**Введение в POM**

* [Что такое ПОМ](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html#What_is_a_POM) ?
* [Супер ПОМ](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html#Super_POM)
* [Минимальный POM](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html#Minimal_POM)
* [Наследование проекта](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html#Project_Inheritance)
  + [Пример 1](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html#Example_1)
  + [Пример 2](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html#Example_2)
* [Агрегация проектов](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html#Project_Aggregation)
  + [Пример 3](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html#Example_3)
  + [Пример 4](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html#Example_4)
* [Наследование проекта против агрегирования проектов](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html#Project_Inheritance_vs_Project_Aggregation)
  + [Пример 5](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html#Example_5)
* [Интерполяция проекта и переменные](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html#Project_Interpolation)
  + [Доступные переменные](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html#Available_Variables)

**Что такое ПОМ ?**

Объектная модель проекта или POM - это основная единица работы в Maven. Это XML-файл, содержащий информацию о проекте и сведения о конфигурации, используемые Maven для создания проекта. Он содержит значения по умолчанию для большинства проектов. Примером этого является каталог сборки target; исходный каталог, который есть src/main/java; каталог исходного кода теста, который есть src/test/java; и так далее. При выполнении задачи или цели Maven ищет POM в текущем каталоге. Он читает POM, получает необходимую информацию о конфигурации, а затем выполняет цель.

Некоторые из конфигураций, которые могут быть указаны в POM, - это зависимости проекта, подключаемые модули или цели, которые могут быть выполнены, профили сборки и т. Д. Также может быть указана другая информация, такая как версия проекта, описание, разработчики, списки рассылки и т. Д.

[[верхняя]](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html)

**Супер ПОМ**

Super POM - это POM Maven по умолчанию. Все POM расширяют Super POM, если явно не установлены, то есть конфигурация, указанная в Super POM, наследуется POM, которые вы создали для своих проектов.

Вы можете увидеть [Super POM для Maven 3.6.3](http://maven.apache.org/ref/3.6.3/maven-model-builder/super-pom.html) в справочной документации Maven Core.

[[верхняя]](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html)

**Минимальный POM**

Минимальные требования для POM следующие:

* project корень
* modelVersion - должен быть установлен на 4.0.0
* groupId - id группы проекта.
* artifactId - id артефакта (проекта)
* version - версия артефакта в указанной группе

Вот пример:

1. <project>
2. <modelVersion>4.0.0</modelVersion>
4. <groupId>com.mycompany.app</groupId>
5. <artifactId>my-app</artifactId>
6. <version>1</version>
7. </project>

POM требует, чтобы были настроены его groupId, artifactId и версия. Эти три значения образуют полное имя артефакта проекта. Это имеет форму <groupId>: <artifactId>: <version>. Что касается приведенного выше примера, его полное имя артефакта - «com.mycompany.app:my-app:1».

Кроме того, как упоминалось в [первом разделе](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html#What_is_a_POM) , если детали конфигурации не указаны, Maven будет использовать их значения по умолчанию. Одно из этих значений по умолчанию - тип упаковки. У каждого проекта Maven есть тип упаковки. Если он не указан в POM, будет использоваться значение по умолчанию «jar».

Кроме того, вы можете видеть, что в минимальном POM *репозитории* не были указаны. Если вы создадите свой проект с использованием минимального POM, он унаследует конфигурацию *репозиториев* в Super POM. Поэтому, когда Maven видит зависимости в минимальном POM, он будет знать, что эти зависимости будут загружены из того, https://repo.maven.apache.org/maven2что было указано в Super POM.

[[верхняя]](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html)

**Наследование проекта**

Элементы в POM, которые объединяются, следующие:

* зависимости
* разработчики и участники
* списки плагинов (включая отчеты)
* выполнение плагина с совпадающими идентификаторами
* конфигурация плагина
* Ресурсы

Super POM - это один из примеров наследования проекта, однако вы также можете ввести свои собственные родительские POM, указав родительский элемент в POM, как показано в следующих примерах.

