

HOL : Configure for Lab

1. JDK 13.x Installation and Configuration
2. Eclipse Download and Configuration
3. Maven 3.6.3 Installation and Configuration
4. Maven Project Creation in Eclipse
5. Tomcat 9.0.33 Installation & Configuration
6. STS 4.6.0.RELEASE Downloads & Configuration
7. Lombok 1.18.12 Library Installation

Task 1. JDK 13.x Installation and Configuration

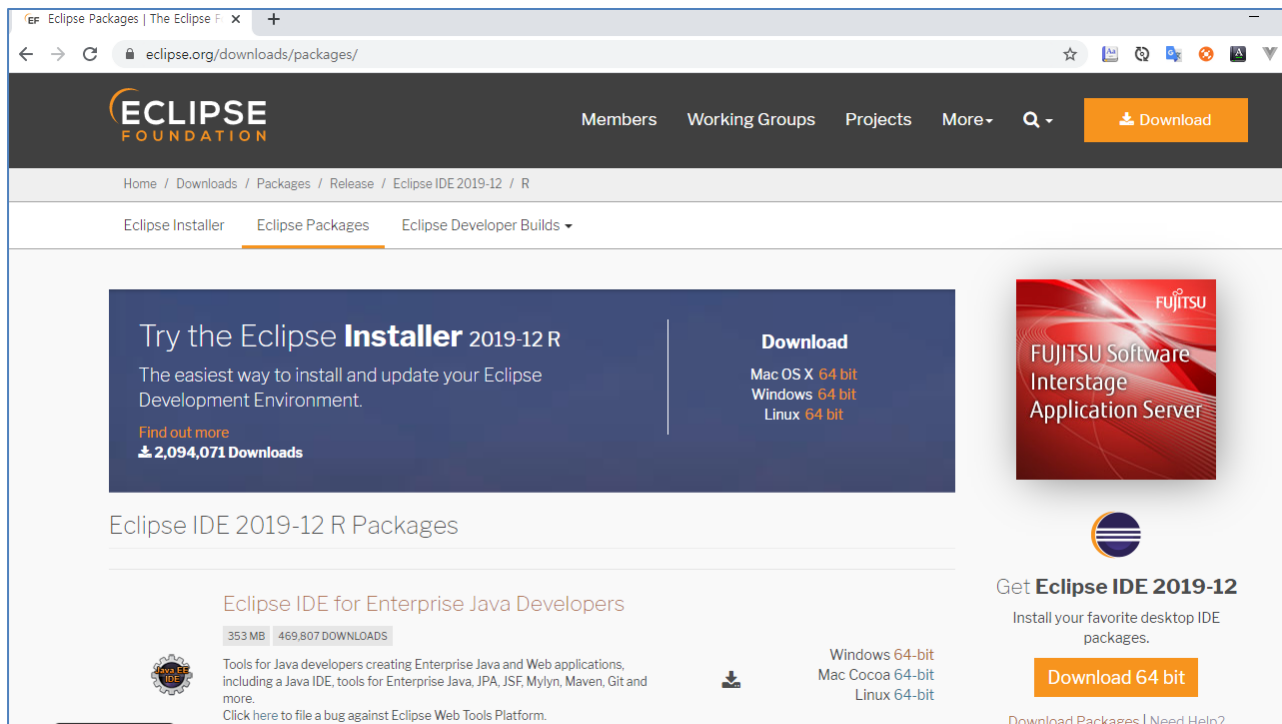
1. JDK : <https://www.oracle.com/java/technologies/javase-downloads.html>
2. Documentation : <https://docs.oracle.com/en/java/javase/13/>
3. API : <https://docs.oracle.com/en/java/javase/13/docs/api/index.html>

```
C:\Users\Instructor>set JAVA_HOME
JAVA_HOME=C:\Program Files\Java\jdk-13.0.2
```

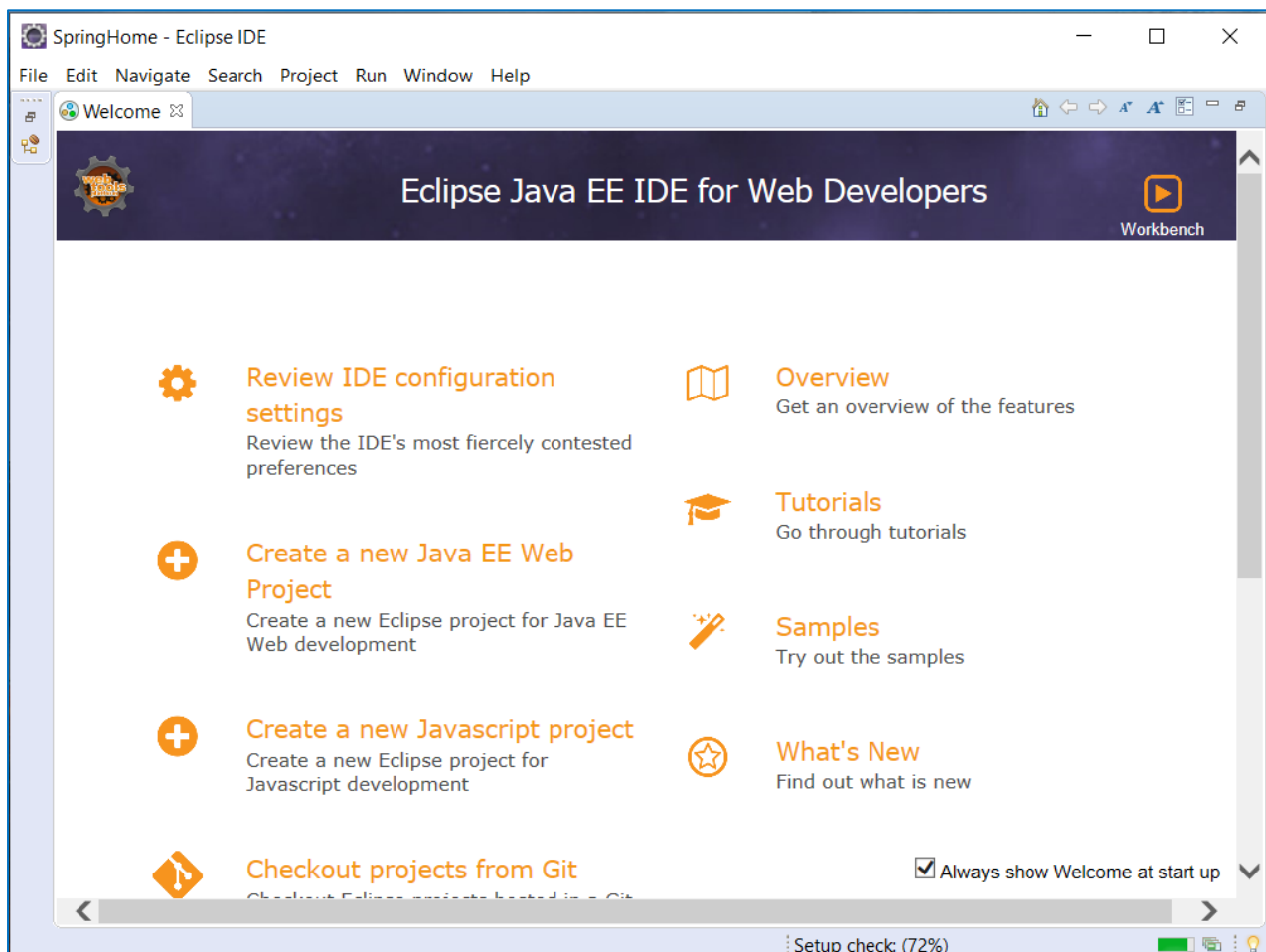
```
C:\Users\Instructor>set PATH
Path=C:\Program Files\Java\jdk-13.0.2\bin;C:\Program Files\apache-maven-3.6.3\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0;C:\Windows\System32\OpenSSH;C:\Users\Instructor\AppData\Local\Microsoft\WindowsApps;C:\Program Files\Bandizip;
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
```

Task 2. Eclipse Download and Configuration

1. <https://www.eclipse.org/downloads/packages/>



2. Eclipse IDE for Enterprise Java Developers



Task 3. Maven 3.6.3 Configuration

1. <http://maven.apache.org/>
2. Refer to : <http://javacan.tistory.com/entry/MavenBasic>
3. Download
 - 1) <http://maven.apache.org/download.cgi>
 - 2) Binary zip archive : apache-maven-3.6.3-bin.zip
 - 3) 환경변수 M2_HOME(C: \ Program Files \ apache-maven-3.6.3) 설정
 - 4) 환경변수 PATH 에 (%M2_HOME% \ bin) 추가

```
C:\Users\Instructor>set M2_HOME
M2_HOME=C:\Program Files\apache-maven-3.6.3

C:\Users\Instructor>set PATH
Path=C:\Program Files\Java\jdk-13.0.2\bin;C:\Program Files\apache-maven-3.6.3\bin;C:\Windows\sy
stem32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windo
ws\System32\OpenSSH\;C:\Users\Instructor\AppData\Local\Microsoft\WindowsApps;C:\Program Files\B
andizip\
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
```

4. 설치 확인

\$ mvn -version

```
C:\Users\Instructor>mvn -version
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: C:\Program Files\apache-maven-3.6.3\bin\..
Java version: 13.0.2, vendor: Oracle Corporation, runtime: C:\Program File
s\Java\jdk-13.0.2
Default locale: ko_KR, platform encoding: MS949
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

5. Maven project 생성하기

- 설치가 끝났다면 Maven project 를 생성해 보자.
- Command prompt 에서 아래 명령어를 실행하면 된다.
- 아래 명령어를 처음 실행할 경우 꽤 오랜 시간이 걸리는데, 그 이유는 Maven 이 필요한 plug-in 과 module 을 download 받기 때문이다.
- Maven 배포판은 최초로 Maven 을 사용하는 데 필요한 module 만 포함하고 있고, 그 외에 archetype plug-in, compiler plug-in 등 Maven 을 사용하는 데 필요한 module 은 포함하고 있지 않다.
- 이들 module 은 실제로 필요할 때 Maven 중앙 Repository 에서 loading 된다.
- 먼저 C:\SpringHome folder 를 생성한 후, folder 로 이동한다.

\$ mvn archetype:generate

```
C:\Windows\system32\cmd.exe

C:\SpringHome>mvn archetype:generate
```

```
C:\Windows\system32\cmd.exe - mvn archetype:generate
type which contains a sample Argo (UIMA) Analysis Engine)
2581: remote -> uk.ac.nactem.argo:argo-reader-archetype (An archetype which contains a sample Argo (UIMA) Reader)
2582: remote -> uk.ac.rdg.resc.edal-ncwms-based-webapp (-)
2583: remote -> uk.co.nemstix:basic-javaee7-archetype (A basic Java EE7 Maven archetype)
2584: remote -> uk.co.solong:angular-spring-archetype (So Long archetype for RESTful spring services with an AngularJS frontend. Includes debian deployment)
2585: remote -> us.fatehi:schemacrawler-archetype-maven-project (-)
2586: remote -> us.fatehi:schemacrawler-archetype-plugin-command (-)
2587: remote -> us.fatehi:schemacrawler-archetype-plugin-dbconnector (-)
2588: remote -> us.fatehi:schemacrawler-archetype-plugin-lint (-)
2589: remote -> ws.osiris:osiris-archetype (Maven Archetype for Osiris)
2590: remote -> xyz.luan.generator:xyz-gae-generator (-)
2591: remote -> xyz.luan.generator:xyz-generator (-)
2592: remote -> za.co.absa.hyperdrive:component-archetype (-)
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): 1498:
```

...

Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): 1498(번호는 다를 수 있다): **엔터**

```
C:\Windows\system32\cmd.exe - mvn archetype:generate
2586: remote -> us.fatehi:schemacrawler-archetype-plugin-command (-)
2587: remote -> us.fatehi:schemacrawler-archetype-plugin-dbconnector (-)
2588: remote -> us.fatehi:schemacrawler-archetype-plugin-lint (-)
2589: remote -> ws.osiris:osiris-archetype (Maven Archetype for Osiris)
2590: remote -> xyz.luan.generator:xyz-gae-generator (-)
2591: remote -> xyz.luan.generator:xyz-generator (-)
2592: remote -> za.co.absa.hyperdrive:component-archetype (-)
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): 1498:
Choose org.apache.maven.archetypes:maven-archetype-quickstart version:
1: 1.0-alpha-1
2: 1.0-alpha-2
3: 1.0-alpha-3
4: 1.0-alpha-4
5: 1.0
6: 1.1
7: 1.3
8: 1.4
Choose a number: 8: 엔터
```

Choose org.apache.maven.archetypes:maven-archetype-quickstart version:

- 1: 1.0-alpha-1
- 2: 1.0-alpha-2
- 3: 1.0-alpha-3
- 4: 1.0-alpha-4
- 5: 1.0
- 6: 1.1
- 7: 1.3
- 8: 1.4

Choose a number: 8: **엔터**

...

...

