개발환경 01

환경변수 설정

1. 우선 개발 전 환경 변수를 설정해야 한다. 설치할 자바 버전을 오라클 사이트에서 내려 받고
2. 환경변수 창에 들어가 편집을 한다.

JAVA HOME과 PATH를 잡아준다.(없으면 생성해서 잡기)

set JAVA_HOME={설치위치}

set PATH=%PATH%;%JAVA_HOME%/bin

JRE와 JDK에 대해서

JRE는 프로그램을 실행하기 위한 최소한의 도구와 라이브러리를 포함하고 있다. JDK는 자바 개발에 필요한 모든 도구를 포함하고 있는데 JDK는 자바 개발에서 필수적인 요소로서 JRE가 JDK에 다 포함되어 있으니 환경 변수를 설정할 때 JDK로 잡는다.

** 자바의 버전에 대해서 pc에 설치할 자바를 선택하는 과정과 개발 시 사용할 버전을 달라도 된다. 즉, jdk 버전 무엇을 쓰든 환경 변수에서 조정을 하면 된다는 의미이다.

MAVEN이란 무엇인가

프로젝트 빌드 관리 및 디펜던시 관리를 자동화하는 빌드 도구 의존성 관리와 라이브러리 관리가 매우 쉬움

메이븐의 라이프 사이클

배포까지의 프로세스를 정의한다는 것. 즉 특정 프로젝트를 빌드하고 배포하는 것이 단계 별로 정의되어 있다는 것임.

claen	설치된 환경을 초기화한다	
package	컴파일 된 코드와 자원 파일들을 jar, war와 같은 배포 형식으로 패키징한다.	패키징에 따라 다름 jar - jar:jar war - war:war pom - site:attach-descriptor ejb - ejb:ejb
install	로컬 리포지토리에 생성된 패키지를 설치한다.	install:install
test	테스트를 실행한다.	surefire:test
mvn clean package -Dmaven.test.skip=true	초기화와 패키징을 동시에 진행하고 test를 스킵한다.	
`mvn -B archetype:generate -DgroupId=com.example -DartifactId=example -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.4`	프로젝트 생성문	mvn archetype:generate -DarchetypeArtifactId=maven-archetype-quickstart

Target 폴더에 대해서

타켓은 메이븐 프로젝트를 빌드하고 나면 생기는 폴더로 jar, war파일을 저장하기 위한 저장소이다. 즉, 빌드한 결과가 담기는 폴더로 프로젝트를 빌드하고 컴파일 된 자바 클래스 파일과 리소스 파일들이 저장되는 곳이라는 것이다.
** 배포할 때 타켓 폴더를 삭제하고 배포하는 이유는 jar, war파일은 실행을 하면 삭제하더라도 다시 생기기 때문이다.

빌드한 프로젝트의 메인 클래스를 실행하고 싶을 때는 타겟 폴더에 있는 옵션을 주고 그 클래스 파일을 실행시키면 된다. 이때 클래스패스를 잡아주어야 한다. 대상 클래스 파일이 있는 경로를 알아야지 실행시킬 수 있으니까.

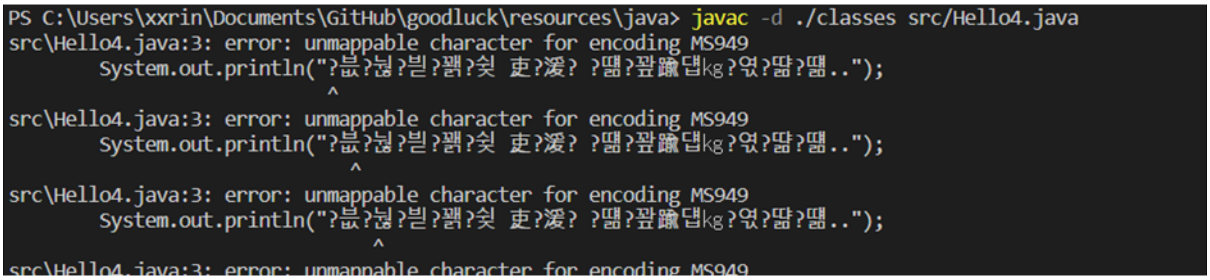
메이븐 골의 생명주기

메이븐의 실행 단위를 골이라고 한다. 골들은 상호 연관해 실행되는데 그러한 관계를 메이븐 골의 생명 주기라고 표현한다.

