**메이븐 설명**

|  |
| --- |
| Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM),  Maven can manage a project's build, reporting and documentation from a central piece of information. |

**메이븐 기능**

  - 빌드  
  - 문서화  
  - 리포팅  
  - 의존관계 관리  
  - 소스코드 관리  
  - 릴리즈  
  - 배포  
  
**메이븐 사용시 이점**  
  - 의존성 관리 (오픈소스 라이브러리 뿐만 아니라 프로젝트 혹은 무듈간의 의존성 또한 관리가 가능함)  
    + Java 개발자의 중요한 능력중에 하나는 오픈소스 라이브러리 혹은 프레임워크의 사용능력이라 생각된다. Java를 사용하여 S/W를 개발하는 경우 다양한 오픈소스 라이브러리를 사용하게 되는데... Maven의 의존성 관리를 이용하면 손쉽게 오픈소스 라이브러리들의 의존성을 관리할 수 있다.  
  - 잘 정의된 convention에 의해서 일관된 프로젝트 구성을 갖을 수 있다.  
  - IDE에 종속적인 부분을 제거할 수 있다. (Eclipse, IntelliJ, NetBeans, JBuilder, JCreator, JDveloper)  
  - 이클립스를 사용하는 경우 maven 이클립스 플러그인을 사용하면 의존성을 갖는 오픈소스의 소스 코드를 F3키를 사용하여 아주 쉽게 확인할 수 있다. (다른 IDE는 사용하지 않아서 잘 모름...)  
  - Maven을 알면 Java를 이용한 많은 오픈소스 프로젝트들이 Maven 프로젝트로 구성되어 있어 오픈소스를 분석할 때 편리하다.  
    + 코딩 스킬을 향상시키기 가장 좋은 방법중에 하나는 좋은 소스코드를 많이 읽오보는 것이다!!!  
  - Maven Profile 기능을 사용하면 배포환경에 따른 설정 파일을 관리하고 배포 파일을 생성할 수 있다.   
  - 의존 라이브러리를 pom.xml 파일을 통해서 관리하므로 버전(형상)관리 시스템으로 공유할 파일의 크기가 줄어든다.   
  
 **메이븐 단점**  
  - maven은 현재 3.x.x 버전까지 왔지만 일부의 기능에서 예상외로 동작하는 경우가 있다. (버그인건지 내가 몰라서 그러는건지?? 누가좀 알려줬으면 한다. ㅠ.ㅠ)  
     + Oracle JDBC Driver 때문에 Local 저장소가 깨진 경우 가 있었는데... 수작업으로 Local저장소를 정리해줘야만 했다. (완전 삽질함...)  
  - 이클립스의 maven 플러그인인 m2e가 가끔 원하는데로 동작을 안한다.  
  
**메이븐 설치**  
  - http://maven.apache.org/download.html에서 메이븐 최신 버전을 다운 받는다.   
  - 다운받은 압축파일을 원하는 경로에 풀어 놓고, 해당 경로를 시스템 환경 변수에 **"MAVEN\_HOME"** 을 추가한다.  
  - 시스템 환경 변수 **"PATH"**에 **"MAVEN\_HOME/bin"**을 추가한다. (mvn 명령을 편하게 사용하기 위함)  
 \* 참고로 maven의 기본 Local 저장소 위치는 **"USER\_HOME/.m2"** 이며,  
   maven 중앙 저장소는 **"http://repo1.maven.org/maven2/"** 임.  
  
**Maven 설정 파일**  
  - MAVEN\_HOME/settings.xml : 모든 사용자에 적용되는 전역적인 메이븐 설정 정보  
  - USER\_HOME/.m2/setting.xml : 특정 사용자에 적용되는 메이븐 설정 정보  
  - pom.xml : 메이븐 프로젝트 설정 파일  
    + 메이븐은 프로젝트와 관련된 정보를 Project Object Model (이하 POM)이라는 이름으로 정의하고 있으며, POM의 이름을 따서 기본 설정 파일은 **"pom.xml"** 파일이다. (다른 이름도 사용할 수 있지만... 일반적인 관례를 따르는게 편하고 협업하기도 좋다고 생각함.)  
  
