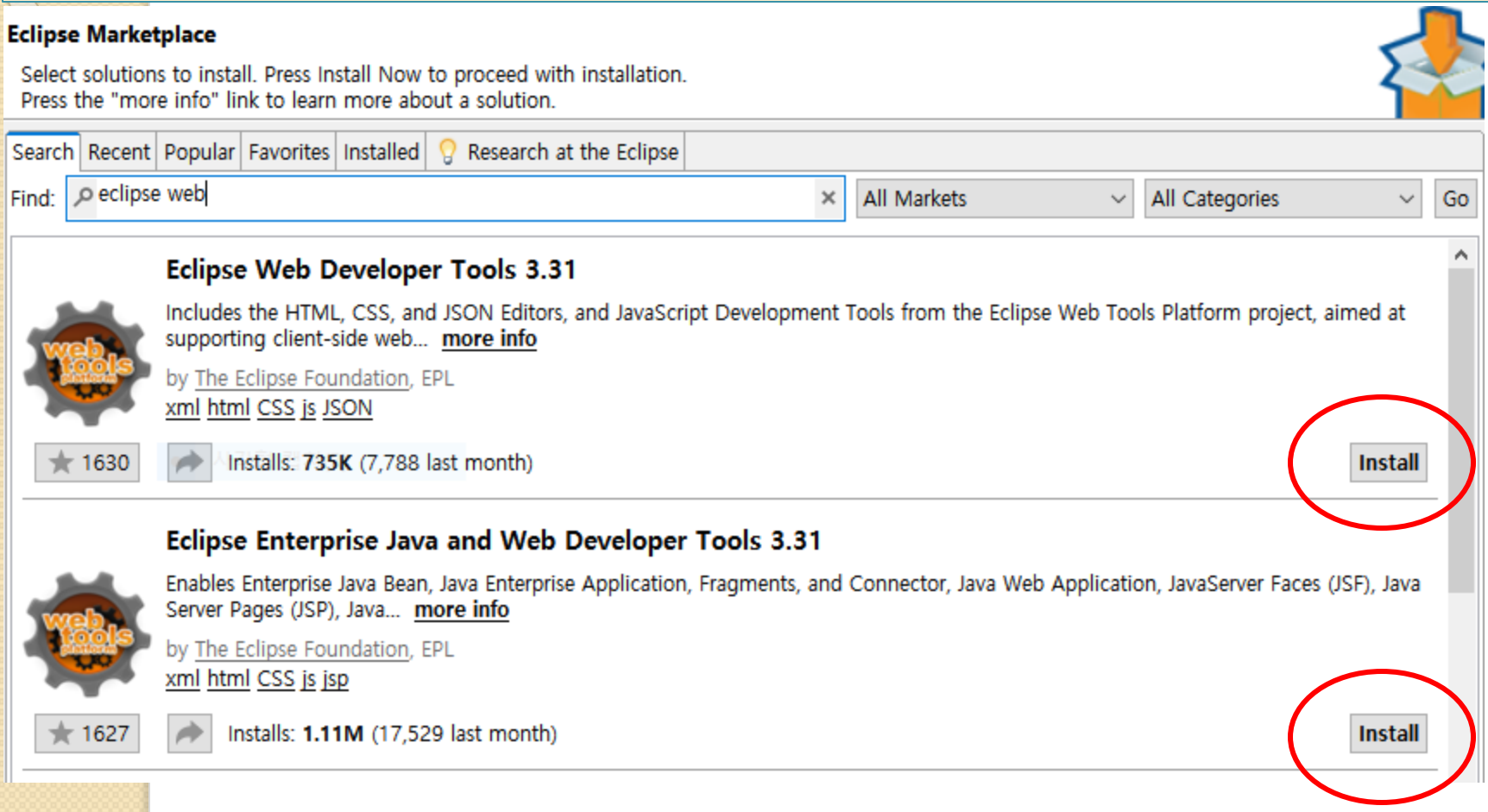
Spring Boot

강사 강태광

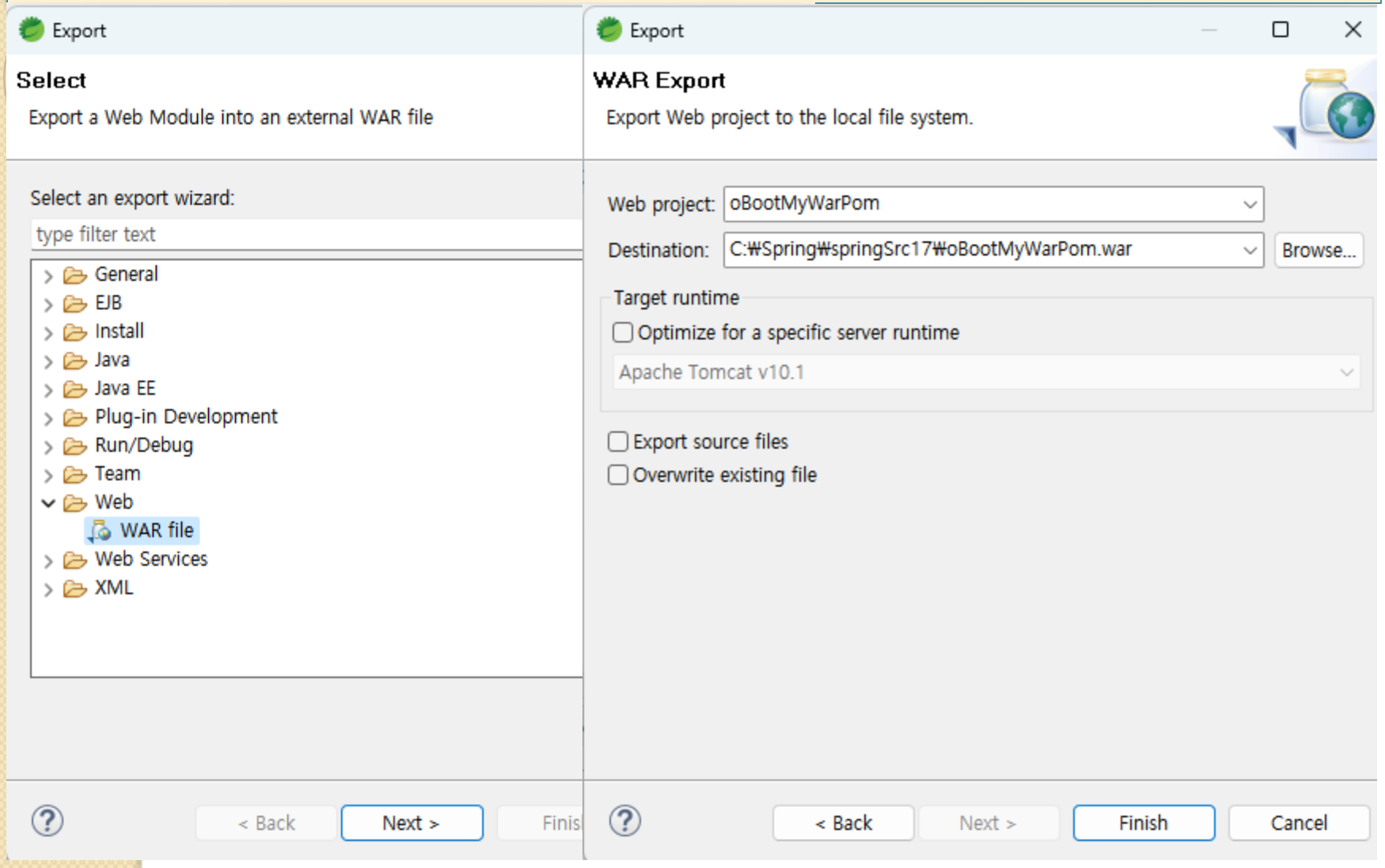
1-1. JSP spring-boot 연계

1. Help Eclipse Marketplace 클릭
2. Find → Eclipse web 입력
3. Eclipse Enterprise Java and Web Develper Tools를 install install
4. 필요하다면 선택해도 되지만 기본만 설치
5. 플러그인 설치후 STS4(이클립스)를 재시작 해주면 다음과 같이 jsp를 추가
6. Trust 메시지가 나오면 반드시 승인



