Основы Maven. Практика

№ урока: 6 **Курс:** Основы Java EE

Средства обучения: InteliJ Idea, MySQL Workbench

Обзор, цель и назначение урока

Знакомство с Maven, как со сборщиком проекта. Рассмотрение шаблона Command и Singleton. Пример веб проекта.

Изучив материал данного занятия, учащийся сможет:

- Научится использовать Maven на базовом уровне.
- Поймет принцип работы шаблонов проектирования Command и Singleton.
- Писать первые веб приложения.

Содержание урока

1. Знакомство с Maven.
2. Структура проекта Maven.
3. Установка Maven.
4. Жизненный цикл Maven.
5. Недостатки Maven.
6. Пример веб проекта.
7. Шаблоны Command и Singleton

Резюме

- **Apache Maven** — фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM, являющемся подмножеством XML. Проект Maven издаётся сообществом Apache Software Foundation, где формально является частью Jakarta Project.
  Название программы, maven, — является словом из языка идиш, смысл которого можно примерно выразить как «собиратель знания».
  Maven обеспечивает декларативную, а не императивную (в отличие от средства автоматизации сборки Apache Ant) сборку проекта. В файлах описания проекта содержится его спецификация, а не отдельные команды выполнения. Все задачи по обработке файлов, описанные в спецификации, Maven выполняет посредством их обработки последовательностью встроенных и внешних плагинов.
  Maven используется для построения и управления проектами, написанными на Java, C#, Ruby, Scala, и других языках.
- **POM** – информация для сборки проекта, поддерживаемого Apache Maven, содержится в XML-файле с названием pom.xml. При запуске Maven проверяет, содержит ли конфигурационный файл все необходимые данные и все ли данные синтаксически правильно записаны.
  Минимальная конфигурация включает версию конфигурационного файла, имя проекта, его автора и версию. С помощью pom.xml конфигурируются зависимости от других проектов, индивидуальные фазы процесса построения проекта (build process), список плагинов, реализующих порядок сборки.
  Крупные проекты могут быть поделены на несколько модулей, или подпроектов, каждый со своим собственным POM. Операции над модулями могут выполняться через общий корневой POM единой командой.
  POM файлы подпроектов могут наследовать конфигурацию от других файлов конфигурации. В то же время все файлы конфигурации обязательно наследуются от «Супер POM» файла по умолчанию. Супер POM обеспечивает конфигурацию по умолчанию, например, стандартная

CyberBionic systematics

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

структура каталогов, используемые по умолчанию плагины, привязка к фазам жизненного цикла и прочее.

- **Архетипы**— Maven использует принцип *Maven-архетипов* (англ. *Archetypes*). Архетип — это инструмент шаблонов, каждый из которых определён паттерном или моделью, по аналогии с которой создаются производные.
  Стандартная структура каталогов — одна из реализаций принципа архетипов в Maven. Следующая структура показывает важнейшие каталоги для проекта на Java:
  Корневой каталог проекта: файл *pom.xml* и все дальнейшие подкаталоги
  - *src*: все исходные файлы
    - *src/main*: исходные файлы собственно для продукта
      - *src/main/java*: Java-исходный текст
      - *src/main/resources*: другие файлы, которые используются при компиляции или исполнении, например, Properties-файлы
    - *src/test*: исходные файлы, необходимые для организации автоматического тестирования
      - *src/test/java*: JUnit-тест-задания для автоматического тестирования
  - *target*: все создаваемые в процессе работы Мавена файлы
    - *target/classes*: компилированные Java-классы
- **Жизненный цикл** проекта — это список поименованных *фаз*, определяющий порядок действий при его построении. Жизненный цикл Maven содержит три независимых порядка выполнения:
  clean — жизненный цикл для очистки проекта. Содержит следующие фазы:
  - pre-clean
  - clean
  - post-clean

  default — основной жизненный цикл, содержащий следующие фазы:
  - validate - выполняется проверка, является ли структура проекта полной и правильной.
  - generate-sources
  - process-sources
  - generate-resources
  - process-resources
  - compile - компилируются исходные тексты.
  - process-test-sources
  - process-test-resources
  - test-compile
  - test - собранный код тестируется заранее подготовленным набором тестов.
  - package - упаковка откомпилированных классов и прочих ресурсов. Например, в JAR-файл.
  - integration-test - программное обеспечение в целом или его крупные модули подвергаются интеграционному тестированию. Проверяется взаимодействие между составными частями программного продукта.
  - install - установка программного обеспечения в локальный Maven-репозиторий, чтобы сделать его доступным для других проектов текущего пользователя.
  - deploy - стабильная версия программного обеспечения распространяется на удаленный Maven-репозиторий, чтобы сделать его доступным для других пользователей.

  site — жизненный цикл генерации проектной документации. Состоит из фаз:
  - pre-site
  - site
  - post-site
  - site-deploy

  Стандартные жизненные циклы могут быть дополнены функционалом с помощью Maven-плагинов. Плагины позволяют вставлять в стандартный цикл новые шаги (например, распределение на сервер приложений) или расширять существующие шаги.

