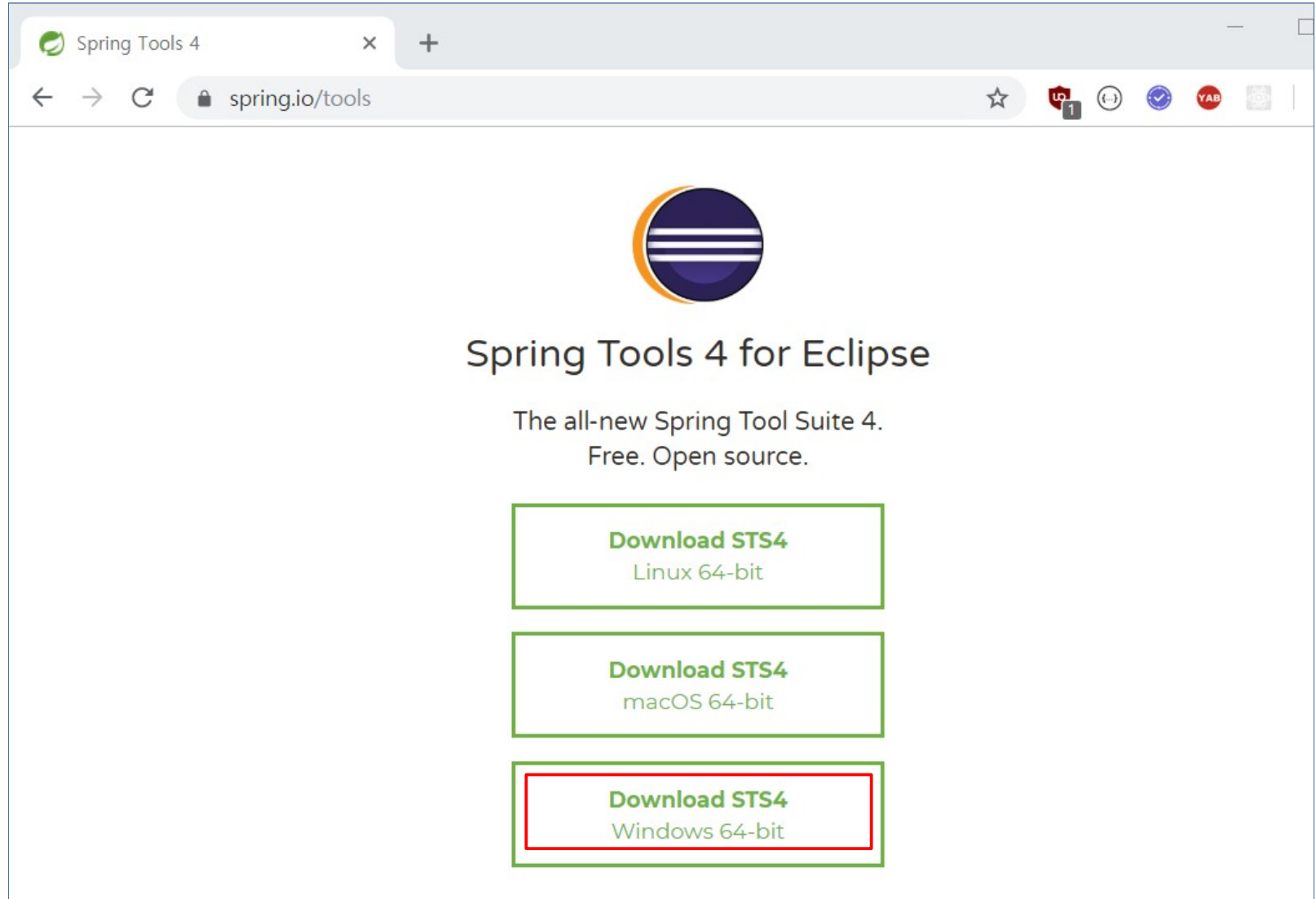


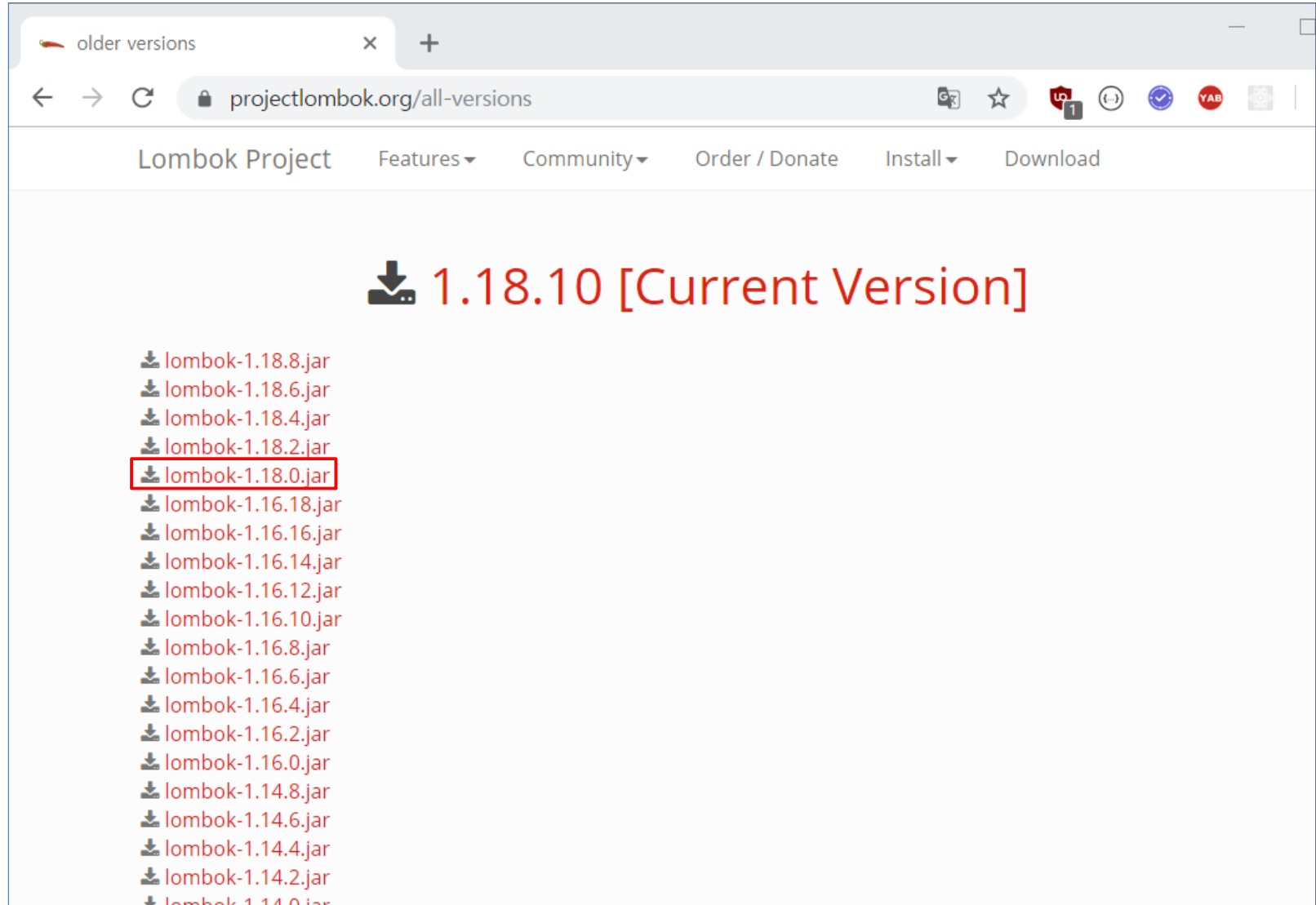
■ STS

● 다운로드 : <https://spring.io/tools>



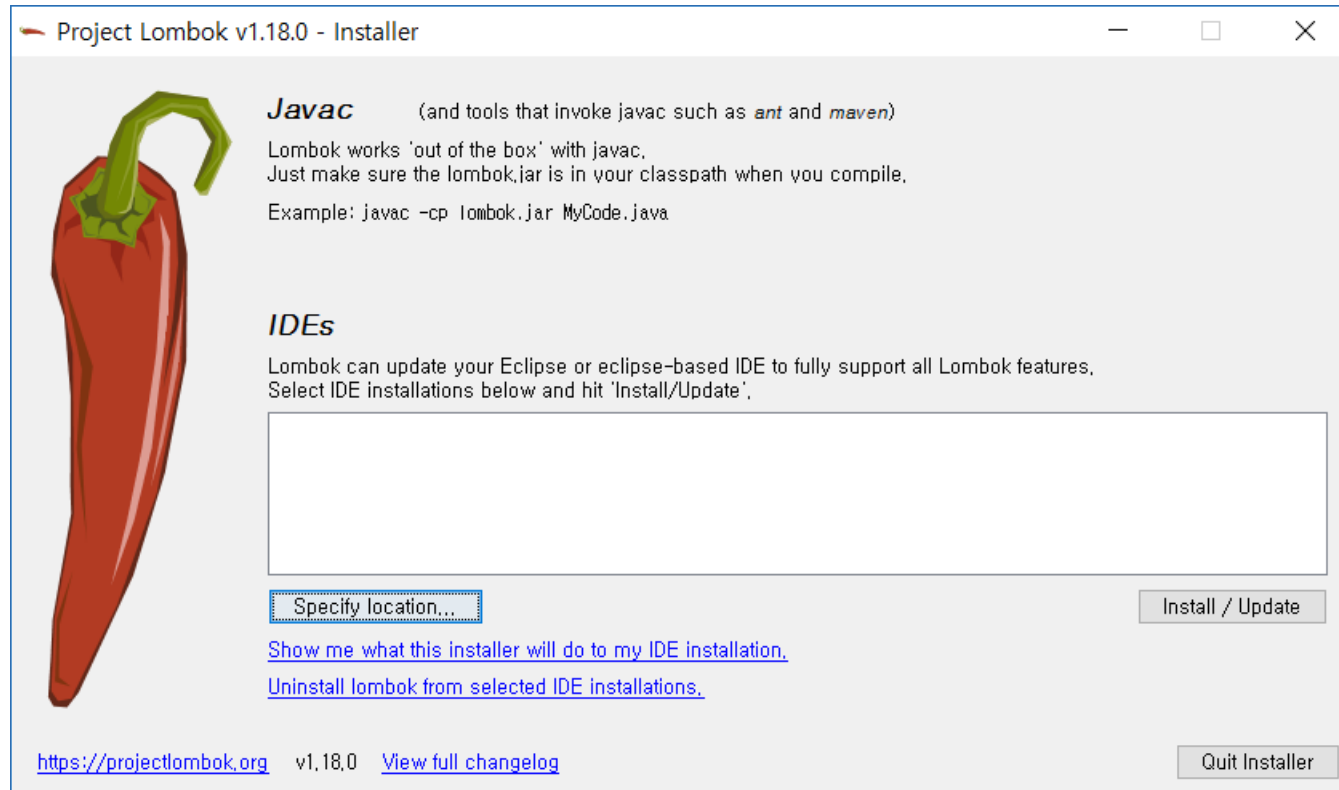
■ Lombok

● 다운로드 : <https://projectlombok.org/all-versions>



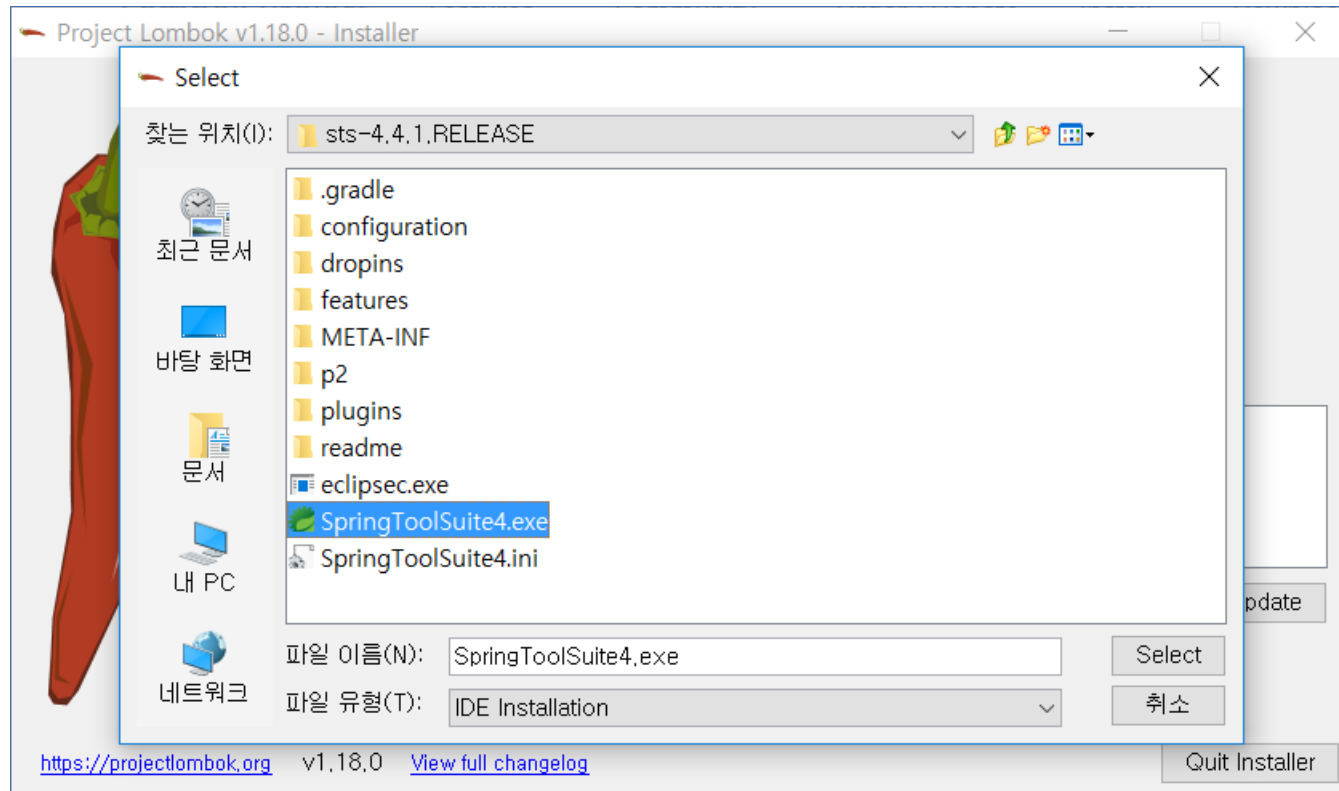
■ Lombok – STS 연동

● JAR 실행



■ Lombok – STS 연동

● STS 실행 파일 선택



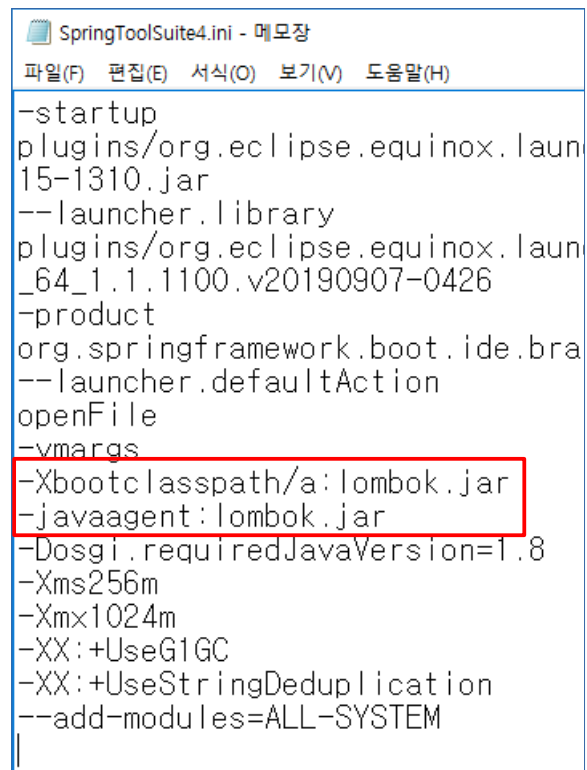
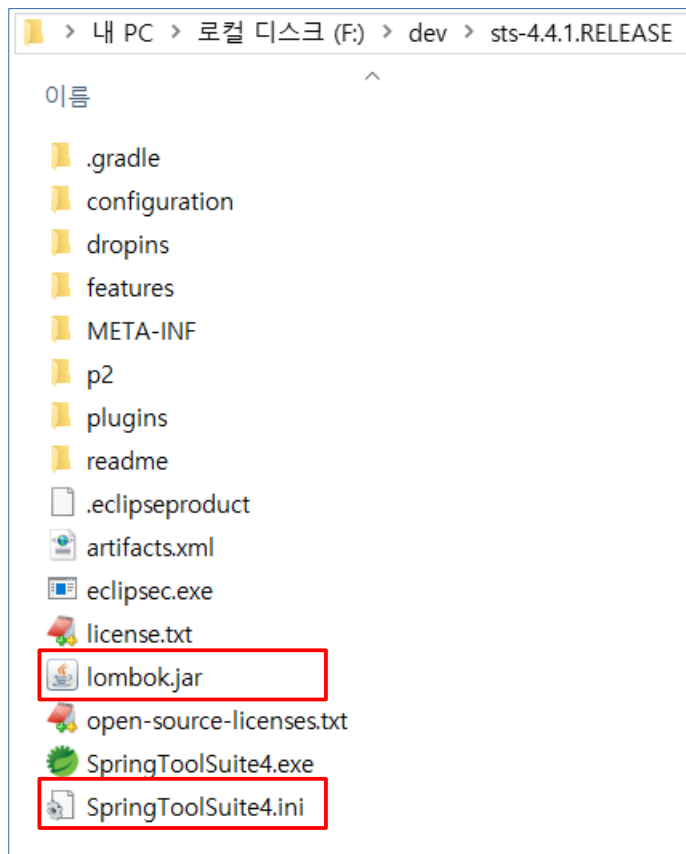
■ Lombok – STS 연동

● STS 실행 파일 선택



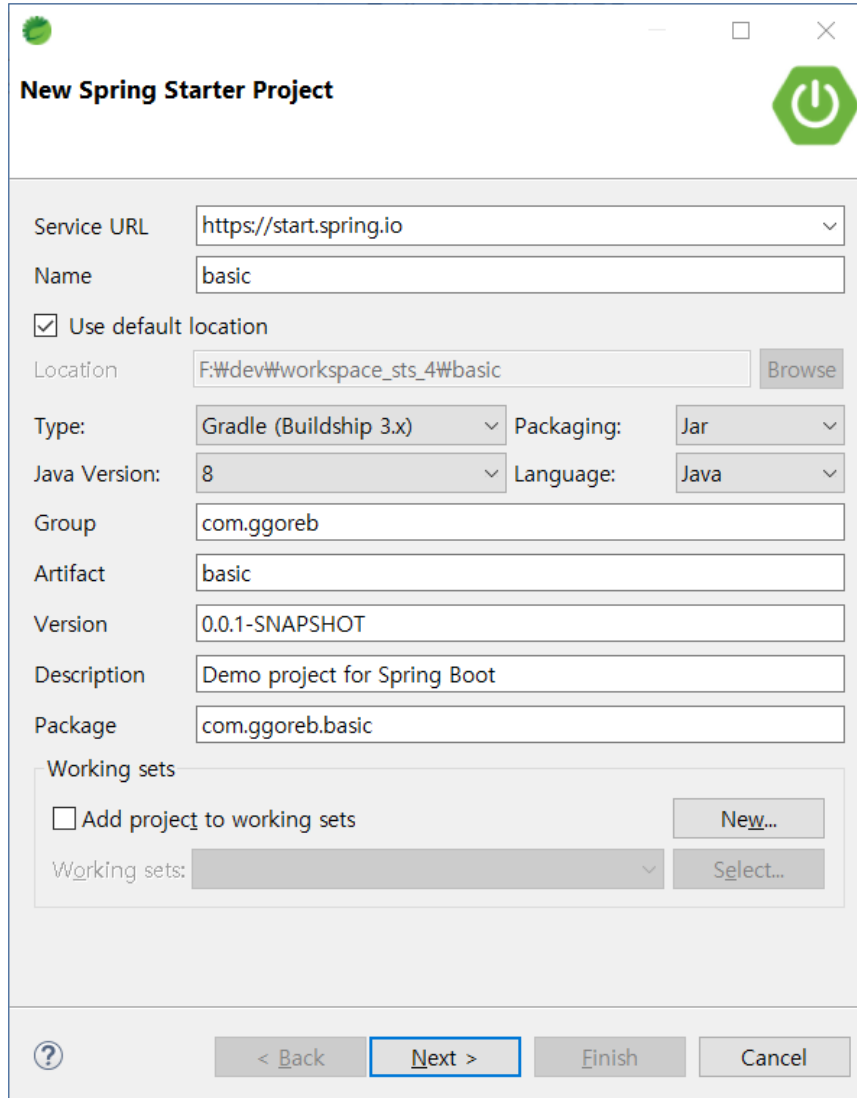
■ Lombok – STS 연동

● 결과 확인



■ 프로젝트 생성

● New – Spring Starter Project



The image shows a 'New Spring Starter Project' dialog box. It has a title bar with a green Spring logo, a maximize button, and a close button. The title 'New Spring Starter Project' is in bold, and there is a green power button icon in the top right corner. The dialog contains several input fields and checkboxes. The 'Service URL' is set to 'https://start.spring.io'. The 'Name' is 'basic'. The 'Use default location' checkbox is checked. The 'Location' is 'F:\dev\workspace_sts_4\basic', with a 'Browse' button next to it. The 'Type' is 'Gradle (Buildship 3.x)', 'Packaging' is 'Jar', 'Java Version' is '8', and 'Language' is 'Java'. The 'Group' is 'com.ggoreb', 'Artifact' is 'basic', 'Version' is '0.0.1-SNAPSHOT', 'Description' is 'Demo project for Spring Boot', and 'Package' is 'com.ggoreb.basic'. There is a 'Working sets' section with an unchecked 'Add project to working sets' checkbox, a 'New...' button, and a 'Working sets:' dropdown with a 'Select...' button. At the bottom, there is a help icon, and buttons for '< Back', 'Next >', 'Finish', and 'Cancel'.

