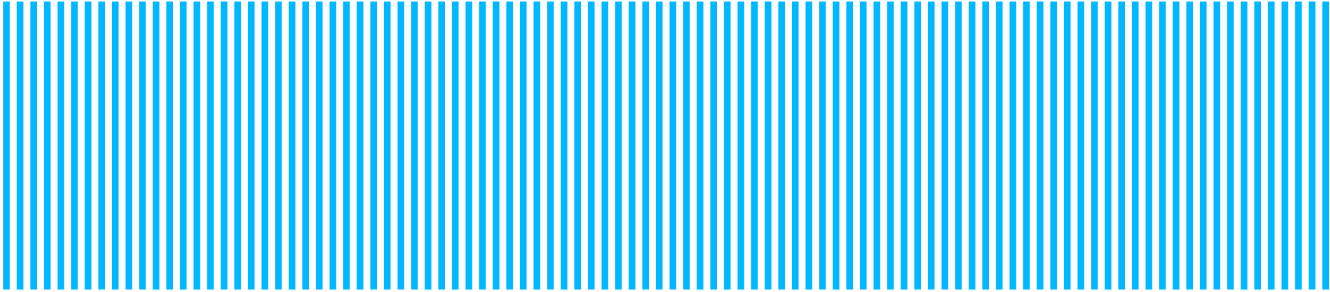




SinauDW “Hey, belajar sendiri itu mudah!”

Sinau Maven

Tutorial ini bebas untuk dicopy / disebarluaskan secara gratis.
Dilarang menggunakan sebagian / keseluruhan isi untuk tujuan komersial tanpa seijin penulis!
Non Commercial use only. Copyright 2012.



Why Maven?

Suatu ketika kita ingin membuat sebuah project java, apa yang kita lakukan pertama kali? Biasanya kita akan langsung membuka IDE lalu mulai membuat project dari sana. Dari sini sekilas nampak tidak ada masalah, namun yang terjadi adalah project yang kita buat dengan IDE tertentu belum tentu bisa dibuka dari IDE yang lain (bisa tapi dengan cara yang cukup melelahkan), karena masing-masing IDE memiliki standarnya sendiri.

Sekarang kita akan coba menambahkan library ke dalam project tadi. Dengan cara manual kita bisa mendownload semua file yang kita butuhkan dan menaruhnya didalam classpath project. Tetapi kadang tidak semudah itu, bisa jadi library itu membutuhkan dependency library yang lain dan dependency library itu membutuhkan dependency library yg lain lagi dan seterusnya. Bayangkan kalo harus mencari satu persatu, tentu sangat melelahkan. Inilah yang dinamakan dengan jar hell.

Masalah belum berhenti sampai disini, misalnya kita ingin berbagi project kita dengan orang lain. Biasanya kebanyakan orang malas untuk mengikuti library dari project yang akan dibagikan tersebut, karena library yang banyak mengakibatkan file membesar dan dengan bandwidth yang pas-pasan akan memakan waktu lama pada saat upload. Akibatnya adalah orang lain yang ingin memanfaatkan project ini harus mencari library yang dibutuhkan satu persatu, sangat tidak praktis.

Untuk mengatasi permasalahan diatas kita bisa menggunakan tools dari apache yang bernama maven. Maven memiliki struktur project standar yang membuat maven ramah terhadap IDE apapun. Maven juga menyediakan online repository yang memudahkan kita untuk download secara otomatis library yang kita butuhkan + dependency-nya.

Sebenarnya ada alternatif lain selain maven, yaitu **ant+ivy** dan **gradle**. Namun karena beberapa alasan saya lebih memilih menggunakan maven. B'coz I love maven's way ^ ^.

Install Maven

Install maven sama seperti instalasi jdk. Baca tutorial instalasi maven pada link berikut ini <http://www.mkyong.com/maven/how-to-install-maven-in-windows/>.

