Spring Framework 맛보기

SOPT 25기 서버파트 양희찬

CONTENTS

01 Server란?

#Frontend #Server #BackEnd #Spring

01. Server란?

Server

Serving

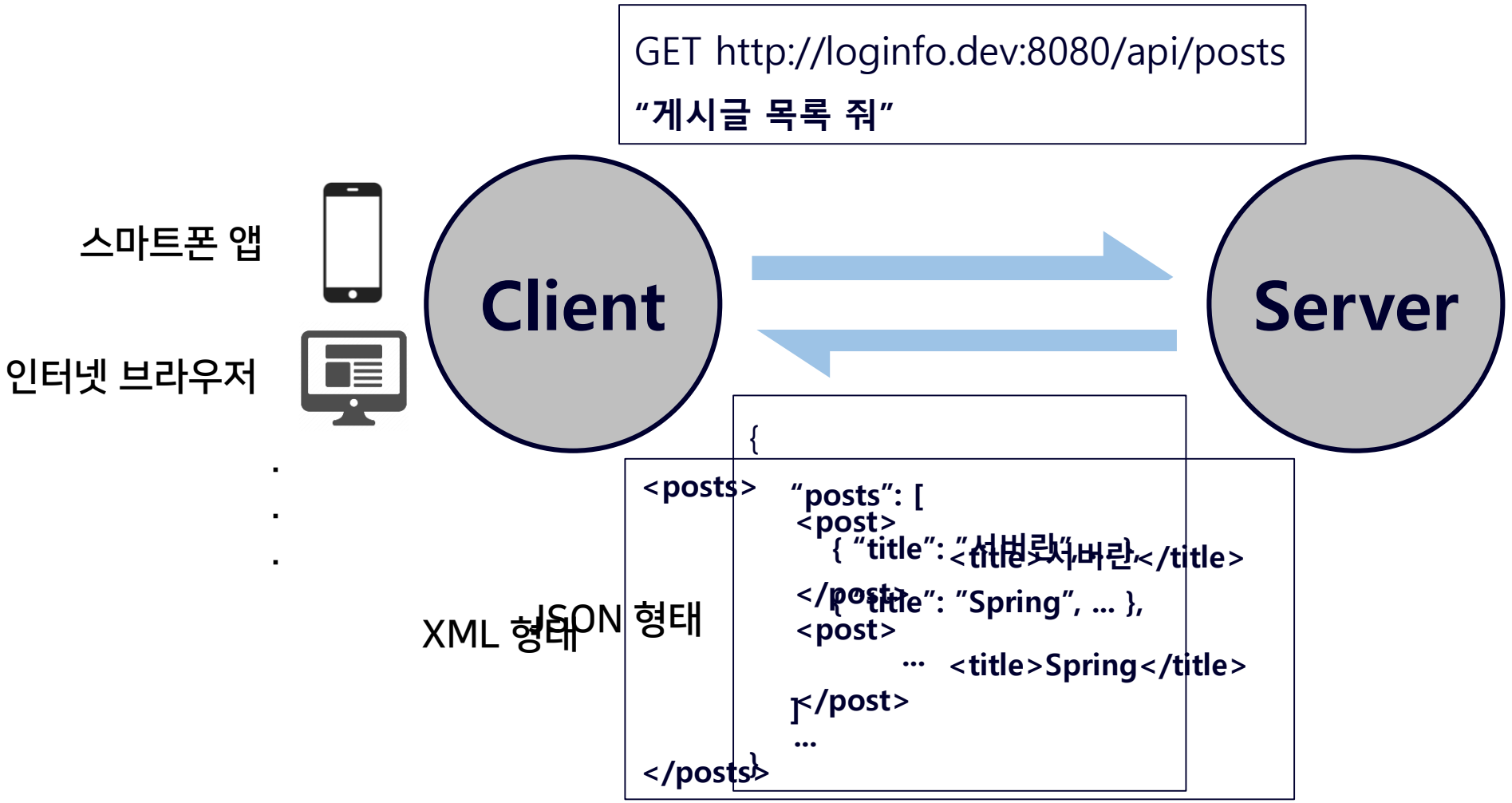
고객(Client)에게 무언가를 제공하는 역할을 하는 무언가

01. Server란?

Server

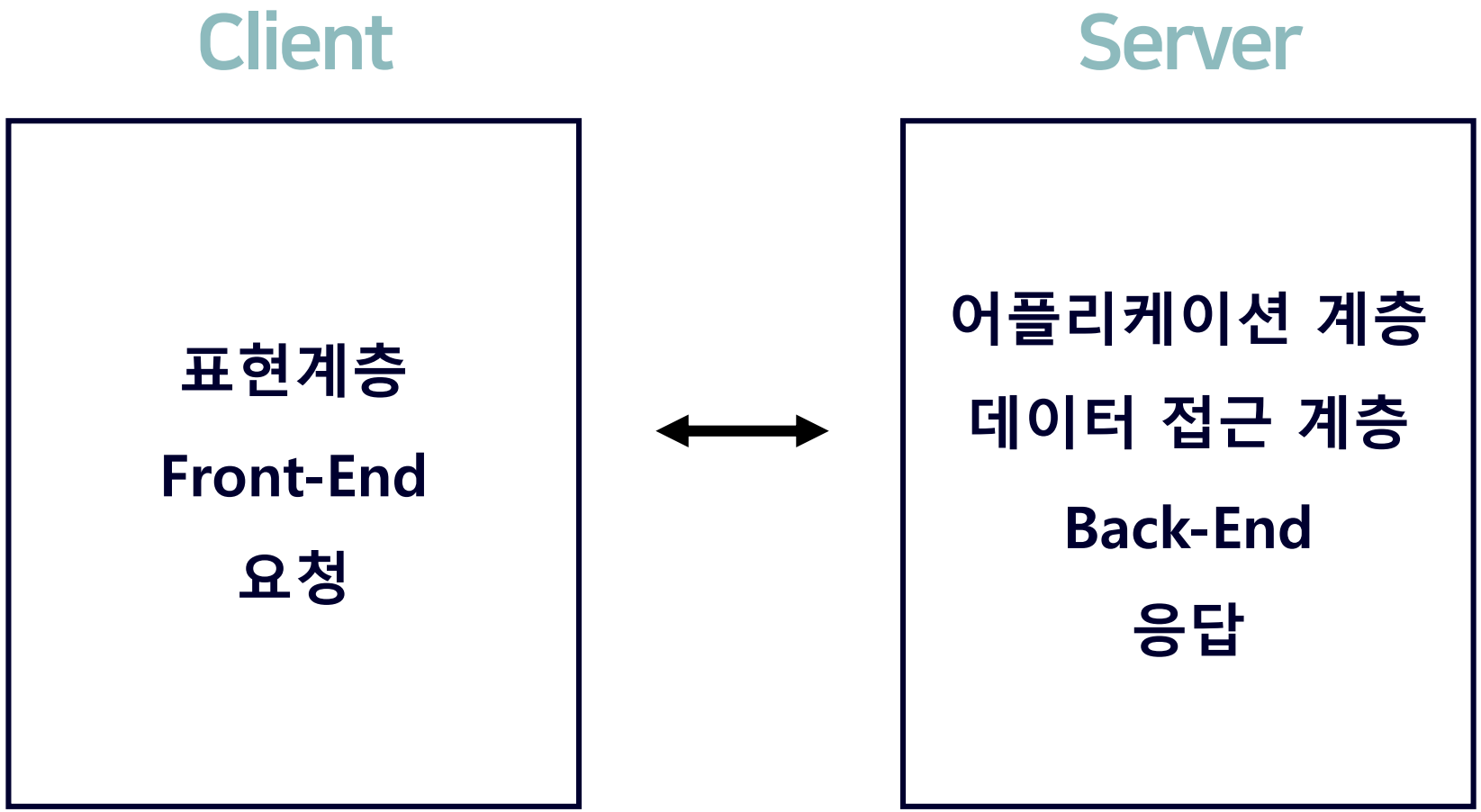
01. Server란?

