Spring MVC Setting

524730-1
2021년 봄학기
5/12/2021
박경신

Spring Framework

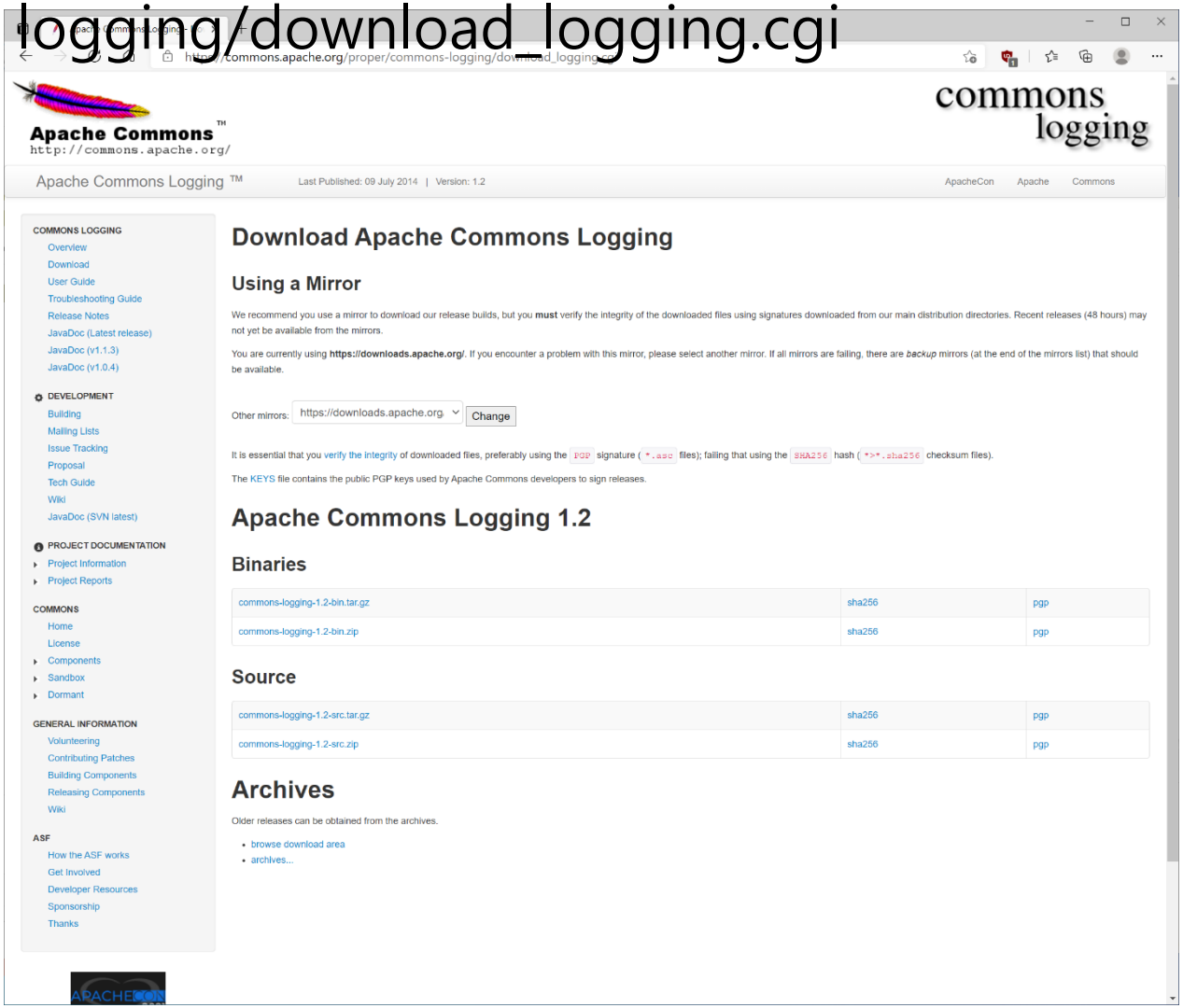
- Spring Framework
  - 웹 서버를 구현하기 위한 어플리케이션 프레임워크
  - IoC (Inversion of Control) 제어의 역행
    - IoC는 메소드나 객체의 호출을 개발자가 결정하는 것이 아니라 외부에서 결정되는 것을 의미
    - Spring IoC 컨테이너는 ApplicationContext 또는 BeanFactory 중 하나를 사용, Bean을 만들고, 의존성을 엮어주며 제공해주는 역할을 수행
  - DI (Dependency Injection) 의존성 주입을 통한 객체 간의 관계
    - IoC가 일어날 때 스프링 내부에 있는 객체들(Bean) 간의 관계를 컨테이너가 자동적으로 연결해주는 것
  - AOP (Aspect Oriented Programming) 관점 지향 프로그래밍
    - 시스템 보안, 로그, 트랙잭션 등 스프링에서는 이러한 '횡단 관심사'를 모듈로 분리해서 제작 가능. 공통 관심 사항을 프로그래밍하여 이 코드를 여러 코드에 짜 넣음(weave). AspectJ로 이미 사용되어 왔음.
  - POJO (Plan Old Java Object) 기반 구조
    - 객체 간의 관계 구성 시 POJO (즉, Bean)의 구성만으로 가능
  - MVC (Model-View-Controller) 구조

Spring MVC Setting

- Spring MVC Quick Start
  - OpenJDK 15.0.2 & Eclipse IDE 2020-12
  - Tomcat 9.0.43
    - Eclipse Server Setting: Eclipse -> Window -> Preferences -> Server -> Runtime Environments -> Tomcat9
  - Common Logging 1.2
    - https://commons.apache.org/proper/commons-logging/download_logging.cgi
    - Download commons-logging-1.2-bin.zip
  - Maven 3.6.3 프로젝트 관리도구
    - https://maven.apache.org/download.cgi
    - Download apache-maven-3.6.3-bin.zip
  - Spring Tool Suite (STS)
    - https://spring.io/tools/ Spring Tools for Eclipse
    - Install Eclipse->Help->"Eclipse Marketplace" -> STS -> Spring Tools 3.9.14.RELEASE