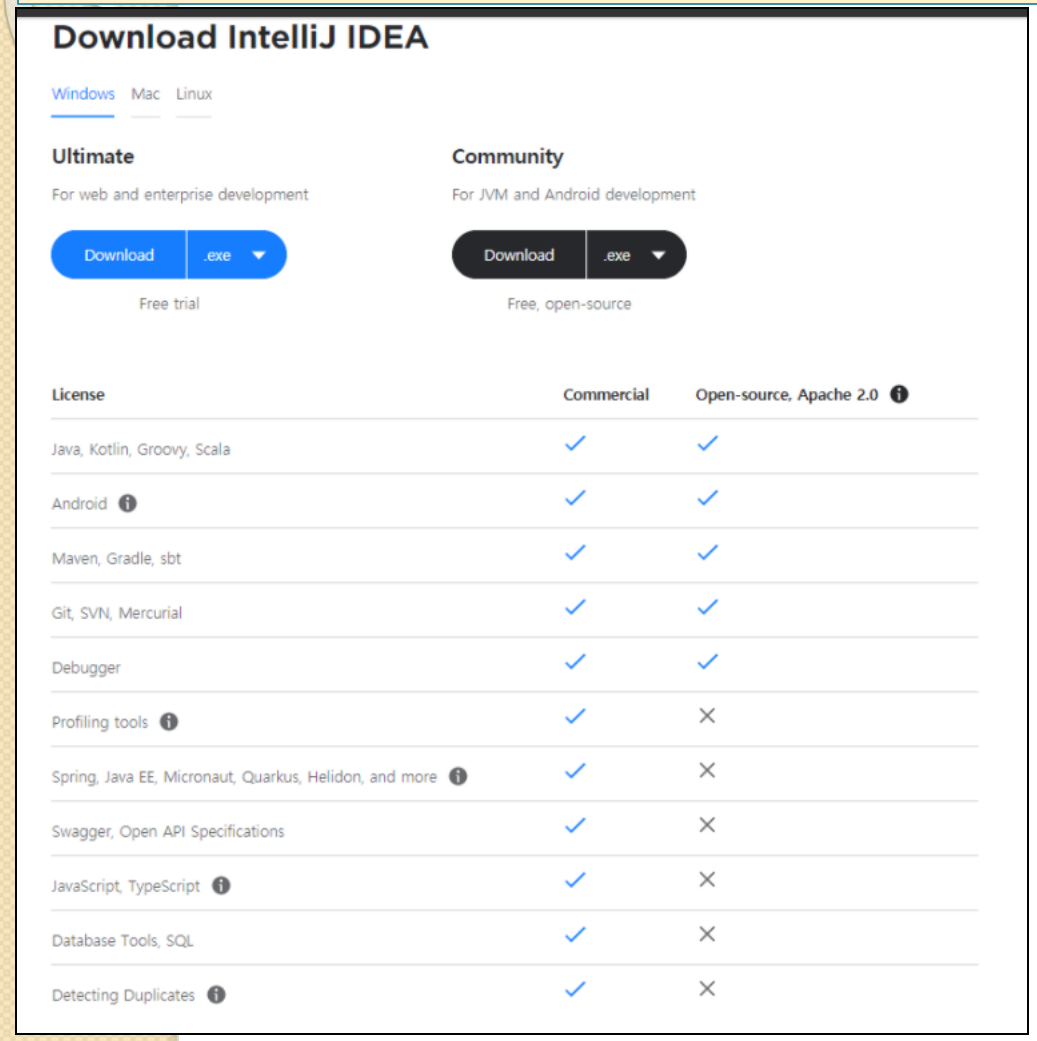
1-2. War 설정시 Project

1. 우클릭-> Prpperties 선택
2. War에 MAVEN 방식으로 설정해야 .jsp인식



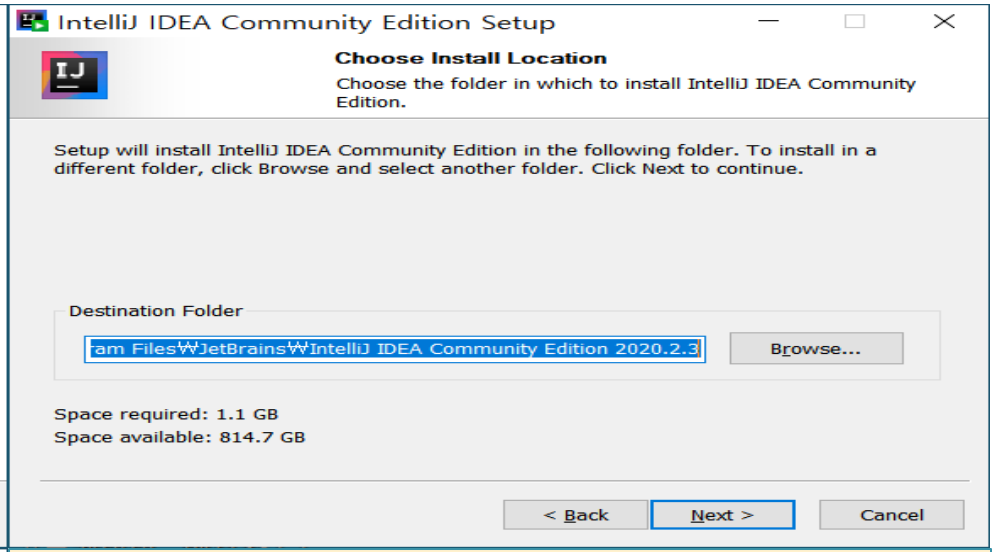
1-4. IntelliJ 설치1

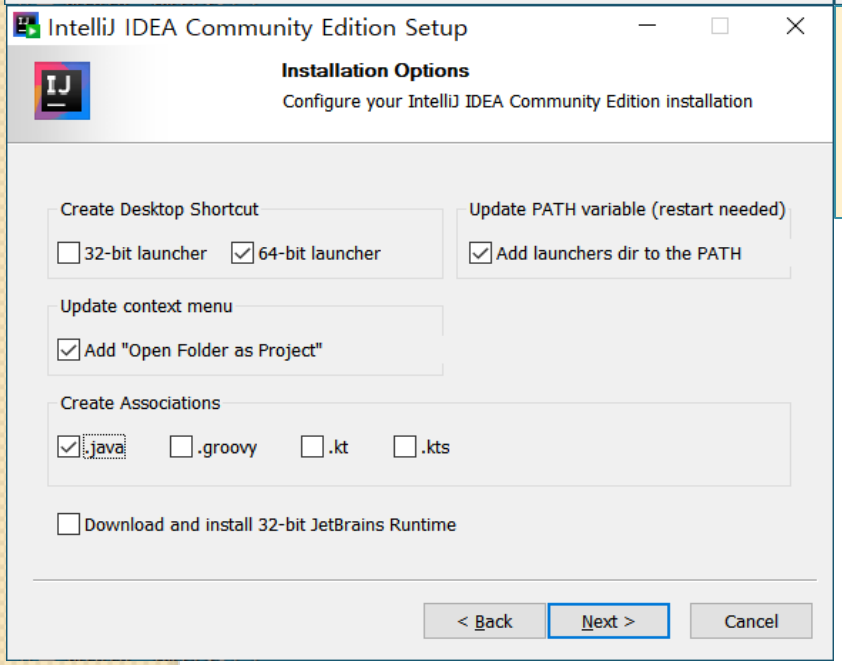
1.젯브레인스(JetBrains) 사에서 제작한 Java 개발을 위한 툴
 - IntelliJ 혹은 IDEA 라고도 함.
2. https://www.jetbrains.com/idea/
3. 버전 선택
 - Ultimate VS Communtity



- IntelliJ IDEA는 기업, 개인에 상관없이 무료로 이용 가능하지만 기능에 제한을 둠
- Ultimate : 완전한 기능 제공 (1달간 무료 체험이 가능)
- Communtity : 제한적 기능 제공
- Web을 제외한 기본적인 JVM 기반 언어와 안드로이드 개발을 지원.
- Java 기반의 REST API 등의 백엔드 개발만 고려한다면 Communtity 에디션 만으로 충분

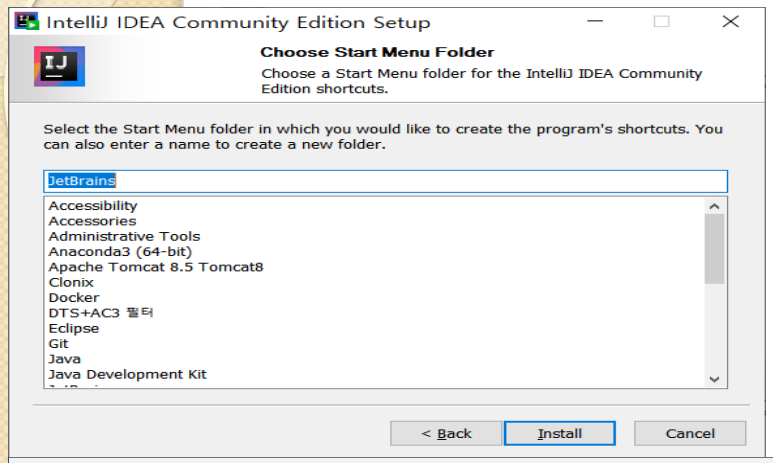
1-4. IntelliJ 설치2

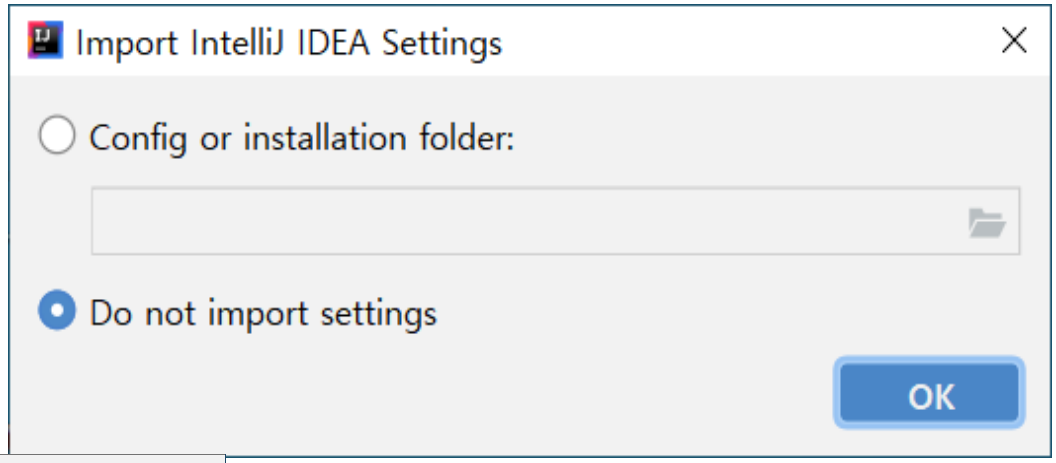


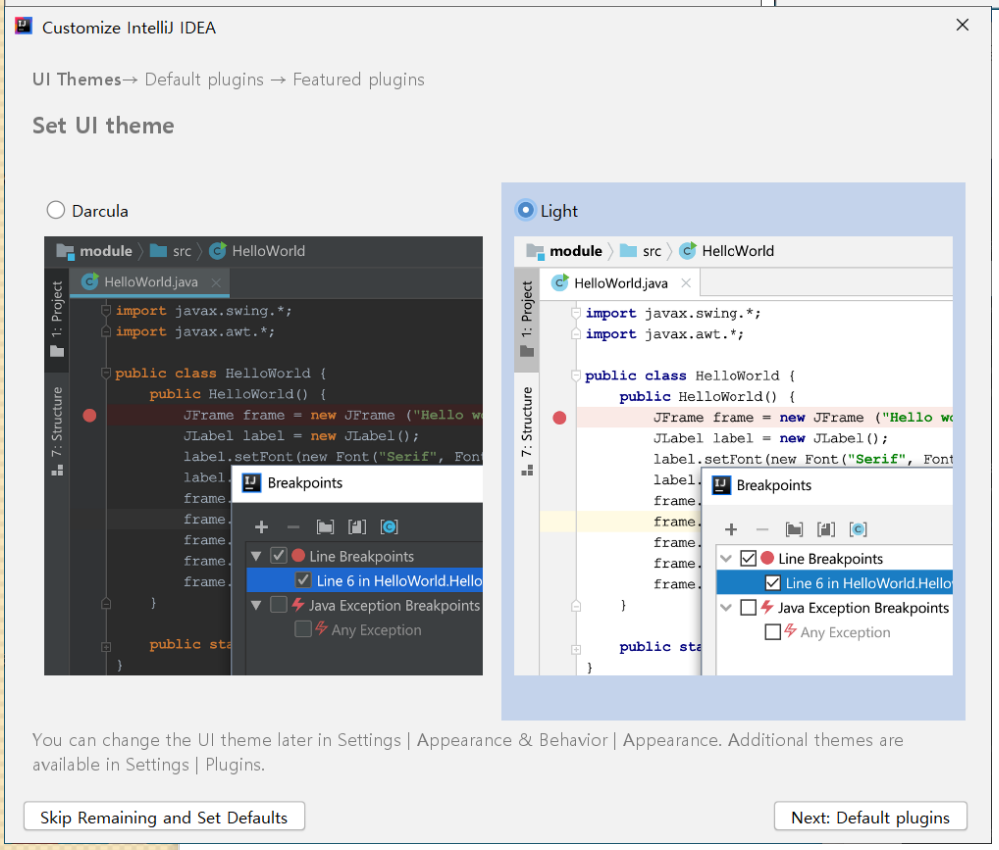


① Create Desktop Shortcut : 윈도우 OS 환경 체크 (64bit)
② Update PATH variable(restart needed) : 윈도우 환경변수에 자동으로 추가 할 수 있도록 체크
③ Update context menu : 프로젝트로 폴더 열기
④ Create Association : 자바 사용