New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

Package:

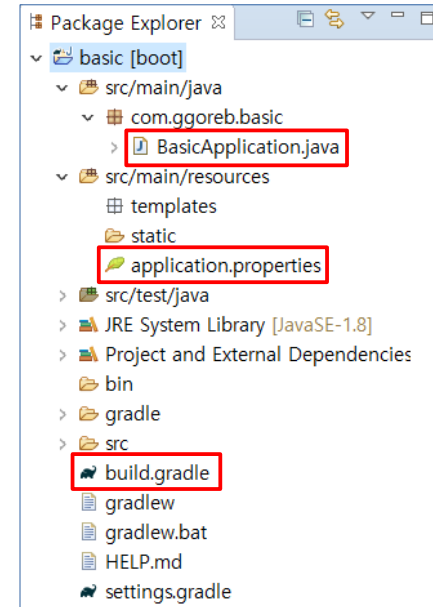
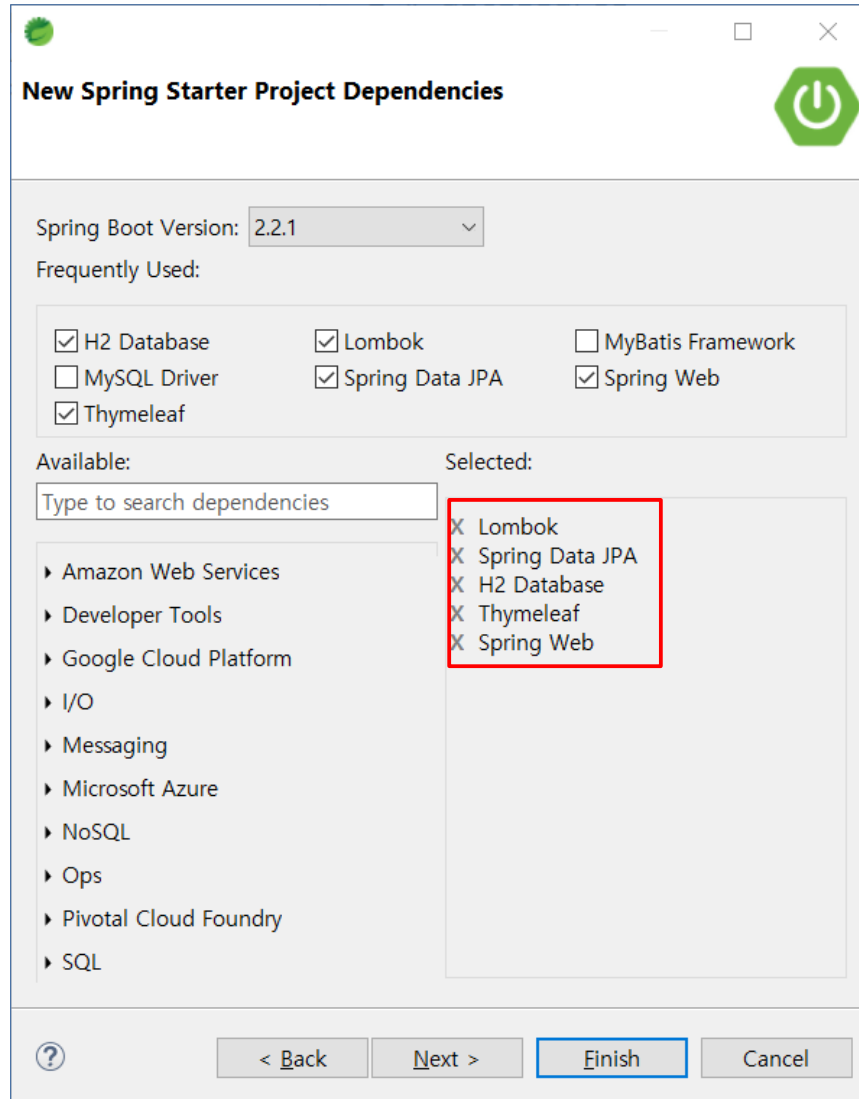
Working sets

☐ Add project to working sets

Working sets:

■ 프로젝트 생성

● New – Spring Starter Project



- Spring Web : 웹 프로젝트
- Thymeleaf : 뷰 템플릿 (HTML)
- JPA : 데이터베이스 템플릿
- H2 : 데이터베이스
- Lombok : 메소드 등 기능 보조

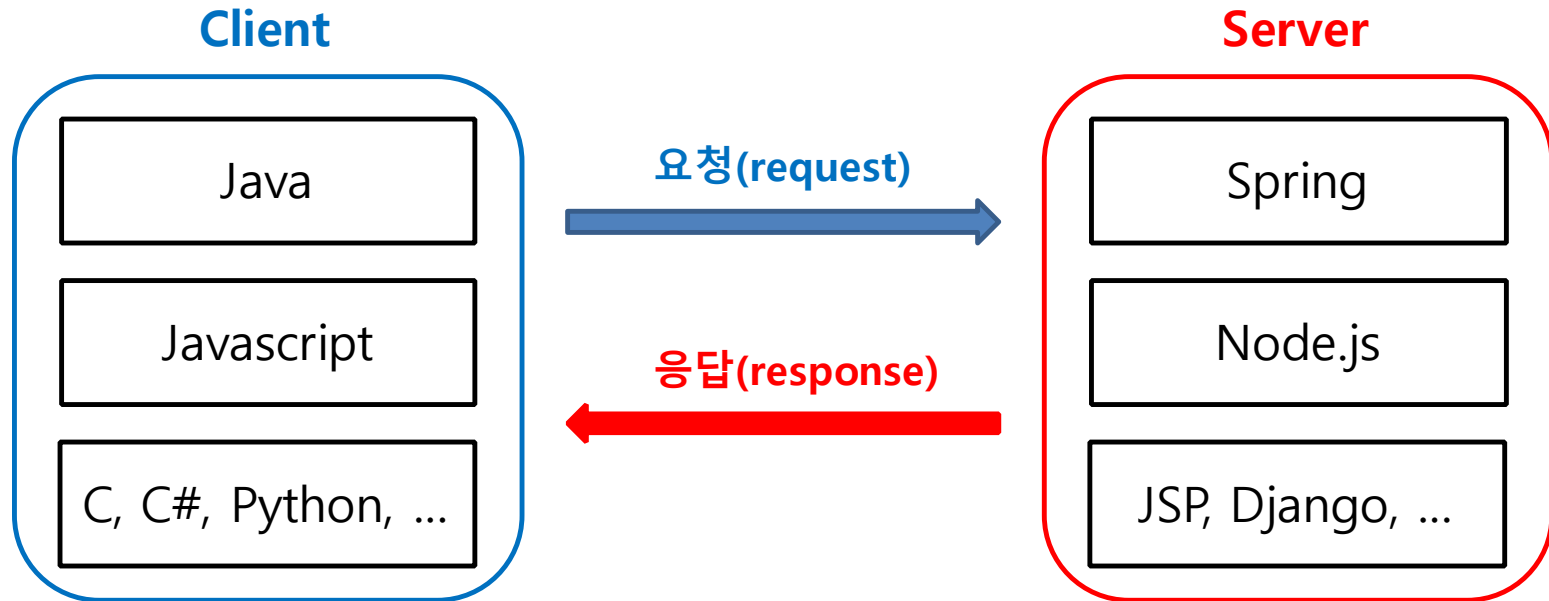
■ 프로젝트 생성

● build.gradle

```
plugins {  
    id 'org.springframework.boot' version '2.2.1.RELEASE'  
    id 'io.spring.dependency-management' version '1.0.8.RELEASE'  
    id 'java'  
}  
  
group = 'com.ggoreb'  
version = '0.0.1-SNAPSHOT'  
sourceCompatibility = '1.8'  
  
...  
  
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    compileOnly 'org.projectlombok:lombok'  
    runtimeOnly 'com.h2database:h2'  
    annotationProcessor 'org.projectlombok:lombok'  
    testImplementation('org.springframework.boot:spring-boot-starter-test') {  
        exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'  
    }  
}  
  
...
```