Commons Logging

- https://commons.apache.org/proper/commons-logging/download_logging.cgi

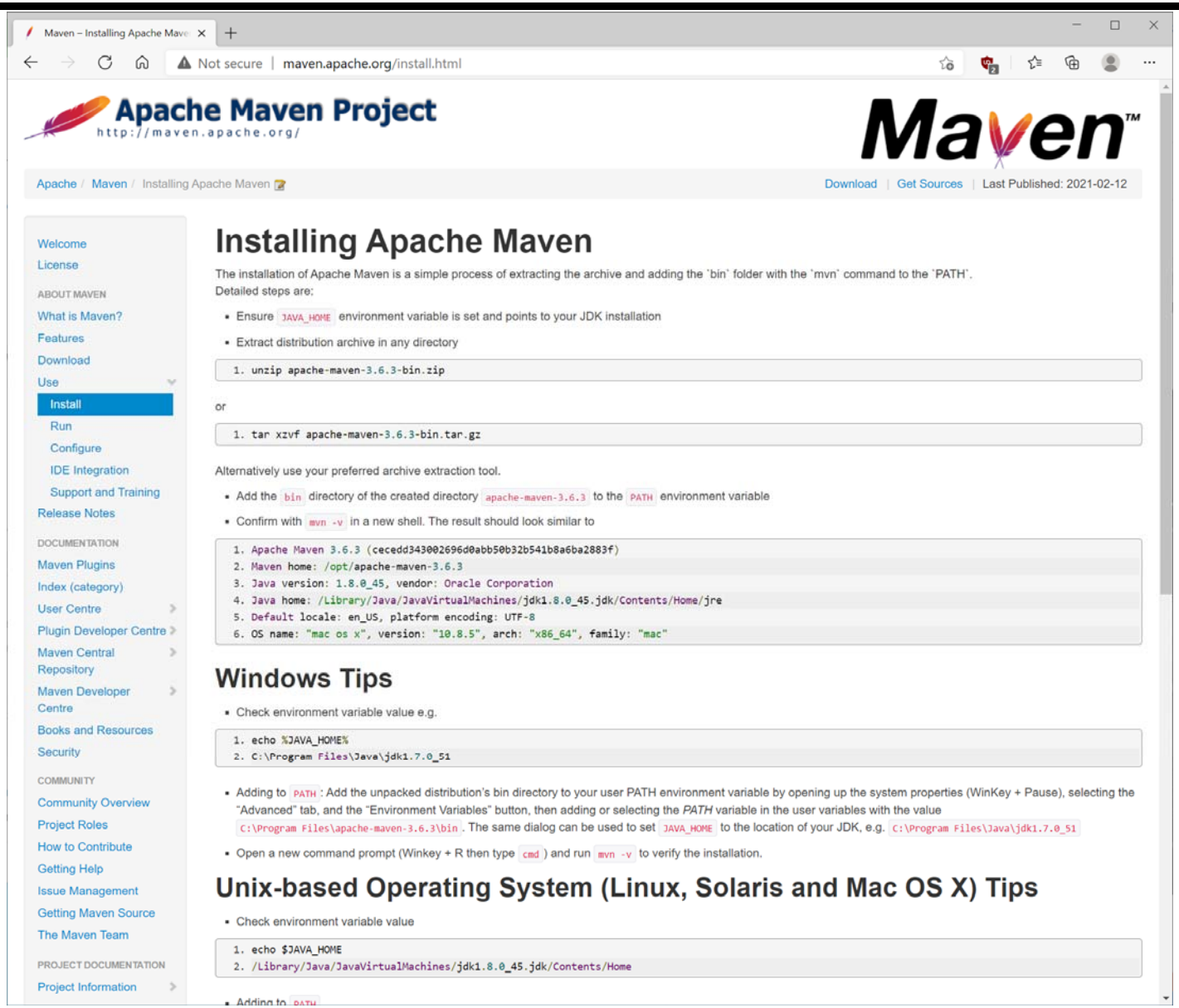


Maven

- https://maven.apache.org/download.cgi

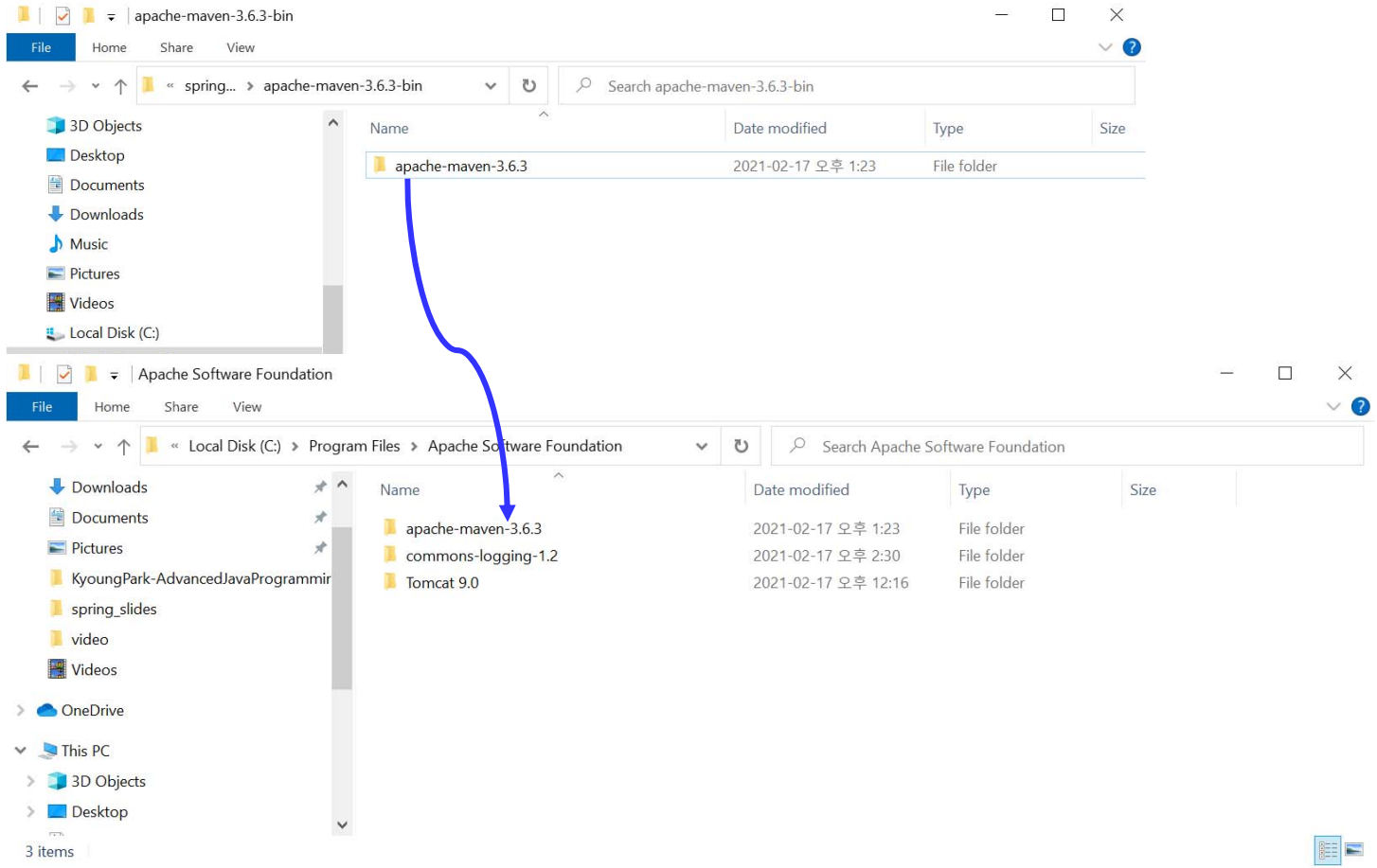
Maven

- Maven


Maven – Installing Apache Mave
Not secure | maven.apache.org/install.html
Apache Maven Project
http://maven.apache.org/
Maven™
Apache / Maven / Installing Apache Maven
Download | Get Sources | Last Published: 2021-02-12
Welcome
License
ABOUT MAVEN
What is Maven?
Features
Download
Use
Install
Run
Configure
IDE Integration
Support and Training
Release Notes
DOCUMENTATION
Maven Plugins
Index (category)
User Centre
Plugin Developer Centre
Maven Central Repository
Maven Developer Centre
Books and Resources
Security
COMMUNITY
Community Overview
Project Roles
How to Contribute
Getting Help
Issue Management
Getting Maven Source
The Maven Team
PROJECT DOCUMENTATION
Project Information
Installing Apache Maven
The installation of Apache Maven is a simple process of extracting the archive and adding the `bin` folder with the `mvn` command to the `PATH`.
Detailed steps are:
Ensure JAVA_HOME environment variable is set and points to your JDK installation
Extract distribution archive in any directory
1. unzip apache-maven-3.6.3-bin.zip
or
1. tar xzvf apache-maven-3.6.3-bin.tar.gz
Alternatively use your preferred archive extraction tool.
Add the bin directory of the created directory apache-maven-3.6.3 to the PATH environment variable
Confirm with mvn -v in a new shell. The result should look similar to
1. Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
2. Maven home: /opt/apache-maven-3.6.3
3. Java version: 1.8.0_45, vendor: Oracle Corporation
4. Java home: /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/jre
5. Default locale: en_US, platform encoding: UTF-8
6. OS name: "mac os x", version: "10.8.5", arch: "x86_64", family: "mac"
Windows Tips
Check environment variable value e.g.
1. echo %JAVA_HOME%
2. C:\Program Files\Java\jdk1.7.0_51
Adding to PATH: Add the unpacked distribution's bin directory to your user PATH environment variable by opening up the system properties (WinKey + Pause), selecting the "Advanced" tab, and the "Environment Variables" button, then adding or selecting the PATH variable in the user variables with the value C:\Program Files\apache-maven-3.6.3\bin. The same dialog can be used to set JAVA_HOME to the location of your JDK, e.g. C:\Program Files\Java\jdk1.7.0_51
Open a new command prompt (Winkey + R then type cmd) and run mvn -v to verify the installation.
Unix-based Operating System (Linux, Solaris and Mac OS X) Tips
Check environment variable value
1. echo $JAVA_HOME
2. /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home
Adding to PATH