**Maven 기본 명령어**  
 - maven [options] [<goal(s)>] [<phase(s)>]   
  
**메이븐 POM 설정 파일**  
 - 프로젝트 기본 정보  
  + 프로젝트 이름, URL, 개발자, 라이선스 등등  
 - 빌드 설정  
  + 기본 빌드 설정을 변경하기 위한 카테고리 (소스, 리소스 디렉토리 변경, 플러그인 설정 변경 및 플러그인 추가)  
 - 프로젝트 관계 설정  
  + 각 프로젝트간의 관계 관리 혹은 모듈간의 의존성 관리  
 - 빌드 환경  
  + 다양한 환경에 따라 달라지는 설정 정보를 관리 (프로파일 기능)  
 - 속성 정보 관리  
  + pom/project 속성, settings 속성, 시스템 환경 변수 속성, 자바 시스템 속성등을 정의 또는 사용할 수 있다.  
 - 메이븐은 기본적인 프로젝트 디렉토리 구조와 플러그인 정보를 포함하고 있는 POM설정파일을 제공하며 모든 POM 설정 파일은 해당 설정파일을 상속한다. (해당 설정 파일은 메이븐에 포함되어 있는 jar 파일을 통하여 배포된다.)  
  
**메이븐 라이프사이클**  
 - 메이븐은 미리 정의하고 있는 빌드 순서를 라이프사이클 이라고 하며, 라이프사이클의 각 빌드 단계를 페이지라고 한다.  
 - 라이프사이클은 여러 단계(페이즈)로 나뉘어져 있으며, 각 페이즈는 의존 관계를 갖는다.  
 - 기본 라이프 사이클  
  + compile : 소스 코드를 컴파일한다  
  + test : 단위 테스트 실행 (기본설정은 단위 테스트가 실패하면 빌드 실패로 간주함)  
  + package : 컴파일된 클래스 파일과 리소스 파일들을 war 혹은 jar와 같은 파일로 패키징  
  + install : 패키징한 파일을 로컬 저장소에 배포 (USER\_HOEM/.m2/)  
  + deploy : 패키징한 파일을 원격 저장소에 배포 (nexus 혹은 maven central 저장소)  
 - clean 라이프 사이클  
  + clean : 메이븐 빌드를 통하여 생성된 모든 산출물을 삭제한다.  
 - site 라이프 사이클  
  + site : 메이븐 설정파일 정보를 활용하여 프로젝트에 대한 문서 사이트를 생성한다.  
  + site-deploy : 생성한 문서 사이트를 설정되어 있는 서버에 배포  
  
**메이븐 페이즈와 플러그인**  
 - 메이븐에서 제공하는 모든 기능은 플러그인 기반으로 동작한다. 메이븐 라이프사이클에 포함되어 있는 페이즈 또한 플러그인을 통하여 실직적인 작업이 실행된다.   
 - <bild>/<plugins>/<plugin> 엘리먼트를 사용하여 사용하고자 하는 플러그인을 추가 및 설정 할 수 있다.   
 - 메이븐 플러그인은 하나의 플러그인에서 여러 작업을 수행할 수 있도록 지원하며, 플러그인에서 실행할 수 있는 각각의 작업을 goal 이라고 정의한다.  
 - 플러그인 goal 실행 방법   
  + mvn groupId:artifactId:version:goal   
   \* 사용하기 완전 구리다 언제 저런걸 다 외우고 있어!!!  
   \* 그래서 친절한 메이븐은 플러그인을 좀더 쉽게 사용할 수 있도록 다은과 같은 규칙이 있다.