1-4. IntelliJ 설치3



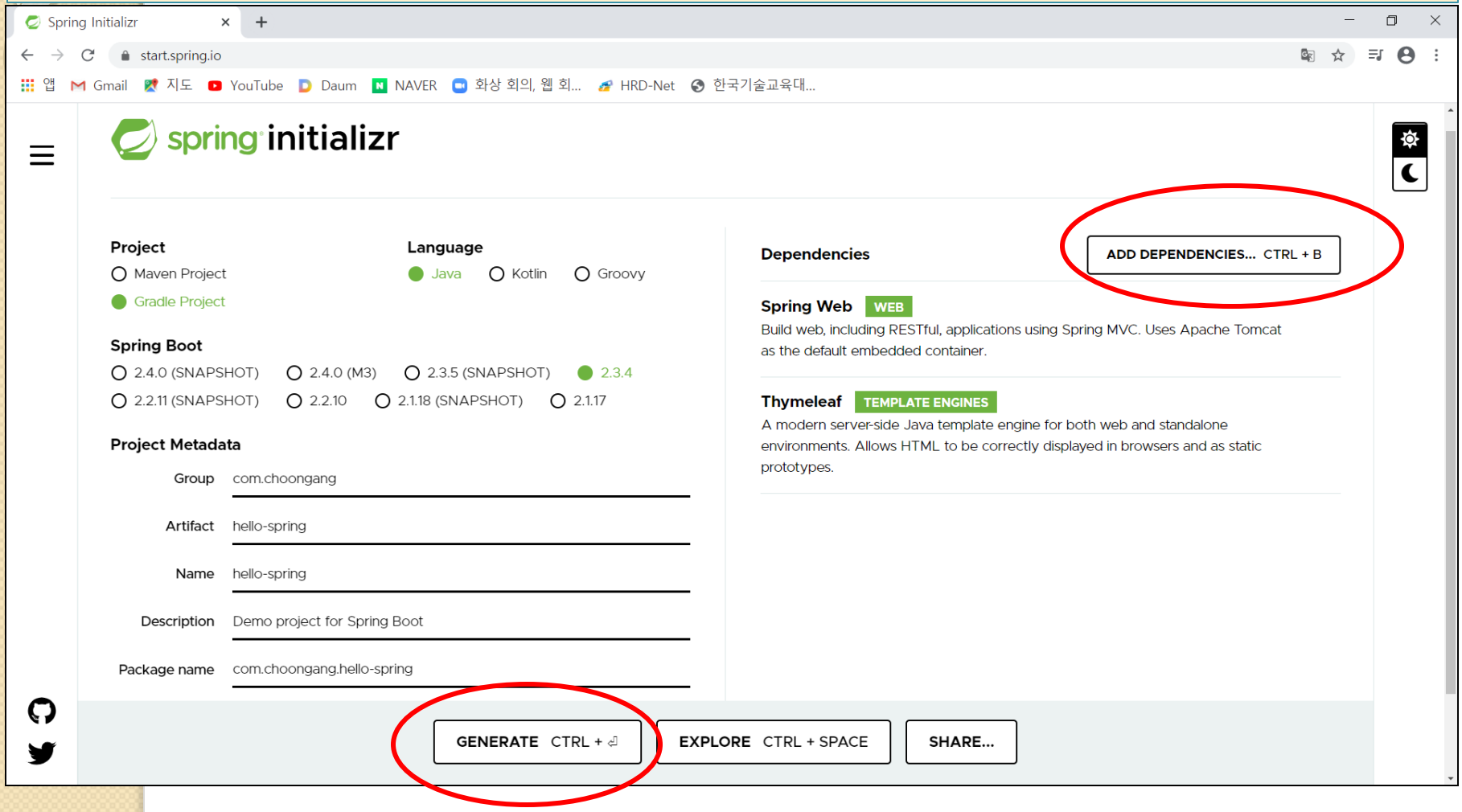




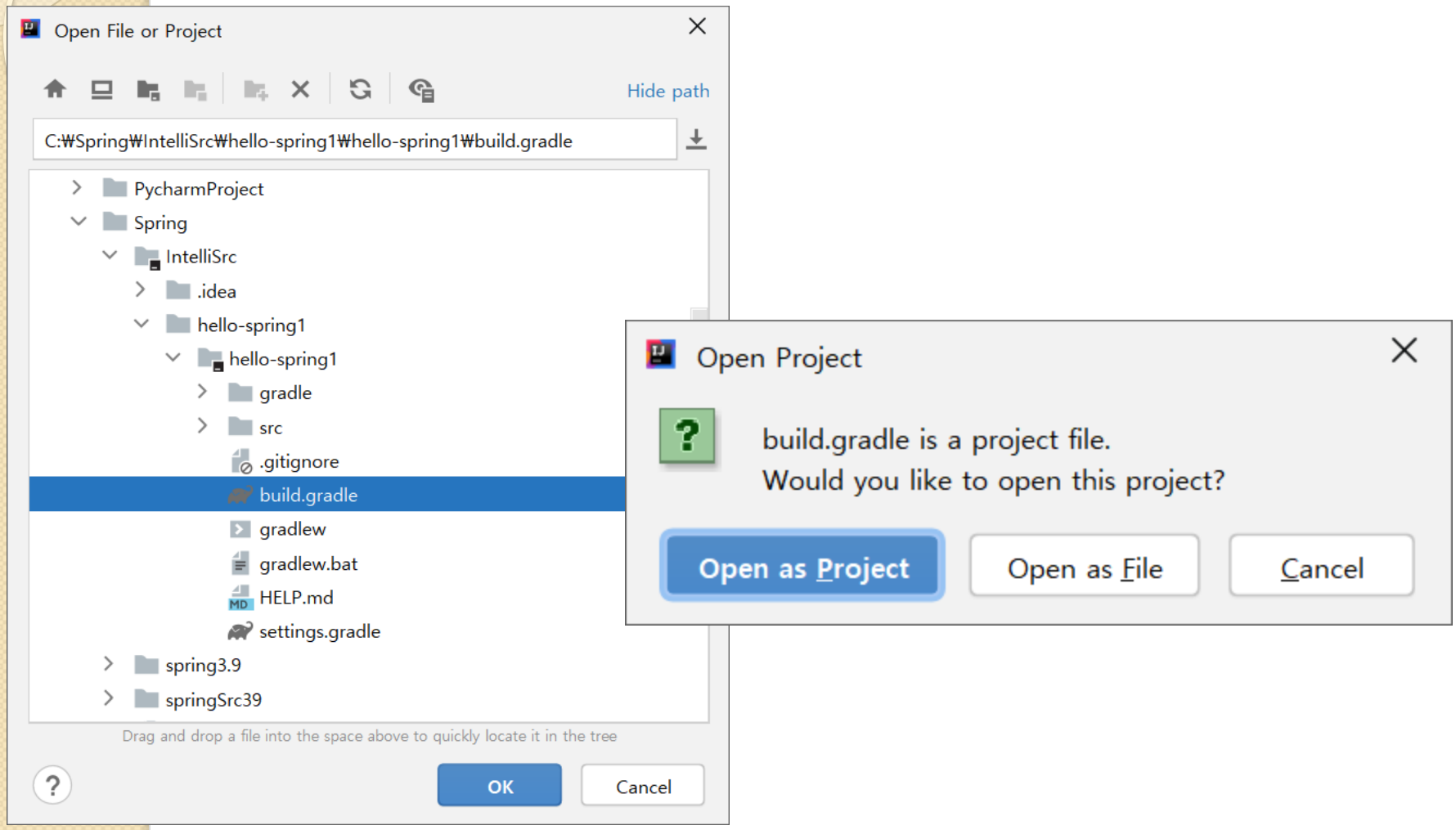
① Create Desktop Shortcut : 윈도우 OS 환경 체크 (64bit)
② Update PATH variable(restart needed) : 윈도우 환경 변수에
자동으로 추가 할 수 있도록 체크
③ Update context menu : 프로젝트로 폴더 열기
④ Create Association : 자바 사용