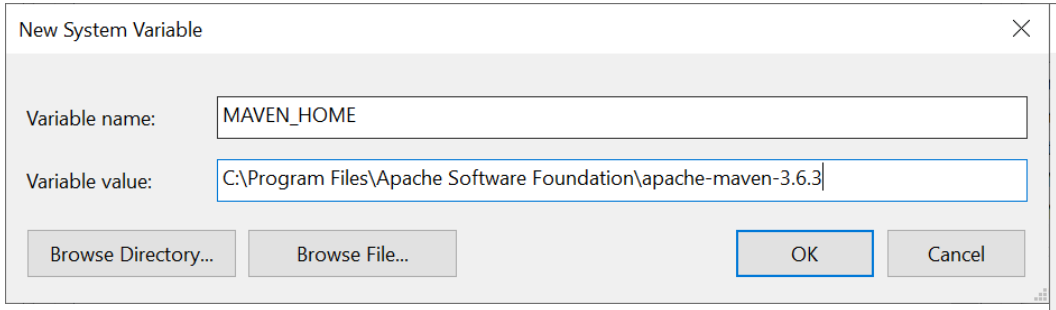
Maven

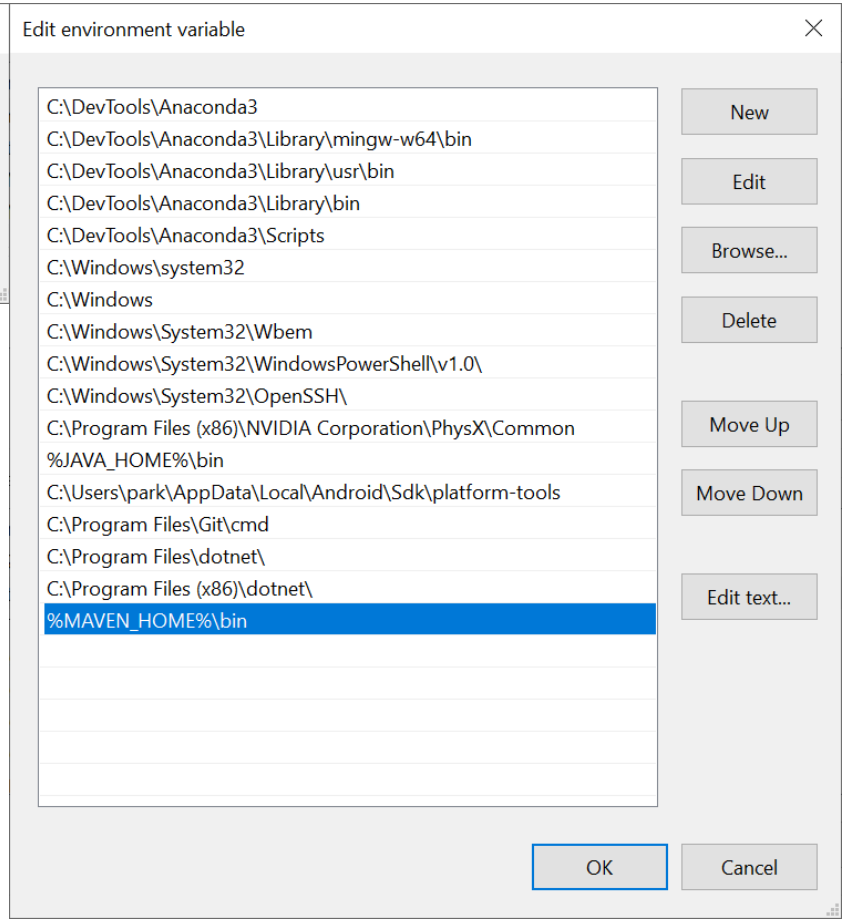
- Unzip apache-maven-3.6.3-bin.zip & Copy “apache-maven-3.6.3” to “C:₩Program Files₩Apache Software Foundation”



