УДК 378.016:004

Иванов А.А., Колташёва Д.Д., Сардак Л.В.

ОБУЧЕНИЕ СТУДЕНТОВ РАБОТЕ С APACHE MAVEN ПРИ РАЗРАБОТКЕ ПРИЛОЖЕНИЙ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ JAVA

Аннотация

В статье обосновывается использование сборщика проектов Apache Maven. Описывается установка и его использование. Обсуждаются основные понятия и принципы работы сборщика. Делается заключение о необходимости обучения студентов, изучающих программирование на языке Java, сборщику проектов Apache Maven.

Ключевые слова: автоматизация сборки проектов, метод проектов, проектная деятельность, студенты, языки программирования.

Ivanov A.A., Koltasheva D.D., Sardak L.V.

STUDENT TRAINING WORKING WITH APACHE MAVEN FOR DEVELOPING APPLICATIONS IN LANGUAGE PROGRAMMING JAVA

Abstract

The article proves the use of the Apache Maven project collector. Describes the installation and its use. The basic concepts and principles of the maven work are discussed. The conclusion is made about the need to train students in the Java language, the project collector Apache Maven.

Keywords: build automation, project method, project activity, students, programming languages.

ПОСТАНОВКА ЗАДАЧИ

В настоящее время в мире компьютерных технологий все сложные приложения представляют собой совокупность большого количества файлов. Во время разработки проектов файлы постоянно требуется компилировать и сортировать по папкам. Почти каждое приложение нуждается в большом списке внешних библиотек, которые могут состоять из отдельных модулей. Все модули библиотек во время сборки программы, зависят друг от друга и требуют определенной последовательности. При сборке проекта необходимо учесть множество малейших нюансы. Именно эта работа требует многократного выполнения, на которое уходит много времени. При самостоятельной сборке проекта может возникнуть целый ряд проблем, для решения которых существуют автоматизированные системы. Такие системы называют сборщиками проектов. Они позволяют автоматизировать следующие действия: сборка бинарного кода, компиляция исходного кода в бинарный код, выполнение тестов, развертка программы на производственной платформе, описание изменений новой версии или написание сопроводительной документации [1].

Зачастую, начальное обучение программированию в высших учебных заведениях, строится на языке Java. В данный момент, этот язык является одним из самых востребованных на рынке труда, имеет долгосрочные перспективы и прост в изучении. Для Java проектов существует два основных сборщика –

Apache Ant и Apache Maven. Оба сборщика разработаны компанией Apache Software Foundation. Проведем их сравнительный анализ. Основное отличие Maven от Ant в том, что он обеспечивает декларативный тип сборки, а не императивный, то есть в файлах описания проекта содержатся не отдельные команды выполнения, а его спецификация. Одной из сильных сторон Maven можно назвать его высокий уровень интеграции с самыми популярными средами разработки, такими как Eclipse, IntelliJ IDEA, NetBeans и др. Это позволяет разрабатывать проект без привязки к конкретной среде разработки. Также положительным качеством Maven является наличие формальных соглашений по разработке, например, общая структура каталогов. Благодаря общей структуре не требуется вручную указывать сборщику где хранятся те или иные файлы, как это приходилось бы делать при использовании Ant [4]. Таким образом, на основе проведенного анализа, можно сделать вывод о том, что Maven является более функциональным и удобным в использовании, в отличие от Ant.

ИСПОЛЬЗОВАНИЕ APACHE MAVEN. ОСНОВНЫЕ ПОНЯТИЯ.

Для того, чтобы использовать Maven в практической деятельности, необходимо скачать бинарный архив с исходными файлами с официальной страницы загрузки Apache Maven https://maven.apache.org/download.cgi [2], после чего установить Maven, выполнив следующие действия:

1. Распаковать скачанный архив.

2. Добавить новую системную переменную среды, указав путь до внутренней папки «bin», которая находится в директории куда была произведена распаковка.

Чтобы убедиться в том, что установка прошла безошибочно, необходимо запустить командную строку и выполнить команду «mvn -version». Установка прошла успешно, если в ответ на команду пришли такие данные, как «Maven home», «Java version», «Java home» и т. д.

Рассмотрим основные понятия и термины Apache Maven.