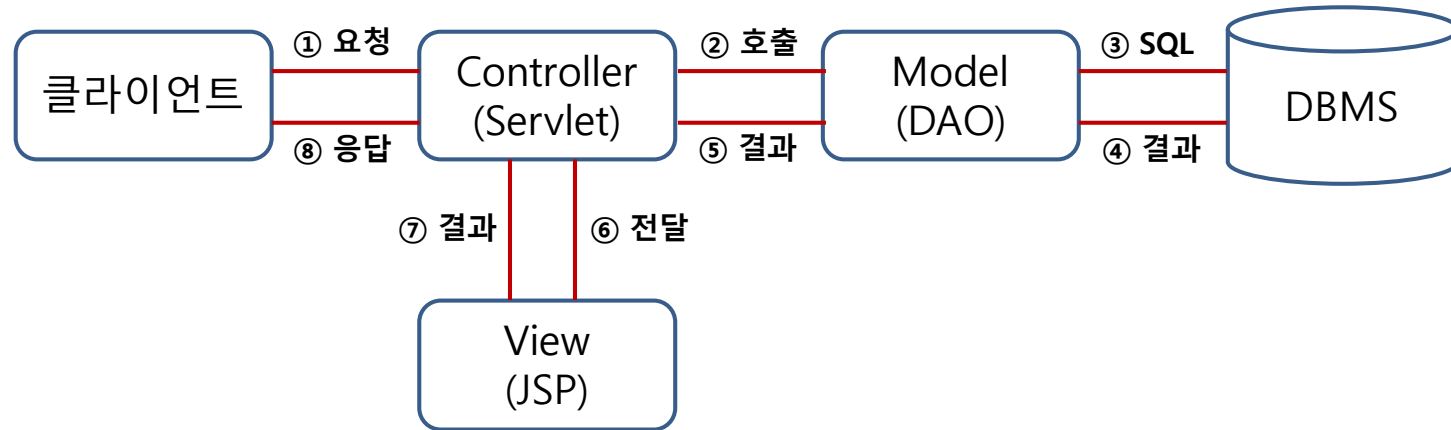
■ 클라이언트 요청 - 서버 응답

● 통신 흐름



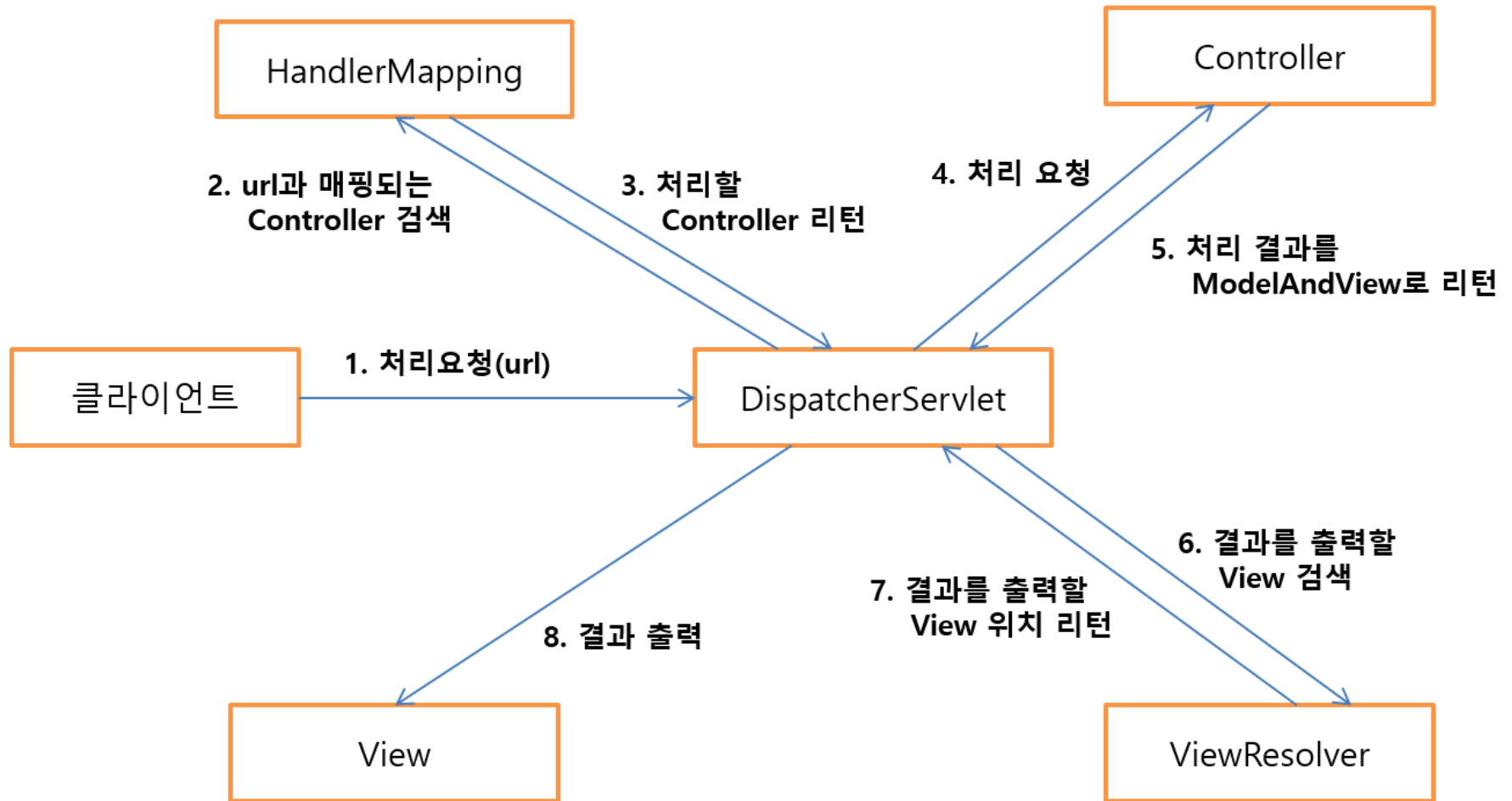
■ 클라이언트 요청 - 서버 응답

● 통신 흐름



■ 클라이언트 요청 - 서버 응답

● 스프링 MVC



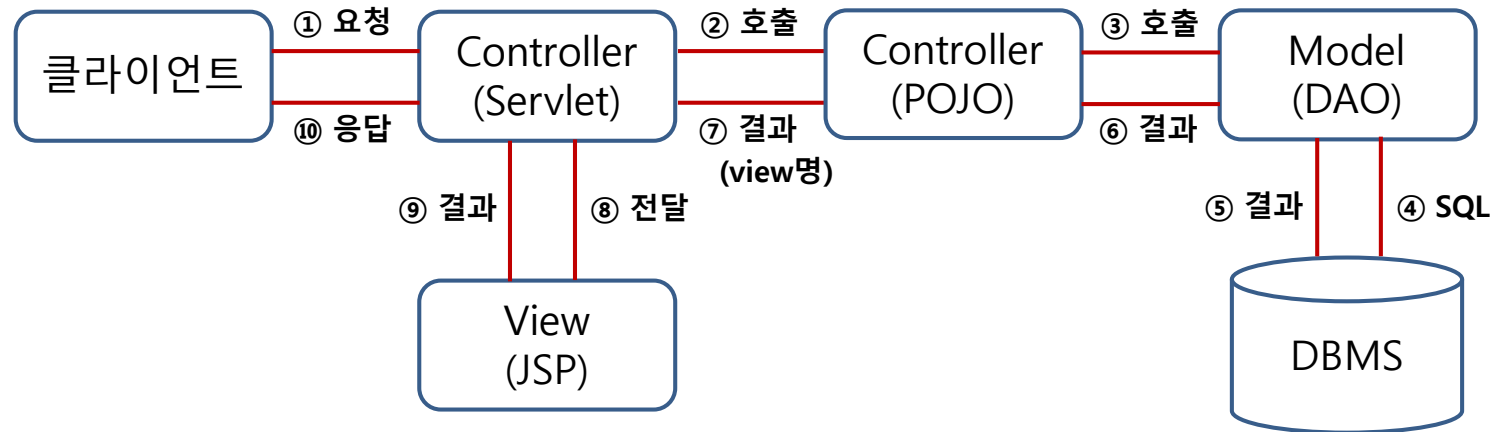
■ 클라이언트 요청 - 서버 응답

● 스프링 MVC

구성요소	설명
DispatcherServlet	클라이언트의 요청을 컨트롤러에게 전달하고 컨트롤러가 리턴하는 결과를 View로 전달
HandlerMapping	클라이언트의 요청을 어떤 컨트롤러가 처리할지 결정
Controller	클라이언트의 요청대로 로직 수행 후 결과 리턴 (Model, View)
ModelAndView	컨트롤러가 수행한 로직 결과 데이터(모델) 및 View 페이지에 대한 정보를 담음
ViewResolver	ModelAndView에서 지정된 View 명을 기반으로 결과 처리 View 결정
View	컨트롤러가 로직을 수행하고 ModelAndView에 저장된 결과를 화면으로 출력 (일반적으로 JSP 또는 Velocity 같은 뷰 템플릿 사용)

■ 클라이언트 요청 - 서버 응답

● 스프링 MVC 통신 흐름



■ Controller

● controller/HomeController.java

```
package com.ggoreb.basic.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class HomeController {
    @RequestMapping("/")
    public String home() {
        return "home";
    }
}
```

● templates/home.html

```
<html xmlns:th="http://www.thymeleaf.org">
<head>
</head>
<body>
    <h1>Home</h1>
</body>
</html>
```

■ Controller

● BasicApplication.java → Ctrl + F11

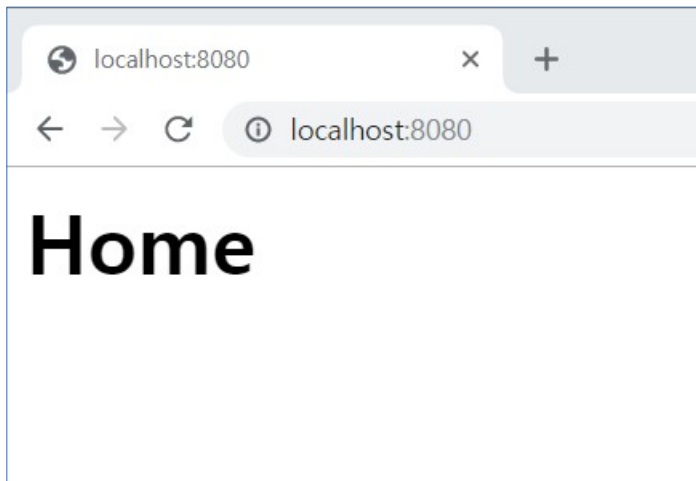
```
package com.ggoreb.basic;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class BasicApplication {

    public static void main(String[] args) {
        SpringApplication.run(BasicApplication.class, args);
    }

}
```



■ Log – Logger 사용

● controller/HomeController.java

```
package com.ggoreb.basic.controller;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class HomeController {
    Logger logger = LoggerFactory.getLogger(HomeController.class);

    @RequestMapping("/")
    public String home() {
        logger.trace("trace!");
        logger.debug("debug!");
        logger.info("info!");
        logger.warn("warn!");
        logger.error("error!");

        return "home";
    }
}
```

```
INFO 213964 --- [nio-8080-exec-1] c.g.basic.controller.HomeController : info!
WARN 213964 --- [nio-8080-exec-1] c.g.basic.controller.HomeController : warn!
ERROR 213964 --- [nio-8080-exec-1] c.g.basic.controller.HomeController : error!
```

■ Log – Logger 사용

● application.properties

```
# log level  
logging.level.com.ggoreb.basic=trace
```

```
TRACE 238424 --- [nio-8080-exec-1] c.g.basic.controller.HomeController : trace!  
DEBUG 238424 --- [nio-8080-exec-1] c.g.basic.controller.HomeController : debug!  
INFO 238424 --- [nio-8080-exec-1] c.g.basic.controller.HomeController : info!  
WARN 238424 --- [nio-8080-exec-1] c.g.basic.controller.HomeController : warn!  
ERROR 238424 --- [nio-8080-exec-1] c.g.basic.controller.HomeController : error!
```

● documentation

– <http://logback.qos.ch/manual/>

■ Log – Slf4j(Lombok) 사용

● controller/HomeController.java

```
package com.ggoreb.basic.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@Controller
public class HomeController {

    @RequestMapping("/")
    public String home() {
        log.trace("trace!");
        log.debug("debug!");
        log.info("info!");
        log.warn("warn!");
        log.error("error!");

        return "home";
    }
}
```

```
TRACE 246412 --- [nio-8080-exec-1] c.g.basic.controller.HomeController : trace!
DEBUG 246412 --- [nio-8080-exec-1] c.g.basic.controller.HomeController : debug!
INFO 246412 --- [nio-8080-exec-1] c.g.basic.controller.HomeController : info!
WARN 246412 --- [nio-8080-exec-1] c.g.basic.controller.HomeController : warn!
ERROR 246412 --- [nio-8080-exec-1] c.g.basic.controller.HomeController : error!
```

■ SpringBoot DevTools – 서버 재기동없이 수정사항 적용

● build.gradle

```
plugins {  
    id 'org.springframework.boot' version '2.2.1.RELEASE'  
    id 'io.spring.dependency-management' version '1.0.8.RELEASE'  
    id 'java'  
}  
  
group = 'com.ggoreb'  
version = '0.0.1-SNAPSHOT'  
sourceCompatibility = '1.8'  
  
...  
  
Dependencies {  
  
    ...  
  
    implementation "org.springframework.boot:spring-boot-devtools"  
}  
  
test {  
    useJUnitPlatform()  
}
```

■ SpringBoot DevTools – 서버 재기동없이 수정사항 적용

● Refresh Gradle Project

The screenshot shows an IDE interface with a context menu open. The menu includes options like Undo, Save, Cut, Copy, Paste, and Run As. The 'Gradle' option is selected, and a sub-menu is visible with 'Refresh Gradle Project' highlighted. In the background, a progress window is open with tabs for Javadoc, Declaration, Console, Progress, and JUnit. The Progress tab is active, showing the text 'Synchronize Gradle projects with workspace' and 'Importing root project: Build' with a progress bar.