Server와 Client



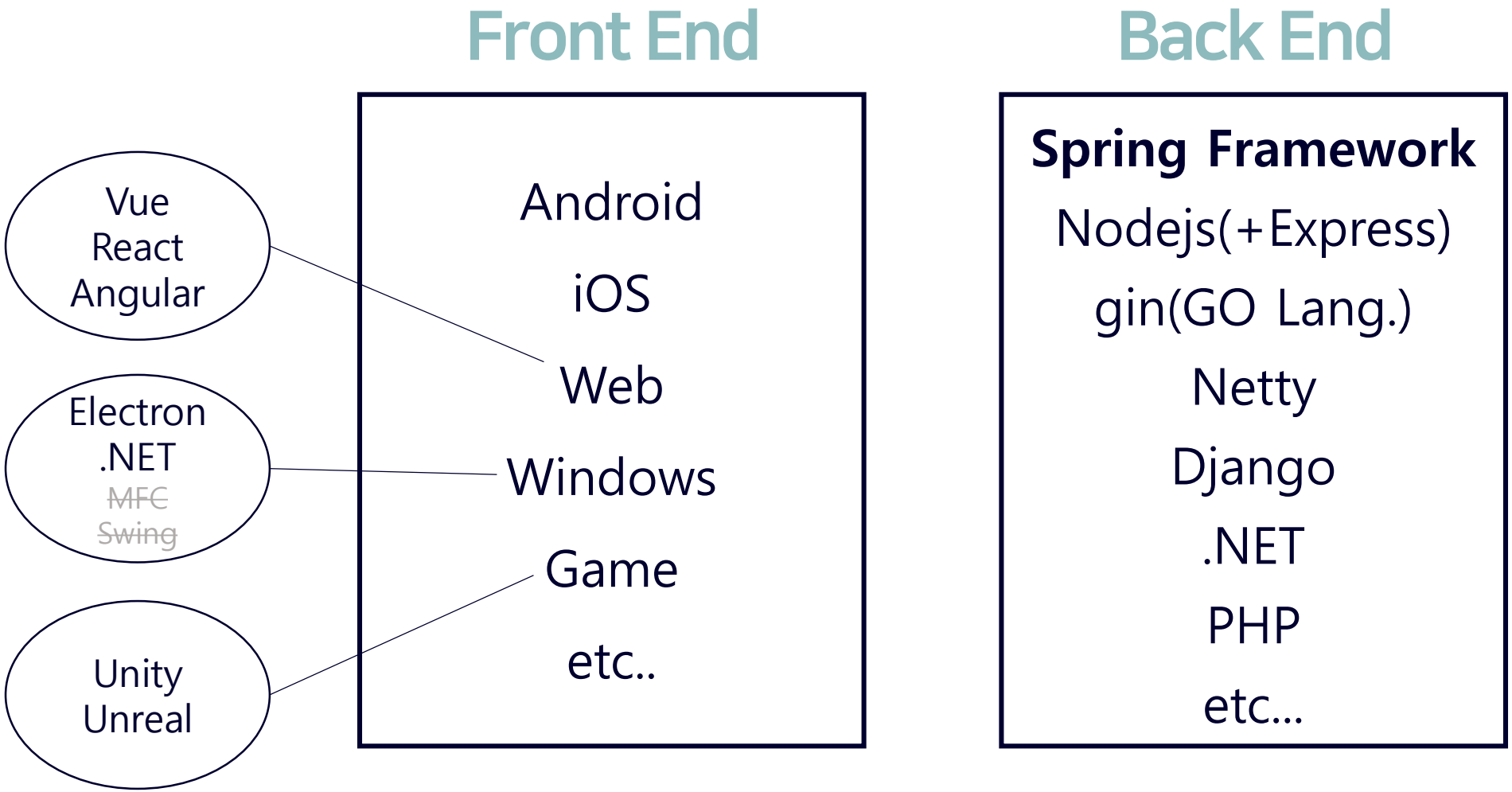
01. Server란?

Server와 Client



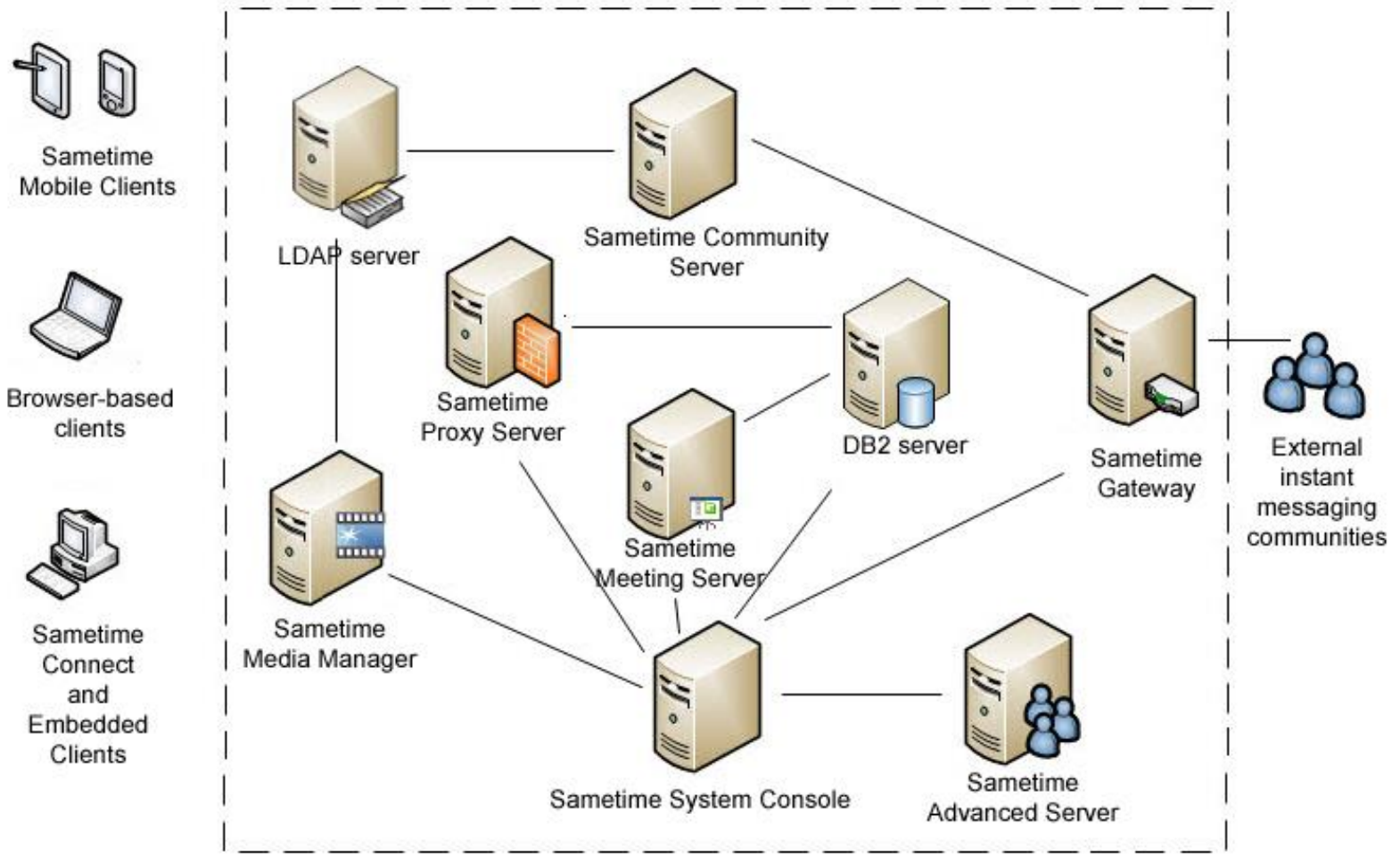
01. Server란?