```
C:\Windows\system32\cmd.exe - mvn archetype:generate
1: 1.0-alpha-1
2: 1.0-alpha-2
3: 1.0-alpha-3
4: 1.0-alpha-4
5: 1.0
6: 1.1
7: 1.3
8: 1.4
Choose a number: 8: 8
Define value for property 'groupId': com.example
Define value for property 'artifactId': demo
Define value for property 'version' 1.0-SNAPSHOT: :
Define value for property 'package' com.example: :
Confirm properties configuration:
groupId: com.example
artifactId: demo
version: 1.0-SNAPSHOT
package: com.example
Y: :
```

Define value for property 'groupId': **com.example**

Define value for property 'artifactId': **demo**

Define value for property 'version' 1.0-SNAPSHOT: : **엔터**

Define value for property 'package' com.example: : **엔터**

Confirm properties configuration:

groupId: com.example

artifactId: demo

version: 1.0-SNAPSHOT

package: com.example

Y: : **엔터**

```
C:\Windows\system32\cmd.exe
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: com.example
[INFO] Parameter: packageInPathFormat, Value: com/example
[INFO] Parameter: package, Value: com.example
[INFO] Parameter: groupId, Value: com.example
[INFO] Parameter: artifactId, Value: demo
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Project created from Archetype in dir: C:\SpringHome\demo
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 04:34 min
[INFO] Finished at: 2020-03-16T17:46:19+09:00
[INFO] -----
C:\SpringHome>
```

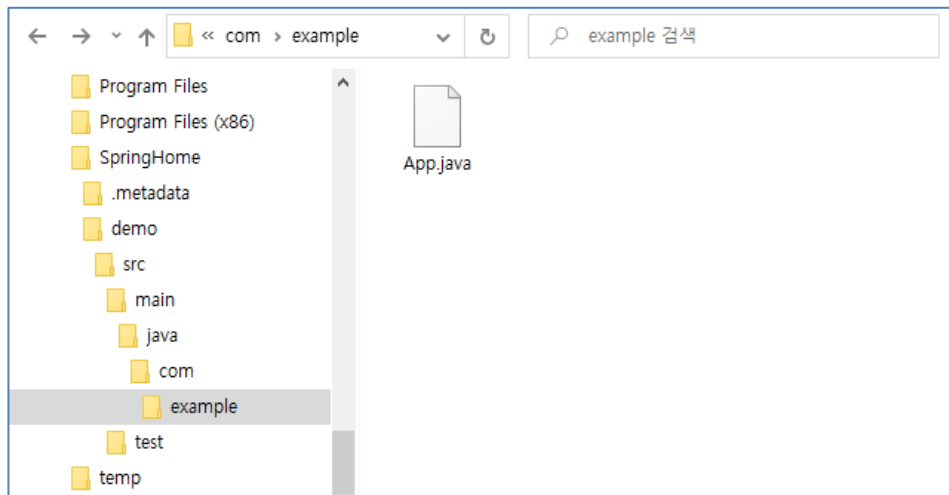
- 위 과정에서 실제로 입력하는 값은 다음과 같다.

- ◆ groupId

- Project 속하는 group 식별 값. 회사, 본부, 또는 단체를 의미하는 값이 오며, package 형식으로 계층을 표현한다.

- 위에서는 com.javasoft 를 groupId 로 이용하였다.
- ◆ artifactId
 - Project 결과물의 식별 값.
 - project 나 module 을 의미하는 값이 온다.
 - 위에서는 demo 을 artifactId 로 이용하였다.
- ◆ version
 - 결과물의 version 을 입력한다.
 - 위에서는 기본 값인 1.0-SNAPSHOT 을 사용하였다.
- ◆ package
 - 기본적으로 생성할 package 를 입력한다.
 - 별도로 입력하지 않을 경우 groupId 와 동일한 구조의 package 를 생성한다.

6. Maven project 의 기본 directory 구조



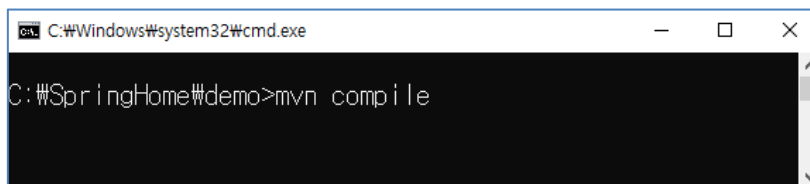
- archetype:generate 이 성공적으로 실행되면, artifactId 에 입력한 값과 동일한 이름의 directory 가 생성된다.
- archetype:generate 명령어는 미리 정의된 template 을 이용해서 Maven project 를 생성하는 기능으로 지금 사용한 Maven Project Template 는 archetype-quickstart 라는 template 이다.
- Template 을 사용하지 않고 직접 directory 를 생성하고 pom.xml file 을 작성해 주어도 동일한 Maven project 가 생성된다.
- 위 경우에는 현재 directory 에 demo 이라는 하위 directory 가 생성된다.
- 위 과정에서 선택한 archetype 은 maven-archetype-quickstart 인데, 이 archetype 을 선택했을 때 생성되는 directory 구조는 다음과 같다.
 - src/main/java
 - ◆ Java source file 이 위치한다.
 - src/main/resources
 - ◆ Property 나 XML 등 resource file 이 위치한다.
 - ◆ classpath 에 포함된다.(생성예정)
 - src/main/webapp
 - ◆ Web application 관련 file 이 위치한다(WEB-INF directory, JSP file 등, 생성예정).

- `src/test/java`
 - ◆ Test Java source file 이 위치한다.
- `src/test/resources`
 - ◆ Test 과정에서 사용되는 resource file 이 위치한다.
 - ◆ Test 시에 사용되는 classpath 에 포함된다.
- 기본적으로 생성되지 않은 directory 라 하더라도 직접 생성해주면 된다.
- 예를 들어, `src/main` directory 에 `resources` directory 를 생성해주면 Maven 은 resource directory 로 인식한다.

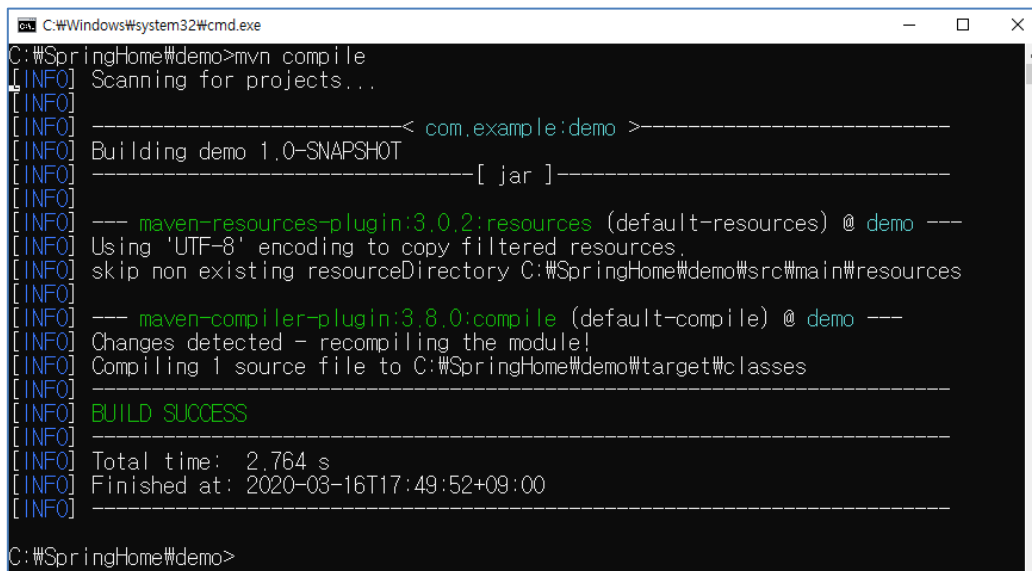
7. Compile 해보기/Test 실행 해보기/Package 해보기

- 이제 간단하게 compile 과 test 를 실행해보자.
- Source code 를 compile 하려면 다음과 같은 명령어를 실행해주면 된다.
- [demo] folder 로 이동 후

\$ mvn compile



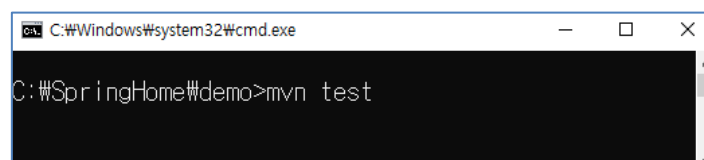
```
C:\SpringHome\demo>mvn compile
```



```
C:\SpringHome\demo>mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.example:demo >-----
[INFO] Building demo 1.0-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @ demo ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\SpringHome\demo\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ demo ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to C:\SpringHome\demo\target\classes
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 2.764 s
[INFO] Finished at: 2020-03-16T17:49:52+09:00
[INFO]
C:\SpringHome\demo>
```

- Compile 된 결과는 `target/classes` directory 에 생성된다.
- Test class 를 실행해보고 싶다면, 다음과 같은 명령어를 사용하면 된다.

\$ mvn test



```
C:\SpringHome\demo>mvn test
```



```
C:\Windows\system32\cmd.exe
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running com.example.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.069 s -
in com.example.AppTest
[INFO] Results:
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 4.785 s
[INFO] Finished at: 2020-03-16T17:52:15+09:00
[INFO] -----
C:\SpringHome\demo>
```

- 그러면, test code 를 compile 한 뒤 test code 를 실행한다.
- 그리고 test 성공 실패 여부를 화면에 출력한다.
- Compile 된 test class 들은 target/test-classes directory 에 생성되고, test 결과 report 는 target/surefire-reports directory 에 저장된다.
- (아무것도 한 것이 없으니 당연하지만) 모든 code 가 정상적으로 만들어지고 test 도 통과했으니, 이제 배포 가능한 jar file 을 만들어보자.
- 아래 명령어를 실행하면 project 를 packaging 해서 결과물을 생성한다.

\$ mvn package

```
C:\Windows\system32\cmd.exe
C:\SpringHome\demo>mvn package
```

```
C:\Windows\system32\cmd.exe
[INFO] Results:
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --- maven-jar-plugin:3.0.2:jar (default-jar) @ demo ---
[INFO] Building jar: C:\SpringHome\demo\target\demo-1.0-SNAPSHOT.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 3.789 s
[INFO] Finished at: 2020-03-16T17:53:25+09:00
[INFO]
C:\SpringHome\demo>
```

- mvn package 가 성공적으로 실행되면, target directory 에 project 이름과 version 에 따라 알맞은 이름을 갖는 jar file 이 생성된다.
- 위 예제의 경우에는 demo-1.0-SNAPSHOT.jar file 이 생성된 것을 확인할 수 있다.

8. POM 파일 기본

- Maven project 를 생성하면 pom.xml file 이 project root directory 에 생성된다.
- 이 pom.xml file 은 Project Object Model 정보를 담고 있는 file 로서, 이 file 에서 다루는 주요 설정 정보는 다음과 같다.
 - ◆ Project 정보
 - Project 의 이름, 개발자 목록, license 등의 정보를 기술
 - ◆ Build 설정
 - Source, resource, lifecycle 별 실행할 plug-in 등 build 와 관련된 설정을 기술
 - ◆ Build 환경
 - 사용자 환경 별로 달라질 수 있는 profile 정보를 기술
 - ◆ POM 연관 정보
 - 의존 project(module), 상위 project, 포함하고 있는 하위 module 등을 기술
 - ◆ archetype:create goal 실행시 maven-archetype-quickstart Archetype 을 선택한 경우 생성되는 pom.xml 파일은 다음과 같다.

[기본으로 생성되는 pom.xml file]

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>demo</artifactId>
  <version>1.0-SNAPSHOT</version>

  <name>demo</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>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
```

