

HOL : Configure for Lab

1. JDK 11.x Installation and Configuration
2. Eclipse JEE Download and Configuration
3. Maven 3.8.3 Installation and Configuration
4. Maven Project Creation in Eclipse
5. Tomcat 10.0.12 Installation & Configuration
6. STS 4.12.0.RELEASE Downloads & Configuration
7. Lombok 1.18.20 Library Installation

Task 1. JDK 11.x Installation and Configuration

1. JDK : <https://www.oracle.com/java/technologies/downloads/>
2. Documentation : <https://docs.oracle.com/en/java/javase/11/>
3. API : <https://docs.oracle.com/en/java/javase/11/docs/api/index.html>

The screenshot shows a Windows Command Prompt window with the following text:

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\instructor>set JAVA_HOME
JAVA_HOME=C:\Program Files\Java\jdk-11.0.12

C:\Users\instructor>set PATH
Path=C:\Program Files\Java\jdk-11.0.12\bin;C:\Program Files\Python39\Scripts\;C:\Program
Files\Python39\;C:\Windows\system32;C:\Windows;C:\Windows\System32\WBem;C:\Windows\System
32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Program Files\MySQL\MySQL Serv
er 5.7\bin\;C:\Program Files\Microsoft VS Code\bin\;C:\Users\instructor\AppData\Local\Micro
soft\WindowsApps\;C:\Program Files\Bandizip\;C:\Program Files\JetBrains\PyCharm Community
Edition 2021.2.1\bin\;
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY;.PYW
```

Task 2. Eclipse Download and Configuration

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

Eclipse IDE 2021-09 R Packages

Eclipse IDE for Java Developers
322 MB 402,372 DOWNLOADS
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration.

Eclipse IDE for Enterprise Java and Web Developers
517 MB 264,645 DOWNLOADS
Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces, YAML, Markdown, Web Services, JPA and Data Tools, Maven and Gradle, Git, and more.
Click here to file a bug against Eclipse Web Tools Platform.
Click here to file a bug against Eclipse Platform.
Click here to file a bug against Maven integration for web projects.
Click here to report an issue against Eclipse Wild Web Developer (incubating).

Windows x86_64
macOS x86_64
Linux x86_64 | AArch64

Get Eclipse IDE 2021-09
Install your favorite desktop IDE packages.
Download x86_64

RELATED LINKS

- Compare & Combine Packages
- New and Noteworthy
- Install Guide
- Documentation
- Updating Eclipse
- Forums
- Simultaneous Release

MORE DOWNLOADS

2. Eclipse IDE for Enterprise Java and Web Developers

SpringHome - Eclipse IDE

File Edit Navigate Search Project Run Window Help

Welcome

Eclipse IDE for Enterprise Java and Web Developers

Hide

Review IDE configuration settings
Create a new Java Web Project
Checkout projects from Git
Import existing projects
Launch the Eclipse Marketplace
Open an existing file

Overview
Tutorials
Samples
What's New

Always show Welcome at start up

Task 3. Maven 3.8.3 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-3.8.3-bin.zip
 - 3) 환경변수 M2_HOME(C:\Program Files\apache-maven-3.8.3) 설정
 - 4) 환경변수 PATH에 (%M2_HOME%\bin) 추가

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\instructor>set M2_HOME
M2_HOME=C:\Program Files\apache-maven-3.8.3

C:\Users\instructor>set PATH
Path=C:\Program Files\Java\jdk-11.0.12\bin;C:\Program Files\apache-maven-3.8.3\bin;C:\Program Files\Python39\Scripts\;C:\Program Files\Python39\;C:\Windows\system32;C:\Windows;C:\Windows\System32\WBem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Program Files\MySQL\MySQL Server 5.7\bin;C:\Program Files\Microsoft VS Code\bin;C:\Users\instructor\AppData\Local\Microsoft\WindowsApps;C:\Program Files\Bandizip\;C:\Program Files\JetBrains\PyCharm Community Edition 2021.2.1\bin;
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY;.PYW
```

4. 설치 확인

\$ mvn -version

```
C:\Users\instructor>mvn -version
Apache Maven 3.8.3 (ff8e977a158738155dc465c6a97ffaf31982d739)
Maven home: C:\Program Files\apache-maven-3.8.3
Java version: 11.0.12, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-11.0.12
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 project를 생성해 보자.
- Command prompt에서 아래 명령어를 실행하면 된다.
- 아래 명령어를 처음 실행할 경우 꽤 오랜 시간이 걸리는데, 그 이유는 Maven이 필요한 plug-in과 module을 download 받기 때문이다.
- Maven 배포판은 최초로 Maven을 사용하는 데 필요한 module만 포함하고 있고, 그 외에 archetype plug-in, compiler plug-in 등 Maven을 사용하는 데 필요한 module은 포함하고 있지 않다.
- 이들 module은 실제로 필요할 때 Maven 중앙 Repository에서 loading된다.
- 먼저 C:\SpringHome folder를 생성한 후, folder로 이동한다.

\$ mvn archetype:generate

```
C:\Windows\system32\cmd.exe
C:\SpringHome>mvn archetype:generate
```

```
C:\Windows\system32#cmd.exe - mvn archetype:generate
2971: remote -> ua.co.gravy.archetype:single-project-with-junit-and-slf4j (Create a single project with jUnit, Mockito and slf4j dependencies.)
2972: remote -> uk.ac.ebi.gxa:atlas-archetype (Archetype for generating a custom Atlas webapp)
2973: remote -> uk.ac.gate:gate-plugin-archetype (Maven archetype to create a new GATE plugin project.)
2974: remote -> uk.ac.gate:gate-pr-archetype (Maven archetype to create a new GATE plugin project including a sample PR class (an empty LanguageAnalyser).)
2975: remote -> uk.ac.nactem.argo:argo-analysis-engine-archetype (An archetype which contains a sample Argo (UIMA) Analysis Engine)
2976: remote -> uk.ac.nactem.argo:argo-reader-archetype (An archetype which contains a sample Argo (UIMA) Reader)
2977: remote -> uk.ac.rdg.resc:edal-ncwms-based-webapp (-)
2978: remote -> uk.co.markg.archetypes:java11-junit5 (An archetype for generate java 11 projects with junit 5.)
2979: remote -> uk.co.nemstix:basic-javaee7-archetype (A basic Java EE7 Maven archetype)
2980: remote -> uk.co.solong:angular-spring-archetype (So Long archetype for RESTful spring services with an AngularJS frontend. Includes debian deployment)
2981: remote -> us.fatehi:schemacrawler-archetype-maven-project (-)
2982: remote -> us.fatehi:schemacrawler-archetype-plugin-command (-)
2983: remote -> us.fatehi:schemacrawler-archetype-plugin-dbconnector (-)
2984: remote -> us.fatehi:schemacrawler-archetype-plugin-lint (-)
2985: remote -> ws.osiris:osiris-archetype (Maven Archetype for Osiris)
2986: remote -> xyz.luan.generator:xyz-gae-generator (-)
2987: remote -> xyz.luan.generator:xyz-generator (-)
2988: remote -> za.co.absa.hyperdrive:component-archetype (-)
2989: remote -> za.co.absa.hyperdrive:component-archetype_2.11 (-)
2990: remote -> za.co.absa.hyperdrive:component-archetype_2.12 (-)
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): 1826:
1826:
```

...
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): 1826(번호는 다를 수

있다): 엔터

```
C:\Windows\system32#cmd.exe - mvn archetype:generate
2977: remote -> uk.ac.rdg.resc:edal-ncwms-based-webapp (-)
2978: remote -> uk.co.markg.archetypes:java11-junit5 (An archetype for generate java 11 projects with junit 5.)
2979: remote -> uk.co.nemstix:basic-javaee7-archetype (A basic Java EE7 Maven archetype)
2980: remote -> uk.co.solong:angular-spring-archetype (So Long archetype for RESTful spring services with an AngularJS frontend. Includes debian deployment)
2981: remote -> us.fatehi:schemacrawler-archetype-maven-project (-)
2982: remote -> us.fatehi:schemacrawler-archetype-plugin-command (-)
2983: remote -> us.fatehi:schemacrawler-archetype-plugin-dbconnector (-)
2984: remote -> us.fatehi:schemacrawler-archetype-plugin-lint (-)
2985: remote -> ws.osiris:osiris-archetype (Maven Archetype for Osiris)
2986: remote -> xyz.luan.generator:xyz-gae-generator (-)
2987: remote -> xyz.luan.generator:xyz-generator (-)
2988: remote -> za.co.absa.hyperdrive:component-archetype (-)
2989: remote -> za.co.absa.hyperdrive:component-archetype_2.11 (-)
2990: remote -> za.co.absa.hyperdrive:component-archetype_2.12 (-)
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): 1826:
1826:
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:
```

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
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:
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.4/maven-archetype-quickstart-1.4.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.4/maven-archetype-quickstart-1.4.pom (1.6 kB at 4.3 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-bundles/1.4/maven-archetype-bundles-1.4.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-bundles/1.4/maven-archetype-bundles-1.4.pom (4.5 kB at 14 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.4/maven-archetype-quickstart-1.4.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.4/maven-archetype-quickstart-1.4.jar (7.1 kB at 24 kB/s)
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

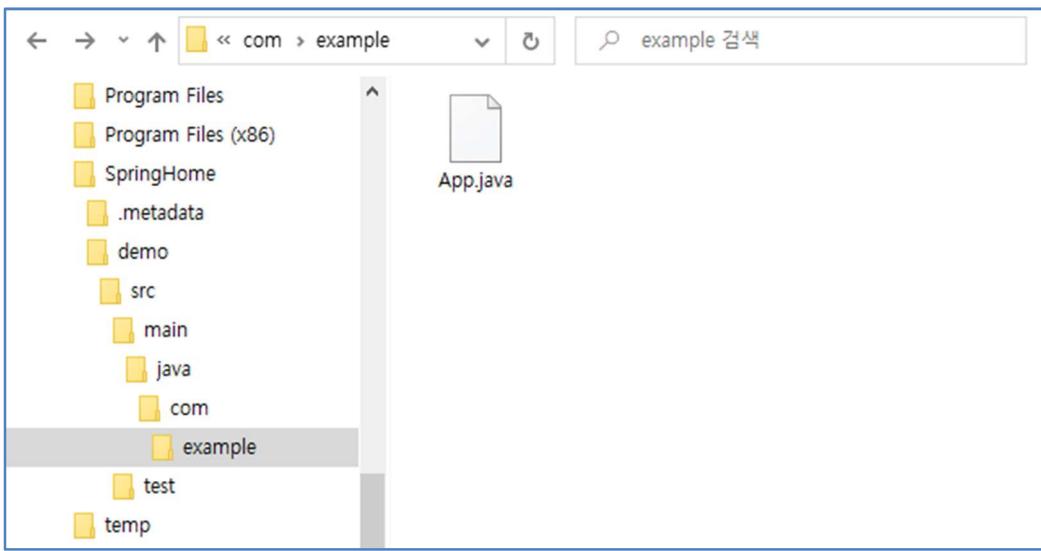
Y: : **엔터**