Server와 Client - 개발



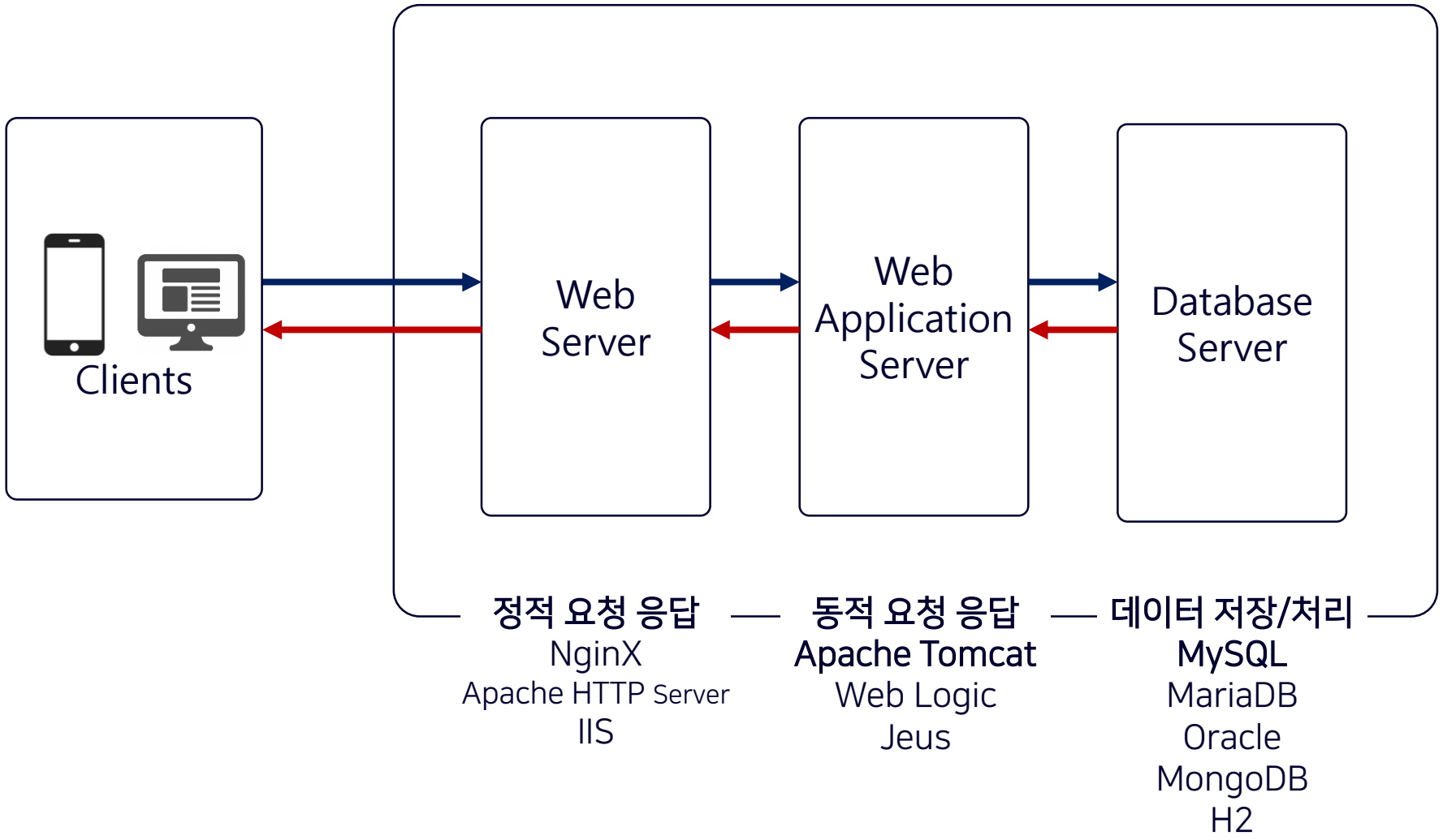
01. Server란?