**Пример 1**

**Сценарий**

В качестве примера давайте повторно используем наш предыдущий артефакт com.mycompany.app:my-app:1. И давайте представим еще один артефакт, com.mycompany.app:my-module:1.

1. <project>
2. <modelVersion>4.0.0</modelVersion>
4. <groupId>com.mycompany.app</groupId>
5. <artifactId>my-module</artifactId>
6. <version>1</version>
7. </project>

И давайте определим их структуру каталогов следующим образом:

.

|-- my-module

| `-- pom.xml

`-- pom.xml

**Примечание:** my-module/pom.xml это POM com.mycompany.app:my-module:1, а pom.xmlэто POM com.mycompany.app:my-app:1

**Решение**

Теперь, если мы превратим com.mycompany.app:my-app:1 в родительский артефакт com.mycompany.app:my-module:1, нам придется изменить com.mycompany.app:my-module: 1 POM в следующую конфигурацию:

**com.mycompany.app:my-module:1's POM**

1. <project>
2. <modelVersion>4.0.0</modelVersion>
4. <parent>
5. <groupId>com.mycompany.app</groupId>
6. <artifactId>my-app</artifactId>
7. <version>1</version>
8. </parent>
10. <groupId>com.mycompany.app</groupId>
11. <artifactId>my-module</artifactId>
12. <version>1</version>
13. </project>

Обратите внимание, что теперь у нас есть добавленный раздел, родительский раздел. Этот раздел позволяет нам указать, какой артефакт является родительским для нашего POM. И мы делаем это, указывая полное имя артефакта родительского POM. При такой настройке наш модуль теперь может наследовать некоторые свойства родительского POM.

В качестве альтернативы, если мы хотим, чтобы groupId и / или версия ваших модулей были такими же, как их родительские, вы можете удалить groupId и / или идентификатор версии вашего модуля в его POM.

1. <project>
2. <modelVersion>4.0.0</modelVersion>
4. <parent>
5. <groupId>com.mycompany.app</groupId>
6. <artifactId>my-app</artifactId>
7. <version>1</version>
8. </parent>
10. <artifactId>my-module</artifactId>
11. </project>

Это позволяет модулю наследовать groupId и / или версию своего родительского POM.

[[верхняя]](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html)

**Пример 2**

**Сценарий**

Однако это сработает, если родительский проект уже установлен в нашем локальном репозитории или находится в этой конкретной структуре каталогов (родительский проект на pom.xmlодин каталог выше, чем у модуля pom.xml).

Но что, если родительский элемент еще не установлен и структура каталогов такая, как в следующем примере?

.

|-- my-module

| `-- pom.xml

`-- parent

`-- pom.xml

**Решение**

Чтобы обратиться к этой структуре каталогов (или любой другой структуре каталогов), нам нужно будет добавить <relativePath>элемент в наш родительский раздел.

1. <project>
2. <modelVersion>4.0.0</modelVersion>
4. <parent>
5. <groupId>com.mycompany.app</groupId>
6. <artifactId>my-app</artifactId>
7. <version>1</version>
8. <relativePath>../parent/pom.xml</relativePath>
9. </parent>
11. <artifactId>my-module</artifactId>
12. </project>

Как следует из названия, это относительный путь от модуля pom.xmlк родительскому pom.xml.

**Агрегация проектов**

Агрегация проектов похожа на [наследование проектов](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html#Project_Inheritance) . Но вместо указания родительского POM из модуля он указывает модули из родительского POM. Таким образом, родительский проект теперь знает свои модули, и если команда Maven вызывается для родительского проекта, эта команда Maven будет выполнена и для родительских модулей. Чтобы выполнить агрегирование проекта, вы должны сделать следующее:

* Измените упаковку родительских POM на значение «pom».
* Укажите в родительском POM каталоги его модулей (дочерних POM).