- pom.xml (POM – Project Object Model) – основной файл, в котором описывается вся структура проекта. Файл pom.xml должен лежать в корневой папке проекта;
- зависимость – это сторонняя библиотека, которая используется в проекте;
- плагин – это дополнительная надстройка, обеспечивающая повышенный контроль над сборкой проекта;
- архетип – стандартное размещение файлов и каталогов для определенного типа проекта.

Для того чтобы создать новый Maven проект, необходимо выполнить из командной строки операцию «mvn archetype:generate». Выполнив эту операцию Apache Maven выведет для выбора список архетипов и их версий. После этого Maven запросит «groupId» и «artifactId» – эти параметры идентифицируют каждый проект. Как правило, параметр «groupId» имеет значение равное наименованию организации и формируется по таким же правилам, как и имена пакетов в Java – имя организации или сайт проекта. Параметр «artifactId» – это название

проекта. Последующие два параметра, номер версии проекта «version» и имя пакета «package», можно пропустить. Впоследствии Maven представит выбранные данные для проекта. По окончанию проделанных действий Maven создаст структуру проекта.

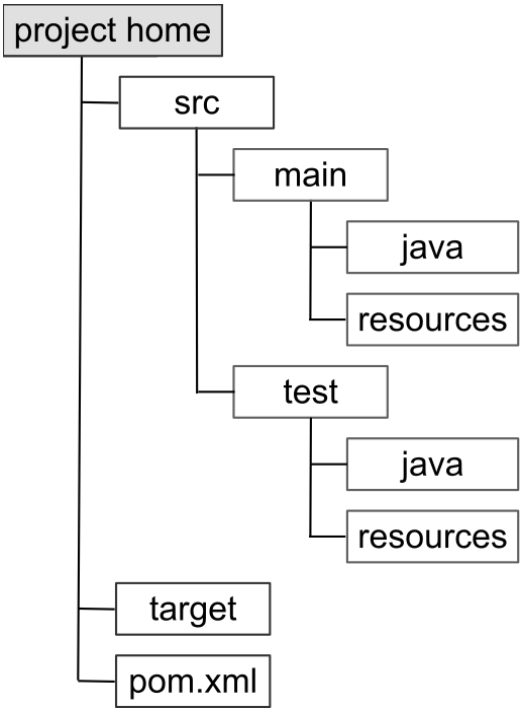

Рис. 1. Структура файлов проекта по умолчанию

Рассмотрим подробнее стандартное размещение каталогов (см. рис.1). В корневой папке расположен файл pom.xml и две директории. В директории «src» находятся все исходные файлы, в директории «target» находятся файлы, которые создает Maven при сборке проекта. Папка «target» может отсутствовать если проект ни разу не был собран. В папках «main/java» и «test/java» лежат исходный Java код и JUnit тесты соответственно. В папках «main/resources» и «test/resources», как правило, находятся вспомогательные файлы, необходимые для компиляции кода, такие как файлы конфигураций, медиа файлы и др. Файл pom.xml является основой проекта и написан на языке XML. В нем описаны основные зависимости, имя проекта, его свойства и пр. Разберем структуру pom.xml более детально.