cmd

JAVA 버전 확인 : JAVA -VERSION

Cp(classpath의 약자)	zip jar 파일의 클래스 검색 경로	
D<name> = <value>	시스템 속성을 설정한다. ex) -DFile Encoding = UTF8	
jar <파일 이름>	jar 파일로 압축되어져 있는 자바 프로그램을 실행시킨다. 클래스 파일이름 대신 jar 파일을 사용해서 압축되어 있는 자바 프로그램을 실행시킬 수 있는데 제대로 실행 되어지기 위해서 jar 파일 안의 maniefast라는 텍스트 파일에 main-class : classname 같은 형태의 텍스트 라인이 포함되어 있어야 한다. 여기에 기술된 class name은 main 함수를 포함하고 있는 클래스 이름이 되어야 한다.	
Javac 파일명.class	클래스를 생성한다	
Java 파일명	코드를 실행한다.	
Javac -d ./classes ./src/파일명.java	클래스 파일의 생성 위치를 지정한다 클래스 파일이 생성될 위치를 클래스로 잡아주고 소스파일이 위치하는 곳을 명시해준 것이다. 그러니까 생성 위치와 소스파일 위치를 떼어줘야 함.	
d directotry	클래스 파일을 생성할 루트 디렉토리를 지정한다. 기본적으로 -d 옵션을 주지 않으면 소스파일이 위치한 디렉터리에 클래스 파일을 생성시킨다. 즉 내가 지정해주지 않으면 자신이랑 같은 곳에 위치하게 한다는 것이다	
Encoding	Encoding utf-8과 같이 한글이 전부 깨질 때 옵션을 주어 인코딩 방식을 설정한다. (아래 참고)	https://www.notion.so/01-555e96d43c1743af8c61e11132d390f7?pvs=4#df392b6d140d4237b4fd853bdc302e04https://www.notion.so/01-555e96d43c1743af8c61e11132d390f7?pvs=4#01a9fd2ac5ae40f2afd7069168d5fb9dhttps://www.notion.so/01-555e96d43c1743af8c61e11132d390f7?pvs=4#9a4ca79b89b04840bf0a71ce1bf47fe1
classpath<path>	컴파일러가 컴파일하기 위해 필요로 하는 참조할 클래스 파일들을 찾기 위해서 컴파일시 파일 경로를 지정해주는 옵션이다. `예를 들어, Hello.java파일이 C:\Java 디렉터리에 존재하고, 필요한 클래스 파일들이 C:\Java\Engclasses에 위치한다면, javac -classpath C:\Java\Engclasses C:\Java\Hello.java로 해주면 된다. 만약 참조할 클래스 파일들이 C:\Java\Engclasses외의 다른 디렉터리에도 존재한다면, C:\Java\Korclasses 일경우, javac -classpath C:\Java\Engclasses;C;\Java\Korclasses C:\Java\Hello.java`	
`javac -d ./classes ./src/hello4.java`	클래스 파일의 생성 위치를 지정한다	javac -d <클래스 파일이 생성될 위치> <소스 파일 위치>
`java -cp ./classes -Dmy.msg={메세지} com.example.main.PropertiesMain {실행 내용}`	실행	C:\Users\xxrin\Documents\GitHub\goodluck\resources\test01\example\targe java -cp ./classes com.example.main.PropertiesMain 1 2 3 4 5 타켓..안에 들 가서 클래스를 찾아야지 이상한데 가서 클래스 폴더 호출하고 실행시키면 안된다..