Maven

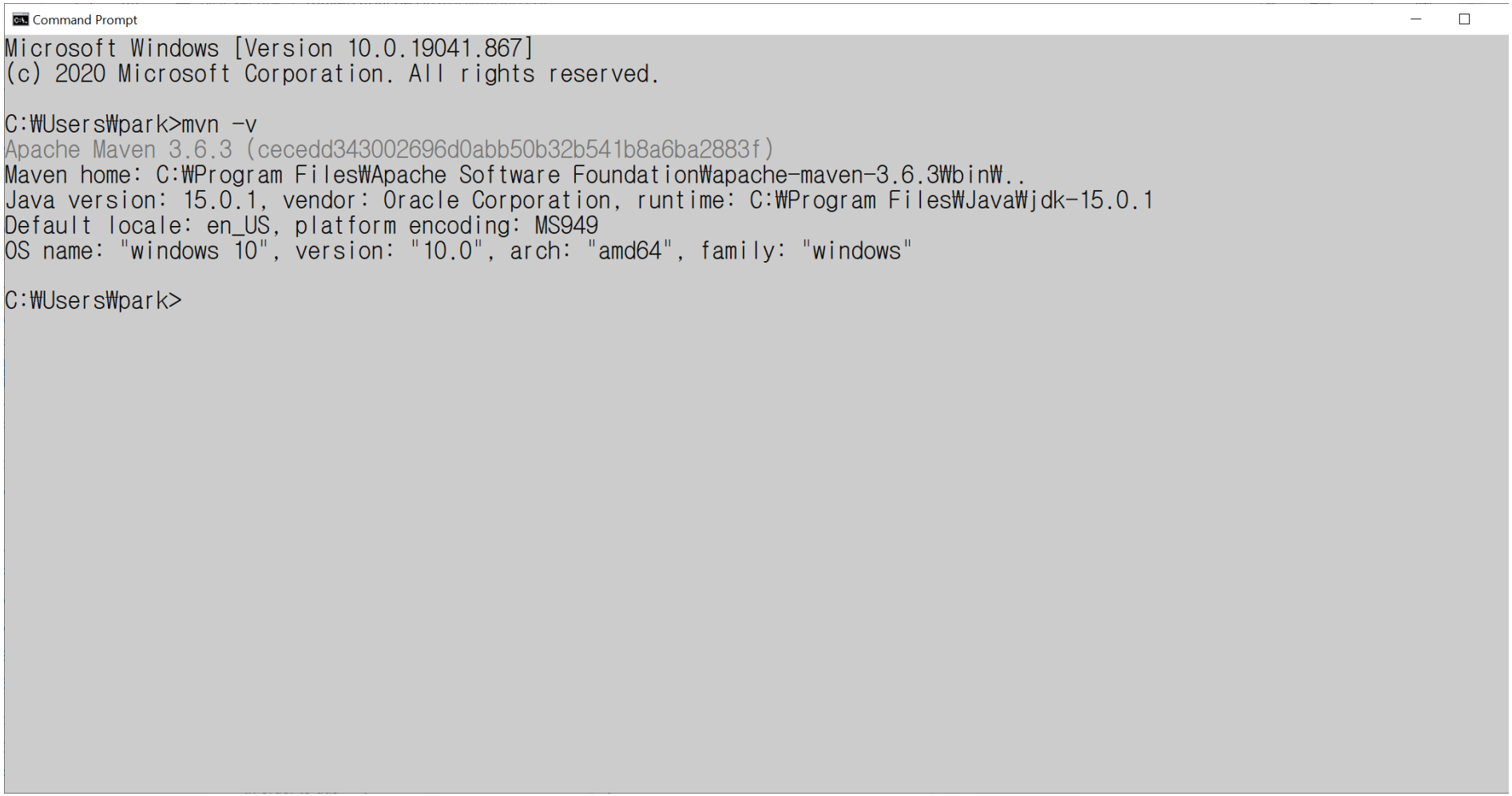
- Set path environment variable





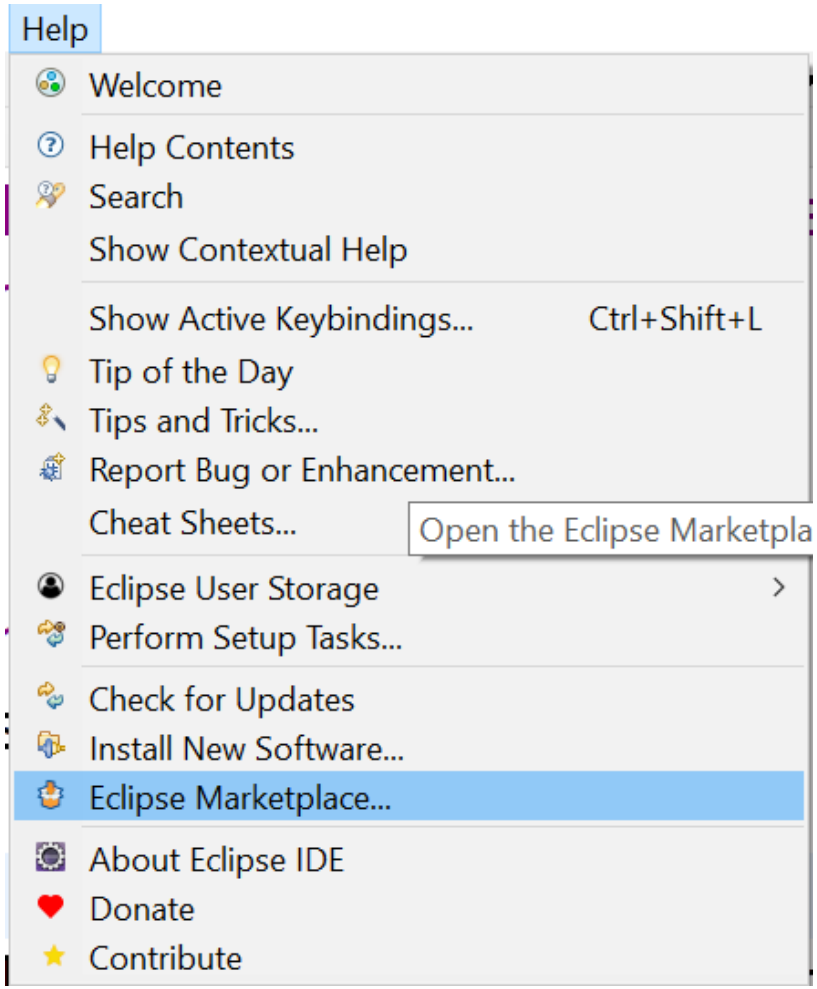
Maven

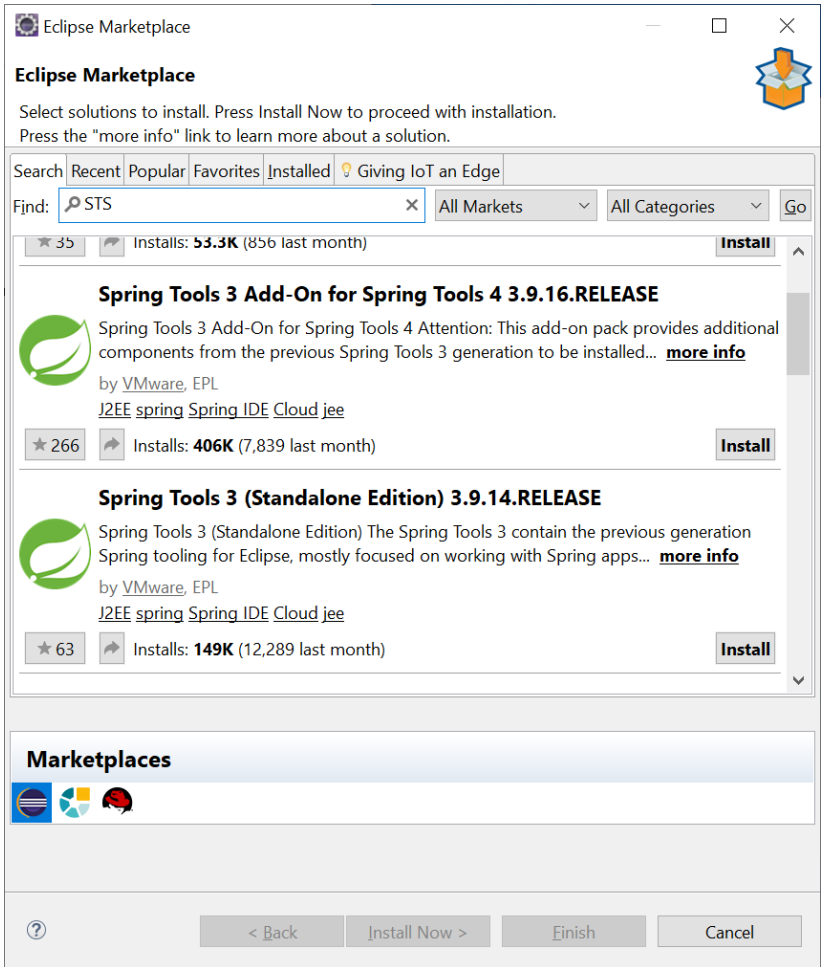
- Run “mvn –v”



Spring Tool Suite (STS)

