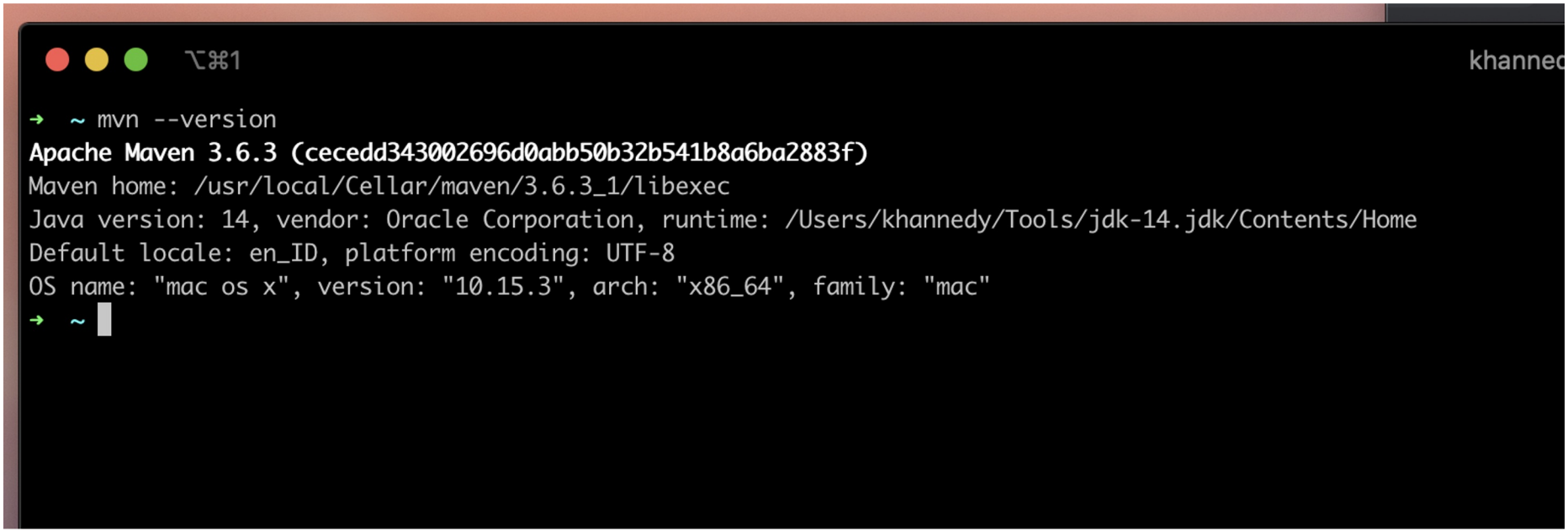
Setting Path

- Windows : https://medium.com/programmer-zaman-now/setting-java-path-di-windows-4da2c65d8298
- Mac & Linux

```
# Add to .bashrc or .zshrc

export MAVEN_HOME="/usr/local/Cellar/maven/3.6.3_1"
export PATH="$MAVEN_HOME/bin:$PATH"
```

Mengecek Apache Maven



Membuat Project

Archetype

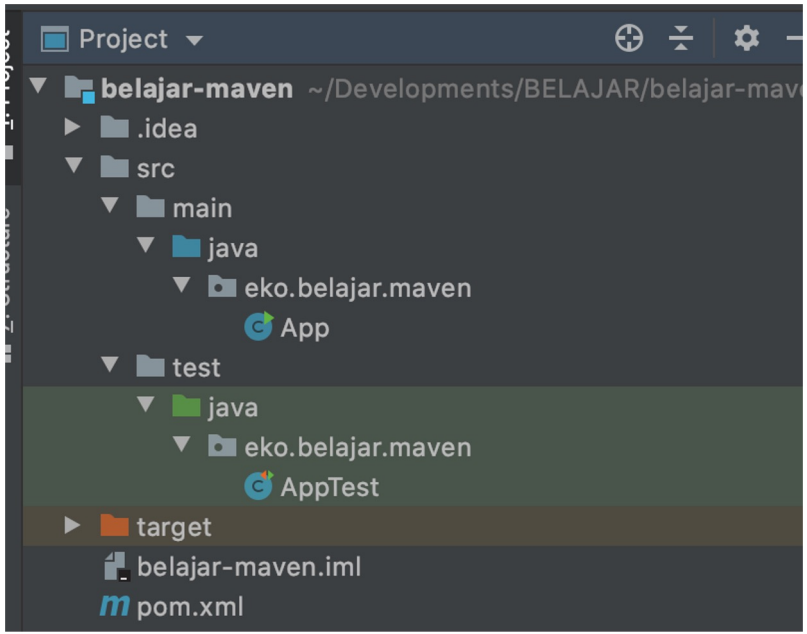
- Maven mendukung pembuatan berbagai macam project dengan mudah
- Pembuatan project di maven menggunakan archetype, archetype adalah template project
- Kita bisa menggunakan yang sudah disediakan oleh maven, atau bahkan bisa membuat template archetype sendiri
- https://maven.apache.org/guides/introduction/introduction-to-archetypes.html

