◆概説 pom.xml／Posted on 2018-12-13 by k.hasunuma
https://www.coppermine.jp/note/2018/12/overviewing-maven-pom/

Apache Maven は Java のプロジェクト構成管理ツールとして最もメジャーなものです。現在では Java 向けに数多くのプロジェクト構成管理ツールが提供されていますが、それらの多くは何らかの形で Maven と関わりを持っています。

Maven は Java を用いたアプリケーション開発サイクルを管理するため、以下のような特徴を持っています。

プロジェクトを groupId / artifactId / version という識別子（“GAV coodinates” と呼ばれることもある）で管理する。
他のプロジェクト（アプリケーション、ライブラリ等）との依存関係を定義することができ、必要に応じて外部（Maven リポジトリ）から自動的にダウンロードすることも可能である。
プラガブル・アーキテクチャとなっており、プラグインによって用意に機能を拡できる。
プラグインも必要に応じて外部（Maven リポジトリ）から自動的にダウンロード可能となっている。
そして、Maven で管理するプロジェクトの定義情報が pom.xml と呼ばれる XML ファイルです。pom.xml は Java 開発者にとって非常に馴染み深い XML ファイルの 1 つですが、「おまじない」「決まりこど」程度の認識で使用している Java 開発者も多いことでしょう。この記事では、私自身の復習の意味も含めて pom.xml ファイルの構造について解説します。

pom.xml の基本構造
まずは Maven のドキュメントに含まれる「POM Reference」を見てみましょう。必要なことは概ねこのドキュメントに記載されています。

pom.xml は <project> … </project> をルート要素とする XML ファイルで、厳密な定義（XML スキーマ）は http://maven.apache.org/xsd/maven-4.0.0.xsd となります。ルート要素を具体的に記述すると以下のようになります。

```
<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">
<!-- Contents of pom.xml -->
```

</project>
<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"> <!-- Contents of pom.xml --> </project>
<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">

  <!-- Contents of pom.xml -->

</project>
pom.xml の内容については「POM Reference」の「Quick Overview」セクションが分かりやすいでしょう。このセクションでは、pom.xml の内容を大きく 4 つのパートに分けています。

The Basics
Build Settings
More Project Information
Environment Settings
普段 Maven を利用しているのであれば、 “The Basics” および “Build Settings” についてはそれなりに記述しているはずですが、 “More Project Information” および “Environment Settings” の各要素について記述することは少ないかと思われます。ただし、Maven リポジトリの総本山である “Maven Central” リポジトリにプロジェクトをアップロードする際には、 “More Project Information” および “Environment Settings” についても必要事項を記述する必要があります。

実は pom.xml の記述内容は、ビルドに必要なものだけでも相当量になります。しかし、その多くにはデフォルト値が存在しており、必ず記述しなければならない要素は決して多くありません。pom.xml のデフォルト値は「The Super POM」に示されている通りですが、要約すると以下のデフォルト値が定められています。

デフォルトで使用する Maven リポジトリ: プロジェクト、プラグインともに “Maven Central” リポジトリ (http://repo.maven.apache.org/maven2) が指定されており、必要に応じて “Maven Central” リポジトリからダウンロードするように設定されていることが分

かります。
プロジェクトのディレクトリ構成: ソース・ファイルのディレクトリ /src/main/java、出力ファイルのディレクトリ /target などが定義されています。
標準でロードされるプラグインとそのバージョンに関する情報: 将来的には削除される予定です。
条件ビルドで使用する「performRelease」プロファイル: 将来的には削除される予定です。
なお、「The Super POM」に示されている値はあくまでデフォルトであり、これらは各プロジェクトで上書きすることができます。

Maven に関する記事で「ソース・ファイルは必ず src/main/java ディレクトリ以下に配置しなければならない」と述べているものもありますが、src/main/java ディレクトリ以下に配置すればソース・ファイルとして認識されるようデフォルト値が設定されているだけであり、必ずしも src/main/java ディレクトリである必要はありません（もっとも、pom.xml の記述量が増えるため、必要がなければデフォルト値に従う方が良いでしょう)。

“Maven Central” リポジトリにはいくつかのミラーサイトがあり、また Sonatype Nexus OSS などをプロキシとして用いることができます。リポジトリをミラーサイトやプロキシに置き換えることで、”Maven Central” マスターサイトの混雑を回避することも可能になります。