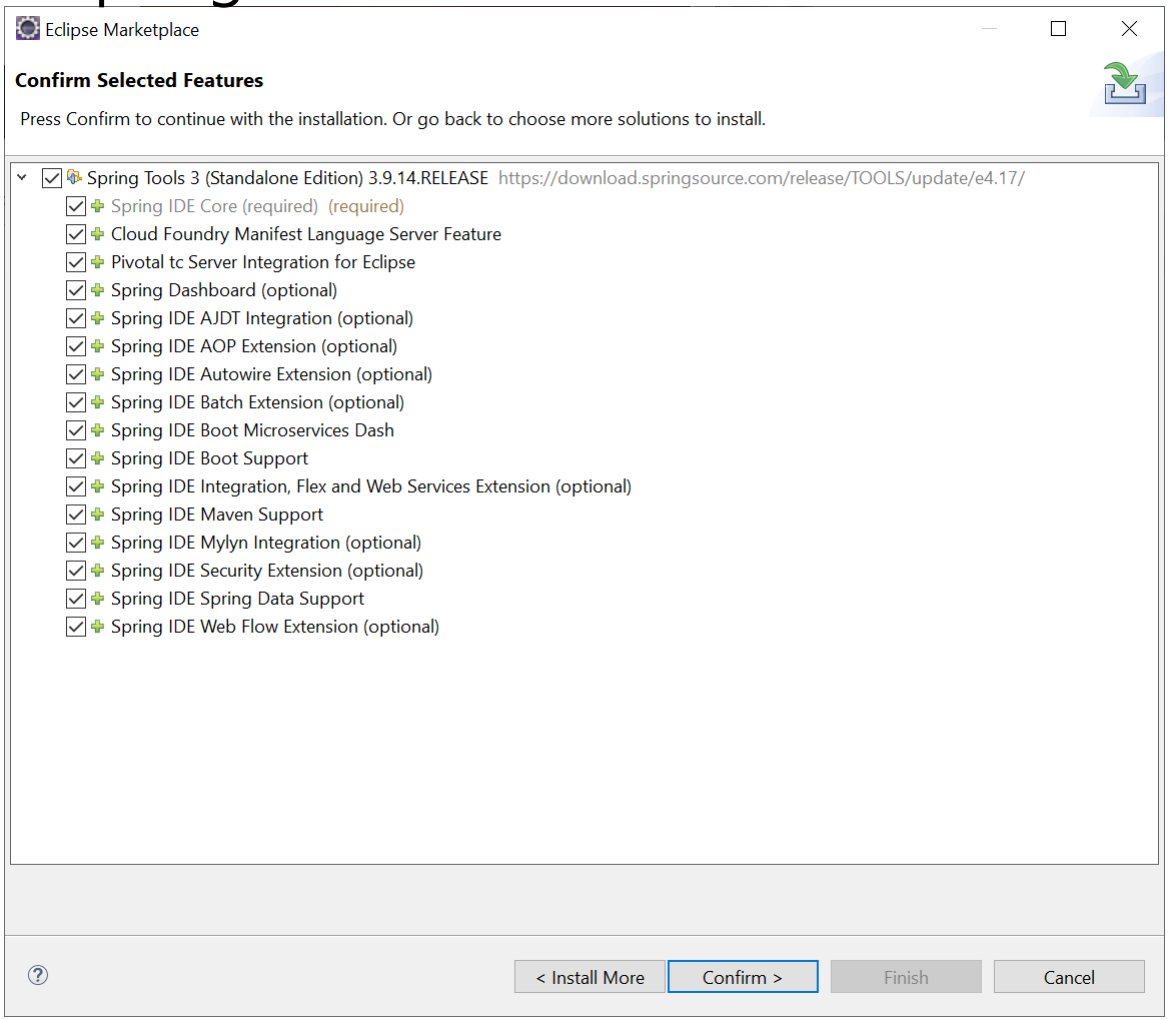
- Eclipse -> Help -> Eclipse Marketplace -> Find: STS -> Spring Tools 3.9.14.RELEASE->Install





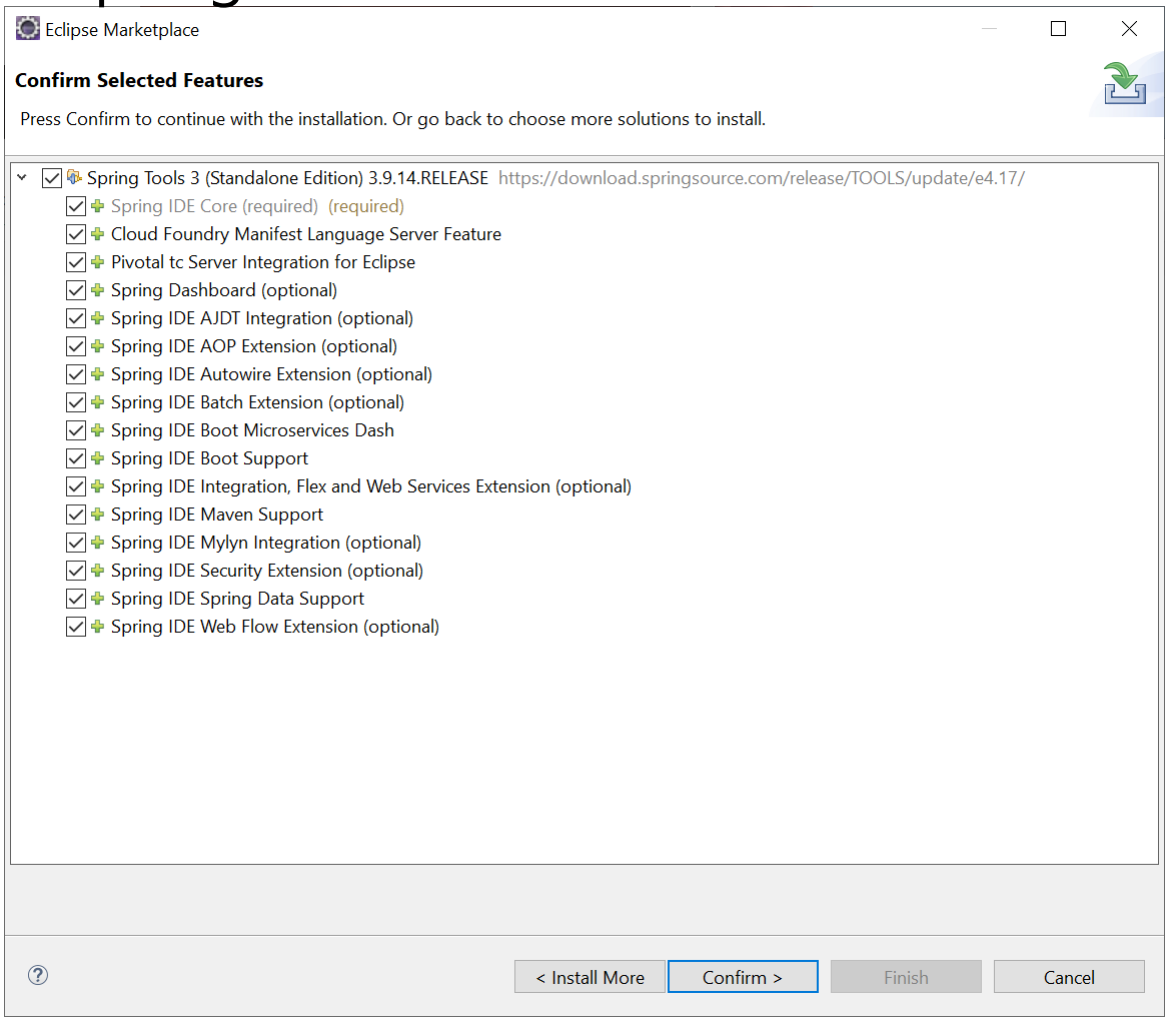
Spring Tool Suite (STS)

- Eclipse -> Help -> Eclipse Marketplace -> Find: STS -> Spring Tools 3.9.14.RELEASE->Install & Restart Eclipse



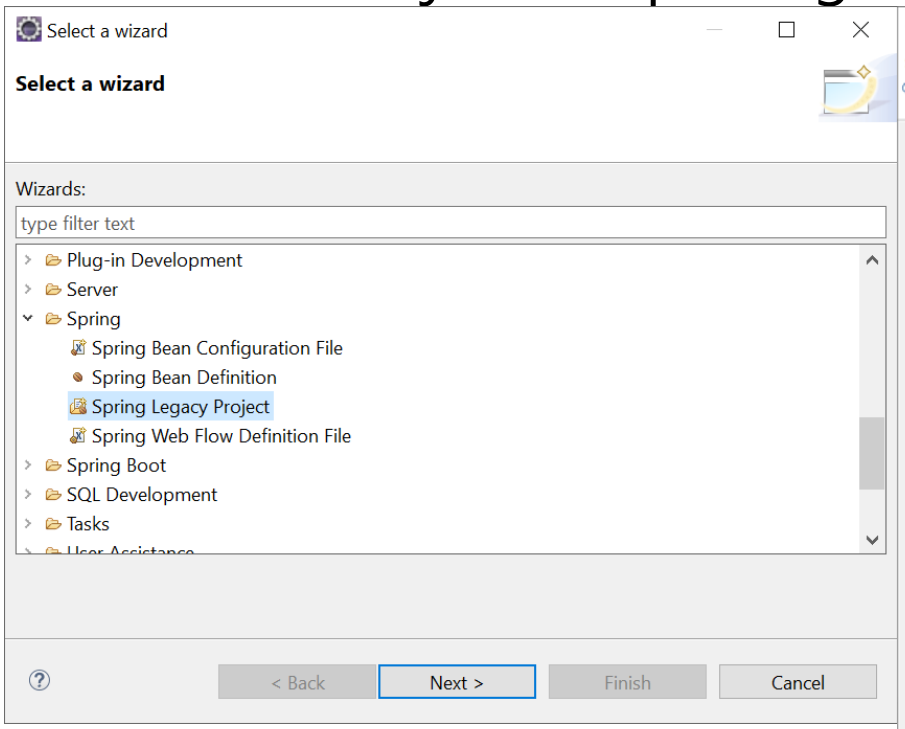
Spring Tool Suite (STS)