|  |
| --- |
| - 로컬 저장소에 설치되어 있는 가장 최신 버전의 플러그인을 실행하기 원한다면 version 정보 생략가능  - artifactId가 'maven-$name-plugin' 과 '$name-maven-plugin' 규칠을 따른다면 groupId:$name:goal 형식으로 실행할  수 있다.  - 메이븐의 중앙 저장소에 위치한 플러그인을 찾기 위한 groupId 목록을 settings.xml 파일에서 관리하며, 해당  groupId에 속해 있으면, groupId 를 생략할 수 있다. ($name:goal 형태이며, 기본설정만으로 왠만한 플러그인들은  groupId를 생략해서 사용한다.) |

 - 페이즈와 플러그인 관계  
  + 각 페이즈의 작업은 메이븐에서 기본으로 포함하고 있는 각 플러그인의 골을 이용해서 수행한다.   
 - 메이븐 플러그인을 검색하고 사용방법을 확인할 수 있는 곳 (아래에도 없다면 구글 검색이 짱!!!)  
  + http://maven.apache.org/plugins/index.html  
  + http://mojo.codehause.org/plugins.html   
  
**메이븐을 이용한 의존성 관리**  
 - 메이븐의 핵심 기능중에 하나이며, 개발자들이 제일 좋아하는 기능이지만 문제가 발생한 가능성이 가장 높은 기능이기도 하다. (ㅠ.ㅠ) 따라서 해당 기능을 잘 이해해야 메이븐을 편하게 잘 사용할 수 있다. 그렇지 못하면 메이븐을 열라 욕하게 될 것 이다. (메이븐 구려서가 아니라 내가 메이븐을 잘 못 이해했기 때문!!!)  
 - 메이븐 저장소 구성  
  + 중앙 저장소 : 오픈소스 라이브러리, 메이븐 플러그인, 메이븐 아키타입을 관리하는 저장소이다. 중앙 저장소는 개발자가 임의로 라이브러리를 뱁포할 수 없다.   
  + 원격 저장소 : 메이븐 중앙 저장소이 외에 각각의 회사 혹은 오픈소스 재단에서 운영 관리하는 저장소   
   ex) http://maven.springframework.org (sprintsource), http://mesir.googlecode.com/svn/trunk/mavenrepo (ojdbc), http://192.168.1.46:5050/nexus (사내 maven 저장소)  
  + 로컬 저장소 : 메이븐을 빌드할 때 다운로드하는 라이브러리, 플러그인을 관리하는 개발자 PC의 저장소 (USER\_HOME/.m2)  
 - 메이븐 저장소 설정 (xml 파일예제..)  
 - 메이븐 의존성 관리  
  + <dependencies>/<dependency> 엘리먼트를 사용하여 의존성을 관리하며, 의존 라이브러리의 groupId, artifactId, version, scope 정보들을 갖는다.  
 - 메이븐 의존성 정보에서 scope 설명  
  + compile : 기본 scope, 컴파일 및 배포할 때 같이 제공해야 하는 라이브러리  
  + provided : servlet.jar 와 같이 컴파일 시점에는 필요하지만 배포할 때에는 포함되지 말아야 하는 라이브러리  
  + runtime: 컴파일 시에는 사용되지 않지만 실행환경에서 사용되어지는 라이브러리  
  + test : JUnit과 같이 테스트 시점에만 사용되는 라이브러리  
  + system : provided와 비슷하다. 단지 우리가 직접 jar 파일을 제공해야 한다. 따라서 이 scope의 jar 파일은 저장소에서 관리되지 않을 수도 있다.  
  + import : 다른 POM설정 파일에 저의되어 있는 의존 관계 설정을 현재 프로젝트로 가져온다.  
 - 의존성 전이  
  + 오픈소스 프레임워크를 의존성에 추가하며, 해당 프레임워크가 의존하고 있는 오픈소스 라이브러리 또한 의존 관계에 자동으로 포함된다.   
  + 의존성 전이 기능은 프로젝트의 의존성을 편리하게 관리할 수 있도록 도와주기도 하지만 불필요한 라이브러리가 추가되거나 의존성이 꼬이게 만드는 원인이 되기도 한다.   
 - 의존성 전이에 대한 설정 변경기능  
  + 의존성 중개 : 버전이 다른 두개의 라이브러리가 의존 관계에 있다면 메이븐은 더 가까운 의존 관계에 있는 pom 설정의 버전과 의존관계를 갖는다. 예를 들어 A 프로젝트가 A -> B -> C -> D2.0 버전, A -> E -> D1.0 버전의 의존 관계가 발생한다면, A 프로젝트는 D1.0 버전과 의존 관계를 갖는다. 만약 D2.0 버전을 사용하고 싶다면 A 프로젝트의 메이븐 설정 파일에 명확하게 의존 관계를 명시해야 한다. (A -> D.20)  
  + 의존성 관리 : <dependencyManagement> 엘리먼트를 사용하여 의존 관계에 있는 라이브러리와 버전을 명시적으로 정의한다.   
  + 의존성 예외 : <exclusion> 엘리먼트를 활용하여 의존성 전이를 예외처리한다.  
  + 기타 : 의존성 스코프, 선택적 의존성 등의 기능이 있음  
  