Server의 종류



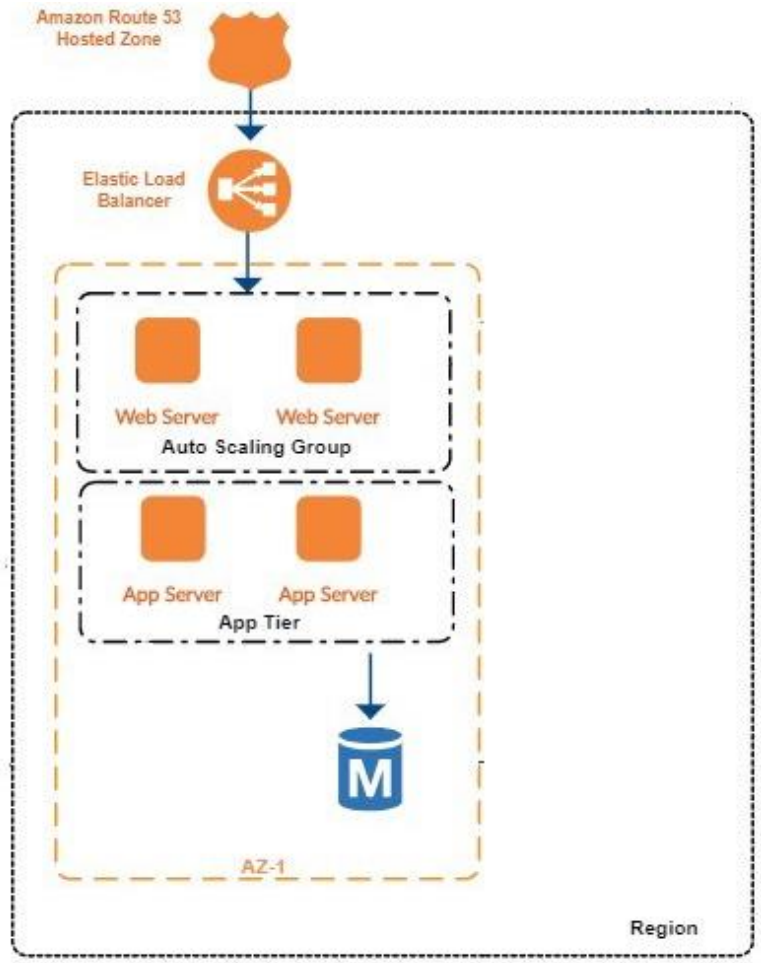
01. Server란?