2.Spring boot Project 생성

1. 스프링 부트 Starter site로 이동 spring Project 생성
   - https://start.spring.io

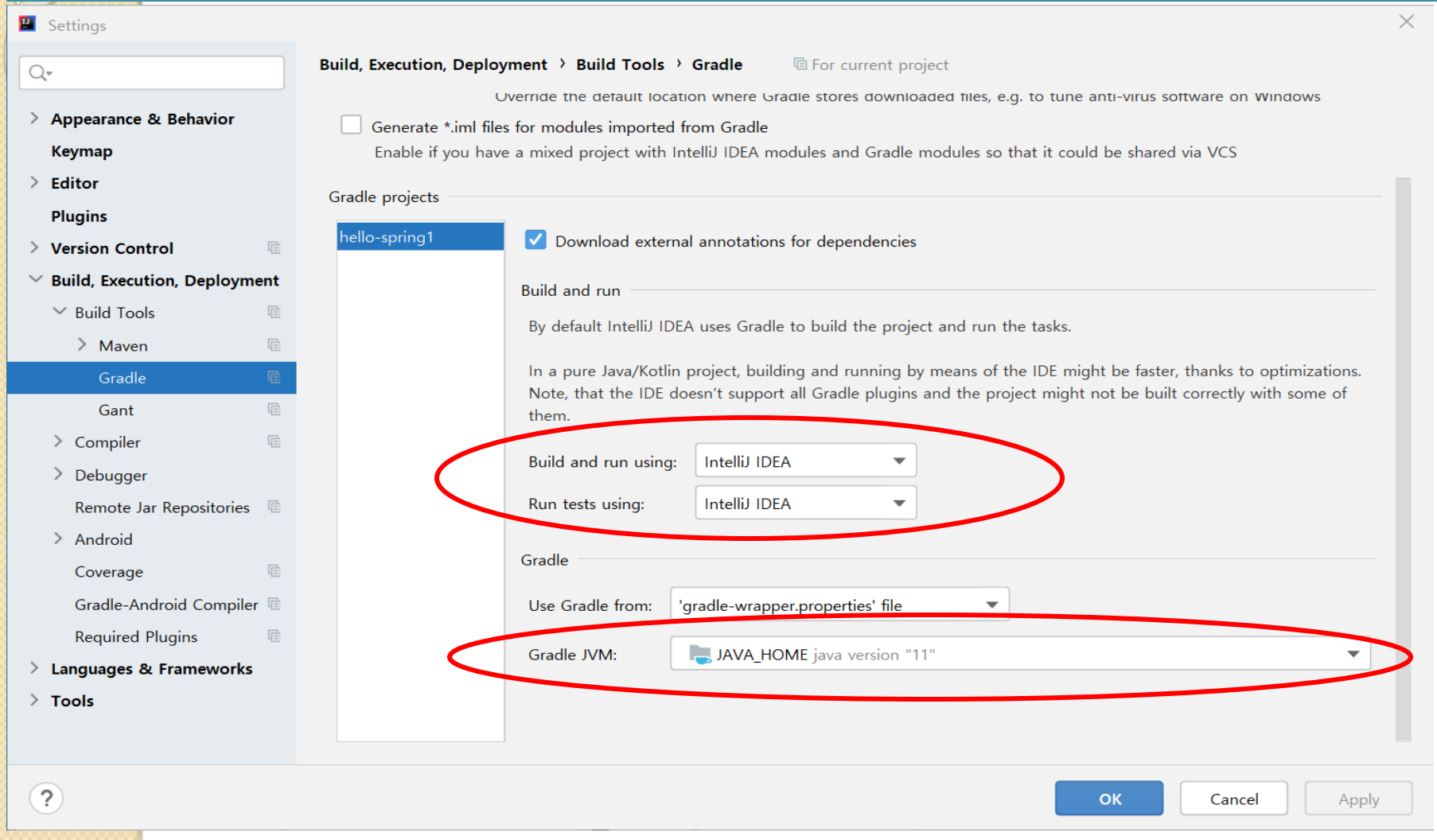


3-1.Spring boot Project intelli-J에서 Open



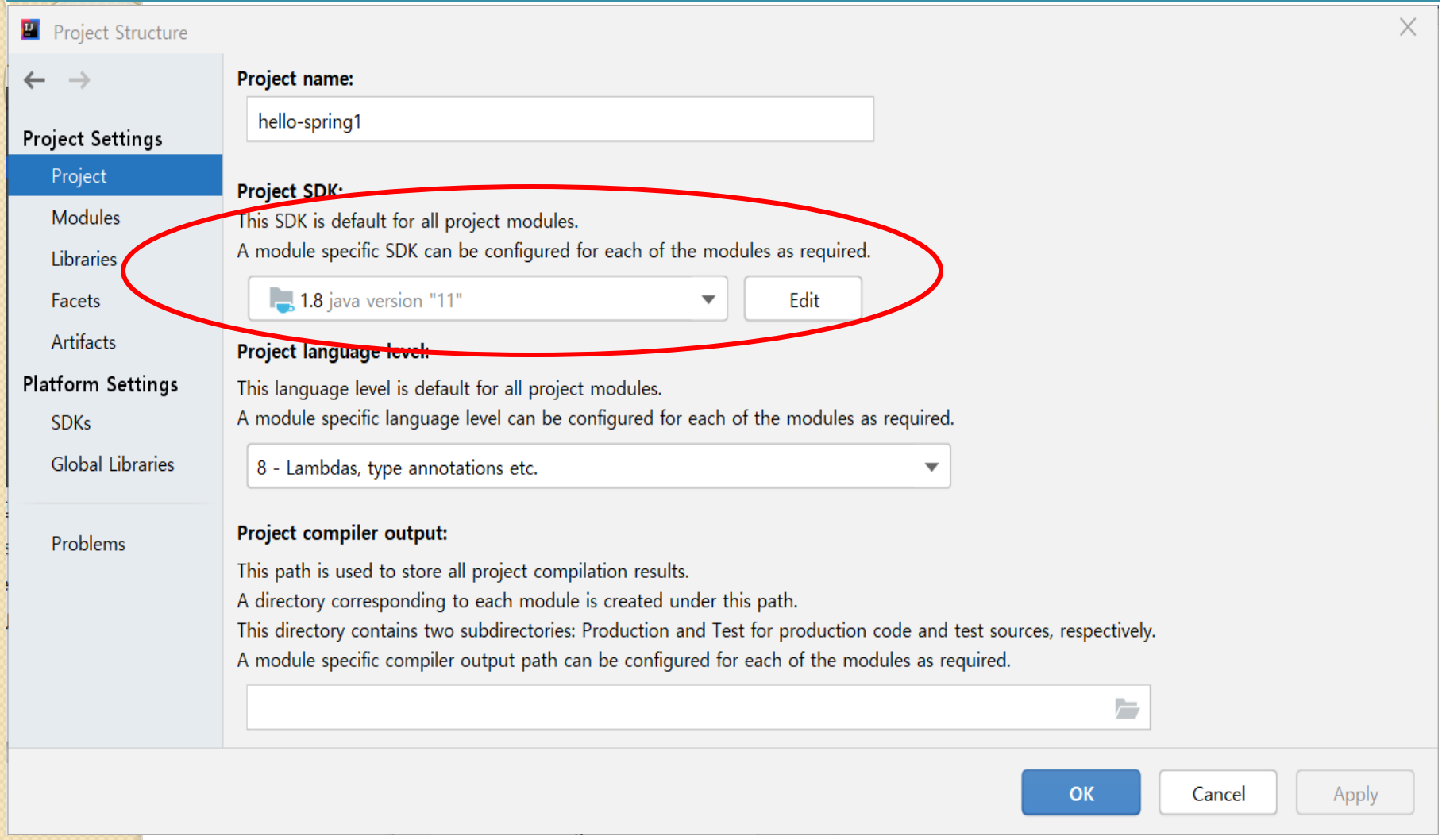
3-2. IntelliJ Gradle 대신에 자바 직접 실행

IntelliJ 버전은 Gradle을 통해서 실행 하는 것이 기본 설정. 이렇게 하면 실행속도가 느림.
다음과 같이 변경하면 자바로 바로 실행해서 실행속도가 더 빠름
File → Settings(Ctrl+Alt+S)



3-3. IntelliJ JDK 설치 확인

File -> Project Structure(Ctrl+Alt+Shift+S)
빨간색 박스의 JDK를 내가 새로 설치한 자바 11로 지정



3-3. IntelliJ 개발 Library

Gradle은 의존관계가 있는 라이브러리를 함께 다운로드

1.스프링 부트 라이브러리
- 1) spring-boot-starter-web
  - ① spring-boot-starter-tomcat: 톰캣 (웹서버)
  - ② spring-webmvc: 스프링 웹 MVC
- 2) spring-boot-starter-thymeleaf: 타임리프 템플릿 엔진(View)
- 3) spring-boot-starter(공통): 스프링 부트 + 스프링 코어 + 로깅
  - ① spring-boot (spring-core)
  - ② spring-boot-starter-logging
    - logback, slf4j

2. 테스트 라이브러리
- 1) spring-boot-starter-test
  - ① junit: 테스트 프레임워크
  - ② mockito: 목 라이브러리
  - ③ assertj: 테스트 코드를 좀 더 편하게 작성하게 도와주는 라이브러리
  - ④ spring-test: 스프링 통합 테스트 지원

4. View 환경설정

1. resources/static/index.html
   1) Web 기본을 실행하면 그냥 수행시켜주는 Page
   2) 스프링 부트가 제공하는 Welcome Page 기능
   3) https://docs.spring.io/spring-boot/docs/2.3.1.RELEASE/reference/html/spring-boot-features.html#boot-features-spring-mvc-welcome-page

```
<!DOCTYPE HTML>
<html>
<head>
    <title>Hello</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
  Hello
   <a href="/Hello">hello</a>
</body>
</html>
```

2. thymeleaf 템플릿 엔진
   1) thymeleaf 공식 사이트: https://www.thymeleaf.org/
   2) 스프링 공식 튜토리얼: https://spring.io/guides/gs/serving-web-content/
   3) 스프링부트 메뉴얼: https://docs.spring.io/spring-boot/docs/2.3.1.RELEASE/reference/ html/spring-boot-features.html#boot-features-spring-mvc-template-engines

5. Controller / View Coding

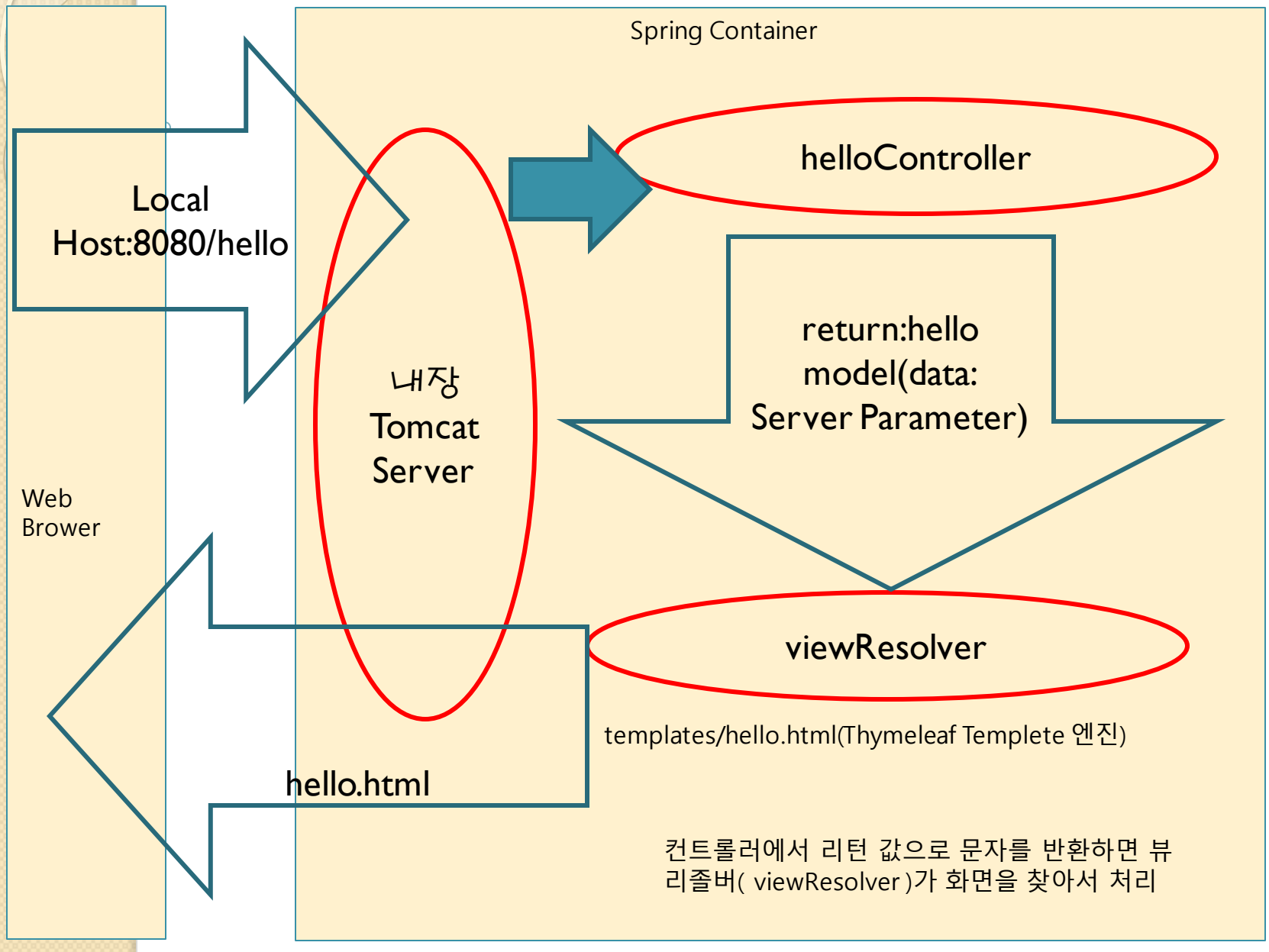
1. Controller (com.choongang.hellospring1.controller)

```
package com.choongang.hellospring1.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HelloController {

    @RequestMapping("Hello")
    public String hello(Model model) {
        System.out.println("HelloController Hello start... ");
        model.addAttribute("data","Server Parameter");
        return "hello";
    }
}
```

2. View(resources/templates/hello.html)

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Hello</title>
</head>
<body>
    <p th:text="안녕..+${data}">
      Boot 프로그램이야
    </p>