```
C:\Windows\system32\cmd.exe
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: :
[INFO] -----
[INFO] Using following parameters for creating project from Archetype: maven-archetype-quickstart:1.4
[INFO] -----
[INFO] Parameter: groupId, Value: com.example
[INFO] Parameter: artifactId, Value: demo
[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: 05:00 min
[INFO] Finished at: 2021-10-05T10:05:46+09:00
[INFO] -----
```

- 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

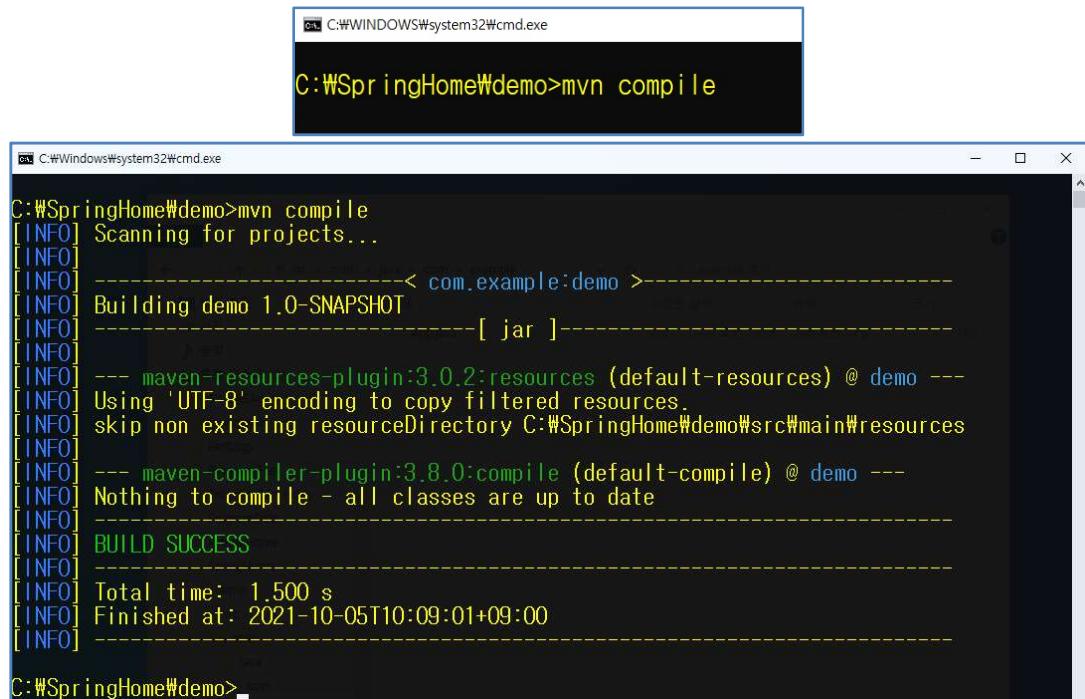


- archetype:generate 이 성공적으로 실행되면, artifactId 에 입력한 값과 동일한 이름의 directory 가 생성된다.
- archetype:generate 명령어는 미리 정의된 template 을 이용해서 Maven project 를 생성하는 기능으로 지금 사용한 Maven Project Template 는 archetype-quickstart 라는 template 이다.
- Template 을 사용하지 않고 직접 directory 를 생성하고 pom.xml file 을 작성해 주어도 동일한 Maven project 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

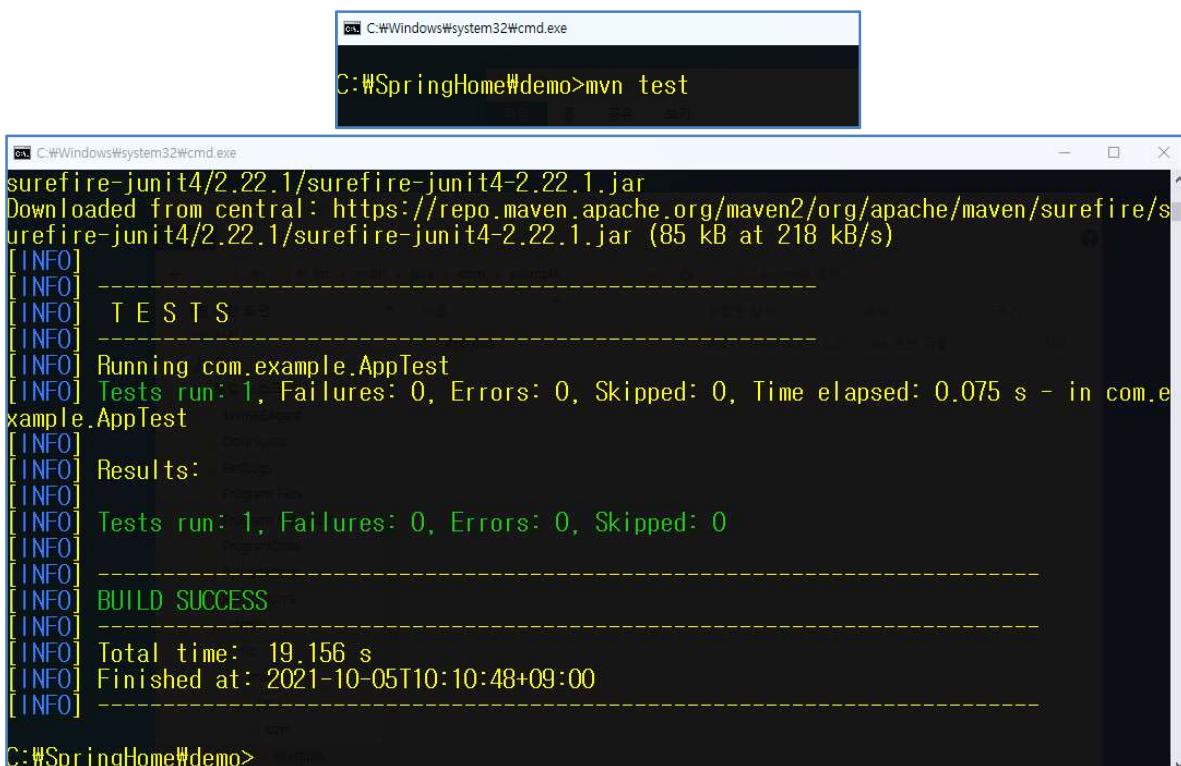


```
C:\Windows\system32\cmd.exe
C:\SpringHome\demo>mvn compile

C:\Windows\system32\cmd.exe
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] Nothing to compile - all classes are up to date
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 1.500 s
[INFO] Finished at: 2021-10-05T10:09:01+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

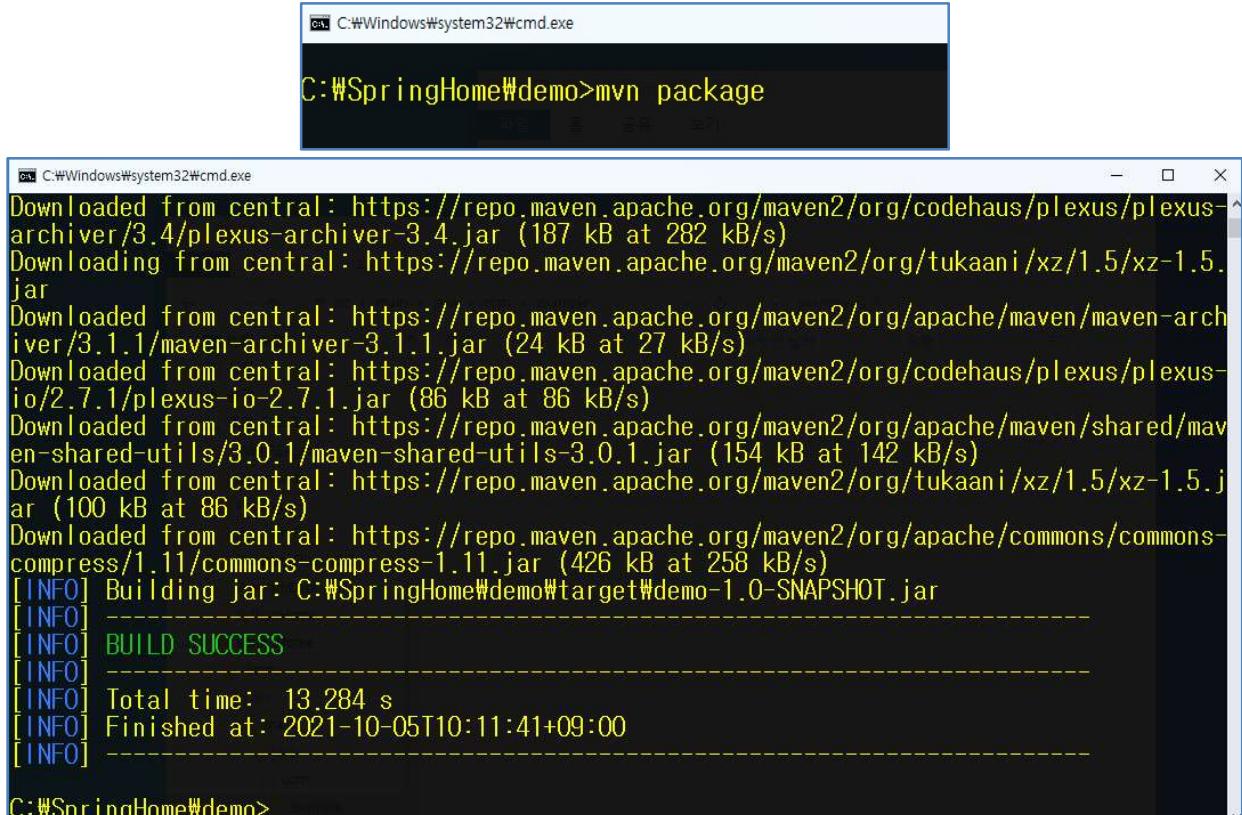


```
C:\Windows\system32\cmd.exe
C:\SpringHome\demo>mvn test

C:\Windows\system32\cmd.exe
surefire-junit4/2.22.1/surefire-junit4-2.22.1.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit4/2.22.1/surefire-junit4-2.22.1.jar (85 kB at 218 kB/s)
[INFO]
[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: 0.075 s - in com.example.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 19.156 s
[INFO] Finished at: 2021-10-05T10:10:48+09:00
[INFO] -----
```