Membuat Java Project

- mvn archetype:generate
- maven-archetype-quickstart

Struktur Project

Struktur Project Apache Maven



Maven Lifecycle

Lifecycle

- Maven bekerja dalam konsep lifecycle
- Untuk menjalankan lifecycle, kita bisa menggunakan perintah : mvn namalifecycle
- Lifecycle akan menjalankan banyak plugin, entah bawaan maven, atau bisa kita tambahkan plugin lain jika mau
- https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

Contoh Lifecycle

- clean, menghapus folder target (tempat menyimpan hasil kompilasi)
- compile, untuk melakukan kompilasi source code project
- test-compile, untuk melakukan kompilasi source code project
- test, untuk menjalankan unit test
- package, untuk membuat distribution file aplikasi
- install, untuk menginstalll project ke local repository, sehigga bisa digunakan di project lain di local
- deploy, deploy project ke remote repository di server

Build Project

Build Project

- Saat kita membuat project biasanya akan ada 2 jenis kode yang kita buat, kode program nya, dan kode testing nya
- Maven mendukung hal tersebut

Menjalankan Kompilasi Program

mvn compile

Menjalankan Unit Test

mvn test

Mem-package Project

mvn package

Dependency

Dependency

- Proyek aplikasi jarang sekali berdiri sendiri, biasanya membutuhkan dukungan dari pihak lain, seperti tool atau library
- Tanpa build tool seperti Apache Maven, untuk menambahkan library dari luar, kita harus melakukannya secara manual
- Apache Maven mendukung dependency management, dimana kita tidak perlu me-manage secara manual proses penambahkan dependency (tool atau library) ke dalam proyek aplikasi kita

Dependency Scope

Saat kita menambahkan dependency ke project Maven, kita harus menentukan scope dependency tersebut, ada banyak scope yang ada di Maven, namun sebenarnya hanya beberapa saja yang sering kita gunakan, seperti

- compile, ini adalah scope default. Compile artinya dependency tersebut akan digunakan untuk build project, test project dan menjalankan project.
- test, ini adalah scope untuk test project, hanya akan di include di bagian test project

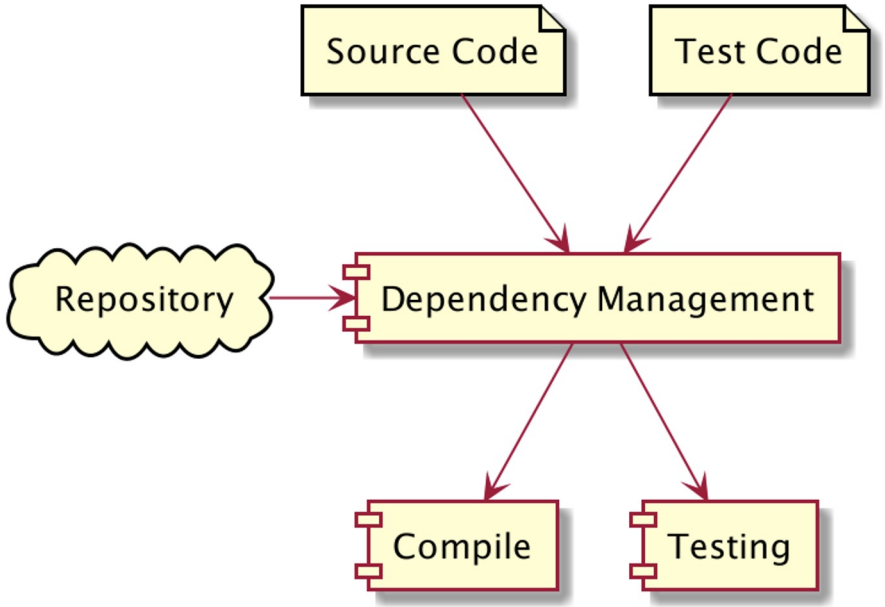
Kode : Menambah Dependency

```
20
21    <dependencies>
22
23        <dependency>
24            <groupId>junit</groupId>
25            <artifactId>junit</artifactId>
26            <version>4.11</version>
27            <scope>test</scope>
28        </dependency>
29
30    </dependencies>
31
```