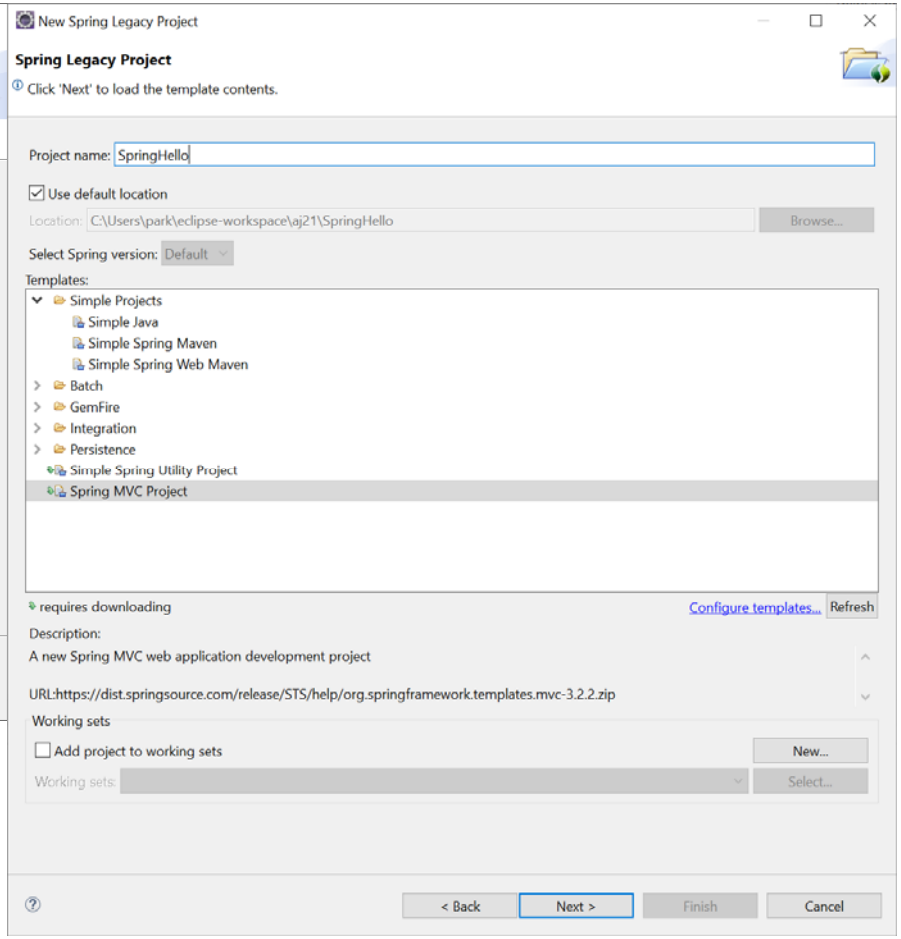
- Eclipse -> Help -> Eclipse Marketplace -> Find: STS -> Spring Tools 3.9.14.RELEASE->Install & Restart Eclipse



Eclipse Spring Legacy Project

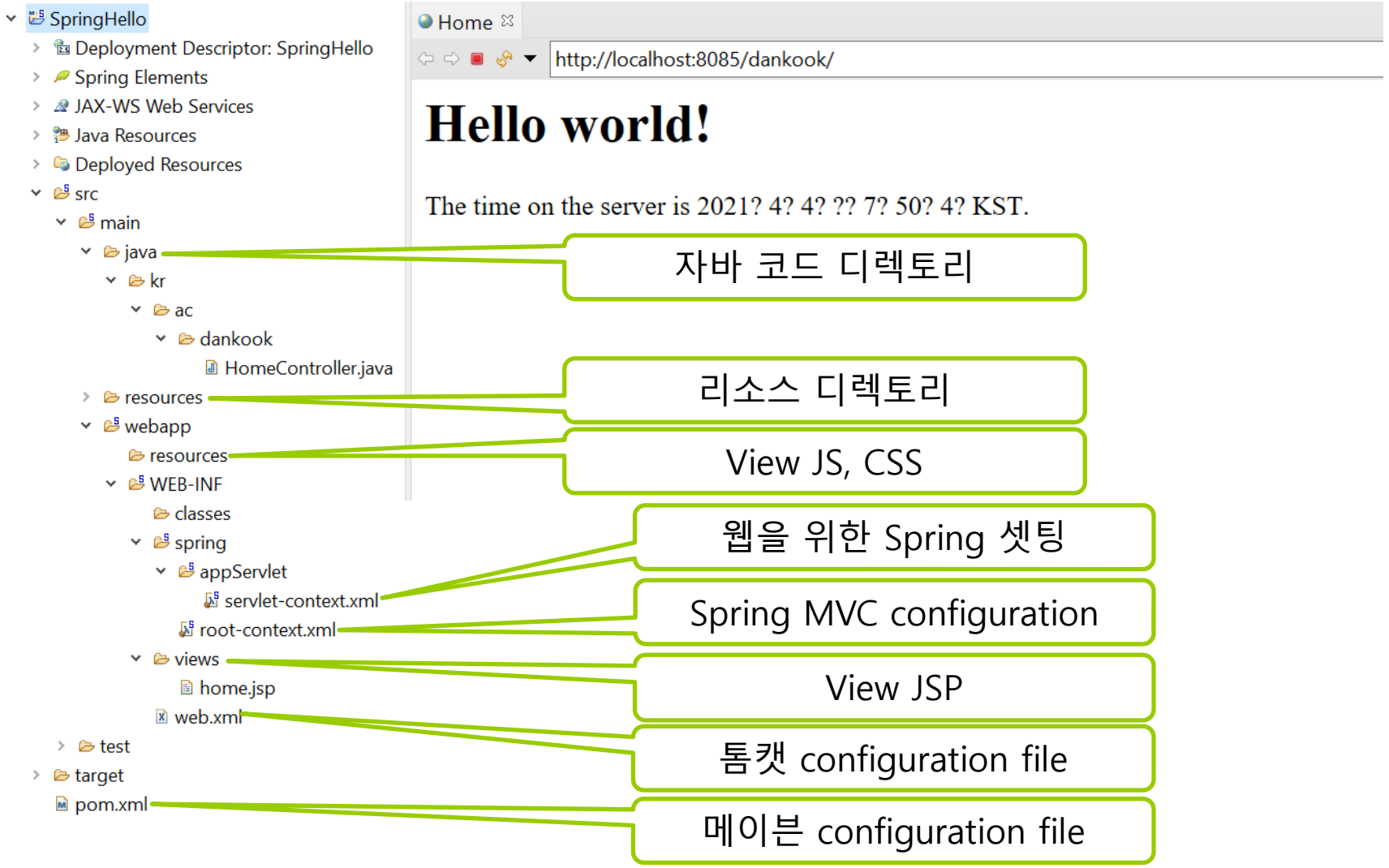
- Eclipse -> File -> New -> Other -> Spring -> Spring Legacy Project ->Project name & Templates: Spring MVC Project -> package name -> Finish





Spring MVC Project

Spring MVC Project 디렉토리 구조



pom.xml 수정

pom.xml

```
<properties>
<java-version>15</java-version>
<org.springframework-version>5.2.8.RELEASE</org.springframework-version>
<org.aspectj-version>1.6.10</org.aspectj-version>
<org.slf4j-version>1.6.6</org.slf4j-version>
</properties>
 <build>
      <plugins>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.8.1</version>
          <configuration>
            <source>15</source>
            <target>15</target>
            <compilerArgument>-Xlint:all</compilerArgument>
            <showWarnings>true</showWarnings>
            <showDeprecation>true</showDeprecation>
          </configuration>
        </plugin>
      </plugins>
  </build>
```