**Maven 기반의 프로젝트 생성**  
 1. mvn 명령어를 이용해서 템플릿 프로젝트 생성

|  |
| --- |
| $ mvn archetype:generate -DgroupId=com.uangel -DartifactId=maven -DarchetypeArtifactId=maven-archetype-quickstart  -DinteractiveMode=false |

|  |
| --- |
| $ mvn archetype:generate -DarchetypeCatalog=internal |

  + 일반적으로 groupId는 생성하는 프로젝트의 도메인명을 사용하며, artifactId는 프로젝트 이름을 사용한다.  
  
 2. 이클립스와 m2e 플러그인을 사용하여 maven 프로젝트 생성 (아마도 제일 쉬운 방법)   
  + m2e 플러그인 Update URL : http://download.eclipse.org/technology/m2e/releases/1.0/1.0.100.20110804-1717  
  + m2e 플러그인 설치 후 "Window > Perferences > java > Build Path > Classpath Variables" 에 **"M2\_REPO"** 설정 추가 확인  
  + 가장 간단한 maven-archetype-quickstart 아키타입 프로젝트 생성   
  
 3. 수작업 (maven을 잘 이해하고 있다면 그리 어렵지 않게 세팅 가능)  
  - pom.xml 파일 생성, pom.xml 파일 설정에 맞게 디렉토리 생성...   
  
**웹서비스 개발을 위한 Maven 프로젝트 구성 및 IDE(이클립스) 통합** - 이클립스와 m2e 플러그인을 사용하여 프로젝트 생성  
  + **"File > New > Other... > Maven Project > Next > maven-archetype-webapp 아키타입선택 > ..."**  
  + **"프로젝트 선택 > 우클릭 > Maven > Update Project Configuration..."**선택 (pom.xml 파일 수정시 이클립스 프로젝스 설정에 반영되도록 설정)  
 - 웹 리소스 디렉토리 경로 변경 (src/main/webapp --> webapp)

|  |
| --- |
| <plugin>  <artifactId>maven-war-plugin</artifactId>  <version>2.1.1</version>  <configuration>  <warSourceDirectory>webapp</warSourceDirectory>  <warName>maven-test-web</warName>  </configuration>  </plugin> |

 - maven eclipse 플러그인 설정 추가

|  |
| --- |
| <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-eclipse-plugin</artifactId>  <version>2.8</version>  <configuration>  <downloadSource>true</downloadSource>  <downloadJavadocs>true</downloadJavadocs>  <wtpversion>2.0</wtpversion>  <wtpContextName>/</wtpContextName>  </configuration>  </plugin>  </plugins> |

- Java 소스 디렉토리 및 리소스 디렉토리 생성  
  + src/main/java  
  + src/main/resources  
  + src/test/java  
  + src/test/resources  
- 메이븐 eclipse 플러그인 실행 (위에서 추가한 설정 및 디렉토리를 m2e 가 제대로 반영하지 못함!!!)  
  + mvn eclipse:clean  eclipse:eclipse   
  
- 생성되어 있는 Maven Project Import 방법  
  + Existing Maven Projects  
  + Check out Maven Projects from SCM  
  