The Basics
“The Basics” はプロジェクトの基本情報を設定するパートです。このうち、groupId、artifactId および version は必ず指定する必要があります（ビルド後の生成物が .jar ファイル以外の場合には、packaging の指定も必要です)。

他の要素についてはオプションです。

dependencies: 他のプロジェクトに依存するケースが多いため、依存関係を定義する dependencies 要素と、その子要素 dependency は非常に良く使用されます。
properties: Maven およびプラグインが参照するプロパティ値で、子要素名がプロパティ名、その値がプロパティ値となります。様々な用途に用いられる、使用頻度の高い要素です。
parent, modules, dependencyManagement: Maven のプロジェクトは入れ子にすることが可能であり、そのための親子関係の定義に用いられます。parent 要素が子→親の参照、modules 要素が親→子の参照を定義します。
parent 要素を定義すると、親プロジェクトに定義されている情報を子プロジェクトが引き継ぎます。これを “Inheritance” といいます。dependencyManagement 要素を利用して、

子プロジェクトの依存関係を親プロジェクト側で管理することも可能です。ただし、parent 要素を使用すると groupId が親プロジェクトの groupId + artifactId に固定され、version を定義することができなくなります (親プロジェクトの version が適用されます)。
modules 要素 (と子要素 module) を定義すると、親プロジェクトのビルドと合わせて子プロジェクトのビルドも行われます。これを “Aggregation” といいます。なお modules 要素を定義する場合、そのプロジェクトは package 要素を “pom” に設定して親プロジェクトであることを宣言する必要があります。
プロジェクトの親子関係を表す方法として、Inheritance と Aggregation の双方を組み合わせることも、いずれか一方のみ使用することも可能です。
以上から、最小の pom.xml は以下のようになります。

```
<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>org.codehaus.mojo</groupId>
<artifactId>my-project</artifactId>
<version>1.0</version>
</project>
<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>org.codehaus.mojo</groupId> <artifactId>my-project</artifactId>
<version>1.0</version> </project>
<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>org.codehaus.mojo</groupId>
  <artifactId>my-project</artifactId>
  <version>1.0</version>
</project>
```

Build Settings

“Build Settings” はビルドに直接関係する情報で、いくつかの要素は必須ですが、Super POM でデフォルト値が設定されているため、事実上はすべてオプション要素となります。

build: ビルドに関する情報です。子要素 plugins (および孫要素 plugin) でプラグインを指定することで、より高度なビルドを行うことができます。いくつかのプラグインは標準でロードされますが、使用されるバージョンは Maven 本体のバージョンに依存するため、特に最新の JDK に合わせてビルドする場合などは新しいバージョンのプラグインを明示する必要があります。.jar ファイル以外をビルドする場合 (.war など) もプラグインを追加する必要があります。

profiles: 条件ビルドを行うためのプロファイルを定義します。ビルドにいくつかのパターンが存在する場合、プロファイルを定義して任意に切り替えることができるようになっています。Maven ではコンパイル、ビルド、インストール、デプロイなどプロジェクトのライフサイクルをサポートしますが、特にデプロイについてはデプロイ先の環境が複数存在することも珍しくないため、そのような場合にプロファイルの切換によって対応することができます。

More Project Information

“More Project Information” は、そのプロジェクトの概要、ライセンス、開発者情報などを明記するパートです。ビルドには一切関係ない情報であるためすべて省略可能ですが、プロジェクトを “Maven Central” にアップロードして公開する場合には、プロジェクトの素性をはっきりさせるために正しく記述することが求められます。

name, description: プロジェクトの名前と説明を記述します。GitHub で公開している場合、おそらく同様の記述を行っているかと思いますので、それに合わせる形で問題ないでしょう。”Maven Central” で公開するのであれば設定しておいた方が良い項目です。

url: プロジェクトの Web サイトの URL を記述します。GitHub で公開している場合はその URL でも良いでしょう。こちらも “Maven Central” で公開するのであれば設定しておいた方が良い項目です。

inceptionYear: プロジェクトの開始年を記述します。覚え書き程度の情報であるため、”Maven Central” での公開を想定する場合でも任意の項目です。

licenses: プロジェクトのライセンスについて記述します。複数のライセンスを設定できるよう、実際のライセンス情報は子要素 license に記述するようになっています。「POM Reference」に Apache License 2.0 の例が記載されていますので、Apache License 2.0 で公開するのであればそのままコピーして使用すれば問題ありません。”Maven Central” で公開する場合、最低限ライセンス (いずれかの OSS ライセンス) の名前と文書への URL が示されていれば問題ありません。