Context Menu Options:

- Undo (Ctrl+Z)
- Revert File
- Save (Ctrl+S)
- Open With >
- Show In (Alt+Shift+W) >
- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Paste (Ctrl+V)
- Quick Fix (Ctrl+1)
- Shift Right
- Shift Left
- Add to Snippets...
- Run As >
- Debug As >
- Profile As >
- Source >
- Find References
- Refactorings >
- Format (Ctrl+Shift+F)
- Code Actions >
- Validate
- Gradle >**
 - Refresh Gradle Project**
- Team >
- Compare With >
- Replace With >
- Preferences...

Progress Window:

- Tabs: @ Javadoc, Declaration, Console, Progress, JUnit
- Progress: Synchronize Gradle projects with workspace
- Importing root project: Build

■ 응답 처리 – View / ViewResolver

- HTML / JSON / XML / Excel / PDF / Image / File (zip, exe 등)

- HTML

 - String / void / Map / Model / ModelAndView / DTO

- JSON

 - Map / DTO / List

Board

조회한 게시물의 번호 1

```
▼ {  
    "id": "아이디",  
    "pw": "비밀번호"  
}
```

This XML file does not appear to have any style

```
▼ <person>  
    <number>1</number>  
    <name>kim</name>  
</person>
```

■ View / ViewResolver

● controller/HtmlController.java

```
package com.ggoreb.basic.controller;

import java.util.HashMap;
import java.util.Map;

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

import com.ggoreb.basic.model.Member;

@Controller
public class HtmlController {
    @GetMapping("html/string")
    public String html() {
        return "html/string";
    }

    @GetMapping("html/void")
    public void htmlVoid() {
    }

    @GetMapping("html/map")
    public Map<String, Object> htmlMap(Map<String, Object> map) {
        Map<String, Object> map2 = new HashMap<String, Object>();
        return map2;
    }
}
```

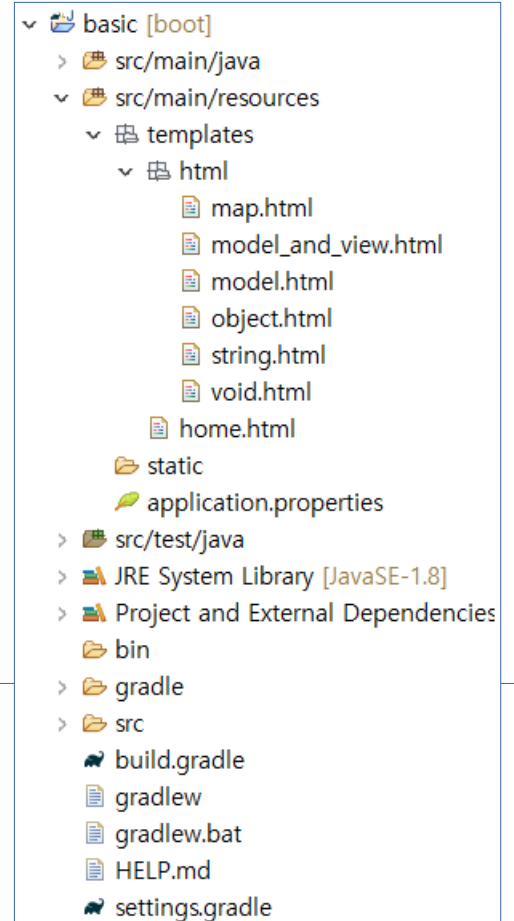
■ View / ViewResolver

● controller/HtmlController.java

```
@GetMapping("html/model")
public Model htmlModel(Model model) {
    return model;
}

@GetMapping("html/model_and_view")
public ModelAndView htmlModel() {
    ModelAndView mav = new ModelAndView();
    mav.setViewName("html/model_and_view");
    return mav;
}

@GetMapping("html/object")
public Member htmlObject() {
    Member member = new Member();
    member.setName("kim");
    return member;
}
}
```



■ View / ViewResolver

● model/Member.java

```
package com.ggoreb.basic.model;

import lombok.Data;

@Data
public class Member {
    private String name;
    private String userId;
    private String userPassword;
}
```

■ View / ViewResolver

● templates/html

```
<html>
<head>
</head>
<body>
    <h1>HTML String</h1>
</body>
</html>
```

← → ↻ ⓘ localhost:8080/html/string

HTML String

```
<html>
<head>
</head>
<body>
    <h1>HTML void</h1>
</body>
</html>
```

← → ↻ ⓘ localhost:8080/html/void

HTML void

```
<html>
<head>
</head>
<body>
    <h1>HTML map</h1>
</body>
</html>
```

← → ↻ ⓘ localhost:8080/html/map

HTML map

```
<html>
<head>
</head>
<body>
    <h1>HTML model</h1>
</body>
</html>
```

← → ↻ ⓘ localhost:8080/html/model

HTML model

■ View / ViewResolver

● templates/html

```
<html>
<head>
</head>
<body>
    <h1>HTML model and view</h1>
</body>
</html>
```

← → ↻ ⓘ localhost:8080/html/model_and_view

HTML model and view

```
<html
    xmlns:th="http://www.thymeleaf.org">
<head>
</head>
<body>
    <h1>HTML object</h1>
    [[${member}]]
    <hr>
    [[${member.name}]]
</body>
</html>
```

← → ↻ ⓘ localhost:8080/html/object

HTML object

Member(name=kim, userId=null, userPassword=null)

kim

■ View / ViewResolver

● controller/Json1Controller.java – @ResponseBody

```
package com.ggoreb.basic.controller;

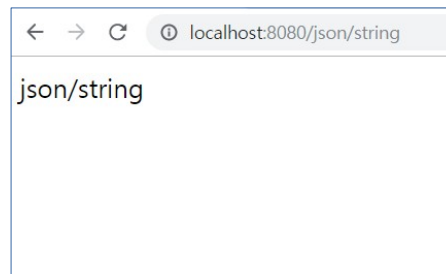
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import com.ggoreb.basic.model.Member;
```

```
@Controller
public class Json1Controller {
    @GetMapping("json/string")
    @ResponseBody
    public String json() {
        return "json/string";
    }

    @GetMapping("json/map")
    @ResponseBody
    public Map<String, Object> jsonMap(Map<String, Object> map) {
        Map<String, Object> map2 = new HashMap<String, Object>();
        map2.put("key1", "value");
        map2.put("key2", 2324);
        map2.put("key3", true);
        return map2;
    }
}
```



■ View / ViewResolver

● controller/Json1Controller.java – @ResponseBody

```
@GetMapping("json/object")
@ResponseBody
public Member jsonObject() {
    Member member = new Member();
    member.setName("kim");
    return member;
}

@GetMapping("json/list")
@ResponseBody
public List<String> jsonList() {
    List<String> list = new ArrayList<>();
    list.add("1");
    list.add("2");
    list.add("3");
    return list;
}
```



```
{
  "name": "kim",
  "userId": null,
  "userPassword": null
}
```



```
[
  "1",
  "2",
  "3"
]
```

■ View / ViewResolver

● controller/Json2Controller.java – @RestController

```
package com.ggoreb.basic.controller;

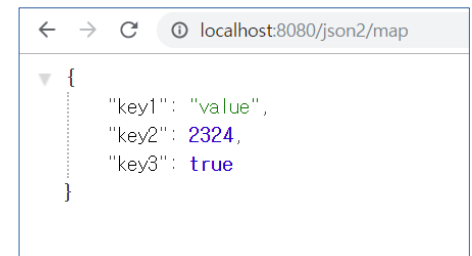
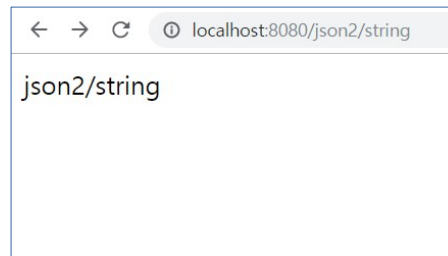
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import com.ggoreb.basic.model.Member;

@RestController
public class Json2Controller {
    @GetMapping("json2/string")
    public String json() {
        return "json2/string";
    }

    @GetMapping("json2/map")
    public Map<String, Object> jsonMap(Map<String, Object> map) {
        Map<String, Object> map2 = new HashMap<String, Object>();
        map2.put("key1", "value");
        map2.put("key2", 2324);
        map2.put("key3", true);
        return map2;
    }
}
```



■ View / ViewResolver

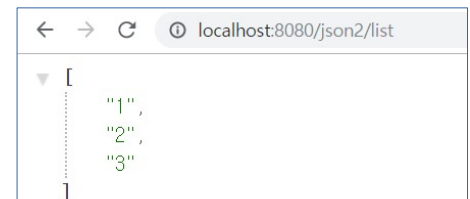
● controller/Json2Controller.java – @RestController

```
@GetMapping("json2/object")
public Member jsonObject() {
    Member member = new Member();
    member.setName("kim");
    return member;
}

@GetMapping("json2/list")
public List<String> jsonList() {
    List<String> list = new ArrayList<>();
    list.add("1");
    list.add("2");
    list.add("3");
    return list;
}
```



```
{
  "name": "kim",
  "userId": null,
  "userPassword": null
}
```



```
[
  "1",
  "2",
  "3"
]
```

■ 크롬 플러그인 설치

● restlet client 검색

The screenshot shows the Chrome Web Store interface. At the top, the search bar contains 'restlet client' and is highlighted with a red box. Below the search bar, a dropdown menu lists several suggestions: 'restlet client', 'restlet clientrest', 'restlet cliente', 'restlet clientextension', 'restlet clientapplication', 'restlet clientchromebook', and 'restlet clientchromebook ver:'. To the right of the search bar, the text '확장 프로그램' (Extension) is visible. Further right, a button says '확장 프로그램 1개 중 1개' (1 of 1 extension). Below the search bar, there are two checkboxes: 'Android에서 사용 가능' (Available on Android) and 'Google 드라이브에서 작동' (Works on Google Drive). At the bottom left, the word '평점' (Rating) is visible. The main content area shows a search result for 'Talend API Tester - Free Edition'. The title is highlighted with a red box. Below the title, it says '제공자: Talend' (Provider: Talend) and 'Visually interact with REST, SOAP and HTTP'. There is a blue button labeled '평가하기' (Review). Below the button, it shows a star rating of 5 stars and the text '3,528 개발자 도구' (3,528 developer tools).

chrome 웹 스토어

seorab81@gmail.com

restlet client

restlet clientrest

restlet cliente

restlet clientextension

restlet clientapplication

restlet clientchromebook

restlet clientchromebook ver:

확장 프로그램

확장 프로그램 1개 중 1개

Talend API Tester - Free Edition

제공자: Talend

Visually interact with REST, SOAP and HTTP

★★★★★ 3,528 개발자 도구

평가하기

Android에서 사용 가능

Google 드라이브에서 작동

평점

■ 크롬 플러그인 설치

● restlet client

The screenshot shows the Restlet Client web application interface. At the top, there is a navigation bar with tabs for CLIENT, REQUESTS (selected), SCENARIOS, Settings, Pricing, Help, and Sign in. Below the navigation bar, there is a search bar with the text "type a name" and buttons for Save, 2 Code, and Reset. A button for "Add an environment" is also present.