Mencari Dependency

- https://search.maven.org/
- https://mvnrepository.com/

Repository


Source Code
Test Code
Repository
Dependency Management
Compile
Testing

Kode : Menambah Repository

```
77
78    <repositories>
79      <repository>
80        <id>bintray-bliblidotcom-maven</id>
81        <name>bintray</name>
82        <url>https://dl.bintray.com/bliblidotcom/maven</url>
83      </repository>
84    </repositories>
85
86  </project>
87
```

Maven Properties

Maven Properties

- Maven mendukung properties untuk menyimpan data konfigurasi
- Fitur ini akan sangat memudahkan kita kedepannya, dibandingkan melakukan hardcode di konfigurasi maven

Kode : Properties

```
13      <url>http://www.example.com</url>
14
15      <properties>
16          <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
17          <maven.compiler.source>14</maven.compiler.source>
18          <maven.compiler.target>14</maven.compiler.target>
19          <junit.version>4.6.2</junit.version>
20      </properties>
21
22      <dependencies>
23
24          <dependency>
```

Kode : Menggunakan Properties

```
    <dependencies>

        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>${junit.version}</version>
            <scope>test</scope>
        </dependency>

    </dependencies>

    <build>
```

Membuat Distribution File

Membuat Distribution File

- Secara default, maven mendukung pembuatan distribution file menggunakan lifecycle package
- Hanya saja, hasil distribution file nya berupa file jar yang berisikan binary code dari project kita
- Dependency lainnya tidak dimasukkan, sehingga tidak bisa langsung dijalankan

Menggunakan Assembly Plugin

- Salah satu plugin yang bisa kita gunakan untuk membuat distribution file beserta dependency yang kita butuhkan adalah Assembly Plugin
- https://maven.apache.org/plugins/maven-assembly-plugin/usage.html
- Tidak hanya Assembly Plugin, sebenarnya masih banyak plugin lain yang bisa kita gunakan untuk membuat distribution file di Maven
- Untuk membuat distribution file, kita bisa menggunakan perintah mvn package assembly:single

Multi Module Project

Multi Module Project

- Saat aplikasi kita sudah sangat besar, kadang ada baiknya kita buat aplikasi dalam bentuk modular
- Misal kita pisahkan module model, controller, view, service, repository, dan lain-lain
- Untungnya, Maven mendukung pembuatan project multi module

Membuat Module Baru

- Untuk membuat module baru, di dalam project yang sudah ada, kita hanya tinggal membuat folder baru, lalu menambahkan setting pom.xml di folder tersebut
- Module harus memiliki parent, dimana parent nya adalah project diatas folder tersebut
- Selanjutnya, di parent nya pun, module harus di include

Konfigurasi Module

```
    <parent>
        <artifactId>belajar-maven</artifactId>
        <groupId>programmer-zaman-now</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>belajar-maven-data</artifactId>


</project>
```

Konfigurasi Parent

```
    <modelVersion>4.0.0</modelVersion>

    <groupId>programmer-zaman-now</groupId>
    <artifactId>belajar-maven</artifactId>
    <packaging>pom</packaging>
    <version>1.0-SNAPSHOT</version>
    <modules>
        <module>belajar-maven-data</module>
    </modules>

    <name>belajar-maven</name>
    <!-- FIXME change it to the project's website -->
```

Include Antar Module

```
    <dependencies>
        <dependency>
            <groupId>programmer-zaman-now</groupId>
            <artifactId>belajar-maven-data</artifactId>
            <version>${project.version}</version>
        </dependency>
    </dependencies>

</project>
```

Dependency Management

Dependency Management

- Saat project kita sudah besar, kadang kita sering menggunakan banyak dependency
- Masalah dengan banyaknya dependency adalah, jika kita salah menggunakan dependency yang sama namun versinya berbeda-beda
- Maven mendukung fitur dependency management, dimana kita bisa memasukkan daftar dependency di parent module beserta versinya, lalu menambahkan dependency tersebut di module tanpa harus menggunakan versinya
- Secara otomatis versi dependency akan sama dengan yang ada di dependency management di parent module

Dependency Management di Parent

```
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.junit.jupiter</groupId>
                <artifactId>junit-jupiter</artifactId>
                <version>${junit.version}</version>
            </dependency>
            <dependency>
                <groupId>programmer-zaman-now</groupId>
                <artifactId>belajar-maven-data</artifactId>
                <version>${project.version}</version>
            </dependency>
```

Dependency di Module

```
<dependencies>
    <dependency>
        <groupId>programmer-zaman-now</groupId>
        <artifactId>belajar-maven-data</artifactId>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter</artifactId>
    </dependency>
</dependencies>
```