# Maven

・http://www.techscore.com/tech/Java/ApacheJakarta/Maven/index/

# 基本

## 環境構築

### ■ Download

・http://maven.apache.org/download.cgi

### ■ Install mac

```
$ brew search maven
$ brew install maven@3.5
```

### ■環境変数

・M2_HOME= インストールフォルダ
・PATH=%PATH%;%M2_HOME%;bin

### ■確認

```
C:\>mvn --version
Apache Maven 3.0.4 (r1232337; 2012-01-17 17:44:56+0900)
Maven home: C:\springsource\apache-maven-3.0.4
Java version: 1.5.0_16, vendor: Sun Microsystems Inc.
Java home: C:\Program Files\Java\jdk1.5.0_16\jre
Default locale: ja_JP, platform encoding: MS932
OS name: "windows vista", version: "6.1", arch: "x86", family: "windows"
```

### ■プロキシ設定

・/conf/settings.xml

```
<proxies>
  <!-- proxy
   | Specification for one proxy, to be used in connecting to the network.
   |-->
  <proxy>
    <id>optional</id>
    <active>true</active>
    <protocol>http</protocol>
    <!--username>proxyuser</username -->
    <!--password>proxypass</password -->
    <host>172.16.xx.xx</host>
    <port>8080</port>
    <!-- nonProxyHosts>127.0.0.1|localhost</nonProxyHosts> -->
  </proxy>
  <!-- -->
</proxies>
```

### ■プロジェクトを作成

・JAR ファイルを作成するプロジェクトを作成

```
mvn archetype:create -DgroupId=info.typea.sample -DartifactId=sample
```

| 引数 | 意味 |
|---|---|
| archetype:create | プロジェクトのスケルトンを作成 |
| groupId | プロジェクトのルートパッケージ名 |
| artifactId | プロジェクト名 |
| -Dkey=value | システムプロパティを設定する |

## ■標準のディレクトリ構成

・Maven では標準のディレクトリ構成が決められている
・構成を理解すれば、 Maven を利用した他のプロジェクトのディレクトリ構成も理解したことになる
・標準のディレクトリ構成に従うことが推奨

| src ディレクトリ以下 | 内容 |
|---|---|
| src/main/java | Java ソースコード |
| src/test/java | テスト用の Java ソースコード |

### pom.xml

・プロジェクトに関する情報を持つ重要なファイル

```
   <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/xsd/maven-4.0.0
.xsd">
   <modelVersion>4.0.0</modelVersion>

   <groupId>info.typea.sample</groupId>
   <artifactId>sample</artifactId>
   <version>1.0-SNAPSHOT</version>
   <packaging>jar</packaging>

   <name>sample</name>
   <url>http://maven.apache.org</url>

   <properties>
     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
   </properties>

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

| 要素 | 内容 |
|---|---|
| modelVersion | POM のバージョン。 特に変更する必要はありません。 |
| groupId | プロジェクトを一意に識別する名前。 プロジェクトのルートパッケージ名を指定するのが一般的です。 |

| | |
|---|---|
| artifactId | プロジェクトの成果物の名前。 作成する JAR や WAR、EAR ファイルなどの名前に使用されます。 |
| packaging | 作成する成果物のパッケージング・タイプ。 jar ( デフォルト )、war、 ear などがあります。 |
| version | プロジェクトのバージョン。 |
| name | プロジェクトの表示名。 ドキュメントを作成するときなどに使用されます。 |
| url | プロジェクトのサイトの URL。 ドキュメントを作成するときなどに使用されます。 |
| dependencies | プロジェクトが依存するライブラリの情報。 |

## 機能

### ■コンパイル

・pom.xml のあるディレクトリで実行する

```
mvn compile
```

### ■ユニットテスト

```
mvn test
```

### ■ドキュメンテーション

```
mvn javadoc:javadoc
```

### ■サイトの作成

・プロジェクトのサイトを簡単に作成

```
mvn site
```

### ■ JAR ファイルの作成

```
mvn package
```

### ■ローカルリポジトリへのインストール

・ローカルリポジトリにインストールすることで、ローカルにある他のプロジェクトから参照可能になる
・デフォルトでは、ユーザーディレクトリの、.m2 フォルダ以下

```
mvn install
```

### ■リモートリポジトリへの配備

・事前にリモートリポジトリの情報を pom.xml に記述
・project/distributionManagement 要素の入れ子として repository 要素を追加

```
mvn deploy
<project ...>
  ...
  <distributionManagement>
    <repository>
      <id>deploy-repository</id>
      <name>deployRepository</name>
      <url>file://${env.M2_HOME}/deployRepository</url>
    </repository>
  </distributionManagement>
  ...
</project>
```

### ■プロジェクトのクリーン

・生成したファイルを削除
・target ディレクトリが削除される

```
mvn clean
```

# 依存性

・プロジェクトの依存するライブラリを自動的にダウンロードし、必要なときにクラスパスの設定を行う
・Maven を用いた開発では、プロジェクトを作成するたびに、ライブラリを一つひとつ手作業でダウンロードする必要がない。
・ライブラリのダウンロードはリモートリポジトリ ( デフォルトでは Maven の セントラルリポジトリ ) から行われる
・必要とするライブラリがリモートリポジトリに存在しない場合や、存在しても必要とするバージョンのものが無い場合は、ライブラリを手作業でダウンロードし、ローカルリポジトリにインストールしなければいけない
・ローカルリポジトリのライブラリはローカルにあるプロジェクトで共有されるので、何度も行う必要はない

## 依存性の指定とスコープ

・依存するライブラリを指定するには、project/dependencies の入れ子として dependency 要素を追加

| 要素 | 内容 |
|---|---|
| compile | scope の指定を省略した場合のデフォルト値です。全ての状況でクラスパスに追加されます。 |
| provided | ライブラリが JDK やコンテナによって提供される場合に指定します。コンパイル時のみクラスパスに追加されます。 |
| runtime | 実行時のみに必要な場合に指定します。テストの実行および通常の実行のときにクラスパスに追加されます。 |

| | |
|---|---|
| test | テストのときのみ必要な場合に指定します。テストのコンパイルと実行のときにクラスパスに追加されます。 |
| system | 明示的にクラスパスに追加する場合に指定します。このスコープのライブラリは常に有効であるとみなされ、リポジトリの検索は行われません。 |

groupId, artifactId, version に指定する値の調べ方

- Maven ではデフォルトのリモートリポジトリとして http://repo1.maven.org/maven2 が指定されている
- groupId や artifactId に指定する値は、通常ここから探す
- 検索用サイト
  - http://mvnrepository.com/
  - http://search.maven.org/#browse

## 依存性の指定を行う

### ■ pom.xml で依存性の指定

### Archetype Catalog

- http://maven.apache.org/archetype/maven-archetype-plugin/specification/archetype-catalog.html
- http://maven.apache.org/archetype/archetype-models/archetype-catalog/archetype-catalog.html

## 依存ライブラリの取得

- http://typea.info/blg/glob/2014/04/java-ee-7-maven-jar.html

```
$ mvn dependency:copy-dependencies
```

### ■出力先のディレクトリを指定

```
$ mvn dependency:copy-dependencies -DoutputDirectory=lib
```

# Tips

### Spring Tool Suite

- Spring Tool Suite Maven プロジェクトの作成
- Spring Tool Suite Maven でネットワークエラー

## トラブル

### ■ Missing Artifact 問題

- Eclipse で Missing Artifact Problem が発生したら、プロジェクト - Maven - Update Project を選択

・Force Update of Snapshot/Release にチェックを入れて実行