- 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

Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-archiver/3.4/plexus-archiver-3.4.jar (187 kB at 282 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/tukaani/xz/1.5/xz-1.5.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-archiver/3.1.1/maven-archiver-3.1.1.jar (24 kB at 27 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-io/2.7.1/plexus-io-2.7.1.jar (86 kB at 86 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-utils/3.0.1/maven-shared-utils-3.0.1.jar (154 kB at 142 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/tukaani/xz/1.5/xz-1.5.jar (100 kB at 86 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/commons/commons-compress/1.11/commons-compress-1.11.jar (426 kB at 258 kB/s)
[INFO] Building jar: C:\SpringHome\demo\target\demo-1.0-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 13.284 s
[INFO] Finished at: 2021-10-05T10:11:41+09:00
[INFO] -----
```

- 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>11</maven.compiler.source>
    <maven.compiler.target>11</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-stdex</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-dbcp 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 사용되는지를 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	Souce 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. 맷음말

- 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 XML이나 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 11 버전을 사용하도록 설정이 필요하다.
- 이를 위해 pom.xml <dependencies> 하위에 아래 코드를 추가한다.

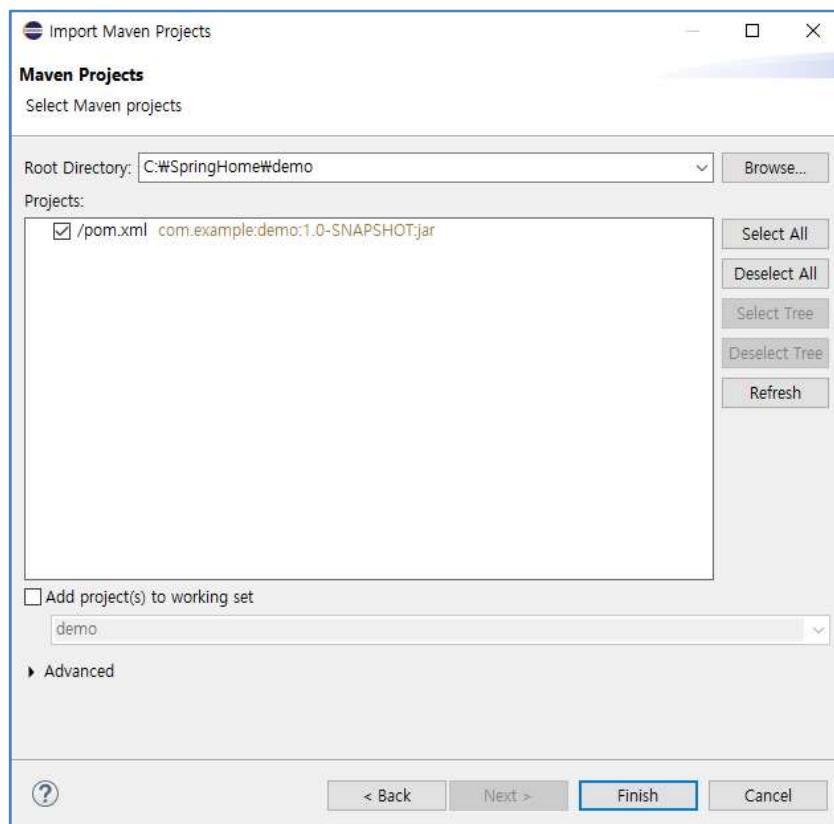
```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13</version>
    <scope>test</scope>
</dependency>
```

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

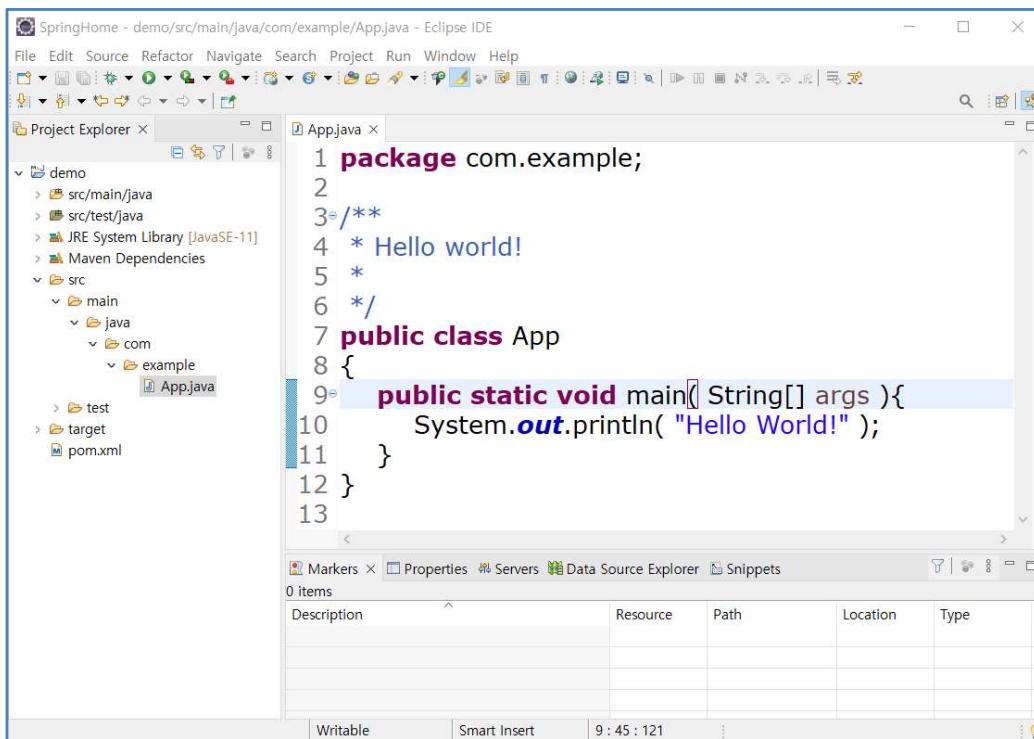
```
<build>
    <pluginManagement>
        <plugins>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>11</source>
                    <target>11</target>
                    <encoding>${project.build.sourceEncoding}</encoding>
                </configuration>
            </plugin>
        </plugins>
    </pluginManagement>
</build>
```

17. Eclipse에서 Maven Project Import하기

- File > Import > Maven > Existing Maven Projects > Next > Browse
- [Root Directory] : C:\SpringHome\demo > OK



- 그러면 자동으로 pom.xml 파일을 기준으로 project를 찾는다. > Finish



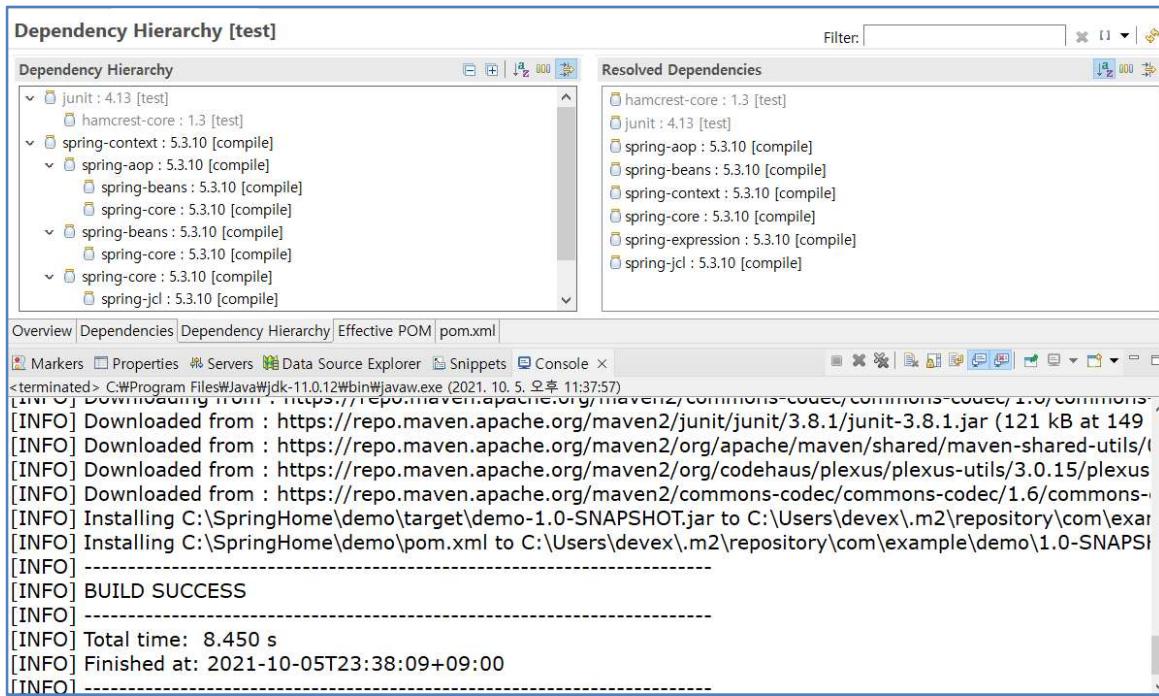
18. pom.xml 의존관계(dependency) 추가

- <https://mvnrepository.com/artifact/org.springframework/spring-context>에서 Spring Context latest version 선택
- Maven에 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>5.3.10</version>
    </dependency>
</dependencies>
```

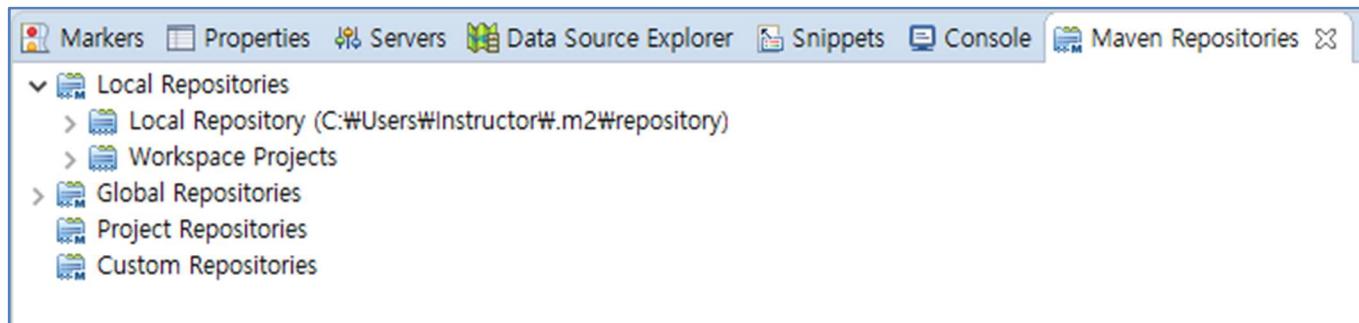
```
17     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
18     <maven.compiler.source>11</maven.compiler.source>
19     <maven.compiler.target>11</maven.compiler.target>
20 </properties>
21
22<dependencies>
23    <dependency>
24        <groupId>junit</groupId>
25        <artifactId>junit</artifactId>
26        <version>4.13</version>
27        <scope>test</scope>
28    </dependency>
29    <dependency>
30        <groupId>org.springframework</groupId>
31        <artifactId>spring-context</artifactId>
32        <version>5.3.10</version>
33    </dependency>
34
35 </dependencies>
36
```

- pom.xml > right-click > Run As > Maven install
- Eclipse가 spring-context module을 포함해서 필요한 jar file들을 download 받기 시작한다.
- 필요한 jar file들을 모두 download 받으면, Eclipse project에 download 받은 jar file들이 Maven 의존 목록(Maven Dependencies)에 표시된다.



19. Maven Repositories View 추가하기

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



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 | 1.4 > Next
- 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

The screenshot shows the Eclipse IDE interface. On the left, the Project Explorer view displays a Maven project structure with a 'demo' parent project containing a 'demo1' module. The 'src' folder of 'demo1' contains 'main' and 'test' folders. Under 'main', there is a 'java' folder which contains a 'com' package, and within that, a 'example' package, which in turn contains a 'demo1' package. Inside the 'demo1' package is an 'App.java' file. On the right, the editor view shows the source code for 'App.java':

```
1 package com.example.demo1;
2
3 /**
4  * Hello world!
5  *
6  */
7 public class App
8 {
9     public static void main(String[] args) {
10         System.out.println("Hello World!");
11     }
12 }
13
```

- 2) Run As > Java Application

- 3) Delete App.java

- 4) Greeter.java 생성

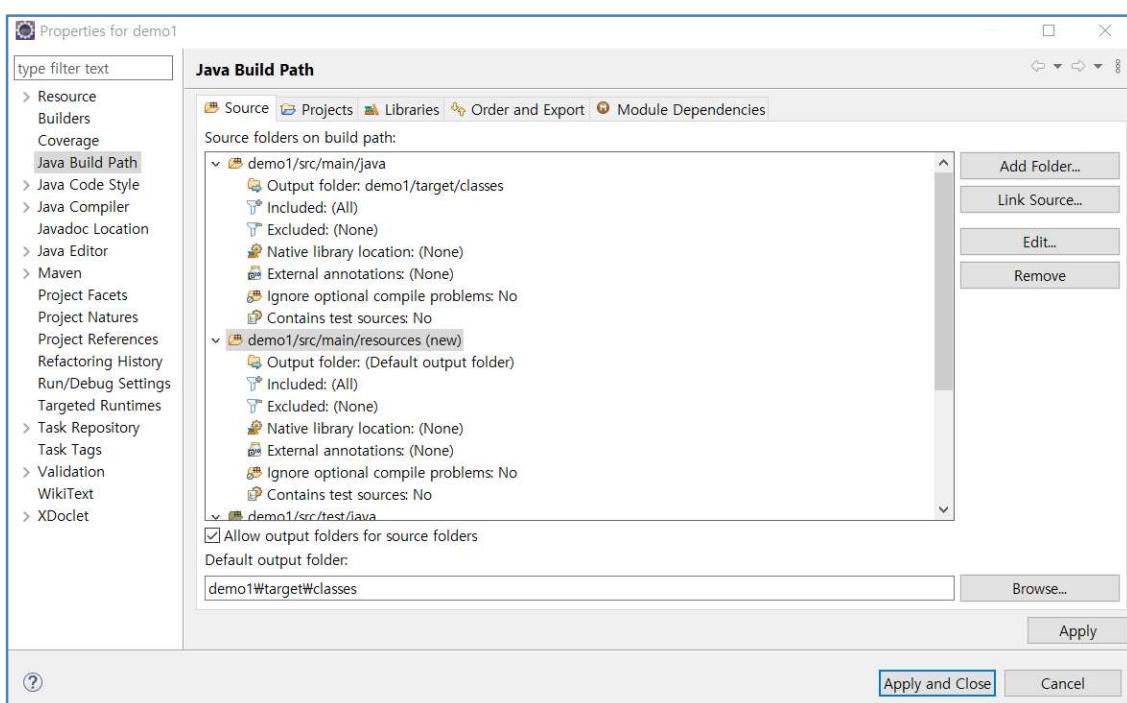
- src/main/java > package com.example.demo1 > New > Create Class Greeter.java