Eclipse Project -> Maven -> Update Project (Update Maven)

home.jsp 수정

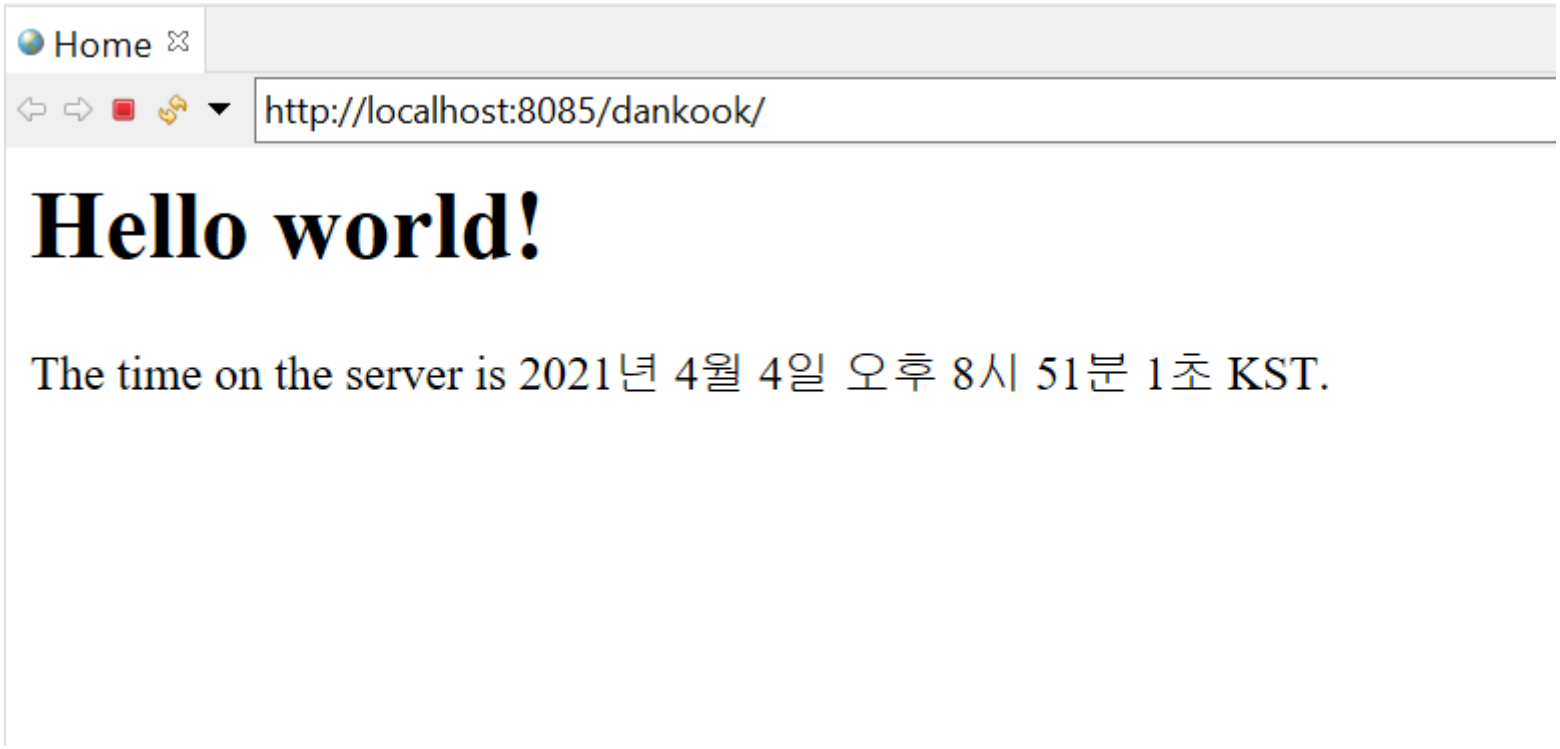
home.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page session="false" %>
<html>
<head>
<title>Home</title>
</head>
<body>
<h1>
Hello world!
</h1>

<P>  The time on the server is ${serverTime}. </P>
</body>
</html>
```

Eclipse Project -> Run As -> Maven Clean & Maven Install & Run on Server

Spring MVC Project 실행



HomeController.java 수정

HomeController.java

```
@Controller
public class HomeController {
    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);

    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
    logger.info("Welcome home! The client locale is {}.", locale);
    Date date = new Date();
    DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG,
    DateFormat.LONG, locale);
    String formattedDate = dateFormat.format(date);
    model.addAttribute("serverTime", formattedDate );
    // add more attribute
    String greetings = "Greetings, Spring MVC";
    model.addAttribute("message", greetings);
    return "home";
    }
}
```

serverTime과 message 속성이 모델에 추가됨

home.jsp 수정

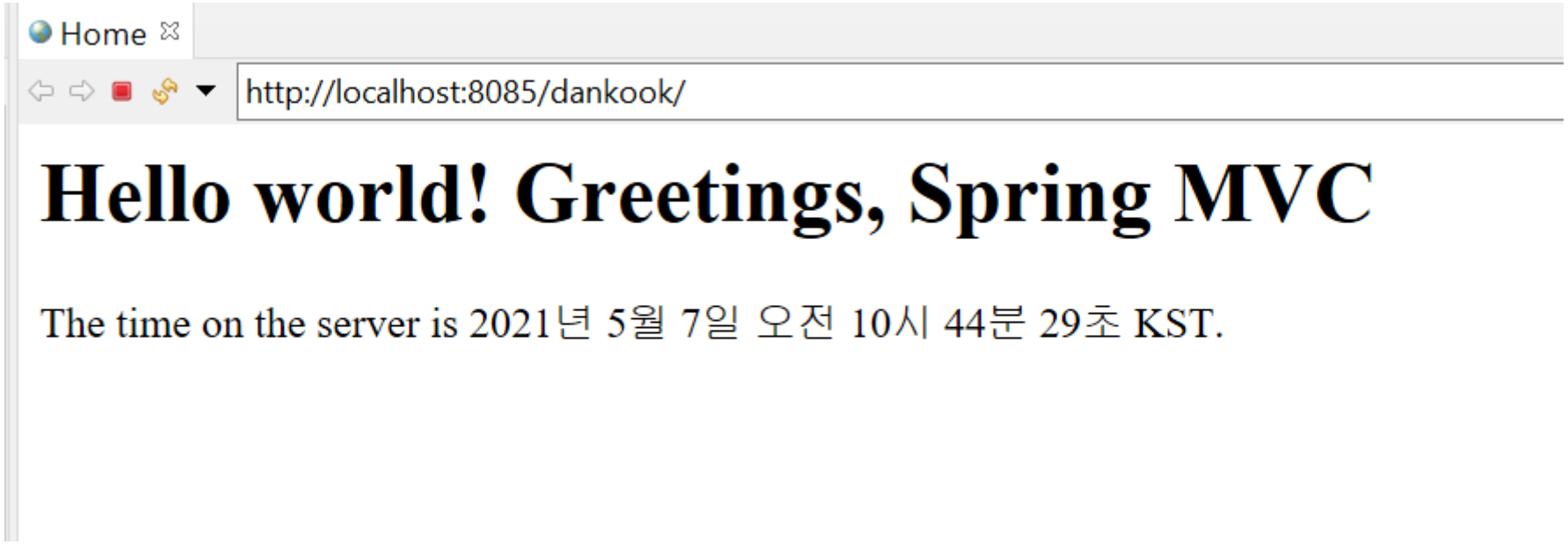
home.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page session="false" %>
<html>
<head>
<title>Home</title>
</head>
<body>
<h1>
Hello world! ${message}
</h1>

<P>  The time on the server is ${serverTime}. </P>
</body>
</html>
```

