maven プロジェクトの作成

非公式ドキュメント by 山崎 昂輔 (TA, 当時M2)

2024 年 5 月 31 日

1　はじめに

この資料では，Maven プロジェクトの作成方法について説明します．Maven は，Java プロジェクトのビルド，依存関係管理，プロジェクト管理を行うためのツールです．

2　Maven と javac の違い

Maven を使用して Java プロジェクトをビルドする際には，通常の javac コマンドを使用したコンパイルといくつかの重要な違いがあります．

2.1　依存関係の管理

Maven は依存関係の管理を自動化します．プロジェクトで使用するライブラリやフレームワークを `pom.xml` ファイルに記述することで，Maven がそれらを自動的にダウンロードし，プロジェクトに追加します．

これは Python での依存関係管理に似ています．例えば，Python では `requirements.txt` ファイルに必要なライブラリを記述し，`pip install -r requirements.txt` コマンドを実行することで，必要なライブラリを自動的にインストールできます．

一方，javac を使用する場合，必要なライブラリを手動でダウンロードし，クラスパスに追加する必要があります．

2.2　プロジェクトの構成

Maven は標準的なプロジェクト構成を提供します．例えば，ソースコードは `src/main/java` に，テストコードは `src/test/java` に配置されます．これにより，プロジェクトの構成が一貫性を持ち，他の開発者がプロジェクトを理解しやすくなります．javac を使用する場合，プロジェクトの構成は開発者に委ねられます．

2.3 ビルドライフサイクル

Maven はビルドライフサイクルを提供し，コードのコンパイル，テストの実行，パッケージング，デプロイなどのタスクを自動化します．これにより，ビルドプロセスが一貫して実行され，手動でのミスが減少します．javac を使用する場合，これらのタスクは手動で実行する必要があります．

2.4 プラグインの利用

Maven は多くのプラグインを提供しており，これらを使用してビルドプロセスを拡張することができます．例えば，コードの静的解析，ドキュメントの生成，デプロイメントの自動化などが可能です．javac は単に Java コードをコンパイルするツールであり，これらの機能は提供されていません．

2.5 まとめ

Maven はプロジェクトのビルドと管理を効率化するための強力なツールです．依存関係の管理，標準的なプロジェクト構成，ビルドライフサイクルの自動化，プラグインの利用など，Maven を使用することで開発プロセスが大幅に改善されます．一方，javac は単純なコンパイルツールであり，これらの機能は提供されていません．

3 Maven のインストール

3.1 Mac の場合

Homebrew を使用して Maven をインストールすることができます．以下のコマンドを実行してください．

```
brew install maven
```

インストールされているかの確認は以下のコマンドで行います．

```
mvn -v
```

3.2 Windows の場合

ブラウザからダウンロードしてインストールすることができます．以下の URL からインストーラをダウンロードしてください．

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

適当な場所で展開し，環境変数を設定してください．（筆者が Windows を使用していないので詳細は検索してください．）

4　プロジェクトの作成

次に，Maven プロジェクトを作成します．コマンドラインで以下のコマンドを実行します．

```
mvn archetype:generate -DgroupId=fuga -DartifactId=hoge -
DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

上記のコマンドにより， fuga というグループ ID と hoge というアーティファクト ID を持つプロジェクトが作成されます．この時ディレクトリ構成は以下のようになります．

```
hoge
├── pom.xml
└── src
    ├── main
    │   └── java
    │       └── fuga
    │           └── App.java
    └── test
        └── java
            └── fuga
                └── AppTest.java
```

これはコマンド以外に，使用するエディタの機能として提供されている場合もあります．

5　依存関係の追加

プロジェクトの依存関係は pom.xml ファイルに記述します．例えば，JUnit を追加するには，以下のように記述します．

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.11</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

- groupId (必須), プロジェクトのグループ ID
- artifactId (必須), プロジェクトのアーティファクト ID

- `version` (必須), 依存関係のバージョン
- `scope` (任意), 依存関係のスコープ

依存関係を追加する際には，`https://mvnrepository.com/` から必要なライブラリを検索し，そのままコピーして `pom.xml` ファイルに貼り付けることができます．

例えば，Apache Flink を追加する場合は，以下の手順で行います．

1. `https://mvnrepository.com/` にアクセスします．
2. 検索バーに「Apache Flink」と入力し，検索します．
3. 必要な Flink のモジュールを選択します．例えば，「flink-java」モジュールを選択します．
4. バージョンを選択し，表示される依存関係のコードをコピーします．
5. `pom.xml` ファイルの `<dependencies>` セクションに貼り付けます．

このようにして，必要なライブラリを簡単に追加することができます．

依存関係の追加もエディタによっては別の方法がサポートされている場合があります．

6 ビルドと実行

プロジェクトをビルドし，実行する手順は以下の通りです．

1. コマンドラインでプロジェクトのルートディレクトリに移動します．
2. 以下のコマンドを実行してプロジェクトをビルドします．

```
mvn package
```

3. ビルドが成功すると， `target` ディレクトリに `demo-1.0-SNAPSHOT.jar` というファイルが生成されます．
4. 以下のコマンドを実行してアプリケーションを実行します．

```
java -cp target/demo-1.0-SNAPSHOT.jar com.example.App
```

Maven にはいくつかの頻出コマンドがあります．以下にそれぞれのコマンドとその説明を示します．

- `mvn compile` - プロジェクトのソースコードをコンパイルします．コンパイルされたクラスファイルは `target/classes` ディレクトリに配置されます．コードのコンパイルが成功するか確認したいときに使用します．
- `mvn package` - プロジェクトをビルドし，パッケージングします．JAR ファイルや WAR ファイルなどの配布可能な形式が `target` ディレクトリに生成されます．配布可能な形式のアーティファクトを作成したいときに使用します．
- `mvn install` - プロジェクトをビルドし，ローカルリポジトリにインストールします．JAR ファイルや WAR ファイルなどの配布可能な形式が `target` ディレクトリに生成され，さら

にローカルリポジトリにインストールされます．他のプロジェクトからこのプロジェクトの成果物を依存関係として使用したいときに使用します．

- `mvn test` - プロジェクトのテストを実行します．これは，プロジェクトの単体テストを実行し，コードが期待通りに動作することを確認するために使用されます．
- `mvn exec:java` - プロジェクトのメインクラスを実行します．`-Dexec.mainClass` オプションでメインクラスを指定します．これは，ビルドされたプロジェクトを実際に実行するために使用されます．
- `mvn assembly:single` - プロジェクトをビルドし，すべての依存関係を含む単一の JAR ファイルを生成します．これは，プロジェクトとそのすべての依存関係を一つの JAR ファイルにまとめるために使用され，配布やデプロイが容易になります．

例えば，`mvn exec:java` コマンドを使用してアプリケーションを実行する場合，以下のようにします．

```
mvn exec:java -Dexec.mainClass="com.example.App"
```

これらのコマンドを使用することで，Maven プロジェクトのビルドや実行が簡単に行えます．