The screenshot shows the Eclipse IDE interface. On the left, the Project Explorer view displays the same Maven project structure as before, with 'demo1' now containing a 'com.example.demo1' package that includes a 'Greeter.java' file. On the right, the editor view shows the source code for 'Greeter.java':

```
1 package com.example.demo1;
2
3 public class Greeter {
4     private String format;
5
6     public String greet(String guest) {
7         return String.format(this.format, guest);
8     }
9
10    public void setFormat(String format) {
11        this.format = format;
12    }
13 }
```

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>
```

The screenshot shows an IDE interface with two tabs: 'Greeter.java' and 'applicationContext.xml'. The 'applicationContext.xml' tab is active, displaying the following XML code:

```
1 <?xml version="1.0" encoding="UTF-8"?>  
2 <beans xmlns="http://www.springframework.org/schema/beans"  
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
4   xsi:schemaLocation="http://www.springframework.org/schema/beans  
5     http://www.springframework.org/schema/beans/spring-beans.xsd">  
6  
7 <bean id="greeter" class="com.example.demo1.Greeter">  
8   <property name="format" value="%s, Hello, World!" />  
9 </bean>  
10  
11 </beans>
```

7) Maven에 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>5.3.10</version>  
  </dependency>  
</dependencies>
```

```

16<properties>
17  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
18  <maven.compiler.source>11</maven.compiler.source>
19  <maven.compiler.target>11</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>5.3.10</version>
34  </dependency>
35
36</dependencies>
37

```

- [demo1] project > right-click > Properties > Project Facets
- Select Java > Select [Runtimes] tab > Check [jdk-11.0.12]
- 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 = new
8             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 }

```

9) Main.java 실행하기

The screenshot shows the Eclipse IDE interface with the Main.java file open in the editor. The code is identical to the one shown above. In the bottom right corner of the editor, the text "Spring, Hello, World!" is displayed, indicating the program's output. The Project Explorer view on the left shows the project structure, including the GreeterJava, applicationContext.xml, demo1/pom.xml files, and the Main.java and Greeter.java source files.

```

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 = new
8             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 }

```

Task 5. Tomcat 10.0.12 Installation & Configuration

1. <http://tomcat.apache.org/>
2. 설치할 version 확인하기 : <http://tomcat.apache.org/whichversion.html>
3. Download Tomcat 10.0.12 from <https://tomcat.apache.org/download-10.cgi>
4. 64-bit Windows zip not 32-bit/64-bit Windows Service Installer(for Windows), tar.gz(for Linux)
5. Unzip apache-tomcat-10.0.12-windows-x64.zip
6. Move apache-tomcat-10.0.12 folder to C:/Program Files/
7. OS 환경변수 등록 : CATALINA_HOME=C:\Program Files\apache-tomcat-10.0.12

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\instructor>set CATALINA_HOME
CATALINA_HOME=C:\Program Files\apache-tomcat-10.0.12

C:\Users\devex>set PATH
Path=C:\Program Files\Java\jdk-11.0.12\bin;C:\Program Files\Python39\Scripts\;C:\Program Files\Python33\;C:\Program Files (x86)\NAT Service;C:\app\devex\product\18.0.0\dbhomeXE\bin;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\WBem;C:\WINDOWS\System32\WindowsPowerShell\v1.0\;C:\WINDOWS\System32\OpenSSH;C:\Program Files\Apache-maven-3.8.3\bin;C:\Program Files (x86)\PuTTY\;C:\ProgramData\chocolatey\bin;C:\Program Files\MongoDB\Server\4.2\bin;C:\Program Files\Git\cmd;C:\Program Files\Apache24\bin;C:\Program Files\Microsoft VS Code\bin;C:\Program Files (x86)\NetSarang\Xshell 7\;C:\Program Files\Docker\docker\;C:\ProgramData\Decker\Decker\app\version-bin;C:\Program Files\MySQL\MySQL Server 5.7\bin;C:\Program Files\apache-tomcat-10.0.12\bin;::\Users\devex\AppData\Local\Microsoft\WindowsApps;C:\Program Files\Bnd\zip\;C:\Program Files\chocolatey\bin
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY;.PYW
```

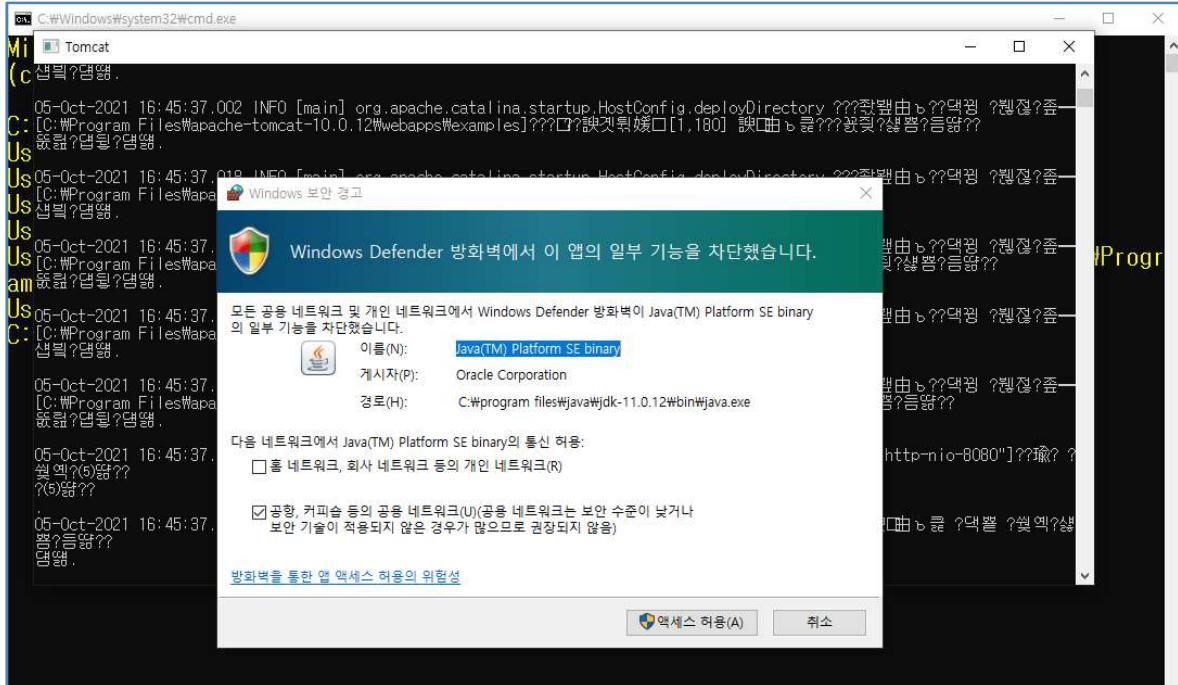
8. Tomcat Service Start

- 1) %CATALINA_HOME%\bin에서 Tomcat Startup & Shutdown

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\instructor>startup.bat
Using CATALINA_BASE:  "C:\Program Files\apache-tomcat-10.0.12"
Using CATALINA_HOME:  "C:\Program Files\apache-tomcat-10.0.12"
Using CATALINA_TMPDIR: "C:\Program Files\apache-tomcat-10.0.12\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk-11.0.12"
Using CLASSPATH:       "C:\Program Files\apache-tomcat-10.0.12\bin\bootstrap.jar;C:\Program Files\apache-tomcat-10.0.12\bin\tomcat-juli.jar"
Using CATALINA_OPTS:   ""
C:\Users\instructor>

C:\Windows\system32\cmd.exe
C:\Users\instructor>shutdown.bat
Using CATALINA_BASE:  "C:\Program Files\apache-tomcat-10.0.12"
Using CATALINA_HOME:  "C:\Program Files\apache-tomcat-10.0.12"
Using CATALINA_TMPDIR: "C:\Program Files\apache-tomcat-10.0.12\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk-11.0.12"
Using CLASSPATH:       "C:\Program Files\apache-tomcat-10.0.12\bin\bootstrap.jar;C:\Program Files\apache-tomcat-10.0.12\bin\tomcat-juli.jar"
Using CATALINA_OPTS:   ""
NOTE: Picked up JDK_JAVA_OPTIONS: --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.base/java.util.concurrent=ALL-UNNAMED --add-opens=java.rmi/sun.rmi.transport=ALL-UNNAMED
C:\Users\instructor>
```



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

The screenshot shows the Apache Tomcat 10.0.12 homepage. At the top, there's a navigation bar with links to Home, Documentation, Configuration, Examples, Wiki, and Mailing Lists. On the right, there's a "Find Help" button. Below the navigation is the Apache logo and the text "APACHE SOFTWARE FOUNDATION". A green banner at the top says "If you're seeing this, you've successfully installed Tomcat. Congratulations!". To the left of the banner is a cartoon cat icon. To the right are three buttons: "Server Status", "Manager App", and "Host Manager". The main content area has a light blue background. It features sections for "Developer Quick Start" with links to Tomcat Setup, First Web Application, Realms & AAA, JDBC Data Sources, Examples, and Servlet Specifications/Tomcat Versions. Below this are three yellow-bordered boxes: "Managing Tomcat" (with links to manager webapp info, tomcat-users.xml, and a note about split access), "Documentation" (with links to Tomcat 10.0 Documentation, Configuration, and Wiki), and "Getting Help" (with links to FAQ and Mailing Lists, and information about mailing lists like tomcat-announce, tomcat-users, tomcat-dev, taglibs-user, and taglibs-dev).

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" />
```