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<pluginManagement>**<!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->**

<plugins>

<plugin>

<artifactId>maven-clean-plugin</artifactId>

<version>3.1.0</version>

</plugin>

<!-- see http://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin_bindings_for_jar_packaging -->

<plugin>

<artifactId>maven-resources-plugin</artifactId>

<version>3.0.2</version>

</plugin>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.0</version>

</plugin>

<plugin>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.1</version>

</plugin>

<plugin>

<artifactId>maven-jar-plugin</artifactId>

<version>3.0.2</version>

</plugin>

<plugin>

<artifactId>maven-install-plugin</artifactId>

<version>2.5.2</version>

```

</plugin>
<plugin>
  <artifactId>maven-deploy-plugin</artifactId>
  <version>2.8.2</version>
</plugin>
<plugin>
  <artifactId>maven-site-plugin</artifactId>
  <version>3.7.1</version>
</plugin>
<plugin>
  <artifactId>maven-project-info-reports-plugin</artifactId>
  <version>3.0.0</version>
</plugin>
</plugins>
</pluginManagement>
</build>
</project>

```

- 위 POM 파일에서 프로젝트 정보를 기술하는 태그는 다음과 같다.
 - ◆ <name>
 - Project 이름
 - ◆ <url>
 - Project Site URL
- POM 연관 정보는 Project 간 연관 정보를 기술하는데, 관련 태그는 다음과 같다.
 - ◆ <groupId>
 - Project 의 group ID 설정
 - ◆ <artifactId>
 - Project 의 Artifact ID 설정
 - ◆ <version>
 - version 설정
 - ◆ <packaging>
 - Packaging type 설정.
 - 위 code 의 경우 project 의 결과 Artifact 가 jar file 로 생성됨을 의미한다.

- jar 뿐만 아니라 Web application 을 위한 war 나 JEE 를 위한 ear 등의 packaging type 이 존재한다.
- ◆ <dependencies>
 - 이 Project 에서 의존하는 다른 Project 정보를 기술한다.
- ◆ <dependency>
 - 의존하는 Project POM 정보를 기술
- ◆ <groupId>
 - 의존하는 Project 의 그룹 ID
- ◆ <artifactId>
 - 의존하는 Project 의 artifact ID
- ◆ <version>
 - 의존하는 Project 의 버전
- ◆ <scope>
 - 의존하는 범위를 설정

9. 의존설정

- <dependency> 부분의 설정에 대해서 좀 더 살펴보도록 하자.
- Maven 을 사용하지 않을 경우 개발자들은 code 에서 필요로 하는 library 를 각각 download 받아야 한다.
- 예를 들어, Apache commons DBCP library 를 사용하기 위해서는 DBCP 뿐만 아니라 common pool library 도 download 받아야 한다.
- 물론, commons logging 을 비롯한 library 도 모두 추가로 download 받아 설치해 주어야 한다.
- 즉, code 에서 필요로 하는 library 뿐만 아니라 그 library 가 필요로 하는 또 다른 library 도 직접 찾아서 설치해 주어야 한다.
- 하지만, Maven 을 사용할 경우에는 code 에서 직접적으로 사용하는 module 에 대한 의존만 추가해주면 된다.
- 예를 들어, commons-dbcp module 을 사용하고 싶은 경우 다음과 같은 <dependency> 코드만 추가해주면 된다.

```
<dependency>
  <groupId>commons-dbcp</groupId>
  <artifactId>commons-dbcp</artifactId>
  <version>1.2.1</version>
</dependency>
```

- 그러면, Maven 은 commons-dbcp 뿐만 아니라 commons-dbcp 가 의존하는 library 를 자동으로 처리해준다.
- Maven 은 commons-dbcp module 을 download 받을 때 관련 POM 파일도 함께 download 받는다.
- 실제로 1.2.1 version 의 commons-dbcp module 의 pom.xml file 을 보면 의존 부분이 다음과 같이 설정되어 확인할 수 있다.

```

<dependencies>
  <dependency>
    <groupId>commons-collections</groupId>
    <artifactId>commons-collections</artifactId>
    <version>2.1</version>
  </dependency>
  <dependency>
    <groupId>commons-pool</groupId>
    <artifactId>commons-pool</artifactId>
    <version>1.2</version>
  </dependency>
  <dependency>
    <groupId>javax.sql</groupId>
    <artifactId>jdbc-stdext</artifactId>
    <version>2.0</version>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>xml-apis</groupId>
    <artifactId>xml-apis</artifactId>
    <version>2.0.2</version>
  </dependency>
  <dependency>
    <groupId>xerces</groupId>
    <artifactId>xerces</artifactId>
    <version>2.0.2</version>
  </dependency>
</dependencies>

```

- Maven 은 commons-dbcx module 을 download 받을 때 관련 POM file 도 함께 download 받는다.
- 그리고 POM file 에 명시한 의존 module 을 함께 다운로드 받는다.

- 즉, commons-dbcp 1.2.1 version 의 경우 commons-collections 2.1 version 과 commons-pool 1.2 version 등을 함께 download 받는다.
- 이런 식으로 반복해서 download 받은 module 이 필요로 하는 module 을 download 받고 이들 module 을 현재 project 에서 사용할 classpath 에 추가해준다.
- 따라서, 개발자는 일일이 필요한 module 을 download 받을 필요가 없으며, 현재 code 에서 직접적으로 필요로 하는 module 에 대해서만 <dependency>로 추가해주면 된다.
- 나머지 의존은 모두 Maven 이 알맞게 처리해준다.

10. 의존의 scope: compile, runtime, provided, test

- 앞의 pom.xml file 에서 <dependency> 부분을 보면 <scope>를 포함하고 있는 것과 그렇지 않은 것이 존재한다는 것을 알 수 있다.
- <scope>는 의존하는 module 이 언제 사용되는 지를 설정할 때 사용되며, <scope>에 올 수 있는 값은 다음의 네 가지가 존재한다.
 - ◆ compile
 - Compile 할 때 필요.
 - Test 및 Runtime 에도 classpath 에 포함된다.
 - <scope>를 설정하지 않을 경우 기본 값은 compile 이다.
 - ◆ runtime
 - Runtime 에 필요.
 - JDBC driver 등이 예가 된다.
 - Project 의 code 를 compile 할 때는 필요하지 않지만, 실행할 때 필요하다는 것을 의미한다.
 - 배포시 포함된다.
 - ◆ provided
 - Compile 할 때 필요하지만, 실제 runtime 때에는 container 같은 것에서 기본으로 제공되는 module 임을 의미한다.
 - 예를 들어, Servlet 이나 JSP API 등이 이에 해당한다.
 - 배포시 제외된다.
 - ◆ test
 - Test code 를 compile 할 때 필요.
 - Mock test 를 위한 module 예이다.
 - Test 시에 classpath 에 포함되며, 배포시 제외된다.

11. 원격 repository 와 local repository

- Maven 은 compile 이나 packaging 등 작업을 실행할 때 필요한 plug-in 이나 pom.xml file 의 <dependency> 등에 설정한 module 을 Maven 중앙 repository 에서 download 받는다.
- 현재 중앙 repository 의 주소는 <http://mvnrepository.com/> 이다.
- 원격 repository 에서 download 받은 module 은 local repository 에 저장된다.

- local repository 는 %USER_HOME%/m2/repository directory 에 생성되며, local repository 에는 다음과 같은 형식의 directory 를 생성한 뒤 download 받은 module 을 저장한다.

[groupId]/[artifactId]/[version]

- 예를 들어, commons-dbcp 1.2.1 version 의 경우, module 및 관련 POM file 이 저장되는 directory 는 다음과 같다.

[USER_HOME]/m2/repository/commons-dbcp/commons-dbcp/1.2.1

- 위 directory 에 저장되는 file 은 packaging 된 module file, pom file 그리고 source code, download option 을 실행한 경우에는 source code 를 포함한 jar file 이 포함된다.
- 일단 원격 repository 로부터 file 을 download 해서 local repository 에 저장하면, 그 뒤로는 local repository 에 저장된 file 을 사용하게 된다.

12. Maven Lifecycle 과 plug-in 실행

- 본 글의 서두에 Maven 은 project 의 lifecycle 기반 framework 를 제공한다고 했다.
- 앞서 project 를 생성한 뒤 compile 하고(mvn compile), test 하고(mvn test), packaging 하는(mvn package) 과정을 정해진 명령어를 이용해서 실행했는데, 이때 compile, test, package 는 모두 build lifecycle 에 속하는 단계이다.
- Maven 은 clean, build (default), site 의 세 가지 lifecycle 을 제공하고 있다.
- 각 lifecycle 은 순서를 갖는 단계(phase)로 구성된다.
- 또한, 각 단계별로 기본적으로 실행되는 plug-in goal 이 정의되어 있어서 각 단계마다 알맞은 작업이 실행된다.
- 아래 표는 default lifecycle 을 구성하고 있는 주요 실행 단계를 순서대로 정리한 것이다.

[표] default lifecycle 의 주요 단계(phase)

단계	설명	단계에 묶인 plug-in 실행
generate-sources	Compile 과정에 포함될 source 를 생성한다. 예를 들어, DB table 과 mapping 되는 Java code 를 생성해주는 작업이 이 단계에서 실행된다.	
process-sources	Filter 와 같은 작업을 source code 에 처리한다.	
generate-resources	Package 에 포함될 자원을 생성한다.	
process-resources	Filter 와 같은 작업을 자원 file 에 처리하고, 자원 file 을 class 출력 directory 에 복사한다.	resources:resources
compile	Source code 를 compile 해서 class 출력 directory 에 class 를 생성한다.	compiler:compile
generate-test-sources	Test source code 를 생성한다. 예를 들어, 특정	

	class 에서 자동으로 test case 를 만드는 작업이 이 단계에서 실행된다.	
process-test-sources	Filter 와 같은 작업을 test source code 에 처리한다.	resources:testResources
generate-test-resources	test 를 위한 자원 file 을 생성한다.	
process-test-resources	Filter 와 같은 작업을 test 자원 file 에 처리하고, test 자원 file 을 test class 출력 directory 에 복사한다.	
test-compile	Test source code 를 compile 해서 test class 출력 directory 에 class 를 생성한다.	compiler:testCompile
test	test 를 실행한다.	surefire:test
package	Compile 된 code 와 자원 file 들을 jar, war 와 같은 배포 형식으로 packaging 한다.	packaging 에 따라 다름, jar - jar:jar war - war:war pom - site:attach-descriptor ejb - ejb:ejb
install	Local repository 에 package 를 복사한다.	install:install
deploy	생성된 package file 을 원격 repository 에 등록하여, 다른 project 에서 사용할 수 있도록 한다.	deploy:deploy

- Lifecycle 의 특정 단계를 실행하려면 다음과 같이 mvn [단계이름] 명령어를 실행하면 된다.
\$ mvn test
\$ mvn deploy
- Lifecycle 의 특정 단계를 실행하면 그 단계의 앞에 위치한 모든 단계가 실행된다.
- 예를 들어, test 단계를 실행하면 test 단계를 실행하기에 앞서 'generate-sources' 단계부터 'test-compile' 단계까지 각 단계를 순서대로 실행한다.
- 각 단계가 실행될 때는 각 단계에 묶인 goal 이 실행된다.
- plug-in 을 직접 실행할 수도 있다.
- mvn 명령어에 단계 대신 실행할 plug-in 을 지정하면 된다.
\$ mvn surefire:test
- 단, plug-in goal 을 직접 명시한 경우에는 해당 plug-in 만 실행되기 때문에 lifecycle 의 단계가 실행되지는 않는다.