</body>
</html>
```

6. 전체 수행 Flow


Spring Container
Local
Host:8080/hello
내장
Tomcat
Server
helloController
return:hello
model(data:
Server Parameter)
viewResolver
templates/hello.html(Thymeleaf Templete 엔진)
hello.html
Web
Brower
컨트롤러에서 리턴 값으로 문자를 반환하면 뷰
리졸버(viewResolver)가 화면을 찾아서 처리

7. Build / Run

1. Build
1) set Java_HOME=C:₩Program Files₩Java₩jdk-17
2) 콘솔로 이동
3) gradlew build
4) cd build/libs

2. **Run(배포)**
1) java -jar hello-spring-0.0.1-SNAPSHOT.jar
2) 실행확인

C:₩Users₩user>**cd C:₩Spring₩IntelliSrc₩hello-spring1₩hello-spring1**

C:₩Spring₩IntelliSrc₩hello-spring1₩hello-spring1>**gradlew build**

Welcome to Gradle 6.6.1!

Here are the highlights of this release:
- Experimental build configuration caching
- Built-in conventions for handling credentials
- Java compilation supports --release flag

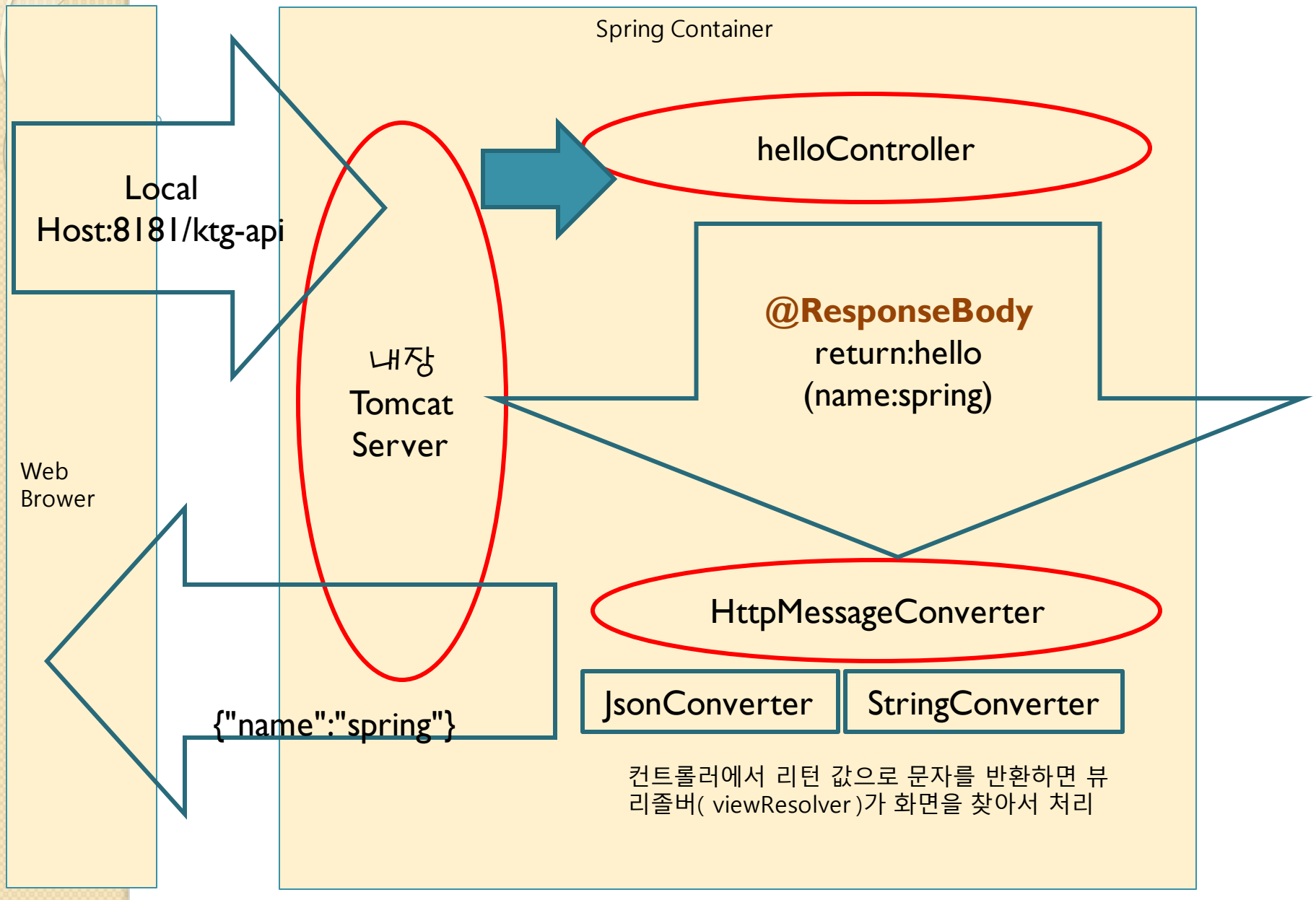
For more details see https://docs.gradle.org/6.6.1/release-notes.html

Starting a Gradle Daemon (subsequent builds will be faster)

> Task :test
2020-10-15 14:07:56.575 INFO 12532 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor : Shutting down ExecutorService 'applicationTaskExecutor'

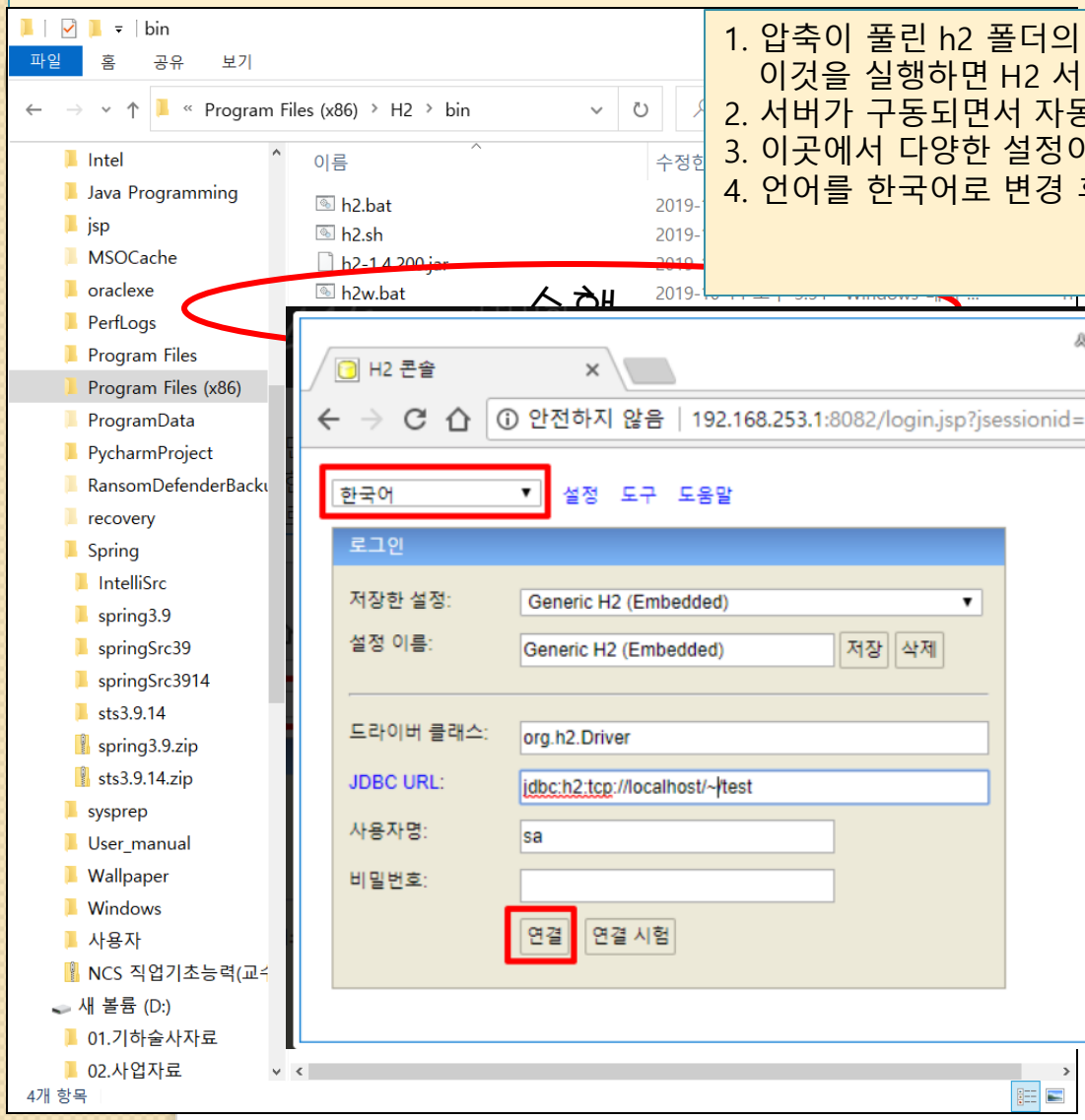
BUILD SUCCESSFUL in 14s
5 actionable tasks: 5 executed
C:₩Spring₩IntelliSrc₩hello-spring1₩hello-spring1>
C:₩Spring₩IntelliSrc₩hello-spring1₩hello-spring1>
C:₩Spring₩IntelliSrc₩hello-spring1₩hello-spring1₩build₩libs>
java -jar hello-spring1-0.0.1-SNAPSHOT.jar (배포 / 실행)

6. ResponseBody Flow



7-1. H2 DB 설치

1. 개발이나 테스트 용도로 가볍고 편리한 DB, 웹 화면 제공
2. https://www.h2database.com 다운로드 및 설치
3. h2 데이터베이스 버전은 스프링 부트 버전에



1. 압축이 풀린 h2 폴더의 **bin 폴더**로 이동하면 **h2w.bat** 파일이 있는데 이것을 실행하면 H2 서버가 구동
2. 서버가 구동되면서 자동으로 웹 기반의 H2 콘솔이 실행.
3. 이곳에서 다양한 설정이 가능.
4. 언어를 한국어로 변경 후 연결

7-2. H2 DB spring-boot 연계

1. 환경설정
   1) build.gradle 파일에 jdbc, h2 데이터베이스 관련 라이브러리 추가
      ① implementation 'org.springframework.boot:spring-boot-starter-jdbc'
      ② runtimeOnly 'com.h2database:h2'

   2) 스프링 부트 데이터베이스 연결 설정 추가(resources/application.properties)
      ① spring.datasource.url=jdbc:h2:tcp://localhost/~/tes
      ② spring.datasource.driver-class-name=org.h2.Driver

   3) 인텔리J 커뮤니티(무료) 버전의 경우 application.properties 파일의 왼쪽이 회색으로 나옴.
      엔터프라이즈(유료) 버전에서 제공하는 스프링의 소스 코드를 연결해주는 편의 기능이 빠진것
      실제 동작하는데는 아무런 문제가 없음

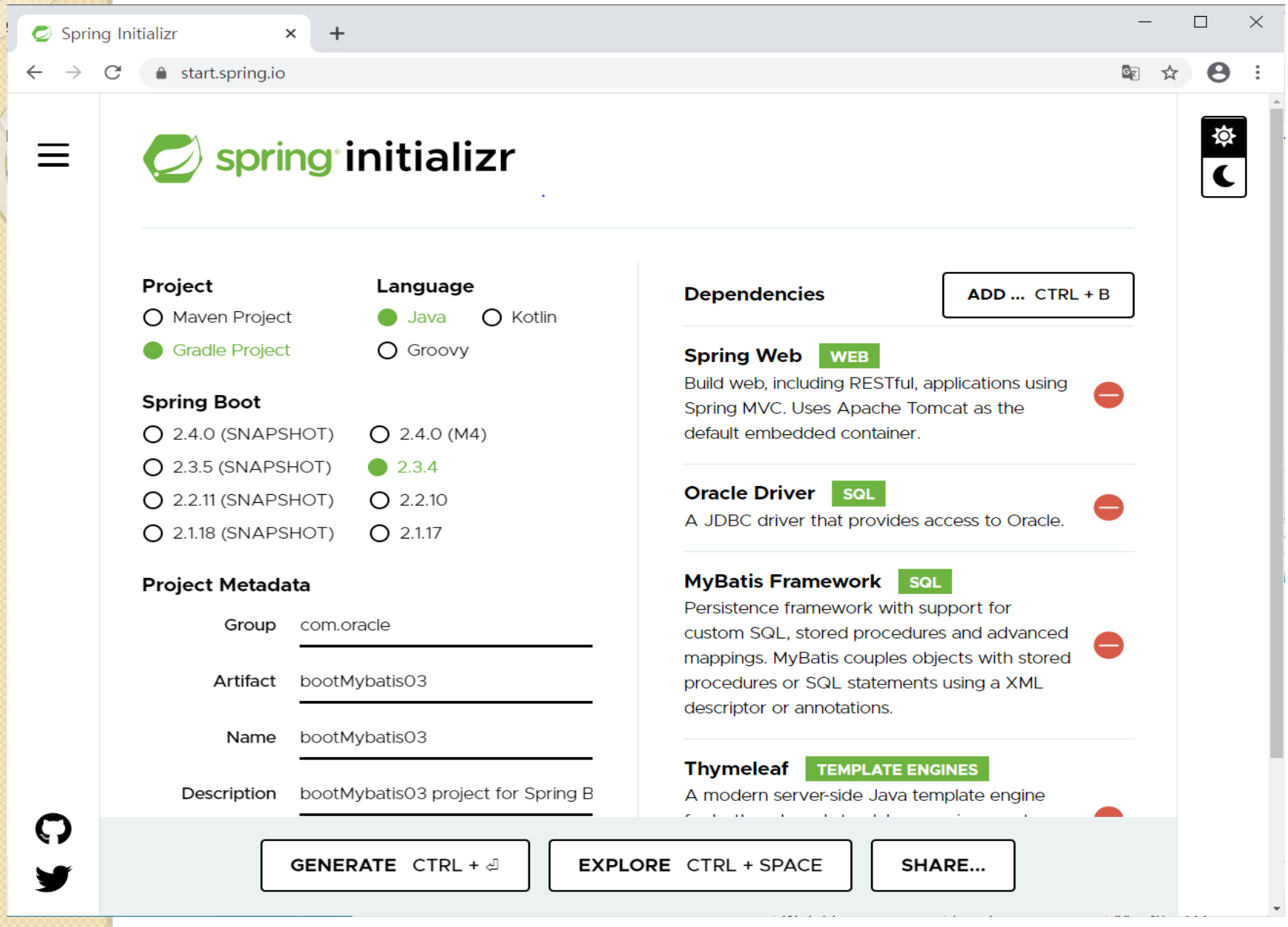
8. JPA spring-boot 연계

1. 사용 배경
 ① JPA는 기존의 반복 코드는 물론이고, 기본적인 SQL도 JPA가 직접 만들어서 실행
 ② JPA를 사용하면, SQL과 데이터 중심의 설계에서 객체 중심의 설계로 패러다임을 전환.
 ③ JPA를 사용하면 개발 생산성을 크게 높일 수 있음.

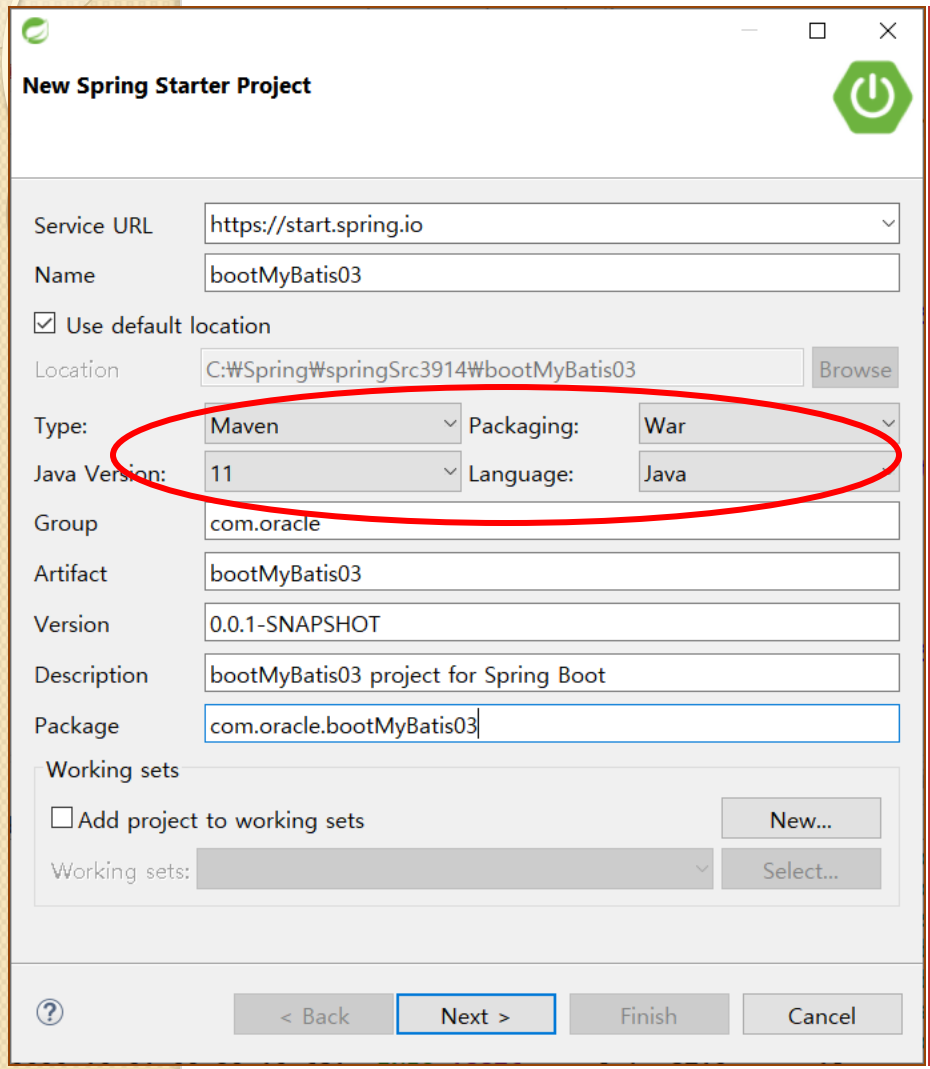
2. 환경설정
  1) build.gradle 파일에 JPA, h2 데이터베이스 관련 라이브러리 추가
   ① // implementation 'org.springframework.boot:spring-boot-starter-jdbc'
        implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
     - spring-boot-starter-data-jpa 는 내부에 jdbc 관련 라이브러리를 포함. 따라서 jdbc는 제거해도 됨
   ② runtimeOnly 'com.h2database:h2'

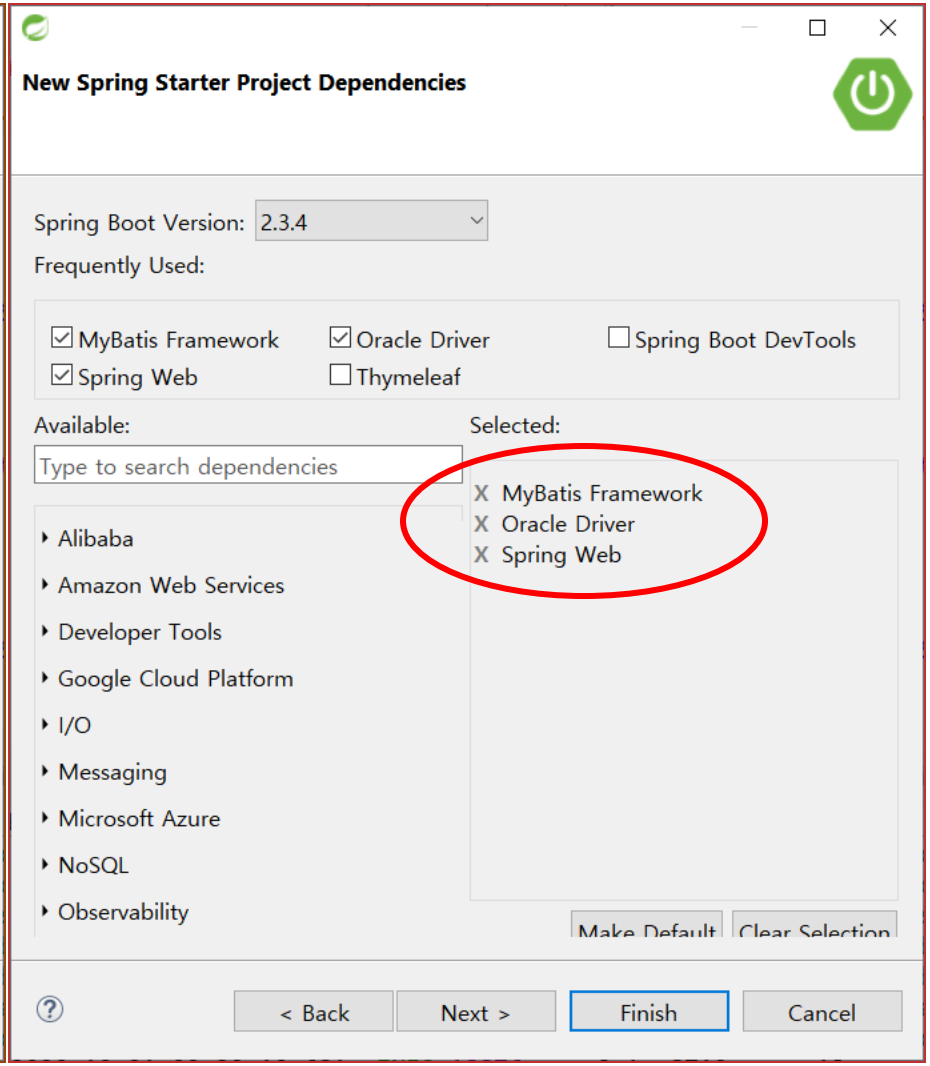
  2) 스프링 부트 JPA 설정 추가 (resources/application.properties)
   ① spring.datasource.url=jdbc:h2:tcp://localhost/~/test
   ② spring.datasource.driver-class-name=org.h2.Driver
   ③ spring.jpa.show-sql=true // JPA가 생성하는 SQL을 출력
   ④ spring.jpa.hibernate.ddl-auto=none
    - JPA는 테이블을 자동으로 생성하는 기능을 제공하는데 none 를 사용하면 해당 기능을 끈다

9-1-1. myBatis spring-boot 연계 (Gradle)



9-2. myBatis spring-boot 연계 (Maven-sts) 1



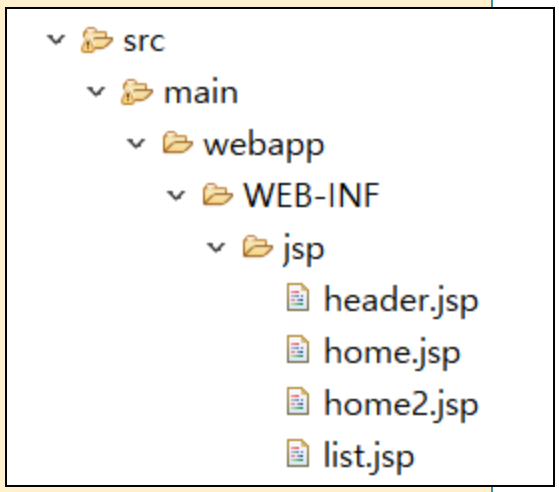


com.oracle.bootMyBatis03 생성 maven / War 작성

9-2. myBatis spring-boot 연계 (Maven-sts) 2

1. application.properties