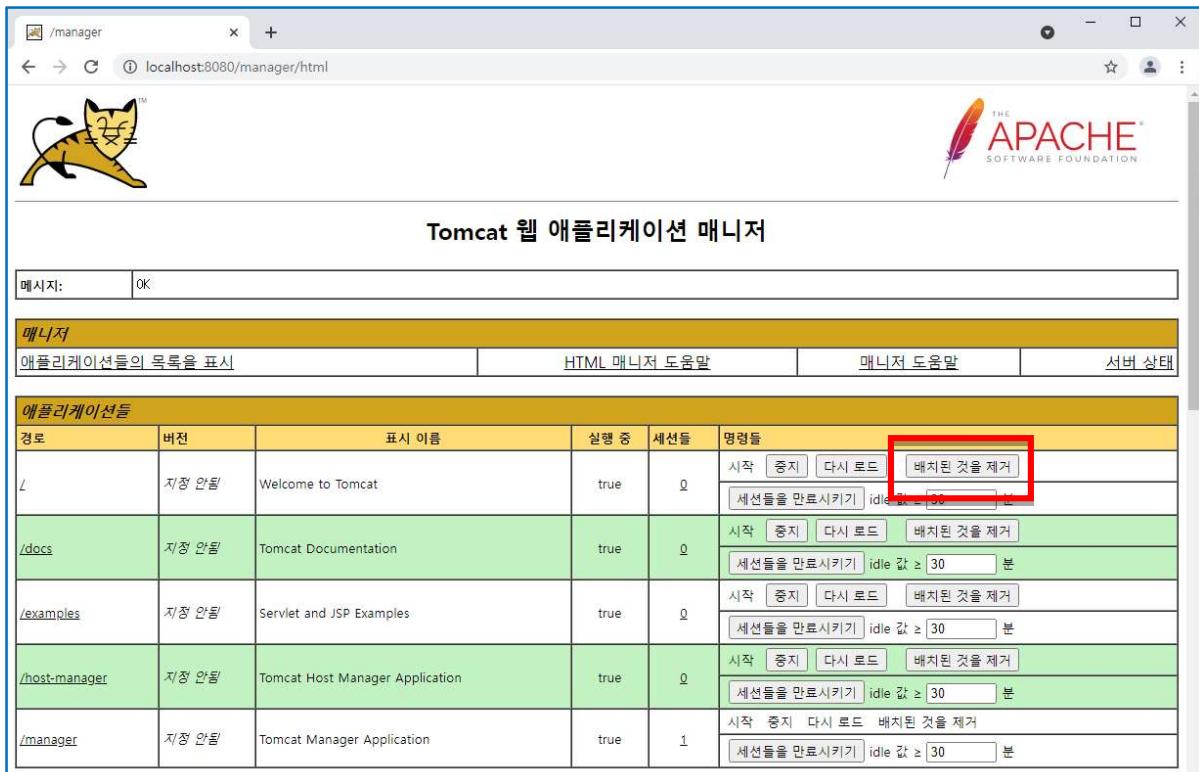
- %CATALINA_HOME%/webapps/ROOT/WEB-INF 복사하여 C:/SpringHome 하위에 붙여넣기
- C:/SpringHome/WEB-INF/web.xml 수정

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
5     https://jakarta.ee/xml/ns/jakartaee/web-app_5_0.xsd"
6   version="5.0"
7   metadata-complete="true">
8
9   <display-name>Welcome to Spring Homepage</display-name>
10  </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>
9       <span style="font-size:30pt;color:blue">Welcome to Spring 5</span>
10    </div>
11  </body>
12 </html>
```

- [Tomcat Web Application Manager]에서 / Path Undeploy



The screenshot shows the Apache Tomcat Web Application Manager interface. At the top, there's a navigation bar with links for Home, Applications, Hosts, and Global Properties. Below the navigation is a search bar and a user icon. The main content area has a title "Tomcat 웹 애플리케이션 매니저". A message box says "메시지: OK". Below it is a toolbar with buttons for "매니저", "애플리케이션들의 목록을 표시", "HTML 매니저 도움말", "매니저 도움말", and "서버 상태". The main table lists applications with columns: 경로 (Path), 버전 (Version), 표시 이름 (Display Name), 실행 중 (Running), 세션들 (Sessions), and 명령들 (Commands). One row for "/manager" is selected and highlighted with a red box around the "배치된 것을 제거" (Remove) button in the Commands column.

경로	버전	표시 이름	실행 중	세션들	명령들
/	지정 안됨	Welcome to Tomcat	true	0	<button>시작</button> <button>중지</button> <button>다시 로드</button> <button>배치된 것을 제거</button> <input type="button" value="세션들을 만료시키기"/> idle 값 > 30 분
/docs	지정 안됨	Tomcat Documentation	true	0	<button>시작</button> <button>중지</button> <button>다시 로드</button> <button>배치된 것을 제거</button> <input type="button" value="세션들을 만료시키기"/> idle 값 > 30 분
/examples	지정 안됨	Servlet and JSP Examples	true	0	<button>시작</button> <button>중지</button> <button>다시 로드</button> <button>배치된 것을 제거</button> <input type="button" value="세션들을 만료시키기"/> idle 값 > 30 분
/host-manager	지정 안됨	Tomcat Host Manager Application	true	0	<button>시작</button> <button>중지</button> <button>다시 로드</button> <button>배치된 것을 제거</button> <input type="button" value="세션들을 만료시키기"/> idle 값 > 30 분
/manager	지정 안됨	Tomcat Manager Application	true	1	<button>시작</button> <button>중지</button> <button>다시 로드</button> <button>배치된 것을 제거</button> <input type="button" value="세션들을 만료시키기"/> idle 값 > 30 분

- [Tomcat Web Application Manager]에서 / Path deploy
 - Context path : /
 - XML Configuration file path : C:\Program Files\Apache-tomcat-10.0.12\webapps\homecontext.xml



The screenshot shows the 'Tomcat 웹 애플리케이션 매니저' (Tomcat Web Application Manager) interface. The application list table has a header row: 경로 (Path), 버전 (Version), 표시 이름 (Display Name), 실행 중 (Running), 세션들 (Sessions), and 명령들 (Commands). The first row, representing the root context ('/'), is highlighted with a red box. Its display name is 'Welcome to Spring Homepage'. The 'Commands' column for this row contains buttons for '시작' (Start), '중지' (Stop), '다시 로드' (Reload), and '배치된 것을 제거' (Remove). Below the table, a message box shows 'OK - 컨텍스트 경로 [/]에 애플리케이션을 배치했습니다.' (The application was deployed successfully at context path [/]).

The screenshot shows a web browser window titled 'Welcome to Spring Homepage' with the URL 'localhost:8080'. The page content is 'Welcome to Spring 5', indicating the application is running correctly.

Task 6. STS 4.12.0.RELEASE Installation and Configuration

1. STS(Spring Tools Suite) 소개

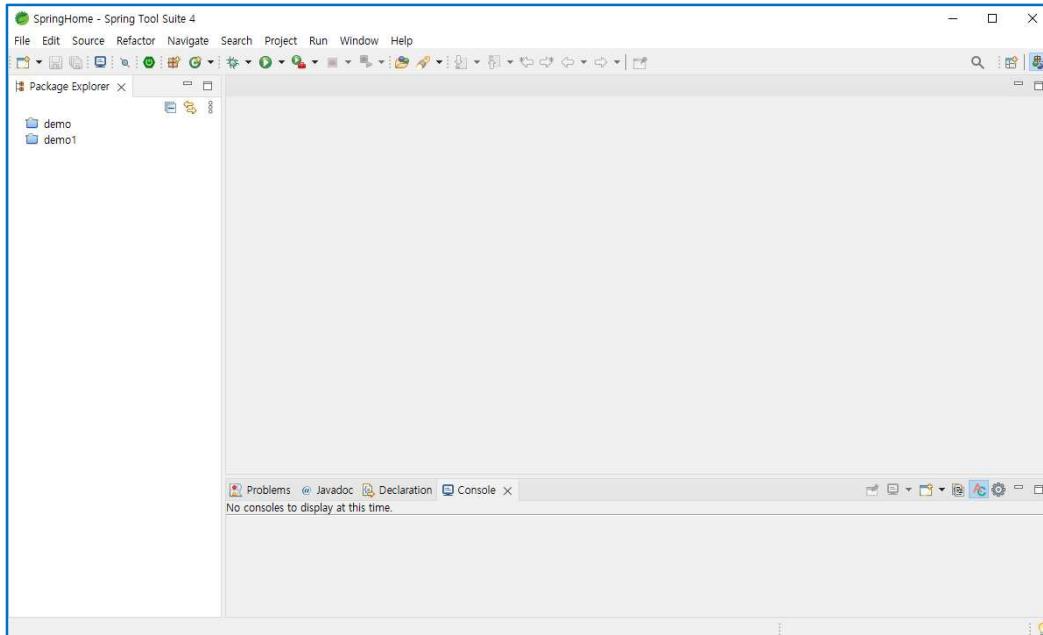
- Spring 개발업체인 SpringSource 가 직접 만들어 제공하는 Eclipse 의 확장판으로 최신 Eclipse 를 기반으로 주요한 Spring 지원 Plug-in 과 관련된 도구를 모아서 Spring 개발에 최적화되도록 만들어진 IDE 이다.
- STS 가 제공하는 기능
 - Bean class 이름 자동완성
 - 현재 project 의 모든 source 와 library, JDK 안의 모든 클래스 중에서 첫 글자가 SDD 로 시작하는 class 를 자동으로 보여줌
 - 설정 file 생성 wizard
 - Bean 설정 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-4-4.12.0.RELEASE-e4.21.0-win32.win32.x86_64.self-extracting.jar

3. STS 시작하기

- Download 받은 spring-tool-suite-4-4.12.0.RELEASE-e4.21.0-win32.win32.x86_64.self-extracting.jar 파일을 double click 하여 압축 푼다. 또는 Command 에서 다음의 명령으로 압축을 푼다.
`$ java -jar spring-tool-suite-4-4.12.0.RELEASE-e4.21.0-win32.win32.x86_64.self-extracting.jar`
- 압축이 풀리면 sts-4.12.0.RELEASE 폴더가 만들어진다.
- 이 폴더를 Cut 해서 C:\Program Files\에 붙인다.
- 바탕화면에 Shortcut 생성을 통해 link 를 C:\Program Files\sts-4.12.0.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-11.0.12/bin/javaw.exe