The main section is titled "REQUEST". It contains a form for creating a request. The METHOD is set to POST. The URL is http://ggoreb.com/ajax/json1.jsp. The length of the request is 32 bytes. There is a Send button.

Below the URL, there is a section for QUERY PARAMETERS. Below that, there is a section for HEADERS and BODY. The HEADERS section has a dropdown menu set to Form. The BODY section has a dropdown menu set to Form. There is a checkbox for "Content-Type" set to application/x-www-form-urlencoded. There is a button for "Add form parameter".

At the bottom, there is a section titled "RESPONSE". It shows a status of 200 OK. There is a message "Cache Detected - Elapsed Time: 74ms".

■ 크롬 플러그인 설치

● restlet client

The screenshot displays a web browser's developer console with the 'RESPONSE' tab selected. A green banner at the top indicates a '200 OK' status. Below this, the 'HEADERS' and 'BODY' sections are visible. The 'HEADERS' section lists: Connection: keep-alive, Content-Length: 48 bytes, Content-Type: application/json; charset=utf-8, Date: 2017 Jul 12 16:10:24 -3s, and Server: nginx. The 'BODY' section shows a JSON object with 'id' and 'pw' fields. A 'Cache Detected - Elapsed Time: 15ms' message is in the top right. A 'length: 48 bytes' label is at the bottom right.

RESPONSE Cache Detected - Elapsed Time: 15ms

200 OK

HEADERS pretty ▼

Connection: keep-alive
Content-Length: 48 bytes
Content-Type: application/json; charset=utf-8
Date: 2017 Jul 12 16:10:24 -3s
Server: nginx

BODY pretty ▼


```
{  
  id : "아이디",  
  pw : "비밀번호"  
}
```

[lines](#) [nums](#) length: 48 bytes

▶ COMPLETE REQUEST HEADERS

■ 크롬 플러그인 설치

● json formatter 검색

 Chrome 웹 스토어

json formatter x

« 홈

☐ 확장 프로그램

☐ 테마

☐ 앱

특성

☐ 오프라인 실행 가능

☐ Google 제공

☐ 무료

☐ Android에서 사용 가능

☐ Google 드라이브에서 작동

평점

☐ ★★★★★


☐ ★★★★★★ 이상

로그인 ⚙

확장 프로그램

확장 프로그램 검색결과 더보기

추가됨




JSON Formatter
callumlocke.co.uk 제공
Makes JSON easy to read. Open source.

★ 평가하기

개발자 도구

★★★★★ (974)




JSON Formatter
www.cafeboy.org 제공
Makes JSON easy to read. Open source.

+ CHROME에 추가

개발자 도구

★★★★★ (2)



In-place JSON Formatter
Ayush Agarwal
This plugin gives you popup on your chrome browser which can be used to validate and format your JSON.

+ CHROME에 추가

개발자 도구

★★★★★ (3)

앱

앱 검색결과 더보기

■ 크롬 플러그인 설치

● JSON Formatter 설치 전



A screenshot of a web browser window. The address bar shows the URL `ggoreb.com/ajax/json1.jsp`. The main content area displays a JSON object without any formatting:

```
{  
    "id" : "아이디",  
    "pw" : "비밀번호"  
}
```

● JSON Formatter 설치 후



A screenshot of a web browser window, identical to the previous one, but with the JSON data formatted. A small downward arrow icon is visible to the left of the opening curly brace. The JSON object is now visually structured with indentation and color-coding:

```
▼ {  
    "id": "아이디",  
    "pw": "비밀번호"  
}
```

■ 요청 처리

- GET – 데이터를 가져오기
- POST – 데이터 저장
- PUT – 데이터 수정
- DELETE – 데이터 삭제

결과	상태코드	설명
Success	200	요청에 대해 정상 응답
Bad request	400	파라미터 등 요구 조건 미충족
Forbidden	403	접근 거부
Not Found	404	잘못되었거나 존재하지 않는 주소
Method Not Allowed	405	사용 불가 메소드 사용
Internal Server Error	500	서버의 프로그램 오류 발생

■ 요청 처리

● GET

요청 메소드 : GET
요청 파라미터 : 1234
제출

요청 후 응답 결과

Method => GET
param => 1234

Name	Status
<input type="checkbox"/> method.jsp?param=1234	200

● POST

요청 메소드 : POST
요청 파라미터 : 1234
제출

요청 후 응답 결과

Method => POST
param => 1234

Name	Status
<input type="checkbox"/> method.jsp	200

● PUT

요청 메소드 : PUT
요청 파라미터 : 1234
제출

요청 후 응답 결과

Method => PUT
param => 1234

Name	Status
<input type="checkbox"/> method.jsp?param=1234	200

● DELETE

요청 메소드 : DELETE
요청 파라미터 : 1234
제출

요청 후 응답 결과

Method => DELETE
param => 1234

Name	Status
<input type="checkbox"/> method.jsp?param=1234	200

■ 요청 처리 – Method

● controller/MethodController.java

```
package com.ggoreb.basic.controller;

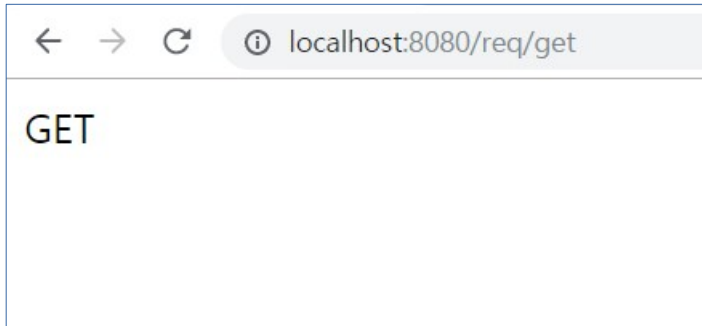
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class MethodController {
    @GetMapping("req/get")
    @RequestMapping(value="req/get", method=RequestMethod.GET)
    public String get() {
        return "GET";
    }

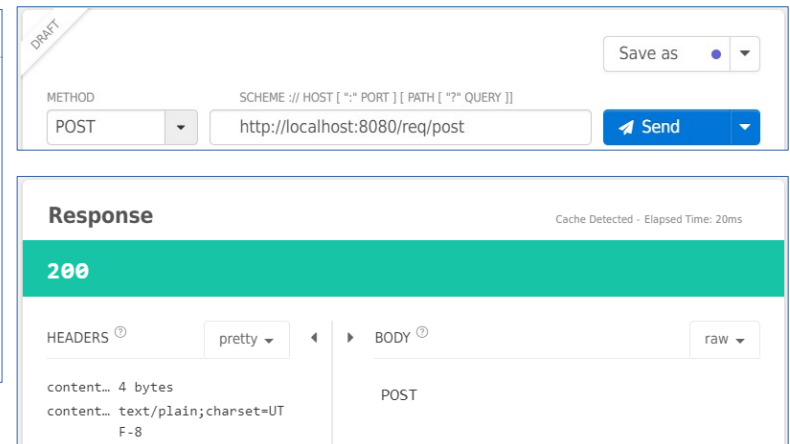
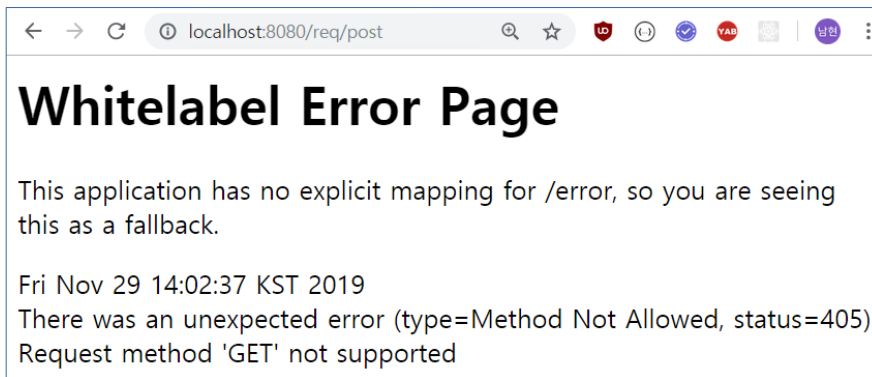
    @PostMapping("req/post")
    @RequestMapping(value="req/post", method=RequestMethod.POST)
    public String post() {
        return "POST";
    }
}
```

■ 요청 처리 – Method

● GET



● POST



■ 요청 처리 – Method

● controller/MethodController.java

```
package com.ggoreb.basic.controller;

import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class MethodController {

    ...

    @PutMapping("req/put")
    @RequestMapping(value="req/put", method=RequestMethod.PUT)
    public String put() {
        return "PUT";
    }

    @DeleteMapping("req/delete")
    @RequestMapping(value="req/delete", method=RequestMethod.DELETE)
    public String delete() {
        return "DELETE";
    }
}
```

■ 요청 처리 – Method

● PUT

DRAFT

Save as

METHOD

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]

PUT

http://localhost:8080/req/put

Send

Response

Cache Detected - Elapsed Time: 17ms

200

HEADERS

pretty

BODY

raw

Content-... text/plain; charset=UTF-8

Content-... 3 bytes

PUT

● DELETE

DRAFT

Save as

METHOD

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]

DELETE

http://localhost:8080/req/delete

Send

Response

Cache Detected - Elapsed Time: 8ms

200

HEADERS

pretty

BODY

raw

content... 6 bytes

content-... text/plain; charset=UTF-8

DELETE

■ 요청 처리

- **HttpServletRequest** - 가장 전통적으로 사용되는 방식
- **RequestParam** (편리함)
 - 파라미터 명칭에 맞게 변수 사용
 - 파라미터 종류 및 개수 상관없이 사용
- **PathVariable** - 요청 주소의 경로명 활용
- **ModelAttribute** (명확함)
 - Model / DTO / VO 등 객체와 연계하여 활용
 - JPA, MyBatis 등 ORM 프레임워크 활용
- **RequestBody**
 - 보편적인 요청 파라미터 형식을 사용하지 않고 JSON 형태의 파라미터 사용
 - 사용 시 메소드 방식을 POST로 지정