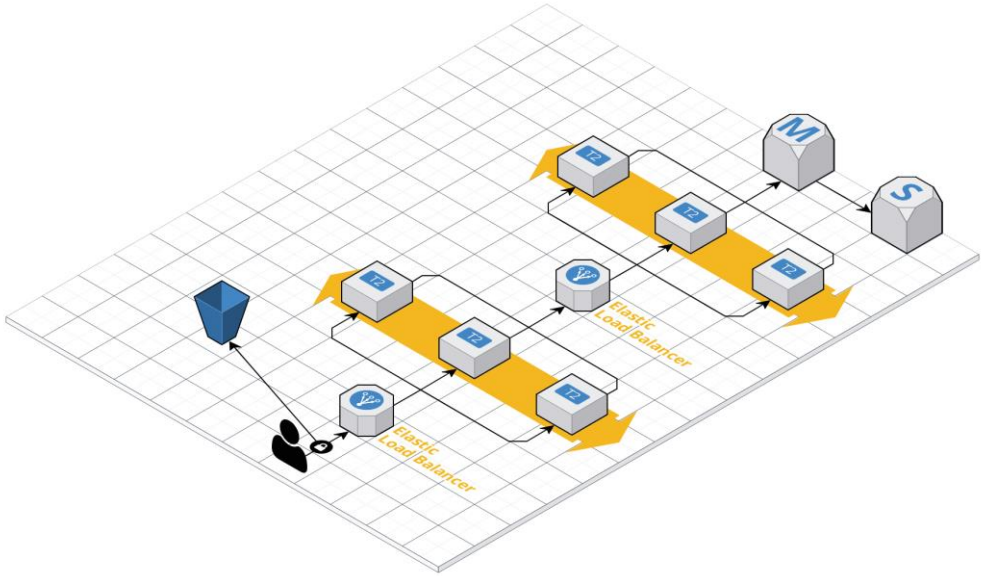
Server와 Client - 아키텍처

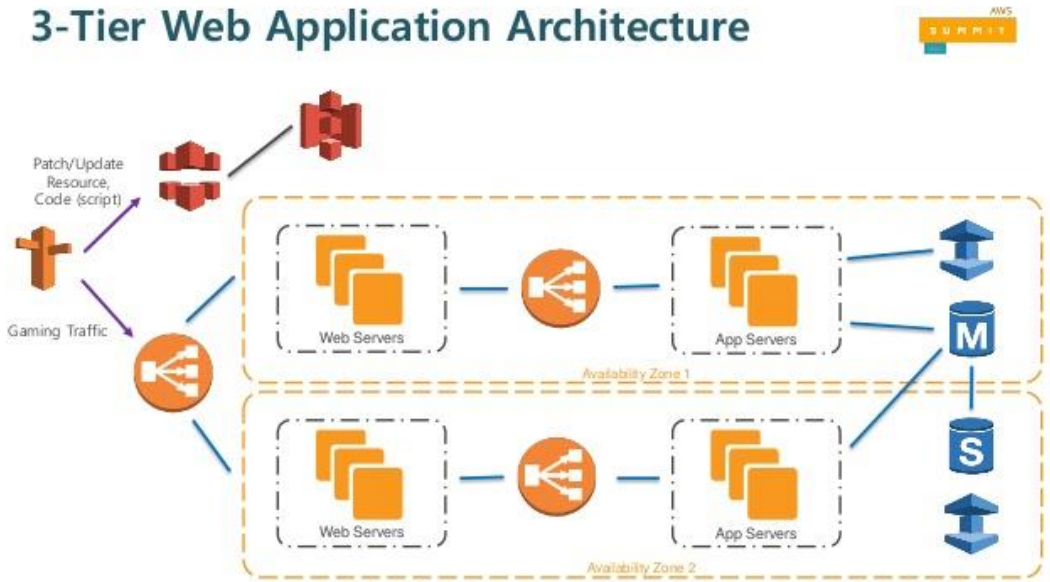


01. Server란?

Server와 Client – 아키텍처 예시

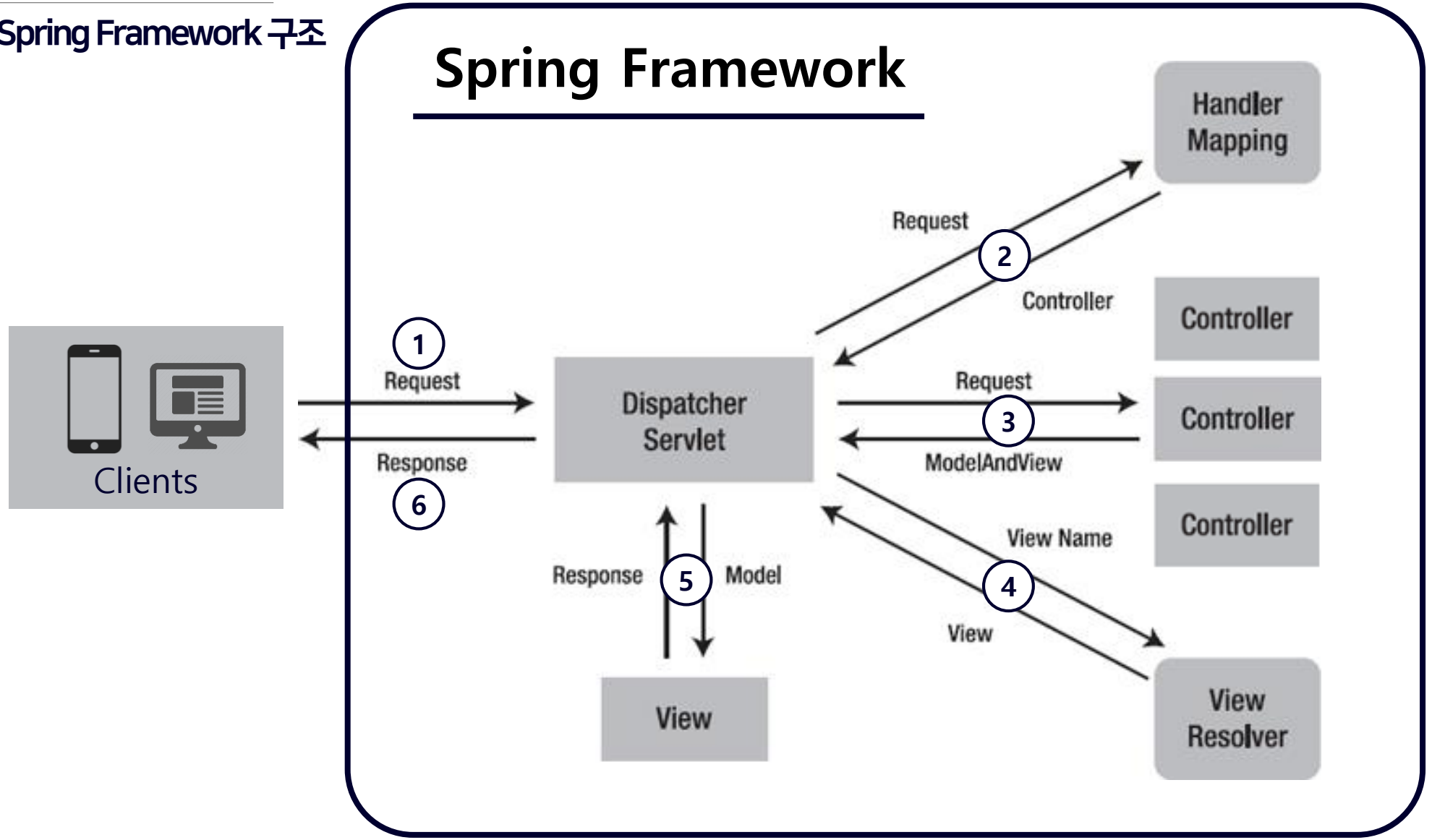






01. Server란?

Spring Framework 구조



01. Server란?

Spring Boot

초기 설정의 어려움

- 필요 라이브러리 모두 import
- 라이브러리간 dependency 설정
- xml 형식의 설정(Bean 설정 등)
- Tomcat 따로 설치 및 구동 필요