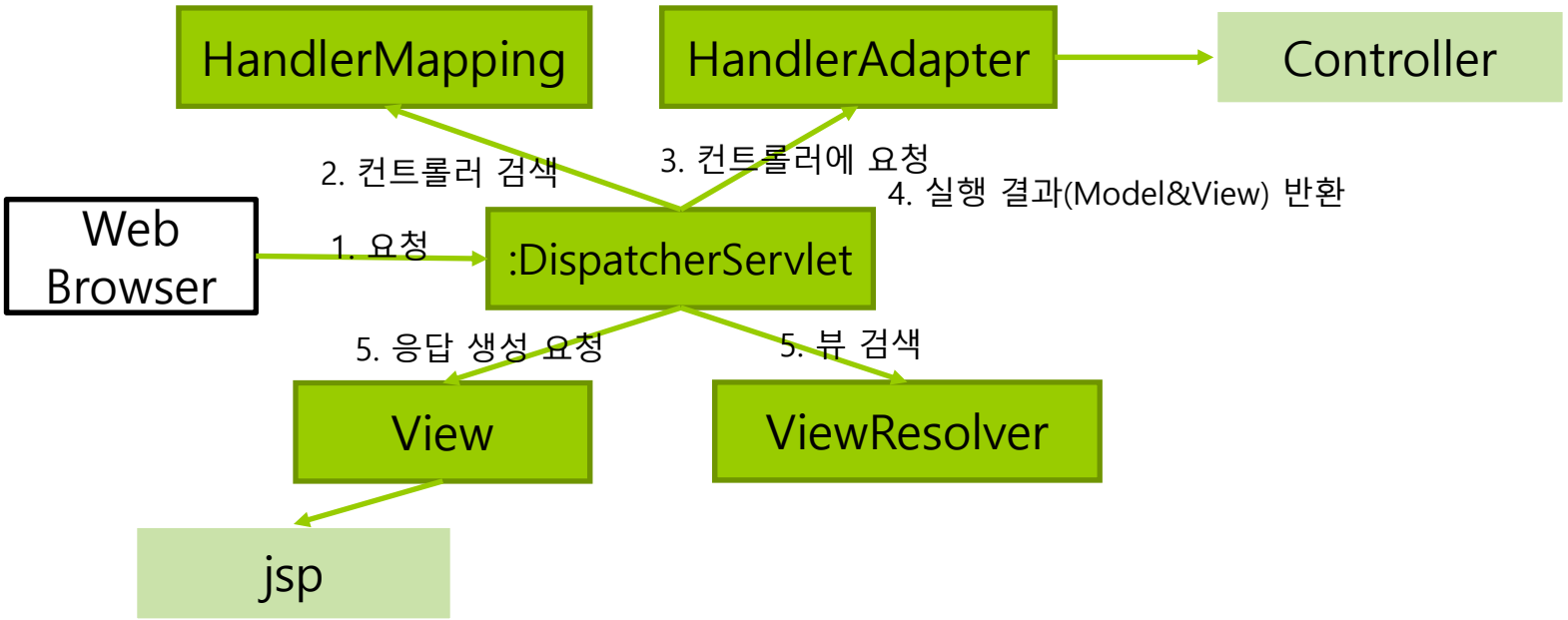
Eclipse Project -> Run As -> Maven Clean & Maven Install & Run on Server

Spring MVC Project 실행



기본 흐름

- MVC 모델 특성상 누가 누구를 호출하는지 직접적으로 표현되지 않음
- 전체 구조를 이해할 필요가 있음



스프링 MVC 설정 과정

- web.xml에 DispatcherServlet 설정
  - 어떤 경로 접근에 어떤 기능(컨트롤러) 할당할 것인지
- web.xml 에 캐릭터 인코딩 필터 설정
- 스프링 MVC 설정
  - HandlerMapping, HandlerAdapter 설정 → 가장 심플하게는 <mvc:annotation-driven>태그를 xml 설정파일에 추가
  - ViewResolver 설정

java 설정 사용시 @EnableWebMvc

@Controller 태그 활성화

DispatcherServlet 설정

- 웹브라우저의 요청(http://localhost:8080/hello) 받아들여서 스프링 컨테이너를 생성
  - 컨테이너 설정 파일로는 /WEB-INF/서블릿이름-servlet.xml이 기본 설정됨
  - 설정 파일 목록을 만들기 위해서는 contextConfigLocation 매개변수를 web.xml에 지정

web.xml

```
<!-- Processes application requests -->
<servlet>
  <servlet-name>appServlet</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

DispatcherServlet 설정

- servlet-context.xml 확인

servlet-context.xml

```
<!-- DispatcherServlet Context: defines this servlet's request-processing
infrastructure -->
<!-- Enables the Spring MVC @Controller programming model -->
<annotation-driven />
<!-- Handles HTTP GET requests for /resources/** by efficiently serving up static
resources in the ${webappRoot}/resources directory -->
<resources mapping="/resources/**" location="/resources/" />
<!-- Resolves views selected for rendering by @Controllers to .jsp resources in the
/WEB-INF/views directory -->
<beans:bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
<beans:property name="prefix" value="/WEB-INF/views/" />
<beans:property name="suffix" value=".jsp" />
</beans:bean>

<context:component-scan base-package="kr.ac.dankook" />
```

캐릭터 인코딩 필터 설정

- 한글 문자열의 전송 패러미터(localhost/사용자/) 처리를 위한 인자
- web.xml의 webapp 태그 내에 다음 내용 추가

```
<filter>
    <filter-name>encodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>encodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

MVC 구성요소 설정 (servlet-context.xml)

- HandlerMapping, HandlerAdapter, ViewResolver 설정
- 내장 매퍼와 어댑터 사용은 다음으로 해결
  - @Controller 어노테이티드 클래스를 컨트롤로 사용할 수 있도록 함

```
<!-- Enables the Spring MVC @Controller programming model -->
<annotation-driven />
```

servlet-context.xml

- 뷰 리졸버는 아래를 통해 구현
  - /web-inf/views/ 아래의 .jsp 파일들을 뷰로 사용

```
<!-- Resolves views selected for rendering by @Controllers to .jsp resources
in the /WEB-INF/views directory -->
<beans:bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
<beans:property name="prefix" value="/WEB-INF/views/" />
<beans:property name="suffix" value=".jsp" />
</beans:bean>
```

컨트롤러 경로 매핑(라우팅) (web.xml)

- 사용자가 요청하는 모든 경로에 대해 appServlet 사용
  - *.do 와 같이 확장자로 기능을 할당하는 경우
  - /users/* 와 같이 경로의 이름으로 기능을 할당하는 경우
    - 다음 코드는 전체 경로에 대해 **appServlet**을 할당하고 있음
    - jsp 요청을 제외한 나머지 모든 요청을 appServlet이 받아들이게 됨
    - 스프링 mvc 설정파일에 <mvc:default-servlet-handler />를 추가하여 톰캣의 404오류 페이지도 스프링이 핸들링하도록 할 수 있음

```
<servlet-mapping>
  <servlet-name>appServlet</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
```

컨트롤러 구현

HomeController.java

```
@Controller  어노테이션
public class HomeController {
    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);

    @RequestMapping(value = "/", method = RequestMethod.GET)  경로매핑
    public String home(Locale locale, Model model) {
    logger.info("Welcome home! The client locale is {}.", locale);
    Date date = new Date();
    DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG,
    DateFormat.LONG, locale);
    String formattedDate = dateFormat.format(date);
    model.addAttribute("serverTime", formattedDate );
    // add more attribute
    String greetings = "Greetings, Spring MVC";
    model.addAttribute("message", greetings);
    return "home";    뷰 이름 리턴
    }
}
```

모델에 데이터 추가

serverTime과 message 속성이 모델에 추가됨