Ex) javac -encoding utf-8 -d classes .\src\Hello4.java

클래스 파일을 만들 때 한글일시 이렇게 오류가 난다. 클래스 파일 경로를 지정해주고 인코딩을
방식을 utf-8로 하겠다는 의미이다.

.;./classes Hello4

```
java -cp ./classes com.example.main.PropertiesMain -Dmy.msg=hello Main 1 2 3 4 5
```

vm이 읽어드린 시스템 환경을 system 클래스에 저장을 해놓음

system은 lang밑에 있어서 따로 import 하지 않아도 쓸 수 있음 - (즉 시스템 클래스라는 것임)

시스템 환경 변수들을 쭉 읽어 들어오는 명령어

mvn package -Dmaven.test.skip=true (테스트 스킵한다는 의미)

```
C:₩Users₩xxrin₩Documents₩GitHub₩goodluck₩resources₩xxrin₩example>
C:₩Users₩xxrin₩Documents₩GitHub₩goodluck₩resources₩xxrin₩example>mvn clean
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< com.example:example >-------------------------
[INFO] Building example 1.0
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:3.1.0:clean (default-clean) @ example ---
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.173 s
[INFO] Finished at: 2023-02-02T14:51:38+09:00
[INFO] ------------------------------------------------------------------------

C:₩Users₩xxrin₩Documents₩GitHub₩goodluck₩resources₩xxrin₩example>
```

한글 출력

자바 클래스 생성시 인코딩 utf8로 한 뒤에 src 경로 이동 후 출력

-encoginf utf-8

```
C:₩Users₩xxrin₩goodluck₩resources₩java₩src>CD ..

C:₩Users₩xxrin₩goodluck₩resources₩java>JAVAC -encoding utf-8 -d ./SRC ./src/Hello3.java

C:₩Users₩xxrin₩goodluck₩resources₩java>cd src

C:₩Users₩xxrin₩goodluck₩resources₩java₩src>dir
 C 드라이브의 볼륨: Windows
 볼륨 일련 번호: 9AB7-E901

 C:₩Users₩xxrin₩goodluck₩resources₩java₩src 디렉터리

2023-02-02  오후 05:10    <DIR>          .
2023-02-02  오후 05:10    <DIR>          ..
2023-02-02  오후 03:57    <DIR>          com
2023-02-02  오후 03:57               105 Hello1.java
2023-02-02  오후 03:57               103 Hello2.java
2023-02-02  오후 05:10               418 Hello3.class
2023-02-02  오후 05:06               124 Hello3.java
               4개 파일                 750 바이트
               3개 디렉터리  371,254,550,528 바이트 남음

C:₩Users₩xxrin₩goodluck₩resources₩java₩src>java Hello3
안냐세요
```

<Pom.xml의 구조>

Project
모든 메이븐 pom.xml 파일의 최상위 요소

Modelversion
폼이 사용중인 개체 모델의 버전 모델 버전을 명시하지 않으면 에러가 난다

Groupid
프로젝트의 주요 식별자 중 하나이며 메이븐에서 관리할 애플리케이션의 분류를 의미함

artifactId
메이븐에서 관리할 고유한 기본 이름. 프로젝트 명과 일치시키며 패키징 시 이 이름을 참조해서 만든다

Version
애플리케이션의 버전을 지정함. 스냅샷이 기본 버전인데 스냅샷이란 완성되지 않은 버전을 의미함. 반대로 release는 완성되어 공식적으로 배포되는 버전을 의미한다.

Packaging
패키징할 방법을 정의. 주로 jar war ear을 사용한다.
** 메이븐 모듈 프로젝트를 작성할 시 부모 pom의 packaging은 pom으로 지정한다. 메이븐은 기본적으로 jar을 사용하는데 부모 프로젝트에서 저렇게 지정해주면 하위 프로젝트에서 패키징 설정을 해주지 않아도 된다. 부모 pom에서 자식들을 호출하기 때문이다.