Springs Boot

초기 설정 단순화

- 관련 라이브러리를 버전에 맞추어 import할 수 있음
- Java Annotation으로 간단히 Bean 설정
- 내장 Tomcat 사용

01. Server란?

참고문서

Front end - Wikipedia

웹 서버와 웹 애플리케이션 서버 - snippet

스프링 부트란 (What is a spring boot?) :: 엉뚱한 마녀의 상상!?

5 Benefits of a 3-Tier Architecture - Izenda

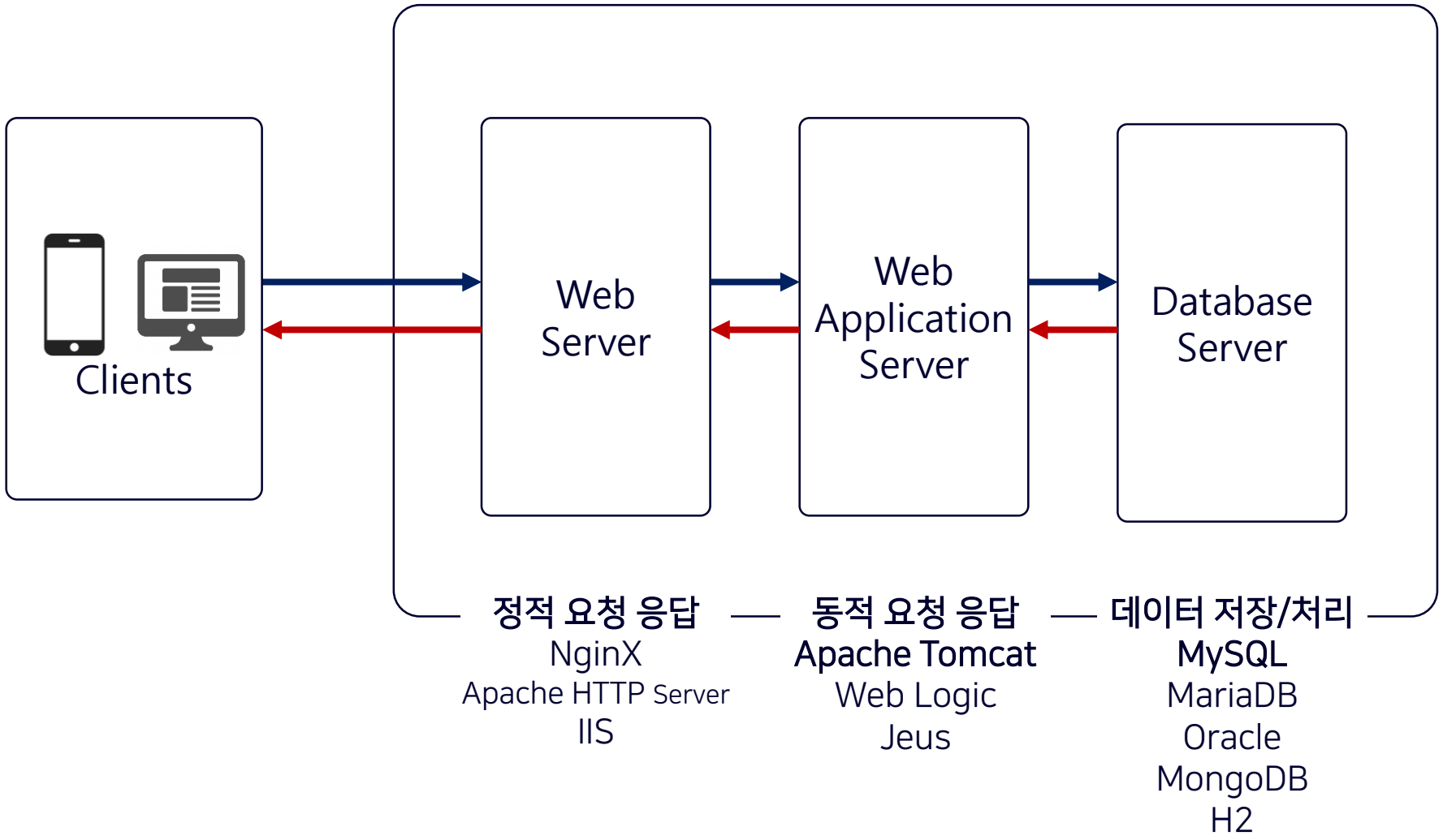
Front and back ends - Wikipedia

02 실습 (1)