[[верхняя]](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html)

**Пример 3**

**Сценарий**

Учитывая предыдущие исходные POM артефактов и структуру каталогов:

**com.mycompany.app:my-app:1's POM**

1. <project>
2. <modelVersion>4.0.0</modelVersion>
4. <groupId>com.mycompany.app</groupId>
5. <artifactId>my-app</artifactId>
6. <version>1</version>
7. </project>

**com.mycompany.app:my-module:1's POM**

1. <project>
2. <modelVersion>4.0.0</modelVersion>
4. <groupId>com.mycompany.app</groupId>
5. <artifactId>my-module</artifactId>
6. <version>1</version>
7. </project>

**структура каталогов**

.

|-- my-module

| `-- pom.xml

`-- pom.xml

**Решение**

Если мы собираемся объединить my-module в my-app, нам нужно будет только изменить my-app.

1. <project>
2. <modelVersion>4.0.0</modelVersion>
4. <groupId>com.mycompany.app</groupId>
5. <artifactId>my-app</artifactId>
6. <version>1</version>
7. <packaging>pom</packaging>
9. <modules>
10. <module>my-module</module>
11. </modules>
12. </project>

В измененном com.mycompany.app:my-app:1 были добавлены разделы упаковки и модулей. Для упаковки было установлено значение «pom», а для раздела модулей у нас есть элемент <module>my-module</module>. Значение <module>- это относительный путь от com.mycompany.app:my-app:1 до POM com.mycompany.app:my-module:1 ( *на практике мы используем artifactId модуля в качестве имени каталога модуля* ).

Теперь, когда команда Maven обрабатывает com.mycompany.app:my-app:1, та же самая команда Maven будет запускаться и против com.mycompany.app:my-module:1. Более того, некоторые команды (в частности, цели) по-разному обрабатывают агрегирование проекта.

[[верхняя]](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html)

**Пример 4**

**Сценарий**

Но что, если мы изменим структуру каталогов на следующую:

.

|-- my-module

| `-- pom.xml

`-- parent

`-- pom.xml

Как родительский POM определил бы свои модули?

**Решение**

Ответ? - аналогично примеру 3, указав путь к модулю.

1. <project>
2. <modelVersion>4.0.0</modelVersion>
4. <groupId>com.mycompany.app</groupId>
5. <artifactId>my-app</artifactId>
6. <version>1</version>
7. <packaging>pom</packaging>
9. <modules>
10. <module>../my-module</module>
11. </modules>
12. </project>

**Наследование проекта против агрегирования проектов**

Если у вас есть несколько проектов Maven, и все они имеют схожие конфигурации, вы можете провести рефакторинг своих проектов, выбрав эти похожие конфигурации и создав родительский проект. Таким образом, все, что вам нужно сделать, это позволить вашим проектам Maven унаследовать этот родительский проект, и эти конфигурации затем будут применены ко всем из них.

А если у вас есть группа проектов, которые создаются или обрабатываются вместе, вы можете создать родительский проект, и этот родительский проект объявит эти проекты как свои модули. Поступая так, вам нужно будет построить только родительский элемент, а остальное приложится.

Но, конечно, у вас может быть как наследование проектов, так и агрегирование проектов. Это означает, что вы можете указать в своих модулях родительский проект и в то же время указать в родительском проекте эти проекты Maven в качестве своих модулей. Вам просто нужно применить все три правила:

* Укажите в каждом дочернем POM, кто является их родительским POM.
* Измените упаковку родительских POM на значение «pom».
* Укажите в родительском POM каталоги его модулей (дочерних POM)

[[верхняя]](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html)

**Пример 5**

**Сценарий**

Учитывая предыдущие исходные POM артефактов снова,

**com.mycompany.app:my-app:1's POM**

1. <project>
2. <modelVersion>4.0.0</modelVersion>
4. <groupId>com.mycompany.app</groupId>
5. <artifactId>my-app</artifactId>
6. <version>1</version>
7. </project>

**com.mycompany.app:my-module:1's POM**

1. <project>
2. <modelVersion>4.0.0</modelVersion>
4. <groupId>com.mycompany.app</groupId>
5. <artifactId>my-module</artifactId>
6. <version>1</version>
7. </project>

и эта **структура каталогов**

.

|-- my-module

| `-- pom.xml

`-- parent

`-- pom.xml

**Решение**

Чтобы выполнить как наследование, так и агрегирование проекта, вам нужно только применить все три правила.