Dependencies
참조할 라이브러리 목록을 지정한다. 개발할 때 라이브러리 참조를 많이 해야하는데 pom파일을 이용해 라이브러리를 관리하니 메이븐의 핵심 부분이다.

Parent
현재 pom의 부모 pom 파일을 지정한다.

Modules
폼 파일을 조합해서 멀티 모듈 기반의 프로젝트를 작성한다.

> ** 여기서 필수적으로 사용되는 키는 groupId와 artifactid version이다. 이 세개의 태그는 메이븐 프로젝트에서 프로젝트 정보를 식별하고 패키징할때도 중요한 정보이니 반드시 정의해줘야 한다. 그래서 생성할 때도 이 세개의 값을 넣게 되어있다.

메이븐 모듈 프로젝트 생성

모듈 프로젝트 생성

1. Maven 기본 프로젝트 생성
2. 모듈 프로젝트 작성(이 과정에서 모듈이 자동으로 추가 되고 모듈 프로젝트에서는 부모와 dependencies를 명시해줘야 한다)

```
http://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation with catalog)
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
  <modelVersion>4.0.0</modelVersion>

  <groupId>example</groupId>
  <artifactId>kCal</artifactId>
  <version>0.0.1-SNAPSHOT</version>
 <packaging>pom</packaging>
 <description>kCal for main project</description>
  <name>kCal</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <modules>
    <module>kCalculator</module>
    <module>k-calculator</module>
  </modules>

</project>

Overview | Dependencies | Dependency Hierarchy | Effective POM | pom.xml
```

3. bat파일 만들기

```
set LIBS=C:\Users\xxrin\Documents\workspace-spring-tool-suite-4-4.17.1.RELEASE\kCal\k-calculator\target\korea-calculator-1.0-SNAPSHOT.jar
set LIBS=%LIBS%;"C:\Users\xxrin\Documents\workspace-spring-tool-suite-4-4.17.1.RELEASE\kCal\kCalculator\target\koreaCalculator-1.0-SNAPSHOT.
java -cp %LIBS% com.example.main.App "%1"
```

set 클래스패스를 지정해주는 이유는 생성한 jar파일을 클래스패스에 포함 시키기 위해서이다.

4. 실행하기
   a. 부모 프로젝트에서 mvn clean package intsall로 모든 프로젝트 jar파일 생성하고 로컬 설치
   b. lib 프로젝트에서 install
   c. bat 파일을 생성하고 실행

```
C:₩Users₩xxrin₩Documents₩workspace-spring-tool-suite-4-4.17.1.RELEASE₩kCal>cal.bat 오백+십삼

C:₩Users₩xxrin₩Documents₩workspace-spring-tool-suite-4-4.17.1.RELEASE₩kCal>set LIBS=C:₩Users₩xxrin₩Documents₩workspace-s
pring-tool-suite-4-4.17.1.RELEASE₩kCal₩k-calculator₩target₩korea-calculator-1.0-SNAPSHOT.jar

C:₩Users₩xxrin₩Documents₩workspace-spring-tool-suite-4-4.17.1.RELEASE₩kCal>set LIBS=C:₩Users₩xxrin₩Documents₩workspace-s
pring-tool-suite-4-4.17.1.RELEASE₩kCal₩k-calculator₩target₩korea-calculator-1.0-SNAPSHOT.jar;"C:₩Users₩xxrin₩Documents₩w
orkspace-spring-tool-suite-4-4.17.1.RELEASE₩kCal₩kCalculator₩target₩koreaCalculator-1.0-SNAPSHOT.jar"

C:₩Users₩xxrin₩Documents₩workspace-spring-tool-suite-4-4.17.1.RELEASE₩kCal>java -cp C:₩Users₩xxrin₩Documents₩workspace-s
pring-tool-suite-4-4.17.1.RELEASE₩kCal₩k-calculator₩target₩korea-calculator-1.0-SNAPSHOT.jar;"C:₩Users₩xxrin₩Documents₩w
orkspace-spring-tool-suite-4-4.17.1.RELEASE₩kCal₩kCalculator₩target₩koreaCalculator-1.0-SNAPSHOT.jar" com.example.main.A
pp "오백+십삼"


오
일
삼
오백+십삼 = 오백일십삼

C:₩Users₩xxrin₩Documents₩workspace-spring-tool-suite-4-4.17.1.RELEASE₩kCal>
```