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>ru.uspu</groupId>
  <artifactId>testProject</artifactId>
  <version>1.0-SNAPSHOT</version>
</project>
```

Рис. 2. Скриншот с минимальной конфигурацией pom.xml

На рис. 2 отображена минимальная конфигурация для работоспособности приложения. Исходя из представленной конфигурации очевидно следующее: pom.xml должен содержать в себе корневой тэг «project», тэг «modelVersion» должен содержать в себе номер версии не ниже 4.0.0, обязательно должны быть указаны идентифицирующие проект тэги «groupId» и «artifactId», а также номер версии приложения в тэге «version». Существует множество необязательных тэгов. Например, для подключения сторонних библиотек к приложению используются тэги «dependencies» и «dependency». Тэг «dependencies» определяет начало блока декларации зависимостей, а тэг «dependency» содержит в себе информацию о какой-либо библиотеке и находится внутри тэга «dependencies». Чтобы использовать стороннюю библиотеку достаточно поместить внутрь тэга «dependencies»

новую зависимость «dependency» и указать обязательные параметры – «groupId», «artifactId» и «version». Список самых востребованных библиотек можно найти на сайте центрального репозитория Maven.

Для сборки проекта Maven использует понятие «жизненный цикл». Это определенный список фаз, который определяет последовательность действий при построении проекта. Следует выделить главные фазы сборки:

- clean – очистка проекта - удаляет все лишние файлы, которые не задействованы в сборке;
- validate – выполнение проверки на полноту и правильность структуры;
- compile – компиляция исходного кода;
- test – JUnit тестирование подготовленными тестами собранного кода;
- package – упаковка ресурсов и откомпилированных классов в исполняемый файл jar;
- integration-test – интеграционное тестирование всего проекта, проверка взаимодействия между различными модулями проекта;
- install – полная сборка проекта;
- deploy – размещение проекта на удаленном сервере [3].

Жизненный цикл в Maven устроен последовательно. Например, если выполнить команду «mvn test», то будут выполнены все фазы, предшествующие этой команде, т.е. фазы clean, validate и compile. Вдобавок, Maven поддерживает составные команды, такую, например, как «mvn clean install». При выполнении такой команды Maven сначала очистит папку «target», а затем соберет проект заново. Для того, чтобы Maven корректно создавал исполняемые файлы, необходимо явно указать pom.xml где лежит файл с точкой входа main. Для этого требуется добавить плагин для сборки jar файлов (см. рис.3).

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-jar-plugin</artifactId>
      <version>3.1.0</version>
      <configuration>
        <archive>
          <manifest>
            <addClasspath>true</addClasspath>
            <classpathPrefix>lib/</classpathPrefix>
            <mainClass>ru.uspu.App</mainClass>
          </manifest>
        </archive>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Рис. 3. Плагин «maven-jar-plugin»

В тэге «mainClass» следует указать groupId проекта и имя файла, в котором находится точка входа в программу. После добавления плагина и выполнения команды «mvn install», Maven создаст исполняемый jar файл в каталоге target. Для того чтобы запустить исполняемый файл необходимо при по-

мощи командной строки перейти в каталог target и выполнить команду «java –jar имя_файла.jar».

МЕТОДИКА ОБУЧЕНИЯ APACHE MAVEN ДЛЯ СТУДЕНТОВ В РАМКАХ КУРСА «ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ».

Для изучения данной темы целесообразно выделить 8 академических часов, из них 4 часа на лекционные и 4 часа на практические занятия. В рамках лекций преподавателю необходимо рассказать студентам о том, с какими проблемами можно столкнуться при создании проектов, а также о том, какие существуют способы решения этих проблем. Необходимо ознакомить обучающихся с существующими сборщиками проектов, провести их сравнение. Также целесообразно рассказать о выбранном сборщике: цели использования, среда эксплуатации, основные понятия, принцип действия и практическое использование на конкретных примерах. На практических занятиях рекомендуем преподавателю вместе со студентами выполнить практические задания, в которых будет наглядно показаны примеры использования выбранного сборщика. Студентам также следует предложить выполнить самостоятельно ряд практических заданий по использованию сборщика. Самостоятельные задания должны быть направлены на разные аспекты и возможности сборщика. Примеры практических заданий для студентов:

- рассмотрение разных архетипов проектов, их назначения и различий;
- подключение различных сторонних библиотек и демонстрация их использования;
- сборка проекта в исполняемые файлы с расширением, отличным от jar;
- написание простых программ и их исполнение;
- тестирование написанного кода.

ВЫВОД

Умение использовать Apache Maven будет полезно при создании студентами курсовых проектов, реализации открытого программного обеспечения, производственной разработке приложений пр. Maven является важнейшим инструментом в списке обязательных программ для любого Java разработчика. Этот инструмент необходимо знать и уметь им пользоваться, потому что большинство реальных проектов, разрабатываемых на Java, используют именно Apache Maven.

ЛИТЕРАТУРА:

1. Автоматизация сборки // ru.wikipedia.org: Википедия – свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/Автоматизация_сборки (дата обращения: 10.04.2018).

2. Downloading Apache Maven 3.5.3 // Apache Maven Project. URL: https://maven.apache.org/download.cgi (дата обращения: 10.04.2018).

3. Maven – автоматизация сборки проекта // habrahabr.ru: Хабрахабр. URL: https://habrahabr.ru/post/77333/ (дата обращения: 10.04.2018).

4. Maven vs Ant // apache-maven.ru: Руководство по maven. URL: http://www.apache-maven.ru/maven_vs_ant.html (дата обращения: 10.04.2018).