```
+SpringToolSuite4.ini - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
-startup
plugins/org.eclipse.equinox.launcher_1.6.300.v20210813-1054.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.2.300.v20210828-0802
-product
org.springframework.boot.ide.branding.sts4
--launcher.defaultAction
openFile
-vm
C:/Program Files/Java/jdk-11.0.12/bin/javaw.exe
plugins/org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149/jre/bin
-vmargs
-Dosgi.requiredJavaVersion=11
-Dosgi.dataAreaRequiresExplicitInit=true
-Xms1024m
-Xmx2048m
--illegal-access=permit
--add-modules=ALL-SYSTEM
-javaagent:C:\Program Files\sts-4.12.0.RELEASE\lombok.jar
```

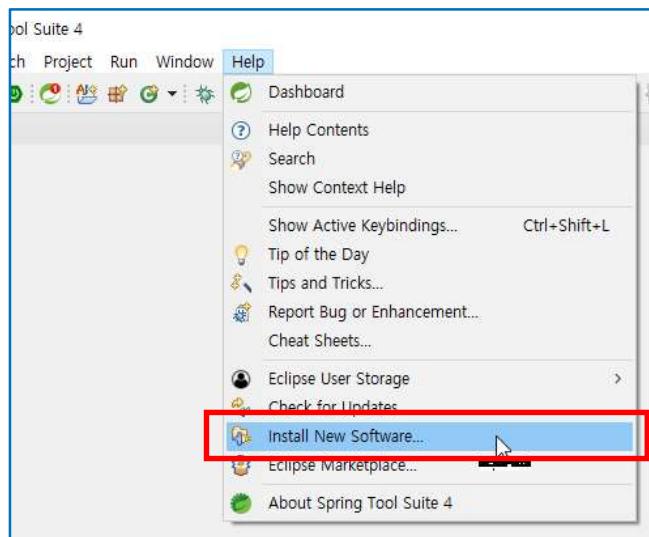
- 파일에 '-vm'이라는 옵션을 설정하는데, 내용은 JDK 경로내의 bin folder 의 javaw.exe 를 지정하는 것이다.
- 또한 실행 메모리가 너무 작을 경우 Xms 를 원하는 크기로 변경한다.

-Xms 1024m

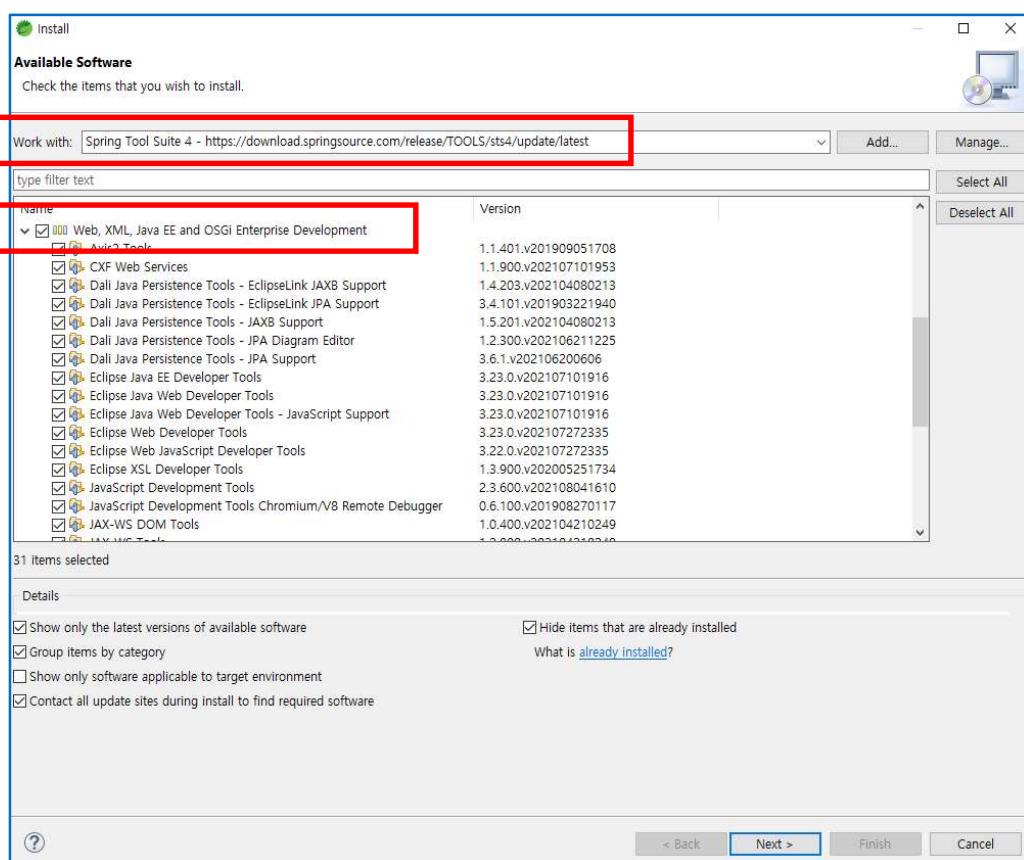
- STS 실행 환경 편집 후 바로가기 아이콘의 문제가 발생할 수 있으므로, 반드시 기존의 '바로가기' 아이콘은 삭제 후 다시 '바로가기'를 추가하는 것이 좋다.

5. Web Install

- 기본적으로 최신 STS 를 설치하면 Web 이 설치되어 있지 않다.
- 그래서 추가적으로 Web 을 설치해야 한다.
- 설치는 Help > Install new Software...

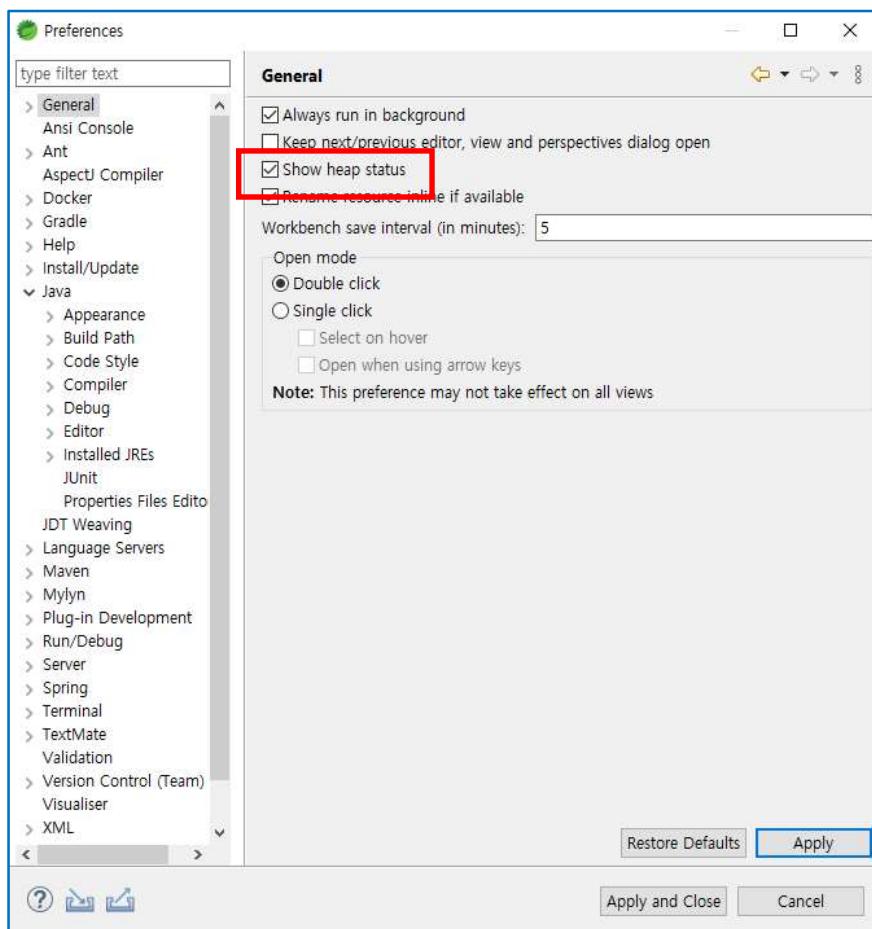


- [Install] 창의 [Available Software] > Work with : 에 Spring Tool Suite 4 – <https://download.springsource.com/release/TOOLS/sts3/update/latest> 를 선택한다.
- 잠시 Pending 후, 리스트에서 Web, XML, Java EE and OSGi Enterprise Development 를 체크한다.

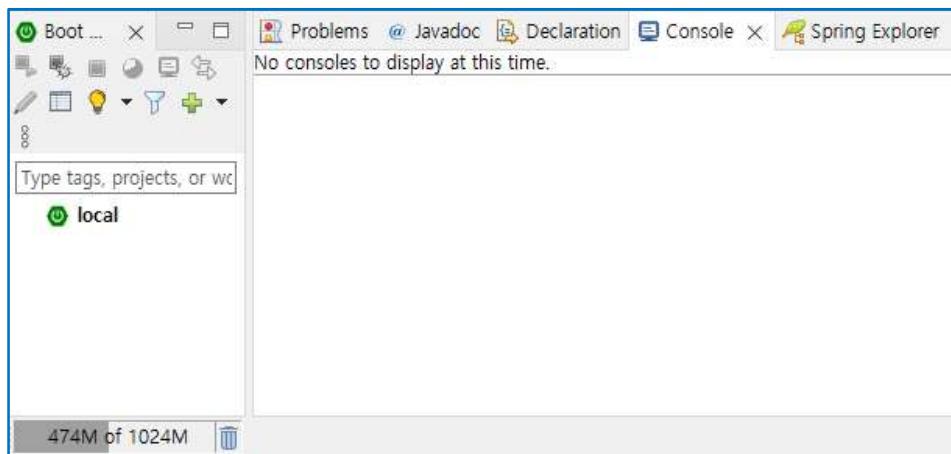


6. Heap 상태 보기

- Heap 메모리 상태를 보기 위한 뷰를 추가한다.
- Preferences > General > Show heap status 를 체크한다.

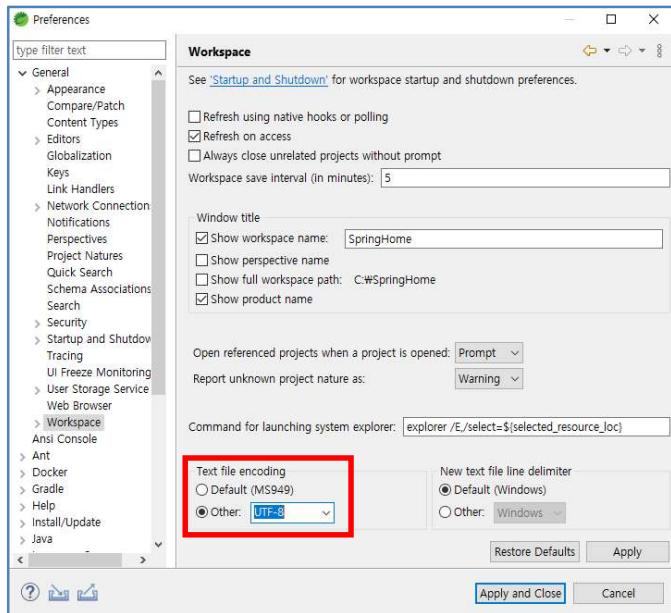


- STS 하단에 heap status 가 생성된 것을 확인할 수 있다.



7. 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

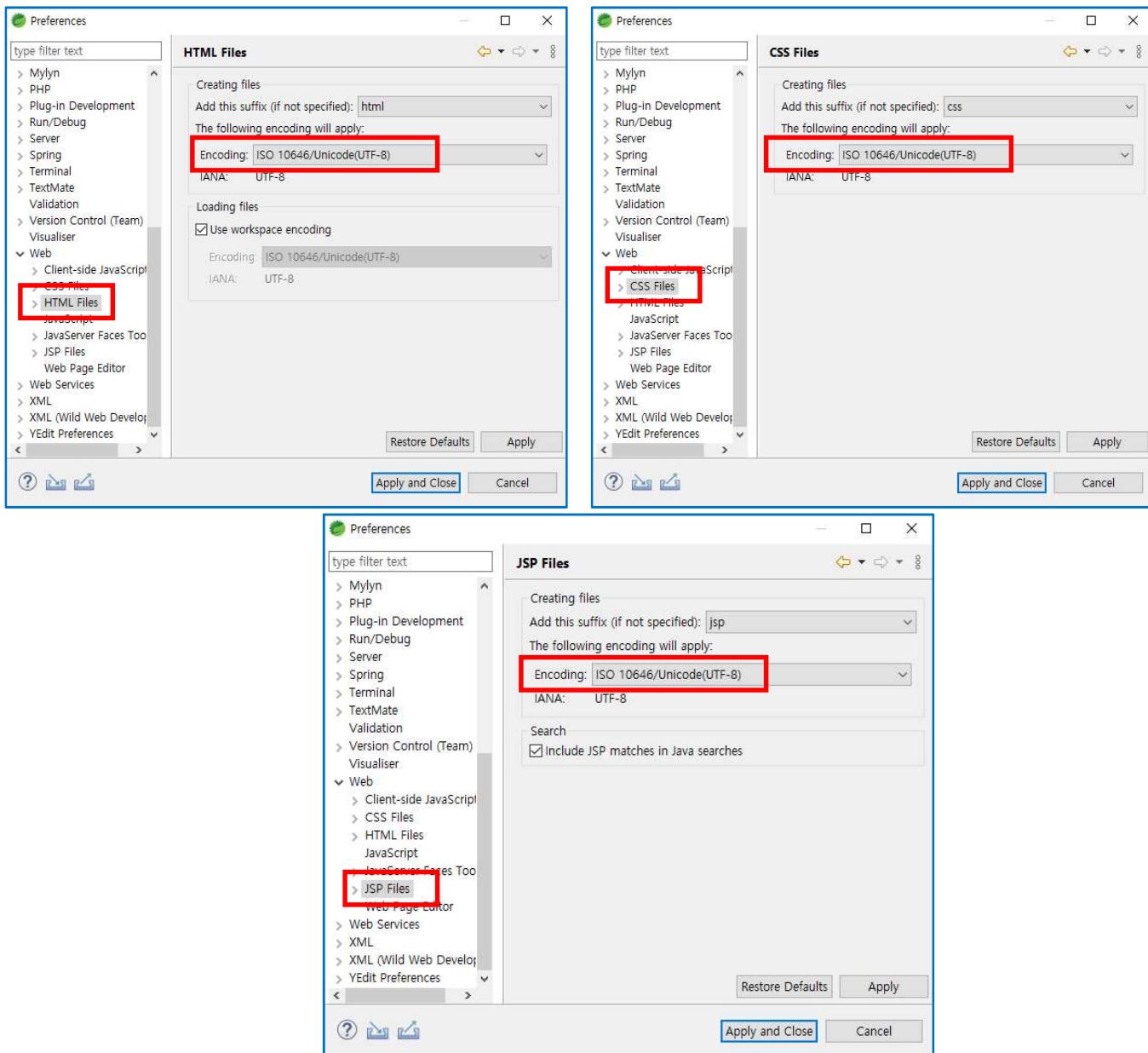


- General > Content Types > Text 와 Java Properties File 그리고 Spring Properties File 도 Default encoding의 값도 UTF-8로 설정하고 [Update] 버튼을 클릭한다.

The screenshots show the 'Content Types' preferences page for three different file types, each with its 'Default encoding' set to 'UTF-8'. The 'Update' button in each dialog is highlighted with a red box.

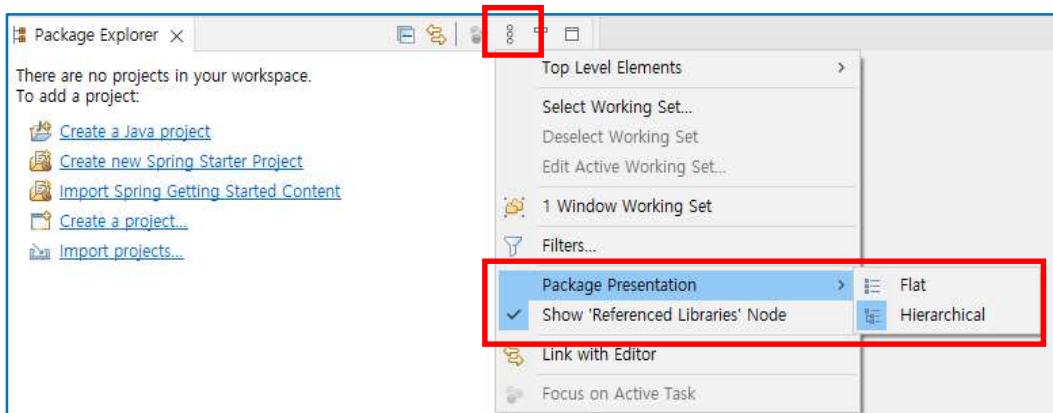
- Text:** Shows 'Text' selected in the content types list. The 'Default encoding' field contains 'UTF-8'.
- Java Properties File:** Shows 'Java Properties File' selected in the content types list. The 'Default encoding' field contains 'UTF-8'.
- Spring Properties File:** Shows 'Spring Properties File' selected in the content types list. The 'Default encoding' field contains 'UTF-8'.

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



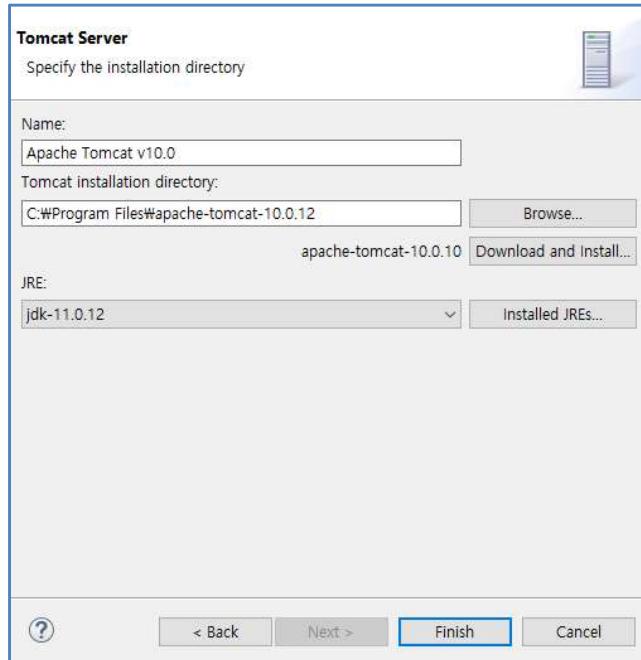
8. 패키지 구조 변경하기

- 일반적으로 STS에서 패키지를 보여줄 때에는 계층적으로 보여지지 않는다.
- Package Explorer의 메뉴에서 Package Presentation을 Flat에서 Hierarchical로 변경한다.

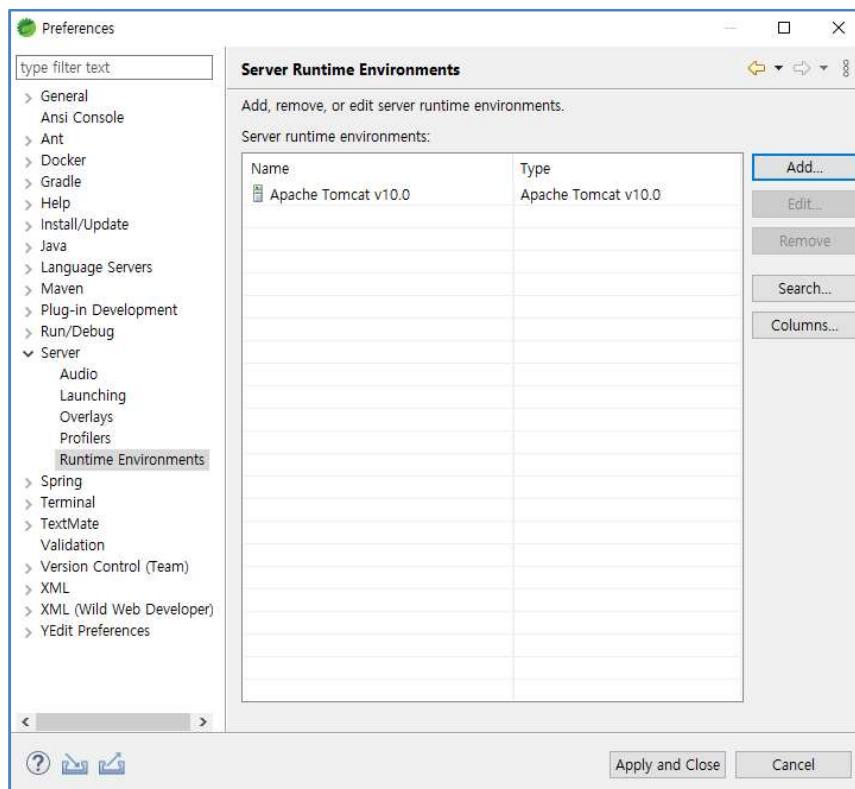


9. STS 에 Tomcat 등록하기

- Window > Preferences > Server > Runtime Environments에서 [Add] 버튼을 click 한다.
- Apache > Apache Tomcat v10.0 > Next
- Tomcat installation directory : C:\Program Files\apache-tomcat-10.0.12
- JRE:jdk-11.0.12

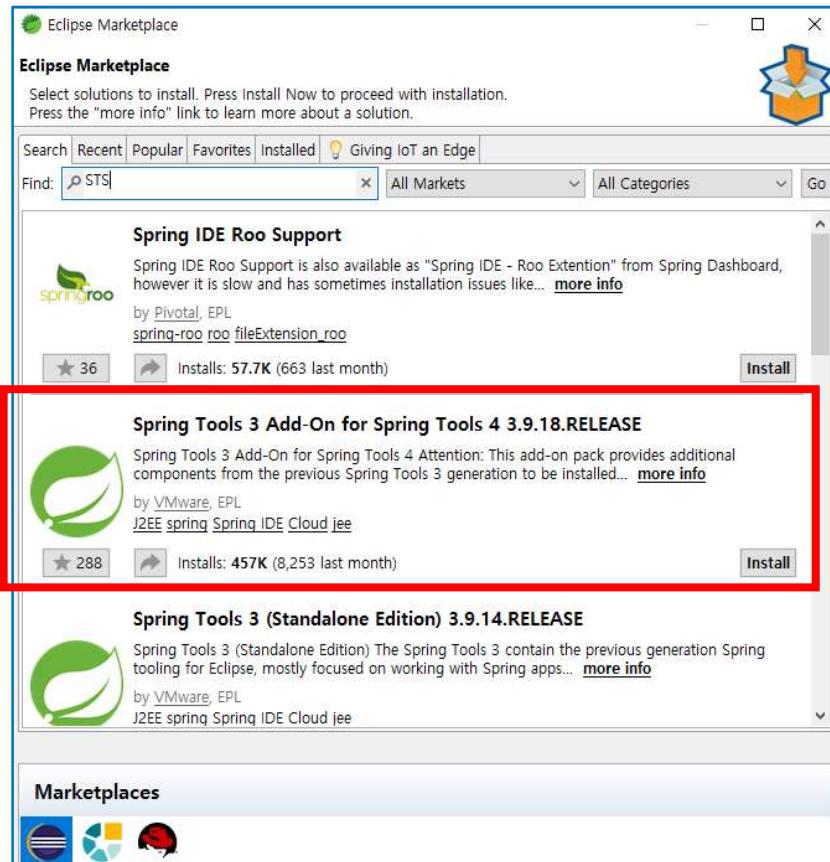


- Finish

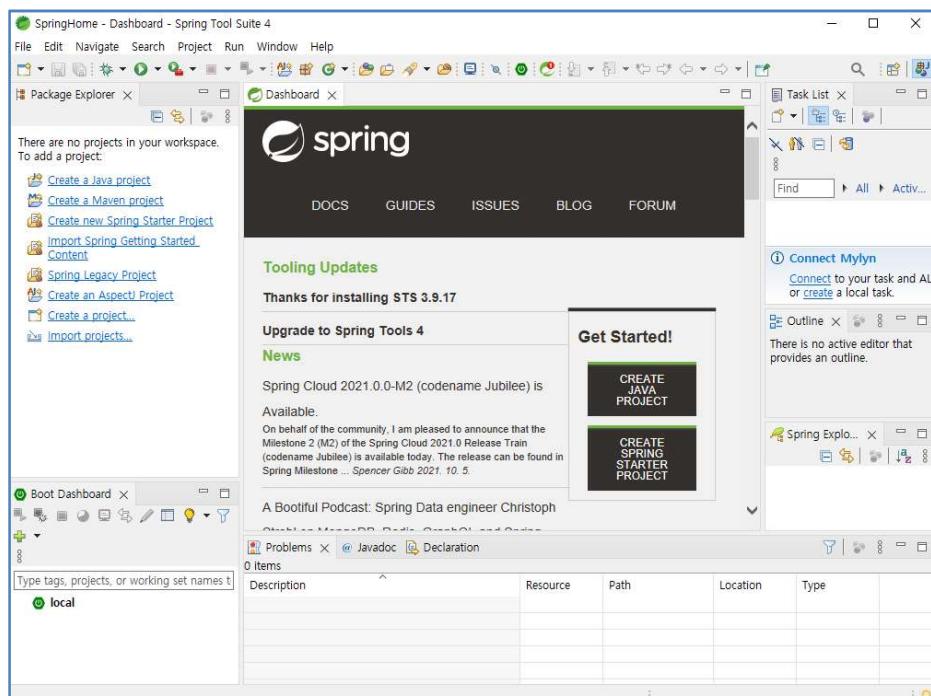


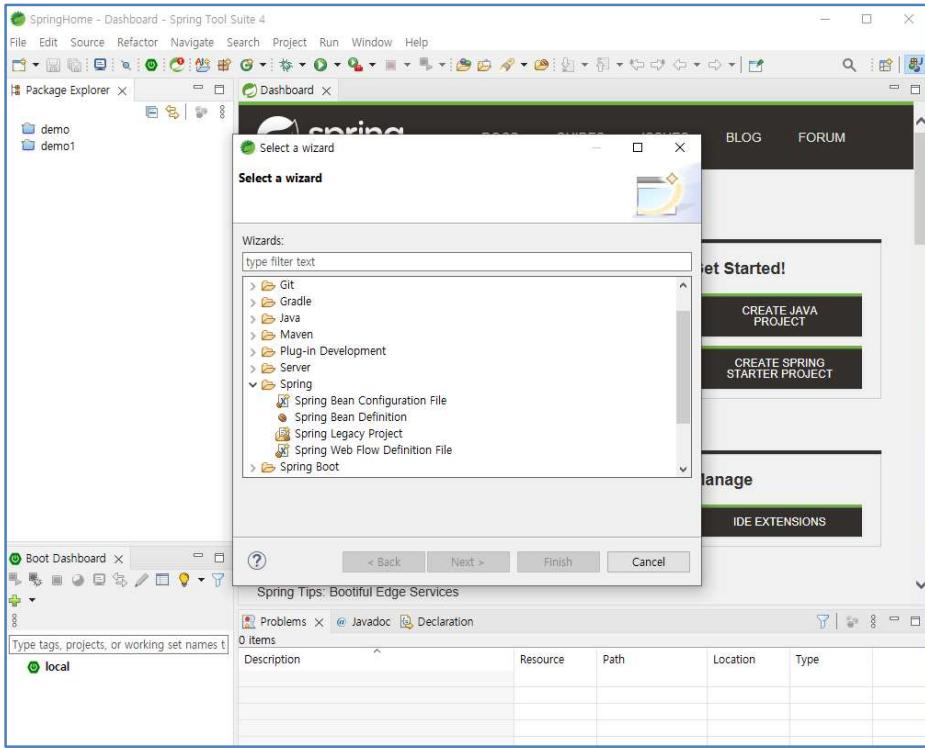
10. STS4에 Spring Legacy Project 생성을 위한 STS3 Add-On 설치하기

- “Spring Legacy Project”는 STS3 까지만 제공되고 STS4에서는 Spring Boot 가 기본이기 때문에 STS3 Add-On 해야 한다.
- STS4 메뉴 > Help > [Eclipse Marketplace...] > “STS”로 검색 > [Spring Tools 3 Add-On for Spring Tools 4 3.9.18.RELEASE] 항목의 [Install] 버튼 Click



- 설치 후 STS4 Restart





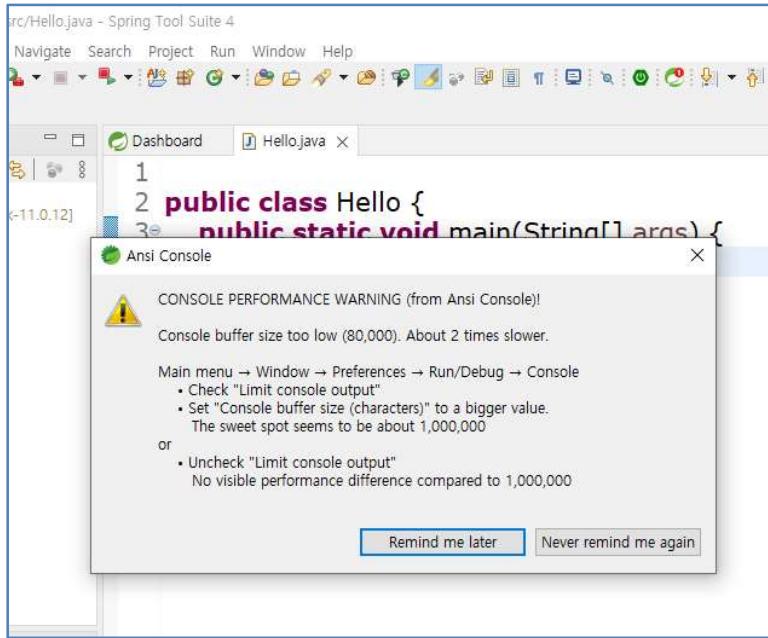
11. STS 로 간단한 Maven Project 만들기

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
- 아래 그림은 가상 머신에서 실행시 Console 의 buffer size 가 너무 작은 경고가 발생할 수 있다. 이럴 경우 다음의 설명대로 Limit 를 제거하면 된다.



2) Maven Project로 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 Project로 전환

- HelloWorld Project > right-click > Spring > Add Spring Project Nature
- <http://mvnrepository.com>에서 'spring context' 검색
- 이 문서를 작성하는 현재 버전은 5.3.10이다.
- 현재 버전의 Dependency를 복사한 다음 pom.xml에 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 -->