```
spring.datasource.driver-class-name=oracle.jdbc.driver.OracleDriver
spring.datasource.url=jdbc:oracle:thin:@localhost:1521/xe
spring.datasource.username=scott
spring.datasource.password=tiger

# dto Location
mybatis.type-aliases-package=com.oracle.bootMyBatis03.model

# xml Location
mybatis.mapper-locations=classpath:mappers/*.xml

# http port set
server.port=8565
#view resolver(Templete ->JSP Use)
spring.mvc.view.prefix=/WEB-INF/jsp/
spring.mvc.view.suffix=.jsp
#server.servlet.jsp.init-parameters.development=true
#custom error page
#server.error.whitelabel.enabled=false
```

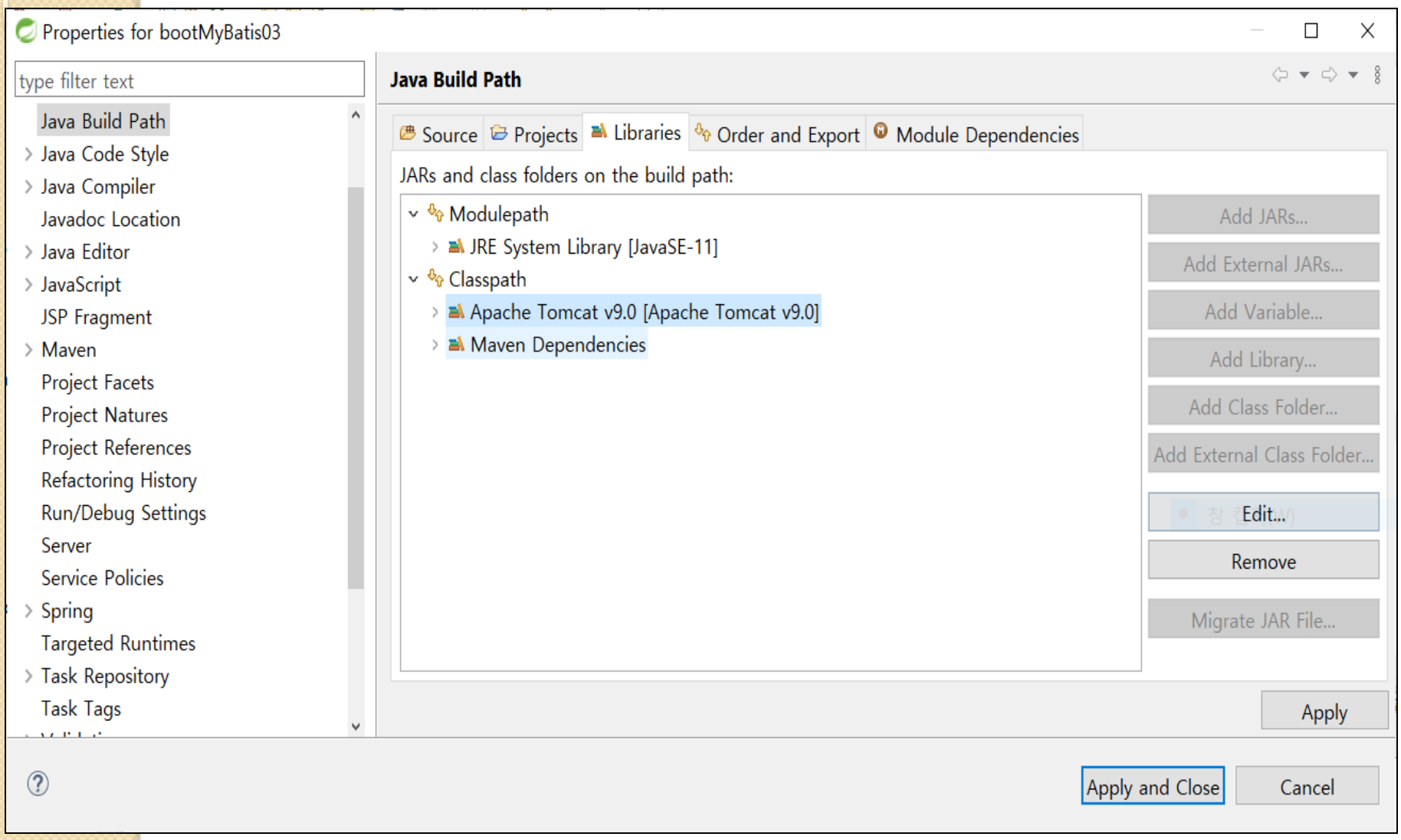


2.pom.xml(dependency) 추가