**메이븐 프로파일 기능**  
 - 메이븐은 서로 다른 환경에 따라 달라지는 설정을 다르게 관리할 수 있는 프로파일 기능을 제공한다.

 - 프로파일은 <profiles> -> <profile> 엘리먼트를 사용하여 설정할 수 있으며, <project> 루트 엘리먼터가 가질 수 있는 모든 엘리먼트를 설정할 수 있다.    
 - 프로파일에 따른 설정 예제

|  |
| --- |
| <profiles>  <profile>  <id>local</id>                 <!-- 기본 Profile 설정 -->  <activation>  <activeByDefault>true</activeByDefault>  </activation>  <properties>  <environment>local</environment>  </properties>  <dependencies>  <!-- JDBC Driver -->  <dependency>  <groupId>com.oracle</groupId>  <artifactId>ojdbc14</artifactId>  <version>10.2.0.4.0</version>  </dependency>  </dependencies>  </profile>  <profile>  <id>development</id>  <properties>  <environment>development</environment>  </properties>  <dependencies>  <!-- JDBC Driver -->  <dependency>  <groupId>com.oracle</groupId>  <artifactId>ojdbc14</artifactId>  <version>10.2.0.4.0</version>  </dependency>  </dependencies>  </profile>    <profile>  <id>service</id>  <properties>  <environment>service</environment>  </properties>  <dependencies>  <!-- JDBC Driver -->  <dependency>  <groupId>com.oracle</groupId>  <artifactId>ojdbc6</artifactId>  <version>11.1.0.7.0</version>  </dependency>  </dependencies>  </profile>  </profiles>  <build>          <!-- Profile 기능을 이용한 리소스 관리 -->  <resources>  <resource>  <directory>src\main\resources</directory>  </resource>  <resource>  <directory>src\main\resources-${environment}</directory>  </resource>  <resource>  <directory>src\main\java</directory>  <excludes>  <exclude>\*\*/\*.java</exclude>  </excludes>  </resource>  </resources>  </build> |

 - 메이븐 사용시 프로파일 지정  
  + mvn -P<profile\_id> pahse[s] goal[s]  
  ex) mvn -Pdevelopment clean package  // 개발서버에 배포할 패키지 파일 생성  
        mvn -Pservice clean package  // 상용 서버에 배포할 패키지 파일 생성  
   
**메이븐 모듈**  
 - 메이븐은 하나의 프로젝트에서 여러 모듈을 관리할 수 있는 기능을 지원한다.   
 - 일반적으로 서비스를 개발하게 되면 관리자 페이지 혹은 CMS(Content Management System) 과 고객 혹은 가입자에게 서비스 제공을 위한 Web Module이 분리가 되며, 두개의 모듈이 핵심 비지니스 로직을 공유하게된다. 이런 경우 메이븐의 모듈 기능을 이용하면 유용하다.

 - 메이븐의 모듈 개념을 이해하고 활용하려면 상속, 집합, 의존 관계 개념을 이해해야 한다.  
 - 상속  
  + 프로젝트에서 공통으로 사용하는 설정은 공통 POM 파일을 만들어 관리하고 하위 모듈에서 이 POM파일을 상속할 수 있다.   
 \* 부모 POM 파일

|  |
| --- |
| <project>    <modelVersion>4.0.0</modelVersion>    <groupId>net.javajigi</groupId>    <artifactId>wikibook</artifactId>   <packaging>pom</packaging>    <version>1.0-SNAPSHOT</version>    [...]  </project> |

 \* 부모 POM 파일을 상속하는 자식 POM 파일

|  |
| --- |
| <project>    <modelVersion>4.0.0</modelVersion>    <parent>      <groupId>net.javajigi</groupId>      <artifactId>wikibook</artifactId>      <version>1.0-SNAPSHOT</version>    </parent>    <groupId>net.javajigi</groupId>    <artifactId>wiki-web-parent</artifactId>    <version>1.0-SNAPSHOT</version>    [...]  </project> |

 - 집합