13. Plug-in Goal

- Maven 에서 plug-in 을 실행할 때에는 'plug-in 이름:plug-in 지원 goal'의 형식으로 실행할 기능을 선택한다.
- 예를 들어, compiler:compile 은 'compiler'는 plug-in 에서 'compile' 기능(goal)을 실행한다는 것을 뜻한다.

14. 맺음말

- 이번 글에서는 Maven 의 기본 사용법을 살펴봤다.
- Maven 이 제공하는 의존 관리는 개발자를 jar 지옥(?)에서 구해준다는 것을 알 수 있었다.
- 또한, Maven 은 표준화된 lifecycle 을 제공하고 있기 때문에 개발자가 Compile-Test-Packaging 등의 과정을 손으로 정의하지 않아도 되며, 개발자는 Maven 이 제공하는 단계 중 필요한 단계만 실행하면 된다.
- 그럼, 나머지 작업(Compile, Test 실행, jar file 생성)은 모두 Maven 이 처리해준다.

15. 관련자료

- Maven 홈 페이지: <http://maven.apache.org>
- Maven: The Definitive Guide (Sonatype, Oreilly)
- Maven compiler version 설정

<출처: <http://javacan.tistory.com/entry/MavenBasic> [자바캔(Java Can Do IT)]>

16. 수동으로 maven project 생성 후 해야 할 작업

- src/main directory 에 resources directory 를 직접 수동으로 추가
- Maven 의 src/main/resources directory 는 classpath 로 사용되는 directory 로서, 이 directory 에는 XML 이나 properties file 과 같은 자원 파일 중에서 classpath 에 위치해야 하는 file 들을 넣는다.
- 그리고 src/test/java directory 에 생성된 AppTest.java 와 src/main/java directory 에 생성된 App.java file 을 필요하지 않으므로 삭제한다.
- Maven project 가 Java code 를 compile 할 때 UTF-8 charset 과 Java 1.8 버전을 사용하도록 설정이 필요하다.
- 이를 위해 pom.xml <dependencies>하위에 아래 코드를 추가한다.

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13</version>
    <scope>test</scope>
</dependency>
```

- <build>하위 <plugins> 하위에도 아래의 코드를 추가한다.

```
<build>
    <pluginManagement>
```

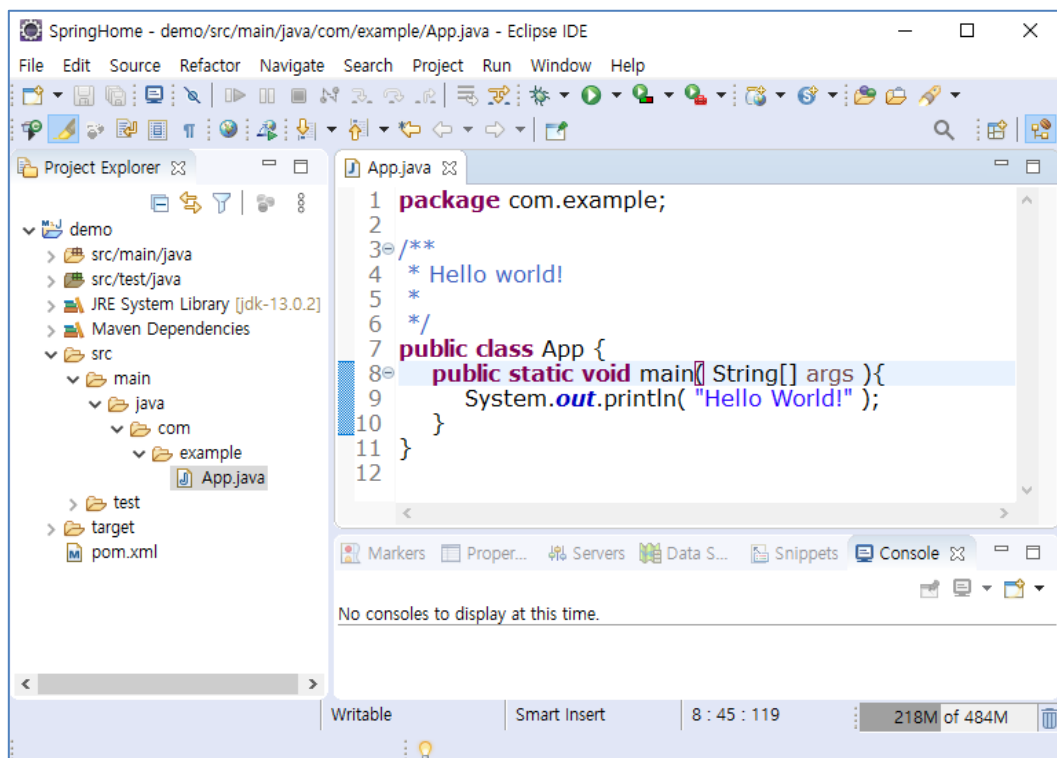
```

<plugins>
  <plugin>
    <artifactId>maven-compiler-plugin</artifactId>
    <configuration>
      <source>1.13</source>
      <target>1.13</target>
      <encoding>${project.build.sourceEncoding}</encoding>
    </configuration>
  </plugin>
</plugins>
</plugManagement>
</build>

```

17. Eclipse 에서 Maven Project Import 하기

- File > Import > Maven > Existing Maven Projects > Next > Browse
- [Root Directory] : C:\SpringHome\demo > OK
- 그러면 자동으로 pom.xml 파일을 기준으로 project 를 찾는다. > Finish



18. pom.xml 의존관계(dependency) 추가

- <https://mvnrepository.com/artifact/org.springframework.spring-context> 에서 Spring Context latest version 선택


- Maven 에 Spring Context 추가하기

- In pom.xml

```

<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13</version>
    <scope>test</scope>
  </dependency>
  <!--아래 코드 추가-->
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.4.RELEASE</version>
  </dependency>
</dependencies>

```



```

14 <url>http://www.example.com</url>
15
16 <properties>
17   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
18   <maven.compiler.source>1.13</maven.compiler.source>
19   <maven.compiler.target>1.13</maven.compiler.target>
20 </properties>
21
22 <dependencies>
23   <dependency>
24     <groupId>junit</groupId>
25     <artifactId>junit</artifactId>
26     <version>4.13</version>
27     <scope>test</scope>
28   </dependency>
29   <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
30   <dependency>
31     <groupId>org.springframework</groupId>
32     <artifactId>spring-context</artifactId>
33     <version>5.2.4.RELEASE</version>
34   </dependency>
35

```

- pom.xml > right-click > Run As > Maven install
- Eclipse 가 spring-context module 을 포함해서 필요한 jar file 들을 download 받기 시작한다.
- 필요한 jar file 들을 모두 download 받으면, Eclipse project 에 download 받은 jar file 들이 Maven 의존 목록(Maven Dependencies)에 표시된다.

Dependency Hierarchy [test] Filter:

Dependency Hierarchy

- ▼ junit : 4.13 [test]
 - hamcrest-core : 1.3 [test]
- ▼ spring-context : 5.2.4.RELEASE [compile]
 - ▼ spring-aop : 5.2.4.RELEASE [compile]
 - spring-beans : 5.2.4.RELEASE [compile]
 - spring-core : 5.2.4.RELEASE [compile]
 - ▼ spring-beans : 5.2.4.RELEASE [compile]
 - spring-core : 5.2.4.RELEASE [compile]
 - ▼ spring-core : 5.2.4.RELEASE [compile]
 - spring-jcl : 5.2.4.RELEASE [compile]
 - ▼ spring-expression : 5.2.4.RELEASE [compile]
 - spring-core : 5.2.4.RELEASE [compile]

Resolved Dependencies

- hamcrest-core : 1.3 [test]
- junit : 4.13 [test]
- spring-aop : 5.2.4.RELEASE [compile]
- spring-beans : 5.2.4.RELEASE [compile]
- spring-context : 5.2.4.RELEASE [compile]
- spring-core : 5.2.4.RELEASE [compile]
- spring-expression : 5.2.4.RELEASE [compile]
- spring-jcl : 5.2.4.RELEASE [compile]

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

Markers | Properties | Servers | Data Source Explorer | Snippets | Console