```

```

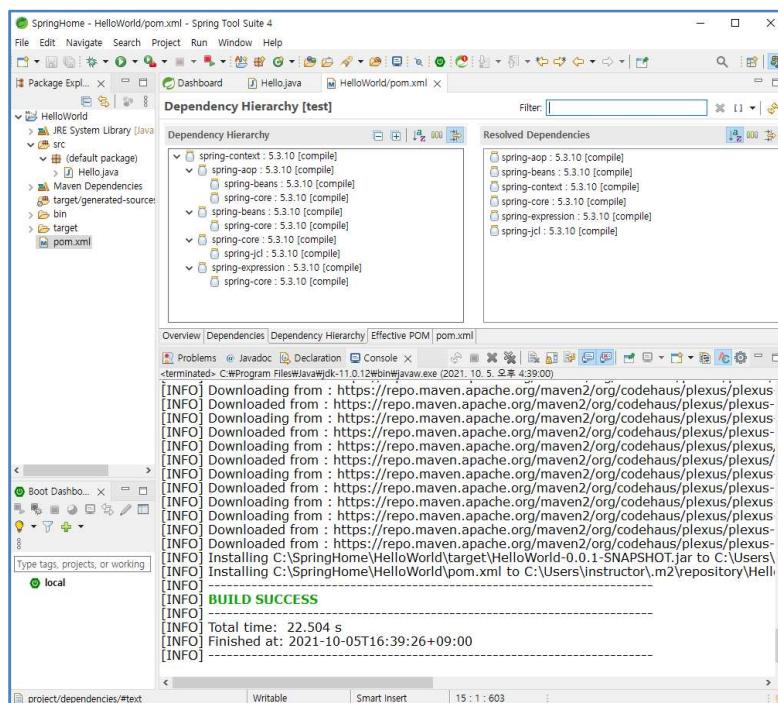
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.10</version>
</dependency>

</dependencies>

<build>
    <sourceDirectory>src</sourceDirectory>
    <plugins>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.1</version>
            <configuration>
                <release>11</release>
            </configuration>
        </plugin>
    </plugins>
</build>
</project>

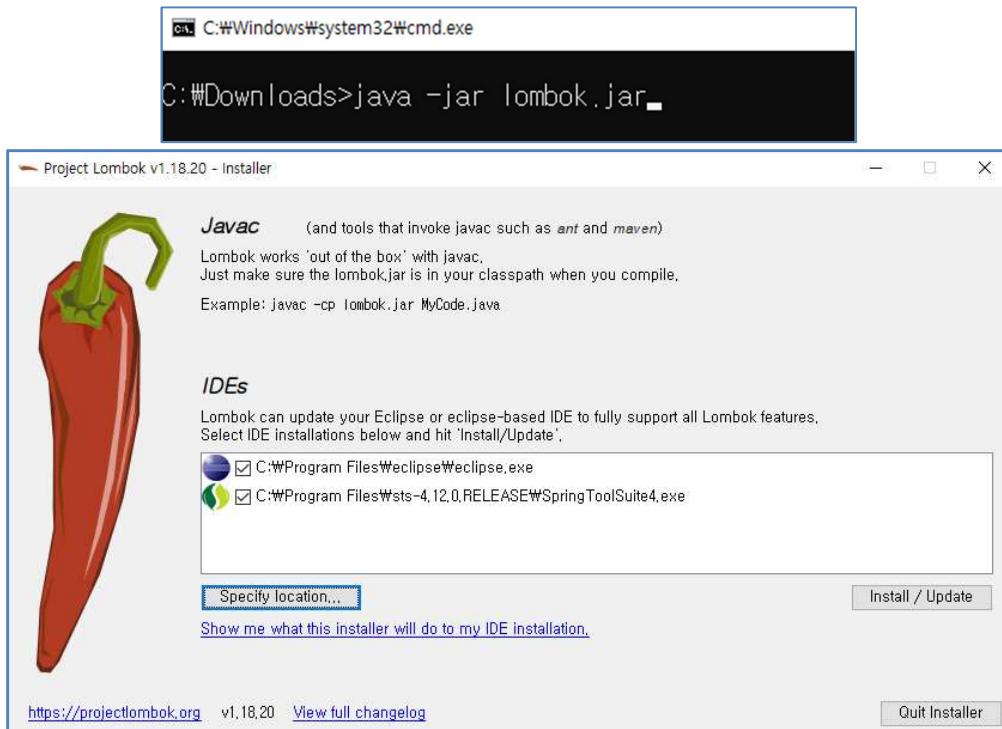
```