- Maven базируется на plugin-архитектуре, которая позволяет применять плагины для различных задач (*compile, test, build, deploy, checkstyle, pmd, scp-transfer*) для данного проекта, без необходимости их в явном виде инсталлировать. Это возможно за счет того, что информация поступает плагину через стандартный вход, а результаты пишутся в его стандартный выход.

CyberBionic systematics
CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Теоретически, это позволяет кому угодно писать плагины для взаимодействия со средствами построения проекта (компиляторы, средства тестирования, и т. п.) для любого другого языка. В реальности, поддержка других языков кроме Java сейчас минимальна. Существует плагин для .NET-фреймворка, а также плагины для C/C++.

Количество доступных плагинов в настоящее время очень велико и включает, в том числе, плагины, позволяющие непосредственно из Maven запускать web-приложение для тестирования его в браузере; плагины, позволяющие тестировать или создавать банки данных; плагины, позволяющие генерировать Web Services. Задачей разработчика в такой ситуации является найти и применить наиболее подходящий набор плагинов.

Плагин обеспечивает достижение ряда целей с помощью следующего синтаксиса:

Например, Java-проект может быть скомпилирован плагином-компилятором путём выполнения команды: `mvn compiler:compile`

mvn [имя плагина]:[имя цели]

Существуют Maven-плагины для построения, тестирования, контроля исходного текста, запуска web-сервера, генерации Eclipse-проектных файлов и множество других.[20] Плагины перечисляются и конфигурируются в секции `<plugins>` файла*pom.xml*. Некоторая базовая группа плагинов включается в каждый проект по умолчанию.

- **Зависимости** В файле pom.xml задаются зависимости, которые имеет управляемый с помощью Maven проект. Менеджер зависимостей основан на нескольких основных принципах:

  Репозитории. Maven ищет необходимые файлы в локальных каталогах или в локальном Maven-репозитории. Если зависимость не может быть локально разрешена, Maven подключается к указанному Maven-репозиторию в сети и копирует в локальный репозиторий. По умолчанию Maven использует Maven Central Repository, но разработчик может конфигурировать и другие публичные Maven-репозитории, такие, как Apache, Ibiblio, Codehaus или Java.Net.

  Транзитивные зависимости. Необходимые библиотеки подгружаются в проект автоматически. При разрешении конфликта версий используется принцип «ближайшей» зависимости, то есть выбирается зависимость, путь к которой через список зависимых проектов является наиболее коротким.

  Исключение зависимостей. Файл описания проекта предусматривает возможность исключить зависимость в случае обнаружения цикличности или отсутствия необходимости в определённой библиотеке.

  Поиск зависимостей. Поиск зависимостей (open-source-библиотек и модулей) ведётся по их координатам (groupId, artifactId и version). Эти координаты могут быть определены с помощью специальных поисковых машин, например, Maven search engine. Например, по поисковому признаку «pop3» поисковая машина предоставляет результат с groupId="com.sun.mail" и artifactId="pop3".

  Менеджеры репозиториев. Репозитории реализуются с помощью менеджеров репозиториев Maven (Maven Repository Manager), таких как Apache Archiva, Nexus (ранее Proximity), Artifactory, Codehaus Maven Proxy или Dead Simple Maven Proxy.

  Область распространения зависимости позволяет включать зависимости только на определённую стадию построения проекта. Существует 6 возможных областей:

  compile. Область по умолчанию. Зависимость доступна во всех путях поиска классов в проекте. Распространяется на зависимые проекты.

  provided. Область аналогична compile, за исключением того, что JDK или контейнер сам предоставит зависимость во время выполнения программы.

  runtime. Зависимость не нужна для компиляции, но нужна для выполнения.

  test. Зависимость не нужна для нормальной работы приложения, а нужна только для компиляции и запуска тестов.

  system. Область аналогична provided за исключением того, что содержащий зависимость JAR указывается явно. Артефакт не ищется в репозитории.

  import (начиная с версии Maven 2.0.9) используется только с зависимостью типа pom в секции `<dependencyManagement>`. Зависимости текущего POM заменяются на зависимости из указанного POM.

CyberBionic systematics

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Закрепление материала

- Что такое Maven?
- Что такое pom в мавен?
- Что такое зависимости в Maven?
- Какой структуры проекта Maven стоит придерживаться?
- Зачем нужен шаблон проектирования Singleton?
- Зачем нужен шаблон проектирования Command?

Самостоятельная деятельность учащегося

Задание 1
Выучите основные понятия, рассмотренные на уроке.

Задание 2
К практическому проекту добавить функционал удобной навигации меню и добавить возможность добавления нового пользователя как админа, который может добавить цветок. Выровнять проект немного через CSS.

Рекомендуемые ресурсы

POM
https://maven.apache.org/guides/introduction/introduction-to-the-pom.html

Уроки от IBM:
https://www.ibm.com/developerworks/java/tutorials/j-mavenv2

Книга
https://books.sonatype.com/mvnex-book/reference/public-book.html

CyberBionic systematics

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str.,5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com