```
<!-- jsp변경 -->
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
</dependency>
<dependency>
  <groupId>org.apache.tomcat.embed</groupId>
  <artifactId>tomcat-embed-jasper</artifactId>
</dependency>
```

9-2. myBatis spring-boot 연계 (Maven-sts) 3

1.외부 Tomcat 사용 지정(default 는 내장되어 있는 tomcat 사용)



9-2. myBatis spring-boot 연계 (Maven-sts) 4

1.GMail 전송 Setting

1) application.properties

```
# gmail Transfer
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=ttaekwang3@gmail.com
spring.mail.password=ktg#@!****
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
```

2.pom.xml(dependency) 추가

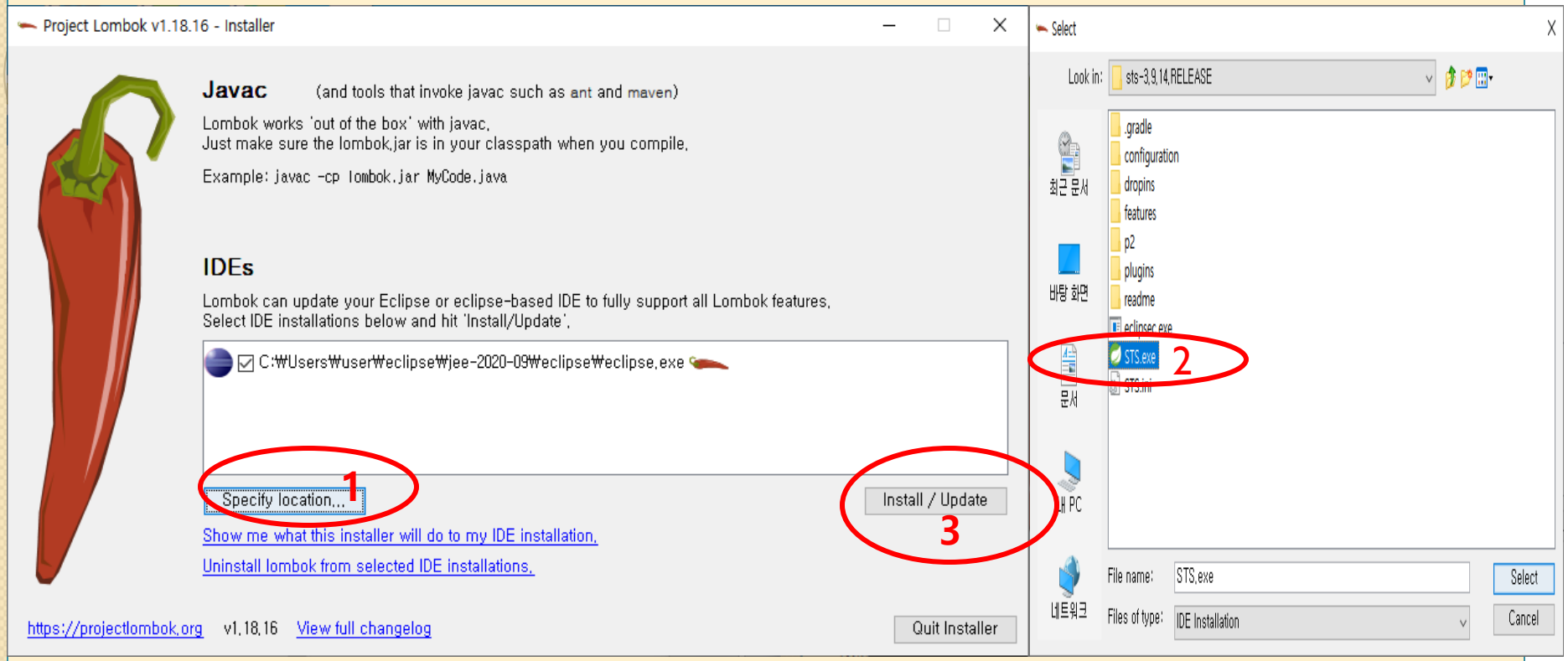
```
<!-- Mail 전송 -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
    <version>2.0.1.RELEASE</version>
</dependency>
```

1.Google Mail 설정

- 2단계 인증을 사용하는 로그인 사용 설정

10-1. Lombok 사용법

1. STS 에 Lombok(롬복)을 설치하는 방법 (**일단 STS 종료 한후 실행**)
 1) https://projectlombok.org/download 로 접속하여 D/L
 2) cmd -> 해당 폴더 이동 -> **java -jar lombok.jar**



3) STS.ini(확인)
 -vmargs
 -javaagent:lombok.jar

10-2. Lombok Project 적용

1. pom.xml에 적용
 - Project 생성시 lombok 추가

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <scope>provided</scope>
</dependency>
```

2) Gradle 적용
 - Project 생성시 lombok 추가 → gradle Refresh
 - build.gradle

```
compileOnly 'org.projectlombok:lombok'
 annotationProcessor 'org.projectlombok:lombok'
```

3) 적용시 장점
 - 기계적으로 작성하는 VO, DTO, Entity 관련 작업 용이
 - getter, Setter, ToString , hashcode 관련 Method 코드 일관성, 가독성 향상

4) 고려사항
 - 모든 팀원이 lombok 설치 해야 함
 - 추가 Annotation사용시 소스코드 분석 난해 가능성 고려

10-3. Lombok annotation

Lombok Annotation	내용
@NonNull	Null 값이 될 수 없다는 것을 명시
@Getter/Setter	코드가 컴파일될 때 속성들에 대해서 Getter/Setter 메소드들을 생성
@ToString	toString() 메소드를 생성
@EqualsAndHashCode	해당 객체의 equals() 와 hashCode() 메소드를 생성
@NoArgsConstructor	파라미터를 받지 않는 생성자를 만들어 줌
@RequiredArgsConstructor	지정된 속성들에 대해서만 생성자를 만들어 줌
@AllArgsConstructor	모든 속성들에 대해서 값을 받는 생성자를 만든다
@Data	@ToString, @EqualsAndHashCode, @Getter(모든 속성), @Setter(final 이 아닌 속성), @RequiredArgsConstructor를 합쳐둔 어노테이션
@Value	불변 클래스를 생성
@Log	자동으로 생기는 log 라는 변수를 이용해서 로그를 찍을 수 있음 자동으로 logging 위한 필드인 private static Logger logger 생성

11. Rest API 1

1. API의 개념
 - 약결합 (Decoupling)적 이며, Platform Agnostic(플랫폼에 종속적이 않음을 뜻한다. 즉, 특정 기기나 OS에서만 돌아가는 것이 아닌 광범위하게 사용)
2. REST API 개념
 - **REST(REpresentational State Transfer)란, "웹에 존재하는 모든 자원(이미지, 동영상, DB 자원)에 고유한 URI를 부여해 활용**"하는 것으로, 자원을 정의하고 자원에 대한 주소를 지정하는 방법론을 의미
 - **API의 Resources , URL이 HTTP Method와 함께 표현되도록 정한 API의 규격**
 - HTTP 프로토콜 장점을 살릴 수 있는 네트워크 기반 아키텍처
 - REST API을 구현하기위해 **HTTP method + 모든 개체 Resource화 + URL 디자인(라우팅)** 필요
 - 라우팅이란 클라이언트의 요청에 대한 결과(응답)을 어떻게 이어줄 것인가를 처리하는 것
 - URI를 이용한 접근 : 모든 개체를 리소스로 보고, 리소스에 고유번호를 부여
 - URL 디자인 원칙 : 자원에 대한 처리를 주소에 나타내지않는다(HTTP method는 내부적으로 처리하도록), 어떤 자원인지 주소에 명확하게 나타냄
 - REST API를 구현하기위해 HTTP 프로토콜에 대한 이해, method 종류, 라우팅에 대한 이해가 있어야 함
 - HTTP method를 클라이언트가 필요한 처리에 맞게 선택하여 서버에 요청

3. Client-Server 구조
 - 전체 구성 : 연결자를 통해 서버에 요청을 보내며, 서버는 해당 요청을 거절하거나 수행하고 클라이언트에 응답
 상태 정보를 저장하지 않으면 각 API 서버는 들어오는 요청의 메시지로만 처리하면 되며, 세션과 같은 Context 정보를 신경 쓸 필요가 없기 때문에 구현이 단순
 - REST 서버: API를 제공하고, 제공된 API를 이용해서 비즈니스 로직 처리 및 저장을 책임
 - 클라이언트의 : 사용자 인증이나 Context(세션, 로그인 정보)등을 직접 관리하고 책임 지는 구조로 역할
4. **Method 종류**
 - 요청의 종류를 서버에 알리기 위해 사용 , CRUD 만들 때 사용

Method 종류	내용
GET	정보를 요청하기위해 사용(Read)
POST	정보를 입력하기위해 사용(Create)
PUT	정보를 업데이트하기위해 사용(Update)
DELETE	정보를 삭제하기위해 사용(Delete)

11. Rest API 2 REST의 특성

1. Uniform Interface

- REST는 HTTP 표준만 따른다면, 어떠한 기술이든 사용이 가능한 인터페이스 스타일.
  ex) HTTP + JSON으로 REST API를 정의했다면, 안드로이드 플랫폼이건, IOS 플랫폼이건, 또는 C나 Java / Python 이건 특정 언어나 기술에 종속 받지 않고 HTTP와 JSON을 사용할 수 있는 모든 플랫폼에 사용이 가능한 느슨한 결합(Loosely Coupling) 형태의 구조

http://localhost:8586/members/new **# id 값이 2인 할 일**

2. Client-Server

- 서비스가 수행되길 기대하는 클라이언트 구성 요소는 연결자를 통해 서버에 요청을 보내며, 서버는 해당 요청을 거절하거나 수행하고 클라이언트에 응답을 보낸다. 그리고 모든 통신은 클라이언트-서버 간의 일대일로 연결

3. **Stateless**

- State가 있다/없다의 의미는 사용자나 클라이언트의 Context를 서버 쪽에 유지 하지 않는다는 의미로, 쉽게 표현하면 HTTP Session과 같은 Context 저장소에 상태 정보를 저장하지 않는 형태를 의미.
- 상태 정보를 저장하지 않으면 각 API 서버는 들어오는 요청의 메시지로만 처리하면 되며, 세션과 같은 Context 정보를 신경 쓸 필요가 없기 때문에 구현이 단순