- pom.xml > right-click > Run As > Maven Install > BUILD SUCCESS <--at Console View
- Dependencies tab에서 spring-context : 5.3.10 설치된 것을 확인 함.

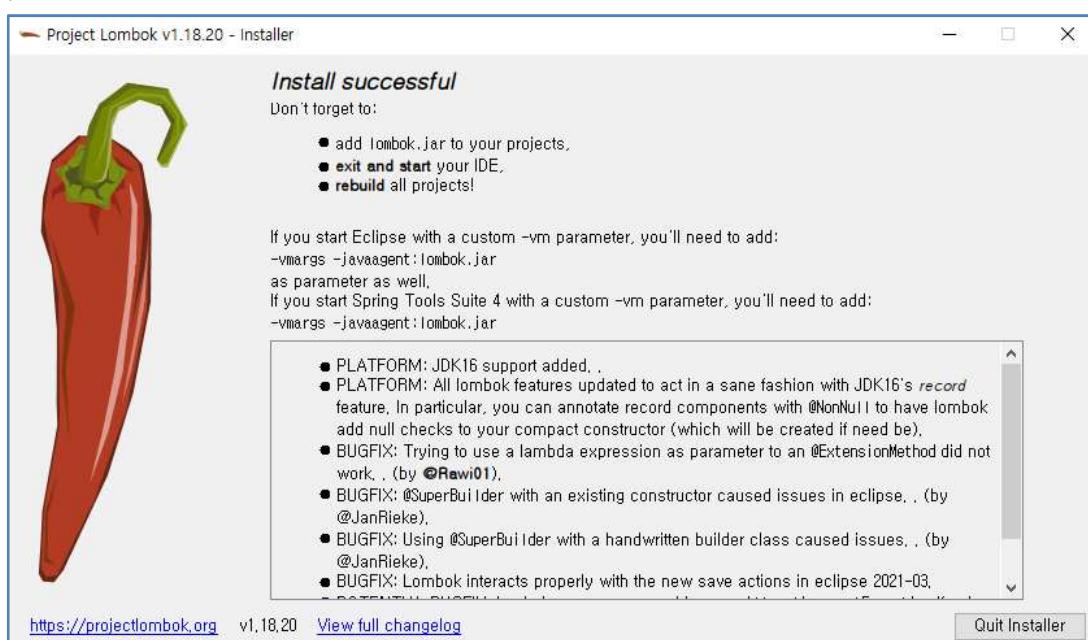


Task 7. Lombok 1.18.20 Library Installation

1. Java 개발할 때 많이 사용하는 getter/setter, `toString()`, 생성자 등을 자동으로 생성해 주는 Library이다.
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 1.18.20 <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



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.