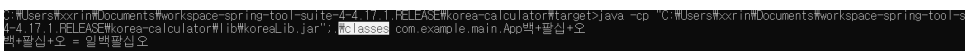
cal.bat {params}를 이용하여 main 클래스를 실행한다.

**배치 파일을 생성하는 이유

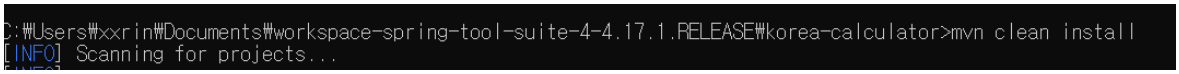
jar 파일을 생성하고 cmd에서 실행하려면 확장자와 클래스를 붙이고 떼는 등 번거로움이 있다. 이러한 불편함을 해결하기 위해 명령어를 한 번에 적어 놓고 실행할 수 있도록 bat 파일을 생성하고 실행한다.

bat 파일 안에서 파라미터를 %1로 받으면 파라미터 하나만 %*로 받으면 파라미터 여러개

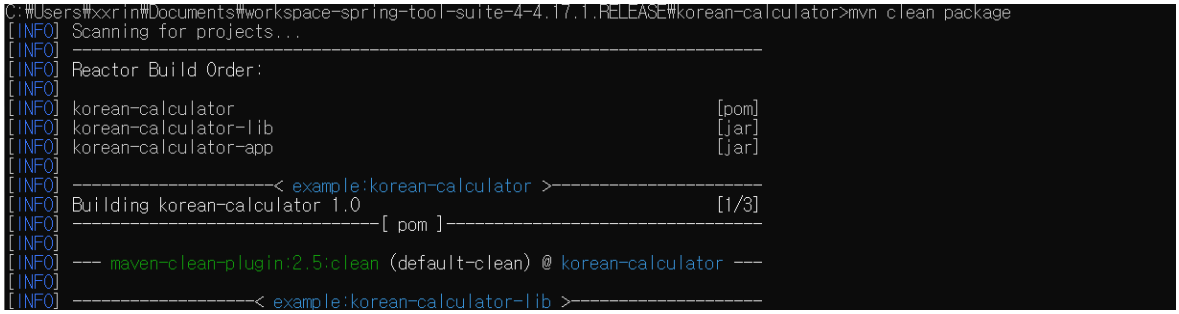
```
set LIBS=C:\Users\xxrin\Documents\workspace-spring-tool-suite-4-4.17.1.RELEASE\korean-calculator\lib\target\korean-calculator-lib-1.0.
set LIBS=%LIBS%;"C:\Users\xxrin\Documents\workspace-spring-tool-suite-4-4.17.1.RELEASE\korean-calculator\app\target\korean-calculator-
java -cp %LIBS% com.whoana.app.Main "%*"
```



\classes까지 폴더를 주면 classes가 어디있든 간에 찾아서 실행을 할 수 있음

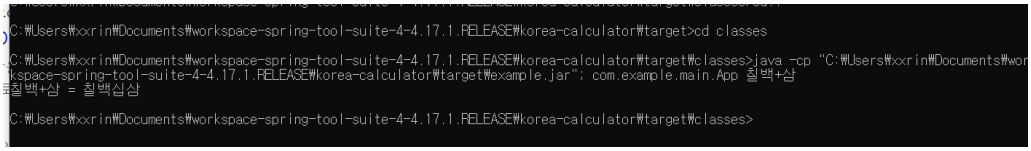


mvn clean install을 하게 되면 내 로컬에 라이브러리를 설치하여 쓸 수 있다.