HelloWorld

Setelah maven berhasil diinstall sekarang kita akan mencoba membuat project pertama menggunakan maven. Buka console & ketikkan perintah ini:

```
$ mvn archetype:generate -DgroupId=test.maven
-DartifactId=testproject -DarchetypeArtifactId=maven-archetype-
quickstart -DinteractiveMode=false
```

arti dari perintah diatas adalah:

- archetype:generate → generate project
- DgroupId=test.maven → buat package didalam project dengan nama test.maven
- DartifactId=testproject → berikan nama project testproject
- DarchetypeArtifactId=maven-archetype-quickstart → artifact project yang digunakan (sampai saat ini ada sekitar 538 jenis artifact maven yang bisa kita gunakan sesuai kebutuhan, untuk melihatnya ketikkan di console mvn archetype:generate)

Bila tidak ada masalah maka kita akan memiliki sebuah project dengan nama testproject. Apabila kita buka didalamnya akan menghasilkan struktur seperti dibawah ini:

```
- testproject
+----- pom.xml
+----- src
    +----- main
    +         |----- java
    +         |----- test
    +         |----- maven
    +         |----- App.java
    +----- test
    +         |----- java
    +         |----- test
    +         |----- maven
    +         |----- AppTest.java
```

isi dari pom.xml adalah sebagai berikut:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>test.maven</groupId>
  <artifactId>testproject</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>testproject</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
```

```
<artifactId>junit</artifactId>
<version>3.8.1</version>
<scope>test</scope>
</dependency>
</dependencies>
</project>
```

Masuk ke dalam direktori testproject dari console & ketikkan perintah:

```
testproject $ mvn compile
```

Perintah ini digunakan untuk meng-compile class-class java didalam project. Untuk menjalankan programnya gunakan perintah:

```
testproject $ mvn exec:java -Dexec.mainClass="test.maven.App"
```

Bila tidak ada masalah maka akan keluar output berupa tulisan "Hello World!". Untuk yang baru pertama kali menggunakan maven proses ini mungkin akan sedikit lama karena maven harus melakukan proses download semua yang dibutuhkan.

Cukup panjang yah perintahnya, sekarang kita akan coba membuat perintah running nya supaya lebih simple. Ubah file pom.xml diatas menjadi seperti dibawah ini:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>test.maven</groupId>
  <artifactId>testproject</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>testproject</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <finalName>charactercount</finalName>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
```

```
<version>2.3.2</version>
<configuration>
  <source>1.5</source>
  <target>1.5</target>
</configuration>
</plugin>

<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>exec-maven-plugin</artifactId>
  <version>1.1.1</version>
  <executions>
    <execution>
      <phase>test</phase>
      <goals>
        <goal>java</goal>
      </goals>
      <configuration>
        <mainClass>test.maven.App</mainClass>
      </configuration>
    </execution>
  </executions>
</plugin>

</plugins>
</build>
</project>
```

Sekarang perintah running nya akan menjadi:

```
testproject $ mvn test
```

Dengan perintah ini kita dapat memastikan bahwa sebelum program dijalankan akan melewati testing terlebih dahulu. (Apa itu testing? Kita akan membahasnya lain waktu). Lebih lanjut mengenai running main class menggunakan maven bisa dibaca dilink berikut

<http://www.vineetmanohar.com/2009/11/3-ways-to-run-java-main-from-maven/>.

Beberapa perintah maven yang sering digunakan antara lain:

- mvn clean : untuk membersihkan compile sebelumnya.
- mvn install : untuk memasang hasil build project ke dalam local repository.
- mvn package : untuk build project.

Semua library yang telah didownload oleh maven akan ditempatkan didalam folder khusus dengan nama .m2. Di linux folder ini ada didalam home & dalam keadaan hidden, di windows cari sendiri ada dimana :D.

Maven Project & IDE

Maven sangat ramah terhadap kebanyakan IDE, sehingga project yang dibuat menggunakan maven mudah untuk dibuka & dimodifikasi dari IDE manapun. Gunakan perintah `mvn eclipse:eclipse` supaya project yang dibuat bisa diimport ke dalam eclipse. Untuk IntelliJ IDEA gunakan perintah `mvn idea:idea`. Untuk netbeans bisa langsung open project tanpa konfigurasi tambahan. Bagaimana mudah bukan ^^.