```
<terminated> C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe (2020. 3. 18. 오전 8:35:43)
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 5.976 s
[INFO] Finished at: 2020-03-18T08:35:52+09:00
```

19. Maven Repositories View 추가하기

- Window menu > Show View > Other > Maven > Maven Repositories
- Local Repository -> C:\Users\W...W.m2\repository 에 저장

Markers | Properties | Servers | Data Source Explorer | Snippets | Console | **Maven Repositories**

- ▼ Local Repositories
 - > Local Repository (C:\Users\Instructor\W.m2\repository)
 - > Workspace Projects
- > Global Repositories
 - Project Repositories
 - Custom Repositories

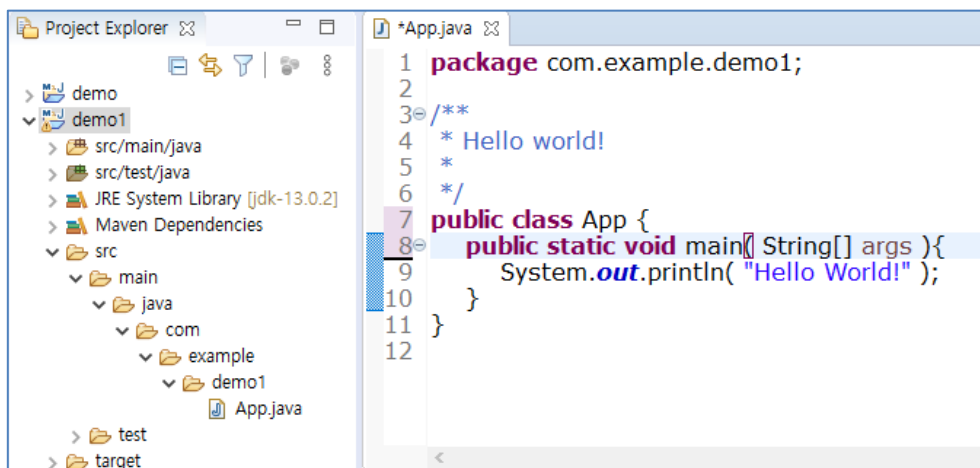
Task 4. Maven Project Creation in Eclipse

1. In Eclipse EE

- 1) New > Other > Maven > Maven Project > Next
- 2) Next > org.apache.maven.archetypes | maven-archetype-quickstart | 1.1 >
- 3) Group Id : com.example
- 4) Artifact Id : demo1
- 5) Version : 0.0.1-SNAPSHOT
- 6) Package : com.example.demo1
- 7) Finish

2. App.class 실행

- 1) src/main/java > package com.example.demo1 > App.java

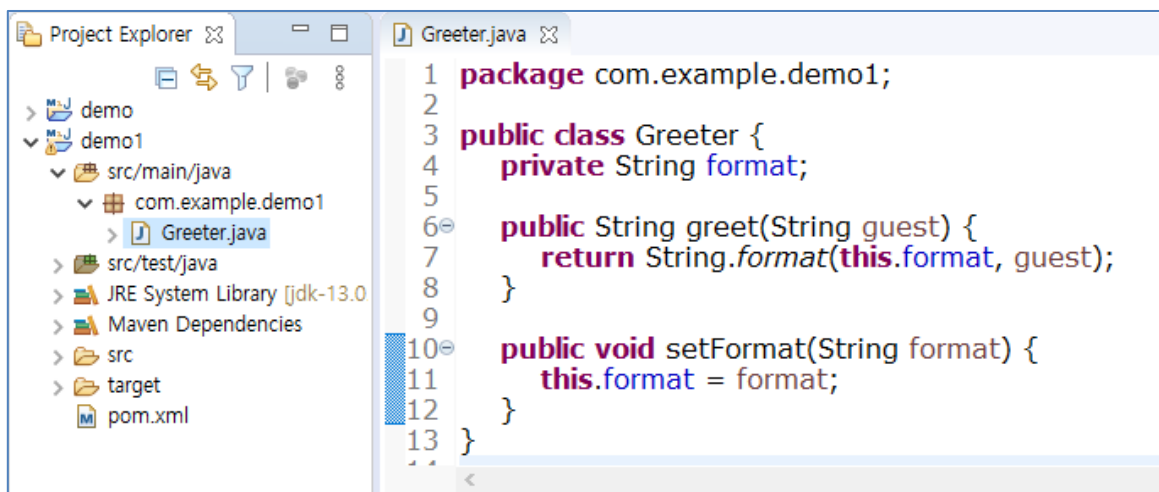


2) Run As > Java Application

3) Delete App.java

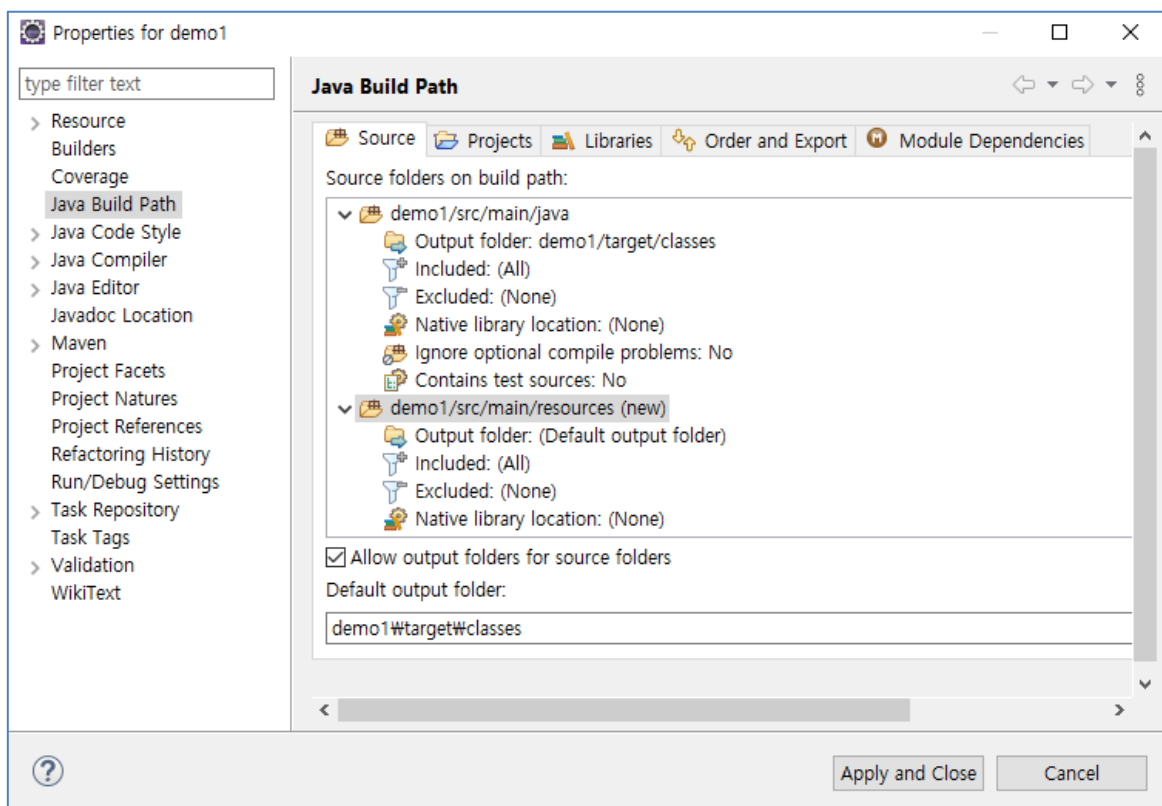
4) Greeter.java 생성

- src/main/java > package com.example.demo1 > New > Create Class Greeter.java



5) [demo1] 프로젝트 > src > main > resources folder Build path 추가하기

- demo1 Project right-click to [Build Path]
- Click to [Configure Build Path]
- Click to [Source] Tab
- Click to [Add Folder]
- Select [main] folder
- Click [Create New Folder]
- Folder name : resources
- Finish
- OK
- Click [Apply and Close]



6) [demo1] > src/main/resources > right-click > New > Other > XML > xml file > Next

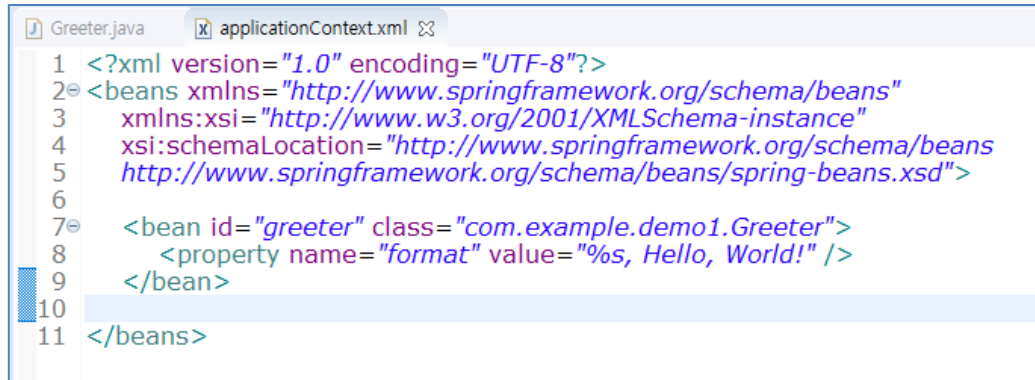
- file name : applicationContext.xml
- Finish

```
<?xml version="1.0" encoding="UTF-8"?>
    <beans xmlns="http://www.springframework.org/schema/beans"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
```

```

        <bean id="greeter" class="com.example.demo1.Greeter">
            <property name="format" value="%s, Hello, World!" />
        </bean>
    </beans>

```



7) Maven 에 Spring Context 설치하기

- In pom.xml

```

<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.13</version> <!-- Version 수정 -->
        <scope>test</scope>
    </dependency>
    <!--아래 코드 추가-->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.2.4.RELEASE</version>
    </dependency>
</dependencies>

```



```

14 <url>http://www.example.com</url>
15
16 <properties>
17   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
18   <maven.compiler.source>1.13</maven.compiler.source>
19   <maven.compiler.target>1.13</maven.compiler.target>
20 </properties>
21
22 <dependencies>
23   <dependency>
24     <groupId>junit</groupId>
25     <artifactId>junit</artifactId>
26     <version>4.13</version>
27     <scope>test</scope>
28   </dependency>
29   <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
30   <dependency>
31     <groupId>org.springframework</groupId>
32     <artifactId>spring-context</artifactId>
33     <version>5.2.4.RELEASE</version>
34   </dependency>
35

```

- [demo1] project > right-click > Properties > Project Facets
- Select Java > Select [Runtimes] tab > Check [jdk1.8.0_221]
- Click [Apply and Close]
- pom.xml > Run As > Maven install
- [Console]에서 '**BUILD SUCCESS**' 확인

8) src/main/java > com.example.demo1 > Main class 생성

```
package com.example.demo1;
```

```
import org.springframework.context.support.GenericXmlApplicationContext;
```

```

public class Main {
    public static void main(String [] args){
        GenericXmlApplicationContext ctx = new
            GenericXmlApplicationContext("classpath:applicationContext.xml");

        Greeter g = ctx.getBean("greeter", Greeter.class);
        String msg = g.greet("Spring");
        System.out.println(msg);
        ctx.close();
    }
}

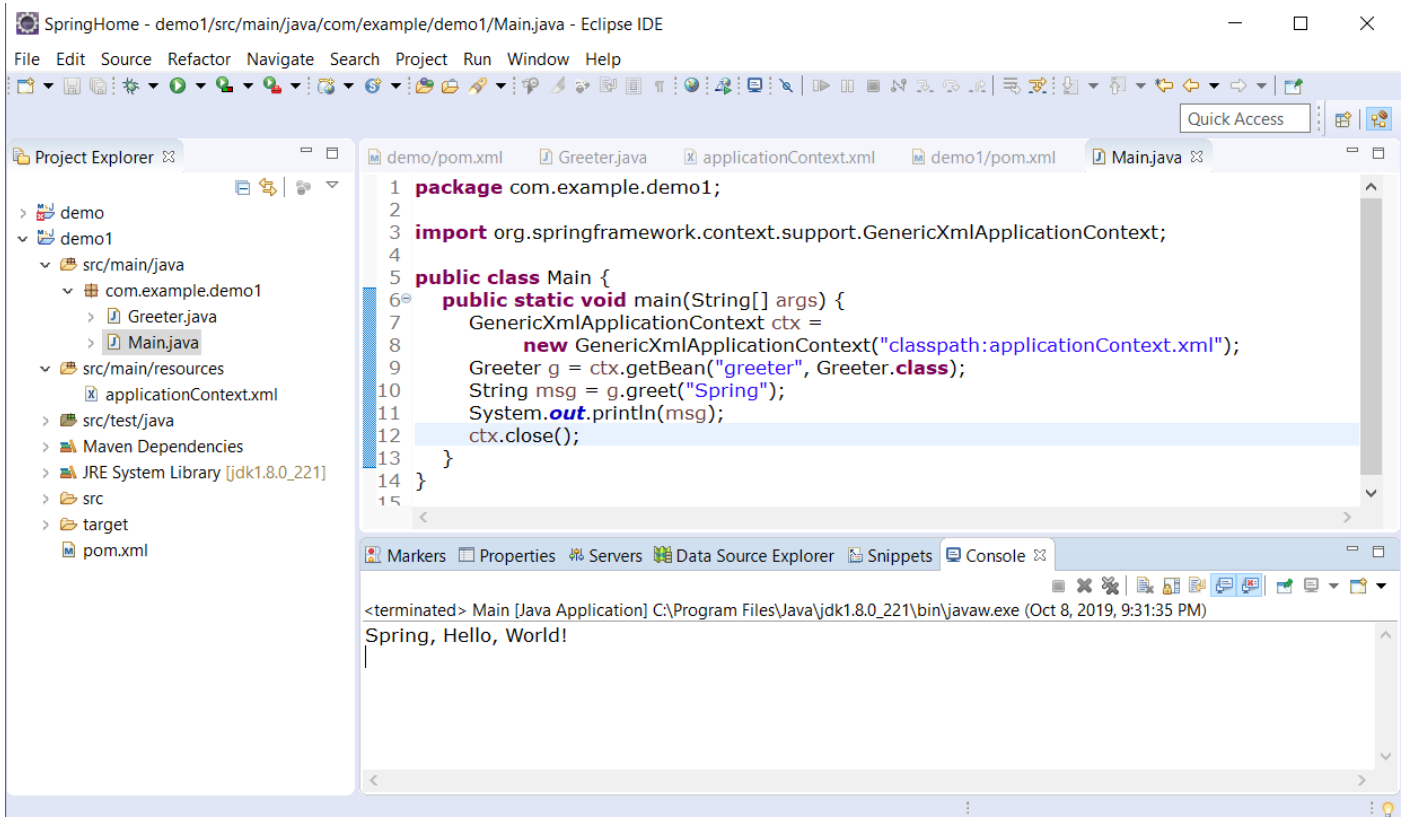
```

```

1 package com.example.demo1;
2
3 import org.springframework.context.support.GenericXmlApplicationContext;
4
5 public class Main {
6     public static void main(String[] args) {
7         GenericXmlApplicationContext ctx =
8             new GenericXmlApplicationContext("classpath:applicationContext.xml");
9         Greeter g = ctx.getBean("greeter", Greeter.class);
10        String msg = g.greet("Spring");
11        System.out.println(msg);
12        ctx.close();
13    }
14 }
15

```

9) Main.java 실행하기



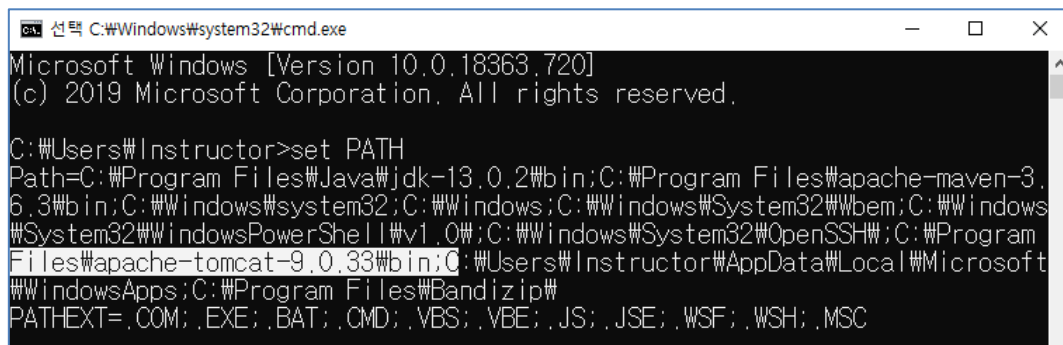
Task 5. Tomcat 9.0.33 Installation & Configuration

1. <http://tomcat.apache.org/>
2. 설치할 version 확인하기 : <http://tomcat.apache.org/whichversion.html>
3. Download Tomcat 9.0.33 from <https://tomcat.apache.org/download-90.cgi>
4. 64-bit Windows zip not 32-bit/64-bit Windows Service Installer(for Windows), tar.gz(for Linux)
5. Unzip apache-tomcat-9.0.33-windows-x64.zip
6. Move apache-tomcat-9.0.33 folder to C:/Program Files/
7. OS 환경변수 등록 : CATALINA_HOME=C:\Program Files\apache-tomcat-9.0.33