organization: プロジェクトの主体となる組織の情報を記述します。個人の場合は自身の名前を組織名にするか、あるいはこの要素を省略することで対応します。
developers: プロジェクトの開発者の情報を子要素 developer に記述します (開発者 1 名に対して developer 要素 1 つが対応します)。「POM Reference」に例が記載されていますが、”Maven Central” で公開するのであれば name + α (email, url, organization/organizationUrl のいずれか) を記載すれば問題ありません。
contributors: プロジェクトのコントリビューターの情報を子要素 contributor に記述します (コントリビューター1 名に対して contributor 要素 1 つが対応します)。developers 要素と似ていますが、developers がプロジェクトの開発主体の情報であるのに対して、contributors は開発主体以外の関係者となる点が異なります。これについては省略しても問題ありません。
Environment Settings
“Enviroinment Settings” は、プロジェクトの開発環境に関する情報を記述するパートです。このパートの情報は “More Project Information” のようなメタ情報としての側面を持つ一方で、ビルド自動化の一環として Maven または外部のツールから参照される可能性もあります。その際に必要な情報が欠けているとビルド・チェーンが失敗してしまうため、使用するツールや環境に合わせて必要な項目を正しく記述する必要があります。言うまでもなく、ローカル環境で単純にビルドするだけであれば一切不要な情報ではあります。

issueManagement: 使用している Issue トラッカーに関する情報を記載します。使用していないのであれば省略して問題ありません。
ciManagement: CI 環境がある場合にはその情報を記載します。基本的には CI ツールが使用するため、CI 環境を使用していないのであれば省略して問題ありません。
mailingLists: メーリングリストを使用している場合にはその情報を記載します。使用していなければ省略して問題ありません。
scm: 使用している SCM ツールの情報を記載します。使用していなければ正直に書かない (省略する) しかありませんが、”Maven Central” で公開するようなプロジェクトであれば何らかの SCM ツールは使用しているはずです。
prerequisites: ビルド時の前提条件を記載します。前提条件をチェックするプラグインを使用している場合には何らかの記述があるはずですが、そうしたものを使用していないのであれば省略可能です。
repositories, pluginRepositories: Super POM では双方のデフォルト値として “Maven Central” が設定されています。もし依存関係にあるモジュールや追加しているプラグインが “Maven Central” 以外で公開されている場合には、子要素 repository でアップロード先の Maven リポジトリに関する情報を追加しなければなりません。すべての依存関係が “Maven Central” で解決できるようであれば、これらは省略できます。

distributionManagement: プロジェクトを Maven リポジトリで公開するために必要な情報を記述します。プロジェクトを外部に公開しないのであれば、この要素は省略することができます。外部に公開する、または組織内のネットワークで共有するためには、そのための Maven リポジトリの情報を記載する必要があります。Sonatype Nexus OSS で Maven リポジトリを構築している場合には Nexus が示す URL を記述するだけで済みます。また、プラグインを使用すれば FTP や WebDAV などを使用して Maven リポジトリを作ることができます。GitHub のリポジトリを Maven リポジトリとして使用することも可能です。例えば Payara では GitHub 上に独自パッチのための Maven リポジトリを構築しています。

“Maven Central”に公開する場合は、“Maven Central”へのアップロードが可能な Maven リポジトリの情報を指定する必要があります。そのような Maven リポジトリの例としては、Sonatype 直営の Nexus Repository Manager が挙げられます。利用するには Sonatype の会員登録と公開領域作成の手続きが必要です（無料）。手続きが完了すると distributionManagement 要素を記述するのに必要な情報が送られてきますので、間違いの内容に記載しましょう。なお、”Maven Central” で公開するためにはプロジェクトの生成物（.jar ファイルなど）に PGP 署名を行わなければなりませんが、それについては別の機会にご説明しようと思います。

まとめに代えて

私が “Maven Central” にアップロードした jakartaee-quickstart-archetype というプロジェクトがあります。JAX-RS と JavaServer Faces のテンプレートを自動生成するためのプロジェクト (Maven では Archetype といいます) です。その pom.xml は一応 “Maven Central” アップロード時の審査(?)をパスしているものですので、とりあえず「このくらい書いておけば大丈夫」というレベルを判定する材料として使えると思います。ご参考まで。