■ 요청 처리 - HttpServletRequest

● controller/RequestController.java

```
package com.ggoreb.basic.controller;

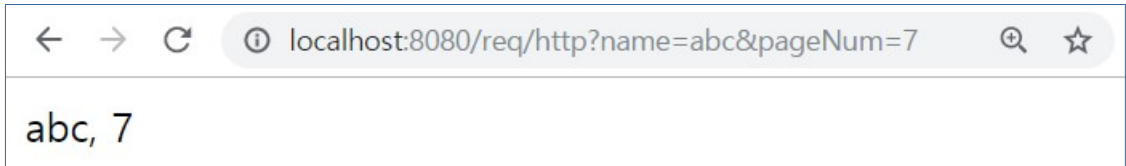
import java.util.Map;

import javax.servlet.http.HttpServletRequest;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.ggoreb.basic.model.Member;

@RestController
public class RequestController {
    @GetMapping("req/http")
    public String http(HttpServletRequest request) {
        String name = request.getParameter("name");
        String pageNum = request.getParameter("pageNum");
        return name + ", " + pageNum;
    }
}
```



■ 요청 처리 - @RequestParam

- 요청 시 지정된 파라미터가 없는 경우 400 오류 (기본값)
- defaultValue 또는 required 옵션으로 파라미터 사용 여부 지정
- 지정 여부와 관계없이 모든 파라미터 사용 시 Map으로 사용
- controller/RequestController.java

```
@GetMapping("req/param1")  
public String param1(  
    @RequestParam("key1") String key1,  
    @RequestParam("key2") String key2) {  
    return key1 + ", " + key2;  
}
```

← → ↻ ⓘ localhost:8080/req/param1?key1=v1&key2=v2 🔍 ☆

v1, v2

```
@GetMapping("req/param2")  
public String param2(  
    @RequestParam Map<String, Object> map) {  
    return map.toString();  
}
```

← → ↻ ⓘ localhost:8080/req/param2?search=java&version=1.8 🔍 ☆

{search=java, version=1.8}

■ 요청 처리 - @PathVariable

- 요청을 처리하는 주소에 {변수명} 형식으로 지정
- @PathVariable을 이용하여 {변수명} 과 동일한 명칭 지정 후 변수 사용
- 명칭 미지정 시 순서대로 입력
- controller/RequestController.java

```
@GetMapping("req/path/{path1}/{path2}")  
public String path(  
    @PathVariable("path1") String path1,  
    @PathVariable("path2") String path2) {  
    return path1 + ", " + path2;  
}
```

← → ↻ ⓘ localhost:8080/req/path/123/456 🔍 ☆

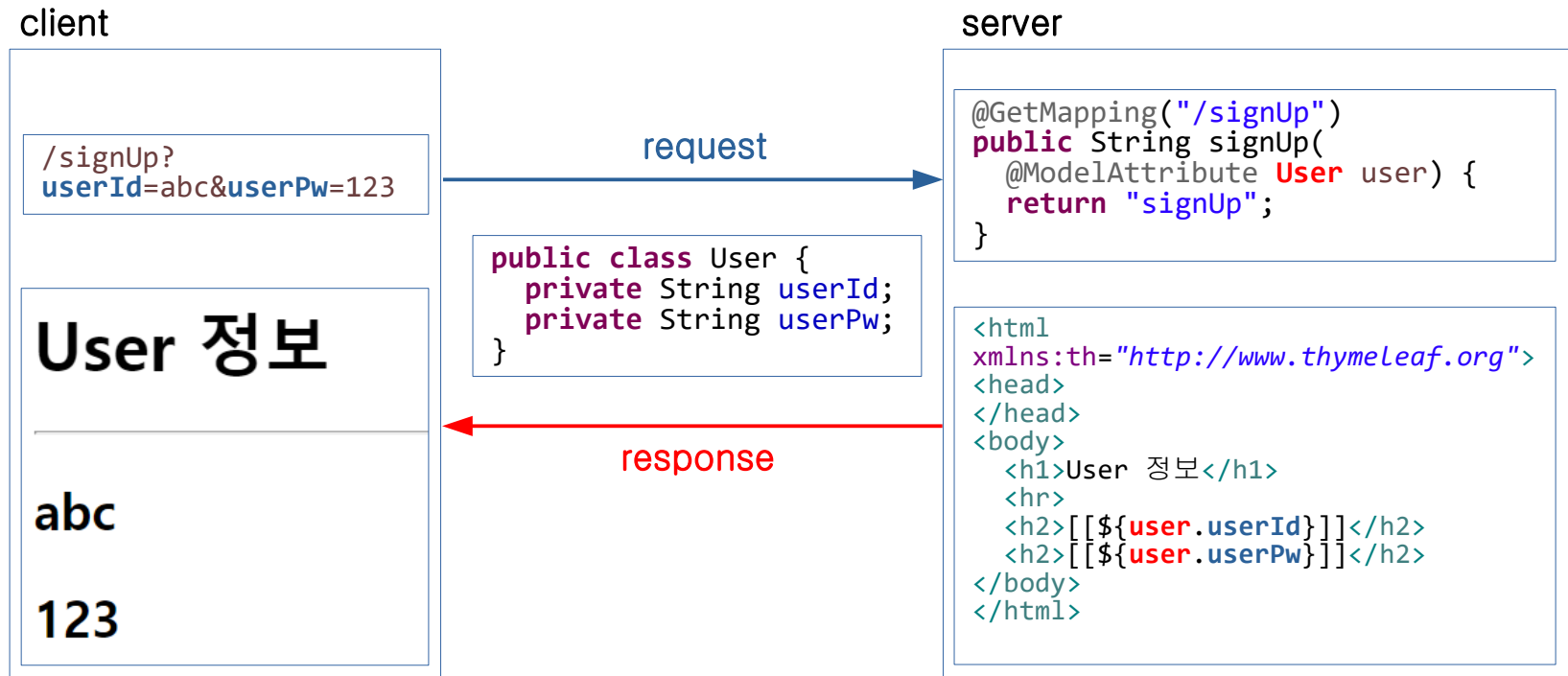
123, 456

← → ↻ ⓘ localhost:8080/req/path/language/java 🔍 ☆

language, java

■ 요청 처리 - @ModelAttribute

- Model에 작성되어 있는 변수/자료형과 파라미터명이 동일하면 자동으로 대입
- JPA/MyBatis 등 ORM 프레임워크 연계 시 편리하게 사용 가능
- 요청 파라미터 저장과 응답 시 사용되는 모델의 역할을 동시에 수행



■ 요청 처리 - @ModelAttribute

● controller/RequestController.java

```
package com.ggoreb.basic.controller;

...

@RestController
public class RequestController {

    ...

    @GetMapping("req/model")
    public String model(
        @ModelAttribute Member member) {
        return member.toString();
    }
}
```

← → ↻ ⓘ localhost:8080/req/model?name=kim&userId=aa&userPassword=11

Member(name=kim, userId=aa, userPassword=11)

■ HTML Template – Thymeleaf

- 스프링 부트에서 권장하는 HTML Template
- HTML5 문법을 사용하는 HTML 태그 및 속성 기반의 Template Engine
- 텍스트, HTML, XML, JavaScript, CSS 등 생성 가능
- Controller에서 View로 넘겨준 Model을 이용하여 데이터 출력

```
@Controller
public class WelcomeController {
    @GetMapping("/welcome")
    public String welcome(Model model) {
        List<String> list = new ArrayList<>();
        list.add("list1");
        list.add("list2");
        model.addAttribute("list", list);

        Map<String, Object> map = new HashMap<>();
        map.put("key1", "map1");
        map.put("key2", "map2");
        model.addAttribute("map", map);

        Member member = new Member();
        member.setUserId("a");
        member.setName("spring");
        member.setUserPassword("1234");
        model.addAttribute("member", member);

        model.addAttribute("message", "thymeleaf");

        return "welcome";
    }
}
```

```
<html xmlns:th="http://www.thymeleaf.org">
<head>
</head>
<body>
<h1>List</h1>
<h3>[[${list}]]</h3>
<h3>[[${list[0]}]]</h3>

<h1>Map</h1>
<h3>[[${map}]]</h3>
<h3>[[${map.key1}]]</h3>
<h3>[[${map['key2']}]]</h3>

<h1>Member</h1>
<h3>[[${member}]]</h3>
<h3>[[${member['userId']}]]</h3>
<h3>[[${member.userPassword}]]</h3>

<h1>Message</h1>
<h3>[[${message}]]</h3>

</body>
</html>
```

■ HTML Template – Thymeleaf

List

[list1, list2]

list1

```
<h3>[[${list}]]</h3>
```

```
<h3>[[${list[0]}]]</h3>
```

Map

{key1=map1, key2=map2}

map1

map2

```
<h3>[[${map}]]</h3>
```

```
<h3>[[${map.key1}]]</h3>
```

```
<h3>[[${map['key2']}]]</h3>
```

Member

Member(name=spring, userId=a, userPassword=1234)

a

1234

```
<h3>[[${member}]]</h3>
```

```
<h3>[[${member['userId']}]</h3>
```

```
<h3>[[${member.userPassword}]]</h3>
```

Message

thymeleaf

```
<h3>[[${message}]]</h3>
```

■ Thymeleaf – Variable Expression / \${ ... }

● controller/ThymeleafController.java

```
package com.ggoreb.basic.controller;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

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

@Controller
public class ThymeleafController {
    @GetMapping("user")
    public String user(Model model) {
        Map<String, Object> user = null;
        user = new HashMap<>();
        user.put("userId", "z");
        user.put("userName", "zoo");
        user.put("userAge", 25);

        model.addAttribute("user", user);

        return "user";
    }
}
```

■ Thymeleaf – Variable Expression / \${ ... }

● templates/user.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
아이디:<span>[[${user.userId}]]</span><br>
이름:<span>[[${user.userName}]]</span><br>
나이:<span>[[${user.userAge}]]</span><br>

<hr>

아이디:<span th:text="${user.userId}"></span><br>
이름:<span th:text="${user.userName}"></span><br>
나이:<span th:text="${user.userAge}"></span><br>

<hr>

아이디:<span data-th-text="${user.userId}"></span><br>
이름:<span data-th-text="${user.userName}"></span><br>
나이:<span data-th-text="${user.userAge}"></span><br>
</body>
</html>
```

아이디:z 이름:zoo 나이:25

아이디:z 이름:zoo 나이:25

아이디:z 이름:zoo 나이:25

■ Thymeleaf – Iteration / th:each

● controller/ThymeleafController.java