```
ca: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Instructor>set CATALINA_HOME
CATALINA_HOME=C:\Program Files\apache-tomcat-9.0.33
```

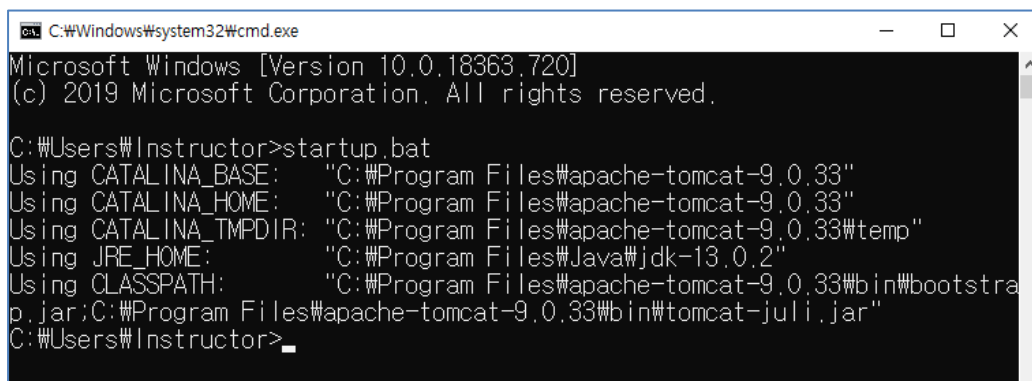


```
ca: 선택 C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Instructor>set PATH
Path=C:\Program Files\Java\jdk-13.0.2\bin;C:\Program Files\apache-maven-3.6.3\bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Program Files\apache-tomcat-9.0.33\bin;C:\Users\Instructor\AppData\Local\Microsoft\WindowsApps;C:\Program Files\Bandizip\
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
```

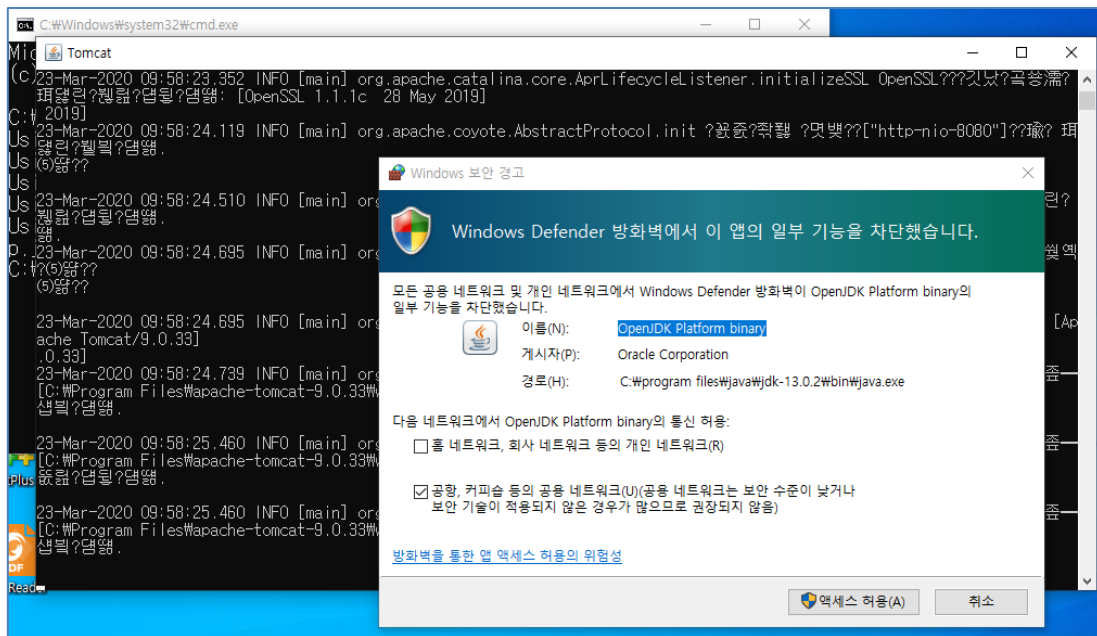
8. Tomcat Service Start

- 1) %CATALINA_HOME%\bin 에서 Tomcat Startup

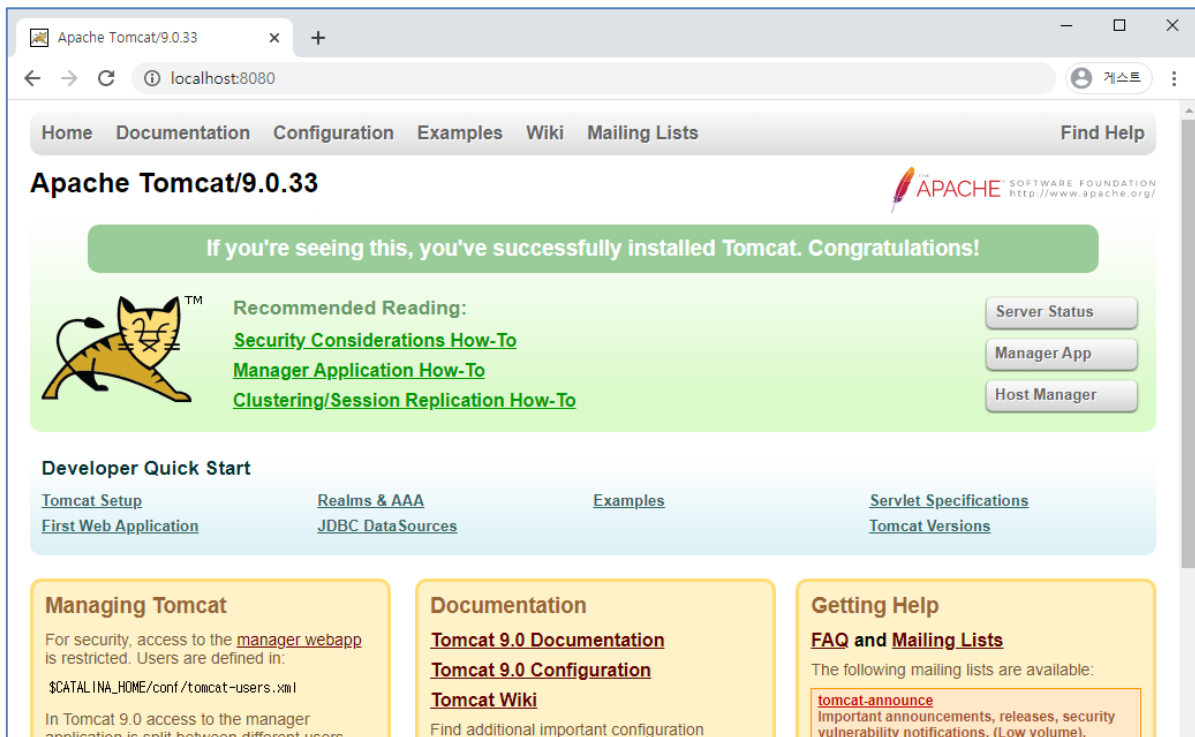


```
ca: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Instructor>startup.bat
Using CATALINA_BASE:   "C:\Program Files\apache-tomcat-9.0.33"
Using CATALINA_HOME:   "C:\Program Files\apache-tomcat-9.0.33"
Using CATALINA_TMPDIR: "C:\Program Files\apache-tomcat-9.0.33\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk-13.0.2"
Using CLASSPATH:       "C:\Program Files\apache-tomcat-9.0.33\bin\bootstrap.jar;C:\Program Files\apache-tomcat-9.0.33\bin\tomcat-juli.jar"
C:\Users\Instructor>
```



2) In Browser, <http://localhost:8080>



9. 관리자 계정 생성

- %CATALINA_HOME%/conf/tomcat-users.xml

```
<user username="admin" password="tomcatadmin" roles="admin-gui,manager-gui"/>
```

- Tomcat Service Restart

10. Tomcat home directory 변경

- %CATALINA_HOME%/webapps/homecontext.xml

```
<Context path="" docBase="C:/SpringHome" debug="0" reloadable="true"
crossContext="true" privileged="true" />
```

- %CATALINA_HOME%/webapps/ROOT/WEB-INF 복사하여 C:/SpringHome 하위에 붙여넣기
- C:/SpringHome/WEB-INF/web.xml 수정

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5                          http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
6      version="4.0"
7      metadata-complete="true">
8
9      <display-name>Welcome to Spring Homepage</display-name>
10
11 </web-app>