**com.mycompany.app:my-app:1's POM**

1. <project>
2. <modelVersion>4.0.0</modelVersion>
4. <groupId>com.mycompany.app</groupId>
5. <artifactId>my-app</artifactId>
6. <version>1</version>
7. <packaging>pom</packaging>
9. <modules>
10. <module>../my-module</module>
11. </modules>
12. </project>

**com.mycompany.app:my-module:1's POM**

1. <project>
2. <modelVersion>4.0.0</modelVersion>
4. <parent>
5. <groupId>com.mycompany.app</groupId>
6. <artifactId>my-app</artifactId>
7. <version>1</version>
8. <relativePath>../parent/pom.xml</relativePath>
9. </parent>
11. <artifactId>my-module</artifactId>
12. </project>

**ПРИМЕЧАНИЕ.** Наследование профиля - та же стратегия наследования, что и для самого POM.

[[верхняя]](http://maven.apache.org/guides/introduction/introduction-to-the-pom.html)

**Интерполяция проекта и переменные**

Одна из практик, которую рекомендует Maven, - *не повторяться* . Однако бывают обстоятельства, когда вам нужно будет использовать одно и то же значение в нескольких разных местах. Чтобы помочь гарантировать, что значение указывается только один раз, Maven позволяет вам использовать как свои собственные, так и предварительно определенные переменные в POM.

Например, чтобы получить доступ к project.versionпеременной, вы должны ссылаться на нее следующим образом:

1. <version> $ {project.version} </version>

Следует отметить, что эти переменные обрабатываются *после* наследования, как описано выше. Это означает, что если родительский проект использует переменную, то в конечном итоге будет использоваться ее определение в дочернем, а не в родительском проекте.

**Доступные переменные**

**Переменные модели проекта**

На любое поле модели, представляющее собой элемент с одним значением, можно ссылаться как на переменную. Так , например, ${project.groupId}, ${project.version}, ${project.build.sourceDirectory}и так далее. Обратитесь к справке по POM, чтобы увидеть полный список свойств.

Все эти переменные обозначаются префиксом " project.". Вы также можете увидеть ссылки с pom.префиксом или полностью опущенный префикс - эти формы теперь устарели и не должны использоваться.

**Специальные переменные**

|  |  |
| --- | --- |
| project.basedir | Каталог, в котором находится текущий проект. |
| project.baseUri | Каталог, в котором находится текущий проект, представленный как URI. *Начиная с Maven 2.1.0* |
| maven.build.timestamp | Метка времени, обозначающая начало сборки (UTC). *Начиная с Maven 2.1.0-M1* |

Формат отметки времени сборки можно настроить, объявив свойство, maven.build.timestamp.formatкак показано в примере ниже:

1. <project>
2. ...
3. <properties>
4. <maven.build.timestamp.format>yyyy-MM-dd'T'HH:mm:ss'Z'</maven.build.timestamp.format>
5. </properties>
6. ...
7. </project>

Шаблон формата должен соответствовать правилам, приведенным в документации API для [SimpleDateFormat](https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html) . Если свойство отсутствует, по умолчанию используется формат значения, уже указанного в примере.

**Свойства**

Вы также можете ссылаться на любые свойства, определенные в проекте, как переменные. Рассмотрим следующий пример:

1. <project>
2. ...
3. <properties>
4. <mavenVersion>3.0</mavenVersion>
5. </properties>
7. <dependencies>
8. <dependency>
9. <groupId>org.apache.maven</groupId>
10. <artifactId>maven-artifact</artifactId>
11. <version>${mavenVersion}</version>
12. </dependency>
13. <dependency>
14. <groupId>org.apache.maven</groupId>
15. <artifactId>maven-core</artifactId>
16. <version>${mavenVersion}</version>
17. </dependency>
18. </dependencies>
19. ...
20. </project>