4. **Cacheable**

- 자기 서술형 메시지 덕분에 각각의 요청에 대한 응답은 그 자체로 해석이 가능. 그렇기 때문에 독립적으로 처리가 가능한데, 이로 인해 Caching이 가능
- 서비스 시스템에서 60%에서 많게는 80% 가량의 트랜잭션이 Select와 같은 조회성 트랜잭션인 것을 감안하면, HTTP의 리소스들을 웹 캐시 서버 등에 캐싱하는 것은 용량이나 성능 면에서 많은 장점
- 구현은 HTTP 프로토콜 표준에서 사용하는 "Last-Modified" 태그나 E-Tag를 이용하면 캐싱을 구현
  E-Tag : HTTP 컨텐츠가 바뀌었는지를 검사할 수 있는 태그 , 메시지에 담겨있는 Entity를 위한 Entity 태그를 제공
- Client가 HTTP GET을 "Last-Modified" 값과 함께 보냈을 때, 컨텐츠가 변화가 없으면 REST 컴포넌트는 "304 Not Modified"를 리턴 하면 Client는 자체 캐쉬에 저장된 값을 사용

5. **Layered System**

- 서버는 클라이언트가 모르게 API 서버에 여러 계층을 추가하여 유연한 구조로 개발가능 .
- 클라이언트 입장에서는 REST API 서버만 호출. 그러나 서버는 다중 계층으로 구성
- 그 앞 단에 사용자 인증(Authentication), 암호화(SSL), 로드밸런싱 등을 하는 계층을 추가해서 구조상의 유연성

6. **Code on Demand(optional)**

- 클라이언트는 리소스에 대한 표현을 응답으로 받고 처리해야 하는데, 어떻게 처리해야 하는지에 대한 Code를 서버가 제공하는 것을 의미

11. Rest API 3

1. PUT 호출 java Script 예시

```
var Memberupdate = function(obj){
                        console.log("update.....");

                        $.ajax({
                             type:'put',
                             url:'/api/v2/members/' + obj.id,
                            dataType:'json',
                            data: JSON.stringify(obj),
                            contentType: "application/json",
                            beforeSend : function(xhr){
                                 xhr.setRequestHeader(obj.csrf.headerName, obj.csrf.token);
                                 },
                            contentType: "application/json",
                            success:location.href="/"
                          });
                        };
```

2. Form에서 PUT 방식으로 전달하기

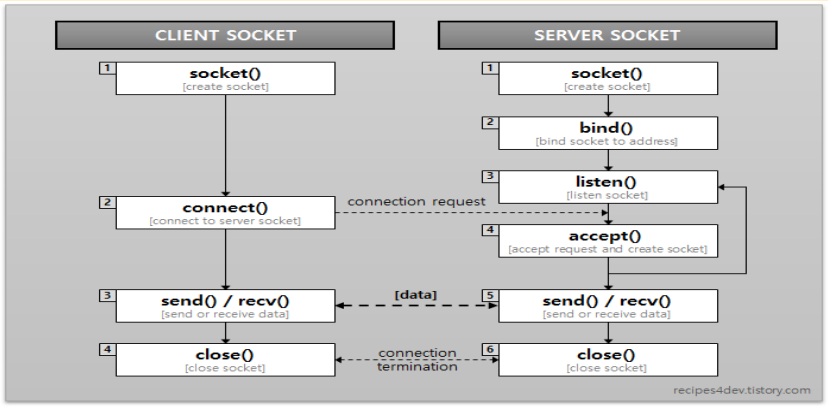
```
  <form id="putMember"    action="/api/v2/members/${id}"    method="POST" >
       <input type="hidden" name="id" value="${id}">
       <input type="hidden" name="_method" value="PUT"/>
       <input type="text" name="name" size = "50">
  </form>
```

12. gradle과 Maven 비교

항목	Gradle	Maven
사용 Project	Spring-Boot, Android	Spring Project, Spring-Boot
사용 용도	별도의 Build 스크립트를 통하여, 사용할 어플리케이션 버전, Library 등의 항목을 설정	내가 사용할 Library 뿐만 아니라 해당 Library가 작동하는데 필요한 다른 Library들까지 관리하여 NW를 통해 자동으로 D/L
적용 이유	Script 언어로 구성되어 있기 때문에, XML과 달리 변수선언, if, else, for등의 Logic 구현 가능하여 간결하게 구성 가능	프로젝트의 전체적인 Life cycle을 관리하는 도구이며, 많은 편리함과 이점이 있어 널리 사용 현재 많이 사용
관리 / Life cycle	**Library 관리** : Maven 레파지토리를 동일하게 사용할 수 있어서 설정된 서버를 통하여 Library를 D/L 받아 모두 동일한 의존성을 가진 환경을 수정 가능 하며 자신이 추가한 Library도 **Repository** 서버에 Up 가능 **Project 관리** : 모든 Project가 일관된 Directory 구조를 가지고 build Process를 유지 support . **단위 테스트** 시 의존성 관리 : junit 등을 사용하기 위해서 명시	**Clean** : 이전 빌드에서 생성된 파일들을 삭제하는 단계 Validate : 프로젝트가 올바른지 확인학고 필요한 모든 정보를 사용할 수 있는 지 확인하는 단계 Compile : 프로젝트의 소스코드를 컴파일하는 단계 Package : 실제 컴파일된 소스 코드와 리소스들을 jar등의 배포를 위한 패키지로 만드는 단계 Verify : 통합테스트 결과에 대한 검사를 실행하여 품질 기준을 충족하는지 확인하는 단계 **Install** : 패키지를 로컬 저장소에 설치하는 단계 Site : 프로젝트 문서를 생성하는 단계 Deploy : 만들어진 Package를 원격 저장소에 release하는 단계
사용 요소	Build.gradle	POM은 pom.xml파일을 말하며 Maven의 기능을 이용하기 위해 POM이 사용
Gradle 비교우위 장점	1. 설정 내용 간결 ,가독성 향상 2. Maven으로 의존관계가 복잡한 프로젝트 설정하기에는 부적절 3. 동적인 빌드는 **Groovy 스크립트**[DSL(Domain Specific Language)]로 플러그인을 호출하거나 직접 Coding 4. Gradle은 Maven보다 성능 향상	

13-1. socket Programming 개념

1. Socket Programming 흐름도



2. Server Socket Programming

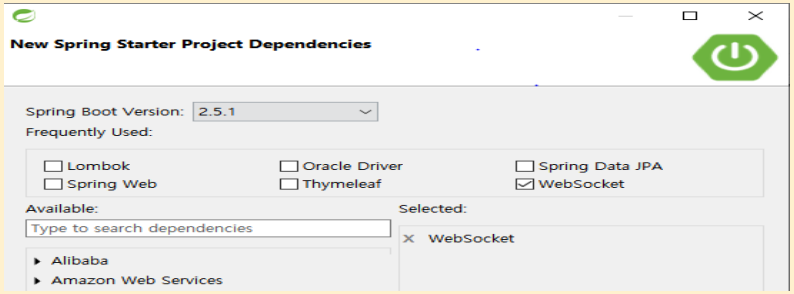
1) Server Socket 생성. (socket())
 - 소켓(Socket)을 생성(create)
2) Server Socket bind
 - O/S 가 특정 포트 번호를 Server Socket이 사용하도록 만들기 위해 Socket 과 Port 번호를 bind(결합)해야 하는데, 이 때 사용하는 API (IP 주소+포트 번호)
3) 클라이언트 연결 요청 대기. (listen())
 - 서버 소켓(Server Socket)에 포트 번호 (또는 IP 주소+포트 번호)를 결합(bind)하고 나면, 서버 소켓(Server Socket)을 통해 클라이언트의 연결 요청을 받아들일 준비
4) 클라이언트 연결 수립. (accept())
 - listen() API가 클라이언트의 연결 요청을 확인하고 문제없이 Return한다고 해서, 클라이언트와의 연결 과정이 모두 완료되는 것은 아님.
 - 최종적으로 연결 요청을 받아들이는 역할을 수행하는것은 accept() API
5) 데이터 송수신. (send()/recv())
 - 실제 Data 전달
6) 소켓 연결 종료. (close())
 - 소켓을 닫기 위해서는 close() API를 호출

3. Client Socket Programming

1) 클라이언트 소켓 생성. (socket())
 - 소켓의 종류를 지정할 수 있는데, TCP 소켓을 위해서는 Stream Type , UDP 소켓을 위해서는 Datagram Type 지정
2) 연결 요청. (connect())
 - connect() API는 "IP주소"와 "포트 번호"로 식별되는 대상(Target)으로 연결 요청
3) 데이터 송수신. (send()/recv())
 - 실제 Data 전달
 - 연결된 소켓을 통해 데이터를 보낼 때는 send(), 데이터를 받을 때는 recv API를 사용
4) 소켓 연결 종료. (close())
 - 소켓을 닫기 위해서는 close() API를 호출

13-2. chating 개요(설정)

1. Project dependencies 생성시 web Socket 설정



2. 부트 프로젝트 설정하기(pom.xml)

```
  <!--웹 소켓 사용  -->
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-websocket</artifactId>
</dependency>

<!-- json simple  json파싱을 위해
      json-simple 라이브러리를 추가 -->
<dependency>
  <groupId>com.googlecode.json-simple</groupId>
  <artifactId>json-simple</artifactId>
  <version>1.1.1</version>
</dependency>
```

3. **WebSocketConfig 설정**

1) 지정 Package
- com.oracle.bootMyBatis03.configuration

2) 웹소켓 구현체와 등록해주는 config파일을 생성, 관련 class

```
//  환경설정 annotation
@Configuration
@EnableWebSocket
public class WebSocketConfig implements WebSocketConfigurer {
       @Autowired
       SocketHandler socketHandler; // SocketHandler   DI
       @Override
        public void registerWebSocketHandlers
                (WebSocketHandlerRegistry registry) {
                     registry.addHandler(socketHandler, "/chating");
         }
}
```

4. **구현체에 등록할 SocketHandler.java**

① afterConnectionEstablished 메소드는 웹소켓 연결되면 동작.
② afterConnectionClosed 메소드는 반대로
웹소켓이 종료되면 동작
③ handleTextMessage 메소드는 메시지를 수신하면 실행
상속받은 TextWebSocketHandler는 handleTextMessage를
실행시키며, 메시지 타입에따라 handleBinaryMessage
또는 handleTextMessage가 실행
④ 웹소켓 세션을 담아둘 맵

```
HashMap<String, WebSocketSession>
  sessionMap =  new HashMap<>();
```

13-3. chating Source 주요 내용

1. 메시지 주고 받는 chat.jsp

```
<script type="text/javascript">
var ws;
// Socket 열 때 사용하는 함수
function wsOpen(){
  var wsUri  = "ws://"+location.host+
               "/bootMyBatis03/chating";
  ws = new WebSocket(wsUri);
  wsEvt();
}

// 소켓이 열리면 동작 또는 메시지를 받으면 동작
function wsEvt() {
  console.log("wsEvt  start... ");
  ws.onopen = function(data){
  //소켓이 열리면 동작
 }
  ws.onmessage = function(data) {
  //메시지를 받으면 동작
 서버에서도 데이터를 JSON형태로 전달해주기
 때문에 받은 데이터를 JSON.parse메소드 활용
 하여 Parsing.
 Parsing한 객체의 type값을 확인하여 getId값이면
 초기 설정된 값이므로 채팅창에 값을 입력하는게
 아니라 추가한 태그 sessionId에 값을 세팅
 }
 document.addEventListener("keypress",
    function(e){
        if(e.keyCode == 13){     // enter press
            send();
        }
```

```
// 사용자 이름을 등록하며 이때 Socket 생성 함수 호출 시켜줌
function chatName(){
  var userName = $("#userName").val();
  console.log("chatName  userName: " + userName);
  if(userName == null || userName.trim() == ""){
     alert("사용자 이름을 입력해주세요.");
     $("#userName").focus();
  }else{
     wsOpen();
  }
}

// Message 보낼때 Json 형식에 맞춰 전송
// JSON형태로 메시지를 보내고 서버에서도 JSON형태의
    메시지를 Parsing하여 구분처리
function send() {
  var option ={
   type: "message",
   sessionId : $("#sessionId").val(),
   userName : $("#userName").val(),
   msg : $("#chatting").val()
  }
  ws.send(JSON.stringify(option))
  $('#chatting').val("");
}
</script>
```