```

- C:/SpringHome/index.html 생성

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Welcome to Spring Homepage</title>
6  </head>
7  <body>
8      <div><span style="font-size:30pt;color:blue">Welcome Spring 5</span></div>
9  </body>
10 </html>

```

- [Tomcat Web Application Manager]에서 / Path undeploy

경로	버전	표시 이름	실행 중	세션들	명령들
/	지정 안됨	Welcome to Tomcat	true	0	시작 중지 다시 로드 배치된 것을 제거 세션들을 만료시키기 idle 값 ≥ 30 분
/docs	지정 안됨	Tomcat Documentation	true	0	시작 중지 다시 로드 배치된 것을 제거 세션들을 만료시키기 idle 값 ≥ 30 분
/examples	지정 안됨	Servlet and JSP Examples	true	0	시작 중지 다시 로드 배치된 것을 제거 세션들을 만료시키기 idle 값 ≥ 30 분
/host-manager	지정 안됨	Tomcat Host Manager Application	true	0	시작 중지 다시 로드 배치된 것을 제거

- [Tomcat Web Application Manager]에서 / Path deploy
 - Context path : /

- XML Configuration file path : C:\Program Files\Apache-tomcat-9.0.31\webapps\homecontext.xml

배치

서버에 있는 디렉토리 또는 WAR 파일을 배치합니다.

컨텍스트 경로: /

버전 (병렬 배치용):



XML 설정 파일 경로: /webapps/homecontext.xml

WAR 또는 디렉토리 경로:

배치

/manager

localhost:8080/manager/html/deploy?org.apache.catalina.filters.CSRF_NONCE=71AB811229365A3BE201E8A5...

Tomcat 웹 애플리케이션 매니저

메시지: OK - 컨텍스트 경로 [/]에 애플리케이션을 배치했습니다.

매니저

애플리케이션들의 목록을 표시 [HTML 매니저 도움말](#) [매니저 도움말](#) [서버 상태](#)

애플리케이션들

경로	버전	표시 이름	실행 중	세션들	명령들
/	지정 안됨	Welcome to Spring Homepage.	true	0	<div>시작 중지 다시 로드 배치된 것을 제거</div> <div>세션들을 만료시키기 idle 값 ≥ 30 분</div>
/docs	지정 안됨	Tomcat Documentation	true	0	<div>시작 중지 다시 로드 배치된 것을 제거</div> <div>세션들을 만료시키기 idle 값 ≥ 30 분</div>
/examples	지정 안됨	Servlet and JSP Examples	true	0	<div>시작 중지 다시 로드 배치된 것을 제거</div> <div>세션들을 만료시키기 idle 값 ≥ 30 분</div>
/host-manager	지정 안됨	Tomcat Host Manager Application	true	0	<div>시작 중지 다시 로드 배치된 것을 제거</div> <div>세션들을 만료시키기 idle 값 ≥ 30 분</div>

Welcome to Spring Homepage

localhost:8080

Welcome Spring 5

Task 6. STS 4.6.0.RELEASE Installation and Configuration

1. STS(Spring Tools Suite) 소개

- Spring 개발업체인 SpringSource 가 직접 만들어 제공하는 Eclipse 의 확장판으로 최신 Eclipse 를 기반으로 주요한 Spring 지원 Plug-in 과 관련된 도구를 모아서 Spring 개발에 최적화되도록 만들어진 IDE 이다.
- STS 가 제공하는 기능
 - Bean class 이름 자동완성
 - 현재 project 의 모든 source 와 library, JDK 안의 모든 클래스 중에서 첫 글자가 SDD 로 시작하는 class 를 자동으로 보여줌
 - 설정 file 생성 wizard
 - Bean 설정 file 생성 wizard 중 사용할 namespace 와 schema version 을 선택하는 화면 제공
 - Bean 의존 관계 graph
 - Spring IDE 는 XML 설정 file 을 읽어서 자동으로 graph 그려줌
 - 각 bean 이 어떻게 참조되고, 어떤 property 를 갖는지 알 수 있음.
 - AOP 적용 대상 표시
 - Spring IDE 의 XML 설정 file 편집기를 이용하면 AOP 의 적용 대상을 손쉽게 확인할 수 있다.

2. Downloads

- Visit to <https://spring.io/tools>
- In [Spring Tools 4 for Eclipse], Click [Windows 64-BIT]
- Filename : spring-tool-suite-4-4.6.0.RELEASE-e4.15.0-win32.win32.x86_64.self-extracting.jar

3. STS 시작하기

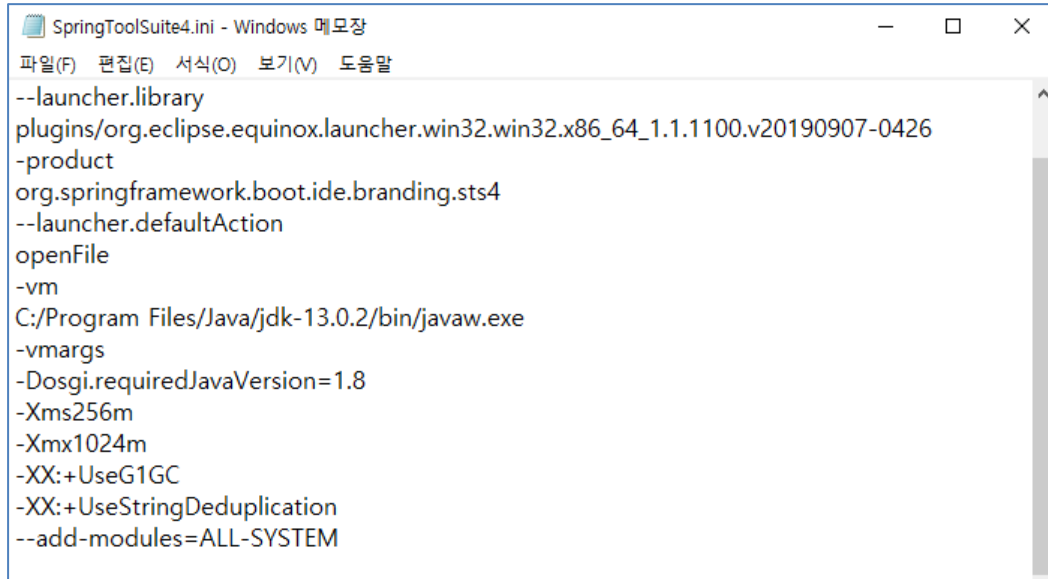
- Download 받은 spring-tool-suite-4-4.6.0.RELEASE-e4.15.0-win32.win32.x86_64.self-extracting.jar 파일을 double click 하여 압축 푼다.
- 압축이 풀리면 sts-4.6.0.RELEASE 폴더가 만들어진다.
- 이 폴더를 Cut 해서 C:\Program Files\에 붙인다.
- 바탕화면에 Shortcut 생성을 통해 link 를 C:\Program Files\sts-4.6.0.RELEASE\SpringToolSuite4.exe 으로 연결한다.
- Shortcut 을 실행한다.
- Workspace 를 C:\SpringHome 으로 잡고 [Use this as the default and do not ask again] check 한 뒤, [OK] button 을 누른다.

4. STS 실행 환경 편집

- 설치한 STS 를 별다른 설정 없이 사용하는 것도 가능하다.
- 하지만, STS(Eclipse 도 마찬가지로)는 기본적으로 JDK 가 아닌 JRE 를 이용해서 실행되기 때문에 이후에 설치할 Lombok library 등의 사용을 위해서 실행 환경을 편집할 필요가 있다.
- STS 설치 folder 에 SpringToolSuite4.ini(Eclipse 인 경우에는 eclipse.ini)를 편집기로 열고 -vmargs 바로 위줄에 아래와 같이 입력 후 저장한다.

-vm

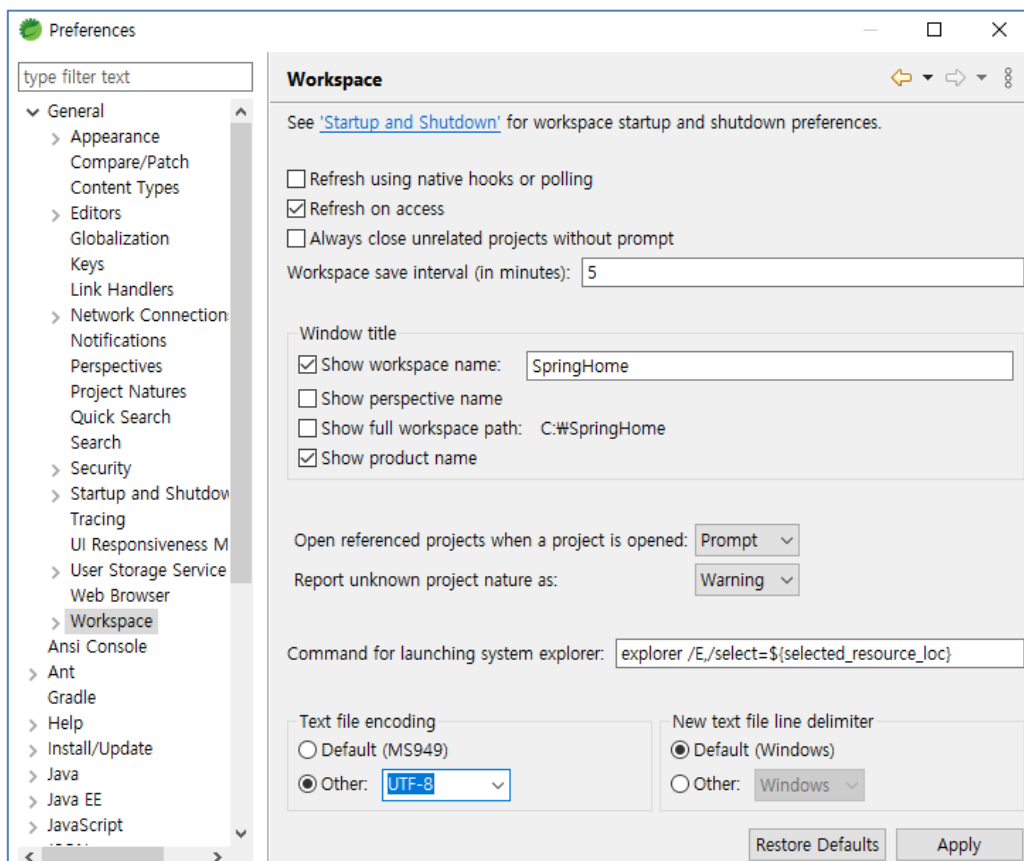
C:/Program Files/Java/jdk-13.0.2/bin/javaw.exe



- 파일에 '-vm'이라는 옵션을 설정하는데, 내용은 JDK 경로내의 bin folder 의 javaw.exe 를 지정하는 것이다.
- STS 실행 환경 편집 후 바로가기 아이콘의 문제가 발생할 수 있으므로, 반드시 기존의 '바로가기' 아이콘은 삭제 후 다시 '바로가기'를 추가하는 것이 좋다.

5. Workspace 의 UTF-8 설정

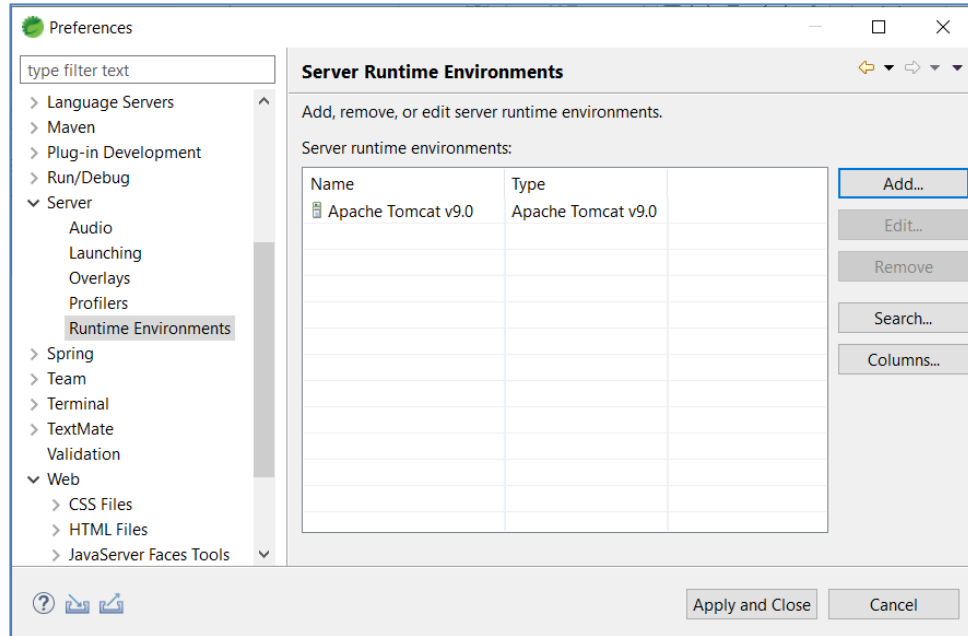
- 특별히 Windows 에서는 encoding 방식이 MS949(or Cp 1252)로 기본 설정되어 있기 때문에 반드시 UTF-8 로 변경해야 한다.
- 변경하기 위해 Window > Preferences > General > Workspace > Text file encoding



- 또한 CSS, HTML, JSP 도 모두 [Korean, EUC-KR]에서 UTF-8 로 변경한다.

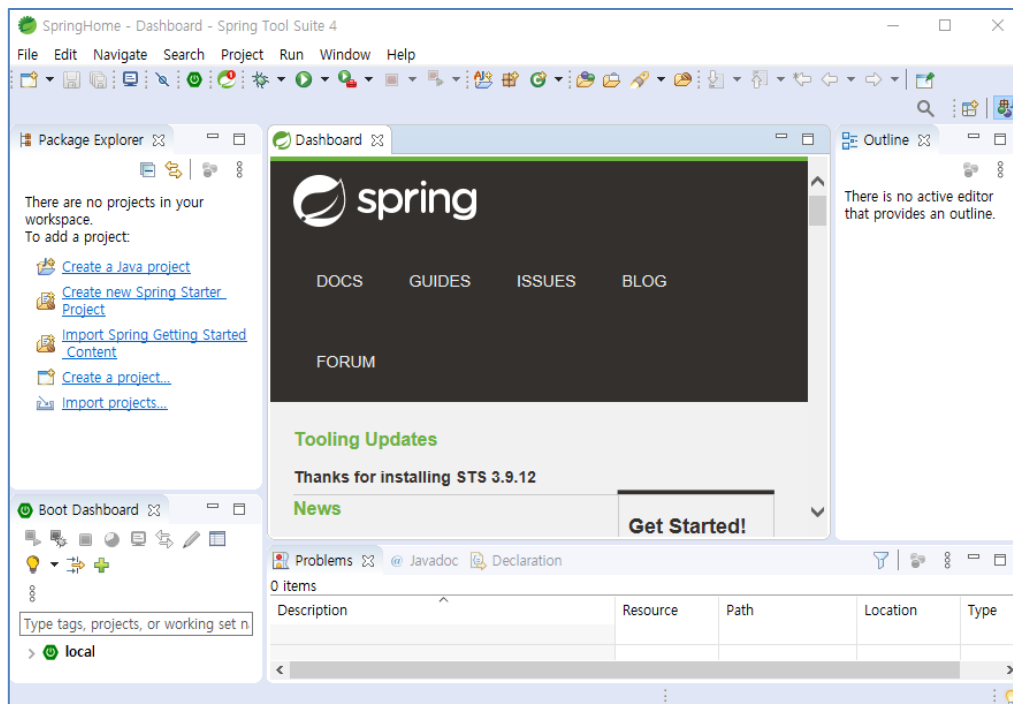
6. STS 에 Tomcat 등록하기

- Window > Preferences > Server > Runtime Environments 에서 [Add] 버튼을 click 한다.
- Apache > Apache Tomcat v9.0 > Next
- Tomcat installation directory : C:\Program Files\Apache-tomcat-9.0.26
- JRE:jdk-13.0.2
- Finish



7. STS4 에 Spring Legacy Project 생성을 위한 STS3 Add-On 설치하기

- 1) "Spring Legacy Project"는 STS3 까지만 제공되고 STS4 에서는 Spring Boot 가 기본이기 때문에 STS3 Add-On 해야 한다.
- 2) STS4 메뉴 > Help > [Eclipse Marketplace...] > "STS"로 검색 > [Spring Tools 3 Add-On for Spring Tools 4 3.9.12.RELEASE] 항목의 [Install] 버튼 Click
- 3) 설치 후 STS4 Restart



8. STS 로 간단한 Maven Project 만들기

1) Java Project 생성

- Project Name : HelloWorld
- Class Name : Hello