외부 라이브러리 추가 방법

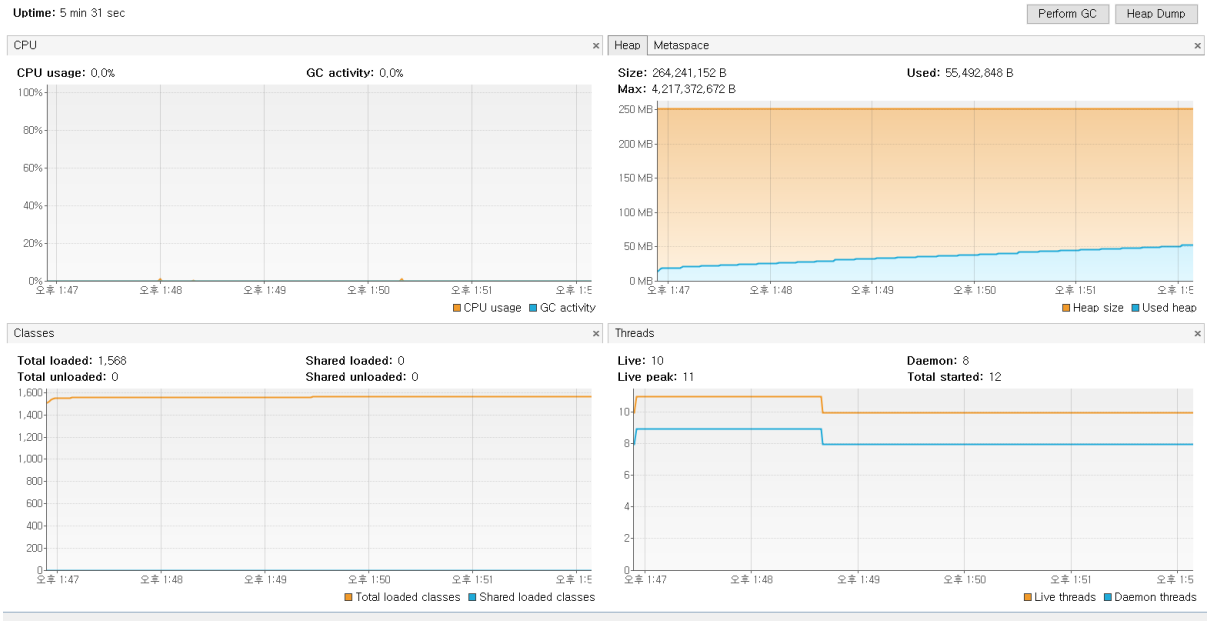
1. 메인 프로젝트 lib 폴더 생성
2. 해당 프로젝트에 생성한 라이브러리 추가
3. mvn install 라이브러리 설치
4. app.java나 메인으로 돌릴 java파일에 사용할 라이브러리 패키지 import



import한 자바 라이브러리를 실행할 때 path를 지정해주고 실행해야 함 그냥 지정하면 안된다.

jvsualvm 이란?

jvmarguments jvm에 내가 설정한 아규먼트와 system properties 기본 설정을 볼 수 있다



실행 시간

스레드로 작성한 코드가 thread에 보임

jconsole이란

실행 방법 - cmd에서 jconsole을 입력하면 실행할 수 있다.

jvm과 함께 제공되는 자바 가상 머신 모니터링 도구.

java 애플리케이션의 성능 모니터링 및 디버깅을 위해 사용된다.

프로젝트를 일부만 빌드하는 법

ex) 하나를 수정하고 배포해야 하는데 모든 파일을 함께 배포하기에는 힘드니 배포할 폴더만 지정해서 빌드를 하는 것이다.

```
mvn clean package -pl mint-common
```

mint common만 빌드하겠다는 의미이며 콤마를 사용하여 여러 프로젝트를 하나씩 빌드할 수 있다.