```
@GetMapping("userList")
public String userList(Model model) {
    List<Map<String, Object>> userList = new ArrayList<>();
    Map<String, Object> user = null;

    user = new HashMap<>();
    user.put("userId", "a");
    user.put("userName", "apple");
    user.put("userAge", 21);
    userList.add(user);

    user = new HashMap<>();
    user.put("userId", "b");
    user.put("userName", "banana");
    user.put("userAge", 22);
    userList.add(user);

    user = new HashMap<>();
    user.put("userId", "c");
    user.put("userName", "carrot");
    user.put("userAge", 23);
    userList.add(user);

    model.addAttribute("userList", userList);

    return "userList";
}
```

■ Thymeleaf – Iteration / th:each

● templates/userList.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
  <table border="1">
    <tr>
      <td>아이디</td>
      <td>이름</td>
      <td>나이</td>
    </tr>
    <tr th:each="user : ${userList}">
      <td th:text="${user.userId}"></td>
      <td th:text="${user.userName}"></td>
      <td th:text="${user.userAge}"></td>
    </tr>
  </table>
  <hr>
  <th:block th:each="pageNumber : ${#numbers.sequence(1, 10)}">
    <span th:text="${pageNumber}"></span>
  </th:block>
</body>
</html>
```

아이디	이름	나이
a	apple	21
b	banana	22
c	carrot	23

1 2 3 4 5 6 7 8 9 10

■ Thymeleaf – Conditional Evaluation / th:if, th:unless, th:switch

● controller/ThymeleafController.java

```
@GetMapping("mode")
public String mode(Model model, @RequestParam Map<String, Object> map) {
    model.addAttribute("name", map.get("name"));
    model.addAttribute("auth", map.get("auth"));
    model.addAttribute("category", map.get("category"));

    return "mode";
}
```

● templates/mode.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
</head>
<body>
    관리자 이름 :
    <span th:if="${name} != null" th:text="${name}"></span>
    <span th:unless="${name} != null" th:text="이름없음"></span>
    <br>
    권한 : <span th:text="${auth} != null ? ${auth} : '권한없음'"></span><br>
    담당 카테고리 :
    <span th:switch="${category}">
        <span th:case="1">커뮤니티</span>
        <span th:case="2">장터</span>
        <span th:case="3">갤러리</span>
    </span><br>
</body>
</html>
```

← → ↻ ⓘ localhost:8080/mode?name=홍길동&auth=gallery&category=3

관리자 이름 : 홍길동
권한 : gallery
담당 카테고리 : 갤러리

■ Thymeleaf – Iteration, Conditional Evaluation

● controller/ThymeleafController.java

```
@GetMapping("pagination")
public String pagination(Model model, @RequestParam(defaultValue="1") int page) {
    int startPage = (page - 1) / 10 * 10 + 1;
    int endPage = startPage + 9;
    model.addAttribute("startPage", startPage);
    model.addAttribute("endPage", endPage);
    model.addAttribute("page", page);

    return "pagination";
}
```

● templates/pagination.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Insert title here</title>
</head>
<body>
    <th:block th:each="pageNumber : ${#numbers.sequence(startPage, endPage)}">
        <span th:if="${page} == ${pageNumber}" th:text="${pageNumber}"
            style="font-weight:bold"></span>
        <span th:unless="${page} == ${pageNumber}" th:text="${pageNumber}"></span>
    </th:block>
</body>
</html>
```

← → ↻ ⓘ localhost:8080/pagination?page=5

1 2 3 4 **5** 6 7 8 9 10

■ Thymeleaf – Link Url Expression / @ { ... }

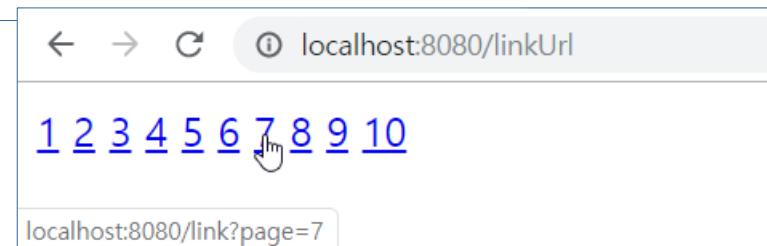
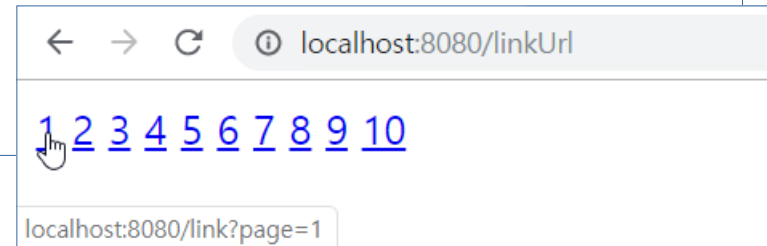
● controller/ThymeleafController.java

```
@GetMapping("linkUrl")
public String linkUrl(Model model, @RequestParam(defaultValue="1") int page) {
    int startPage = (page - 1) / 10 * 10 + 1;
    int endPage = startPage + 9;
    model.addAttribute("startPage", startPage);
    model.addAttribute("endPage", endPage);
    model.addAttribute("page", page);

    return "linkUrl";
}
```

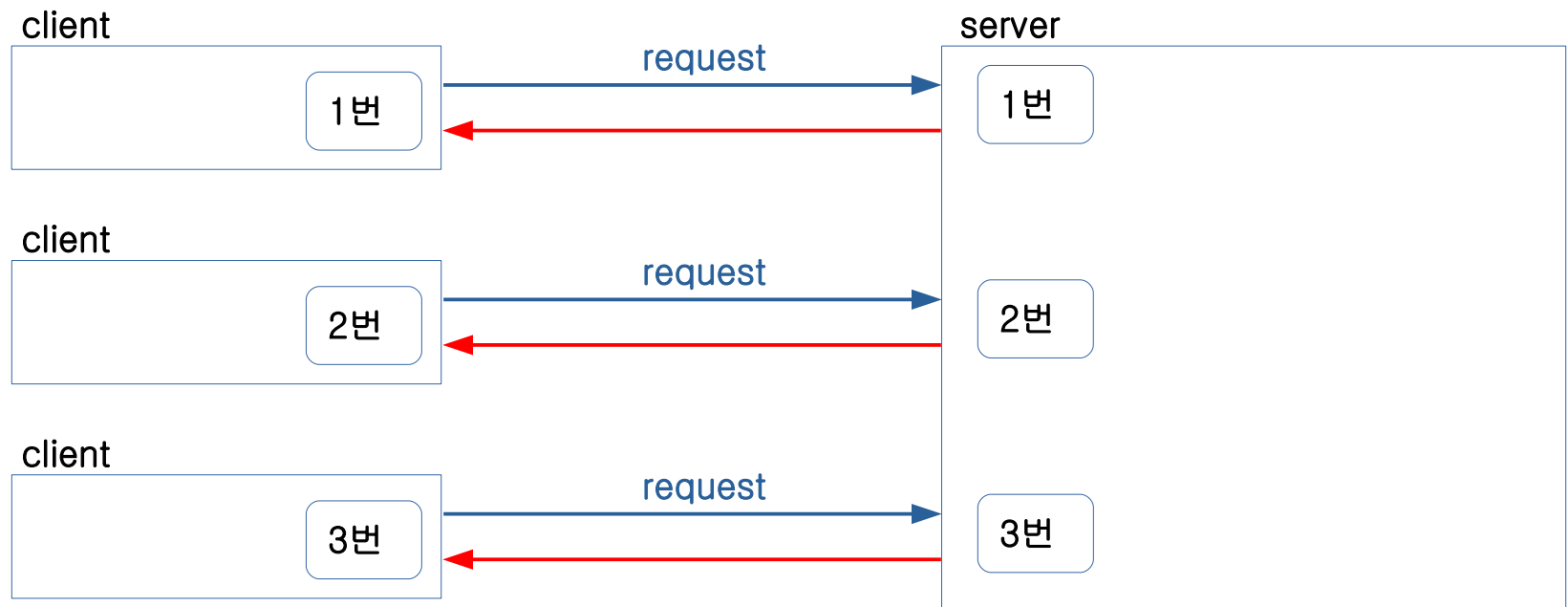
● templates/linkUrl.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <th:block th:each="pageNumber : ${#numbers.sequence(1, 10)}">
        <a th:href="@{/LinkUrl(page=${pageNumber})}" th:text="${pageNumber}"></a>
    </th:block>
</body>
</html>
```



■ Session

- 클라이언트에 대한 정보를 서버에 저장할 수 있는 공간
- 접속하는 클라이언트 당 하나의 세션 생성
- 사물함과 같은 형식으로 저장이 되며 사물함의 번호를 클라이언트로 전송
- 번호를 분실하는 경우 새로운 세션을 생성하고 다시 클라이언트로 전송
- 설문조사와 같이 여러단계로 정보 입력 시, 로그인 후 사용 내역 저장 등 활용



■ Thymeleaf – session

● controller/SessionController.java

```
package com.ggoreb.basic.controller;

import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

import com.ggoreb.basic.model.User;

@Controller
public class SessionController {
    @GetMapping("/login")
    public String login() {
        return "login";
    }

    @PostMapping("/login")
    public String loginPost(User user, HttpSession session) {
        session.setAttribute("user", user);
        return "redirect:/main";
    }

    @GetMapping("/main")
    public String main() {
        return "main";
    }
}
```

■ Thymeleaf – session

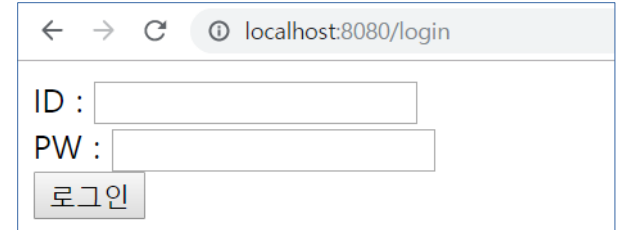
● model/User.java

```
package com.ggoreb.basic.model;  
  
import lombok.Data;  
  
@Data  
public class User {  
    private String userId;  
    private String userPw;  
}
```

■ Thymeleaf – session

● templates/login.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
  <form action="/login" method="post">
    ID : <input type="text" name="userId"><br>
    PW : <input type="password" name="userPw"><br>
    <input type="submit" value="로그인">
  </form>
</body>
</html>
```



← → ↻ localhost:8080/login

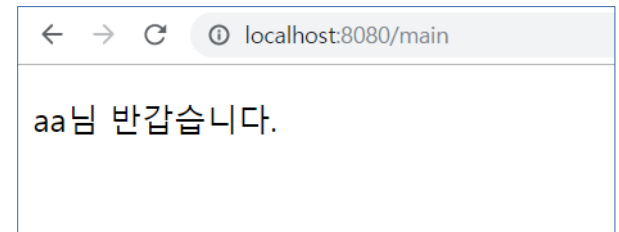
ID :

PW :

로그인

● templates/main.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
  <p th:if="${session.user} != null"
    th:text="${session.user.userId} + '님 반갑습니다.'"></p>
  <p th:unless="${session.user} != null">로그인되어 있지 않습니다.</p>
</body>
</html>
```



← → ↻ localhost:8080/main

aa님 반갑습니다.