```
public class Hello {  
    public static void main(String [] args){  
        System.out.println("Hello, World");  
    }  
}
```

- 실행 확인

2) Maven Project 로 전환

- HelloWorld Project > right-click > Configure > Convert to Maven Project
- Project : /HelloWorld
- Group Id : HelloWorld
- Artifact Id : HelloWorld
- version : 0.0.1-SNAPSHOT
- Packaging : jar
- Finish

3) Spring Project 로 전환

- HelloWorld Project > right-click > Spring > Add Spring Project Nature
- <http://mvnrepository.com> 에서 'spring context' 검색
- 이 문서를 작성하는 현재 버전은 5.2.4.RELEASE 이다.
- 현재 버전의 Dependency 를 복사한 다음 pom.xml 에 붙여 넣는다.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"  
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  
        http://maven.apache.org/xsd/maven-4.0.0.xsd">  
    <modelVersion>4.0.0</modelVersion>  
    <groupId>HelloWorld</groupId>  
    <artifactId>HelloWorld</artifactId>  
    <version>0.0.1-SNAPSHOT</version>  
    <dependencies>
```

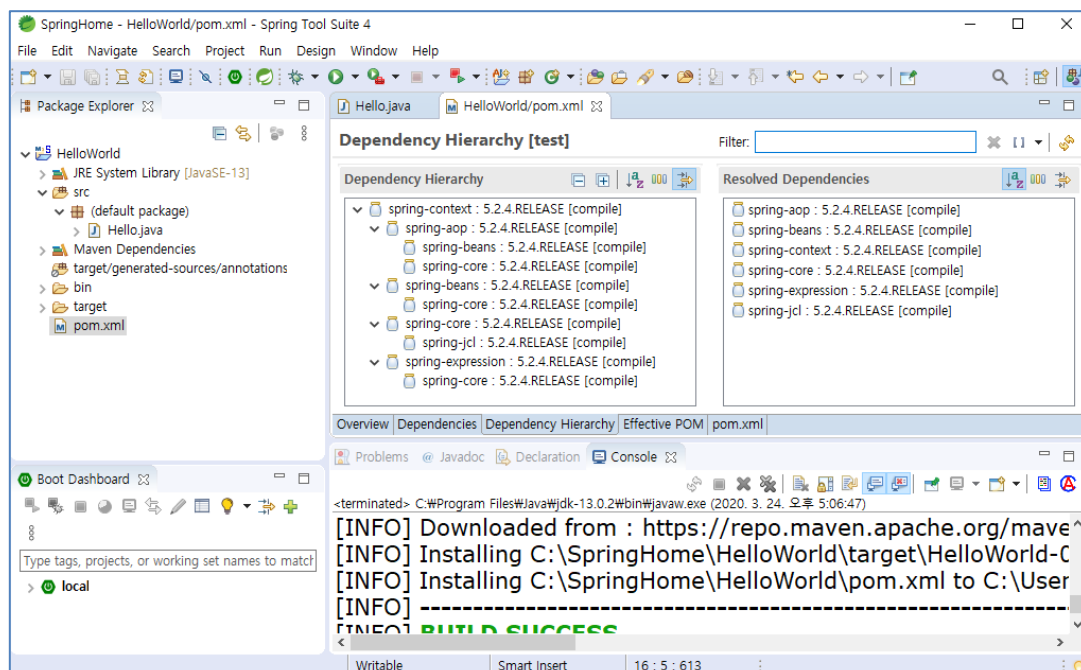
```

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.4.RELEASE</version>
</dependency>

</dependencies>
<build>
    <sourceDirectory>src</sourceDirectory>
    <plugins>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.0</version>
            <configuration>
                <release>13</release>
            </configuration>
        </plugin>
    </plugins>
</build>
</project>

```

- pom.xml > right-click > Run As > Maven Install > BUILD SUCCESS <--at Console View
- Dependencies tab 에서 spring-context : 5.2.4.RELEASE 설치된 것을 확인 함.

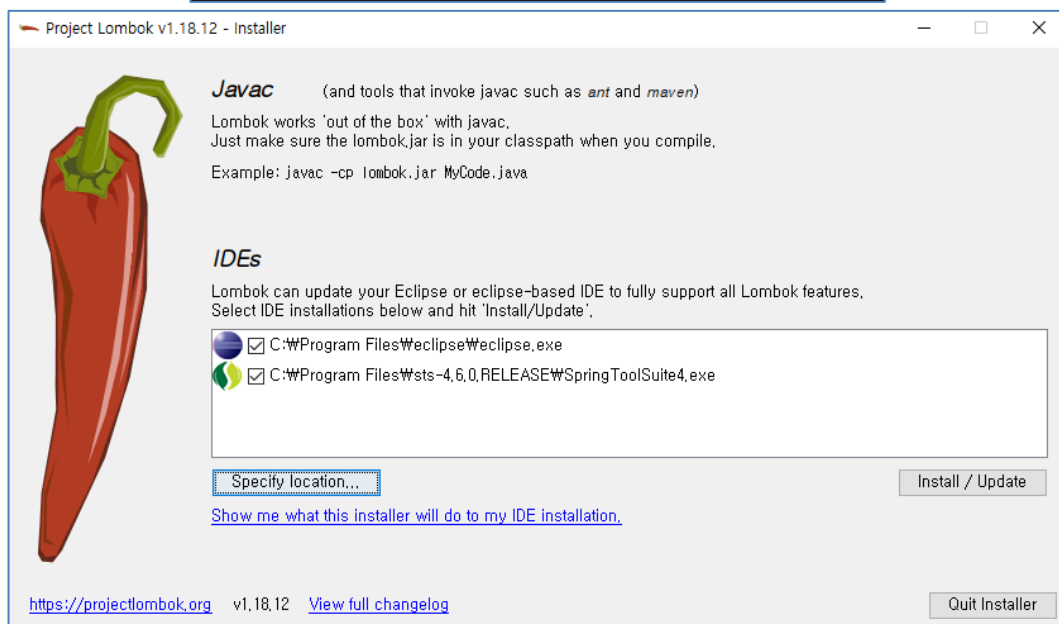


Task 7. Lombok 1.18.12 Library Installation

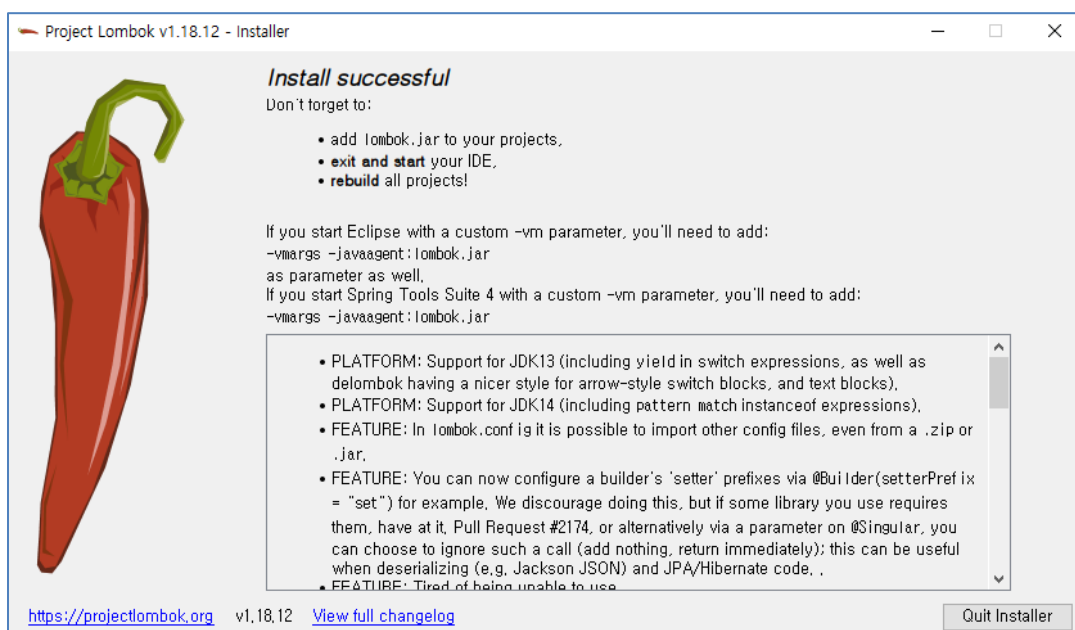
1. Java 개발할 때 많이 사용하는 getter/setter, toString(), 생성자 등을 자동으로 생성해 주는 Library 이다.
2. 다른 jar file 들과 다르게 Project 의 code 에서만 사용되는 것이 아니고, STS(Eclipse) 내에서도 사용되어야 하기 때문에 별도로 설치한다.
3. <https://projectlombok.org/>
4. Download 1.18.12 : <https://projectlombok.org/download>
5. Lombok.jar 를 download 한 다음, Download 받은 folder 에서 다음과 같이 실행할 수 있다.
6. C:/Downloads>java -jar lombok.jar

```
C:\Windows\system32\cmd.exe

C:\Downloads>java -jar lombok.jar
```



7. 실행되는 화면에는 필요한 IDE 를 선택할 수 있다.
8. 만일 Eclipse 나 STS 의 설치 경로를 찾지 못할 경우에는 [Specify location...] 버튼을 눌러서 직접 지정한다.
9. [Install/Update] button 을 click 한다.



10. [Install successful] 창에서 [Quit Installer] button 을 click 하여 창을 닫는다.

11. 설치가 끝나면 Eclipse 와 STS 의 실행 경로에 lombok.jar file 이 자동으로 추가된 것을 확인할 수 있다.

