Maven Project란?

: 자바 소스 파일(war 또는 jar 파일)을 빌드, 라이브러리 의존성, 컴파일, 배포를 해결해주는 도구입니다. 쉽게 말해 자바용 프로젝트 관리도구이며, 이것을 사용하는 프로젝트가 바로 Maven Project입니다.

Pom.xml

: pom.xml(Project Object Model)은 Maven의 기본 작업 단위이며, 프로젝트를 빌드하기 위해 Maven에서 사용하는 프로젝트 및 구성 세부사항이 포함된 XML파일이다.

* <property> : pom.xml 내의 dependency 환경설정을 위한 변수를 SET
* source, target : default 자바 버전 설정
* <dependency> : 여기다가 프로젝트에서 사용할 라이브러리들을 명시

(명시된 라이브러리들은 “메이븐 중앙 저장소”에서 자동으로 다운받아준다.)

* <build>내 <plugin> : Maven project build를 위한 라이브러리 명시

WAR, JAR

* Jar : 동작에 톰캣이 필요없는 애플리케이션으로 bin 내에 있는 .class 파일을 패키지로 묶은 것
* War : 톰캣 위에서 동작하는 애플리케이션으로 webapp폴더 아래에 있는 파일들이 war로 묶여진다.(webapp 폴더는 크게 assets 폴더와 WEB-INF 폴더로 구성된다.)

Webapp/WEB-INF 폴더 아래로 이동되는 파일들

* Src/main/java : java파일이 컴파일되어 생성된 .class 파일은 /WEB-INF/classes 폴더 하위에 옮겨져 저장된다.
* Src/main/resource : 아래에 있는 xml 파일 및 mybatis 등의 파일들은 전부 복사되어 /WEB-INF/classes 폴더 하위에 저장된다.
* Maven Dependencies : Maven에 설정한 라이브러리들은 /WEB-INF/libs 폴더 하위에 저장된다.

프레임워크와 라이브러리의 차이

* 라이브러리

: 라이브러리를 사용하는 애플리케이션 코드는 애플리케이션 흐름을 직접 제어한다. 단지 동작하는 중에 필요한 기능이 있을 때 능동적으로 라이브러리를 사용할 뿐이다.

* 프레임워크

: 거꾸로 애플리케이션 코드가 프레임워크에 의해 사용된다. 보통 프레임워크 위에 개발한 클래스를 등록해두고, 프레임워크가 흐름을 주도하는 중에 개발자가 만든 애플리케이션 코드를 사용하도록 만드는 방식이다.

어노테이션 기법과 new 할당기법의 차이(결과는 동일해 보임)

: new 할당 기법으로 사용 시, 할당할 때마다 ‘동등’한 객체가 생성되지만,

어노테이션 기법으로 스프링 할당 시, 싱글턴 기법이 적용되 있기 때문에 ‘동일’한 객체가 가져와진다.(참조되어 진다, 객체는 하나만 생성)

------------------------- maven repository 경로 설정 ------------------------------

C:\Users\사용자\.m.2 경로에 들어가 setting.xml 파일 생성(localRepository에 경로지정)

* 내용

<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"

 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">

    <localRepository>D:\tools\repository</localRepository>

    <interactiveMode>true</interactiveMode>

    <offline>false</offline>

</settings>

이클립스 Preference -> maven -> user setting -> User setting에 setting.xml 경로 지정 및 Update

---------------------------- Junit 설정 ---------------------------------------------

Junit을 설정하는 방식은 Maven을 통한 설정과 이클립스 Library를 추가로 설정하는 두가지 방식이 있다.

Maven 방식과 이클립스 Library 추가 방식과의 차이는 환경설정에 있다.

Maven 방식은 프로젝트에 포함되는 파일인 “pom.xml”에 적용되어 형상관리 툴(Svn, Git)로 관리되는 프로젝트에서는 동일한 개발 환경으로 개발이 된다.

그러나 이클립스 Library 추가 방식은 개발자가 각자의 이클립스에 설정하는 것이기 때문에 형상관리 툴로 관리되는 프로젝트에서는 설정을 하지 않으면 오류가 발생한다.

* Maven 설정

https:://mvnrepository.com에 JUnit 검색(4: Junit, 5: JUnit (Jupiter, Vintage, Platform))

해당 dependencies 적용

* Library 설정

해당 프로젝트 property 진입

Java Build Path의 Library 탭에서 add library 클릭

Junit 4로 설정

* Junit 템플릿 설정(Ctrl + space)

Window -> preference -> Java -> Editor -> Template -> New

Name : 해당 이름을 적고 ctrl shift를 하면 자동 불러오기 됨

Description : 설명

Pattern : 입력되는 라이브러리들

Ex) ${junit1:importStatic('org.hamcrest.CoreMatchers.\*')}

${그냥id:importStatic 스태틱이면 스태틱 그냥이면 import}

-------------------------- DB Connection execute(), ExecuteQuery 등 ---------------------------------

* Execute // CREATE, DELETE 작업, INSERT 작업

: 수행결과로 Boolean 타입의 값을 반환한다.

모든 구문을 수행할 수 있다.

* ExecuteQuery // SELECT 작업

: 수행결과로 ResultSet 객체의 값을 반환한다.

SELECT 구문을 수행할 때 사용되는 함수이다.

* ExecuteUpdate // INSERT / DELETE / UPDATE에

반영된 레코드의 건수를 반환

: 수행결과로 Int 타입의 값을 반환한다. // CREATE / DROP -1 반환

SELECT 구문을 제외한 다른 구문을 수행할 때 사용되는 함수이다.

(INSERT, DELTE, UPDATE 관련 구문에서는 반영된 레코드의 건수를 반환합니다.