#백견이 불여일타

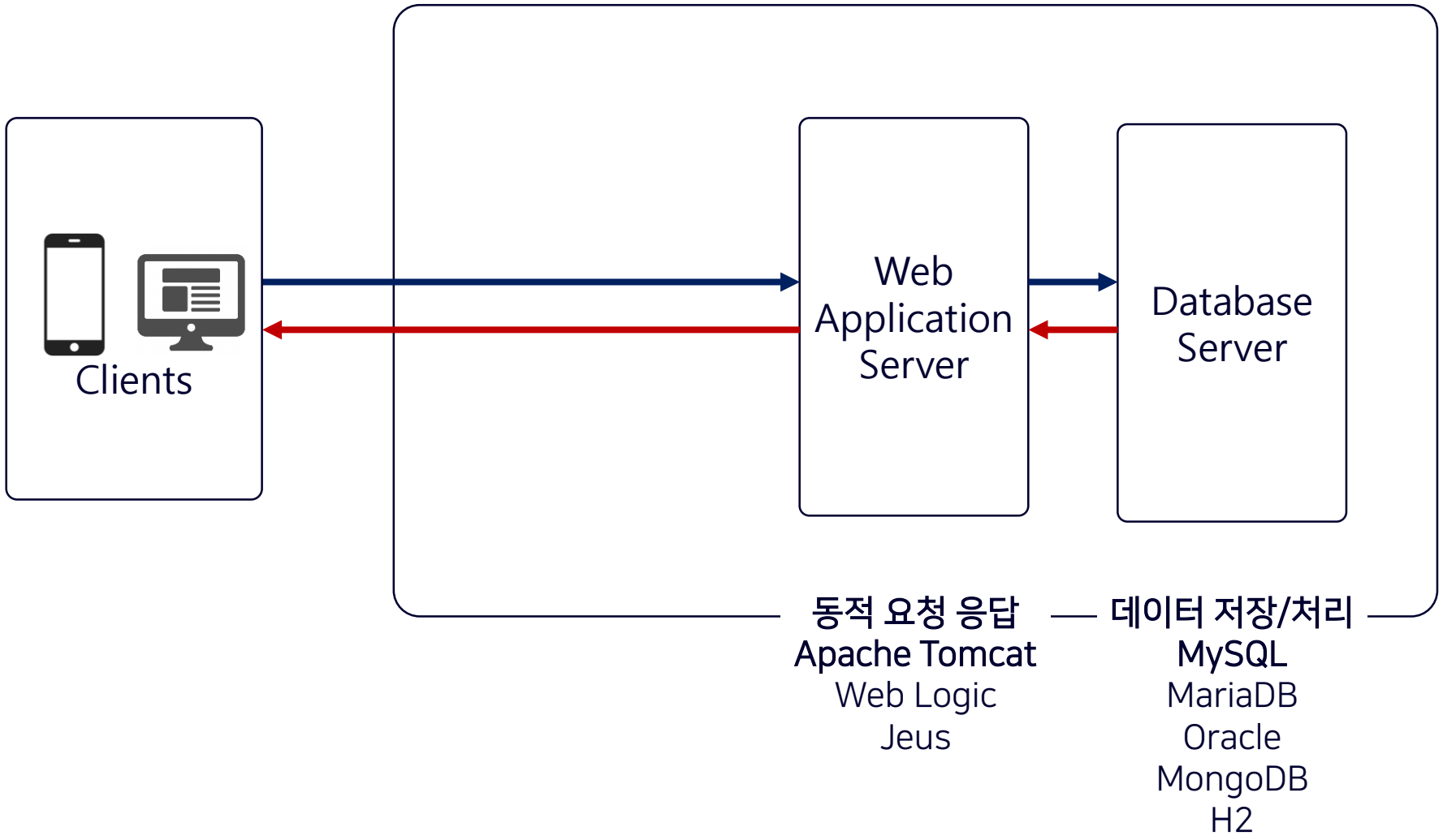
02. 실습 (1)

실습할 아키텍처



02. 실습 (1)

실습할 아키텍처



02. 실습 (1)

개발 환경

백엔드 개발 프레임워크 : Spring Framework, **Spring Boot**

프로그래밍 언어 : **Java** or Kotlin

빌드툴 : Maven or **Gradle**

Persistence Framework : JPA or **MyBatis**

데이터베이스 : **MySQL**

IDE : **IntelliJ** or Eclipse or STS

DB 구축/운영툴 : **MySQL Workbench**

API 테스트툴 : **Postman** or Insomnia

02. 실습 (1)

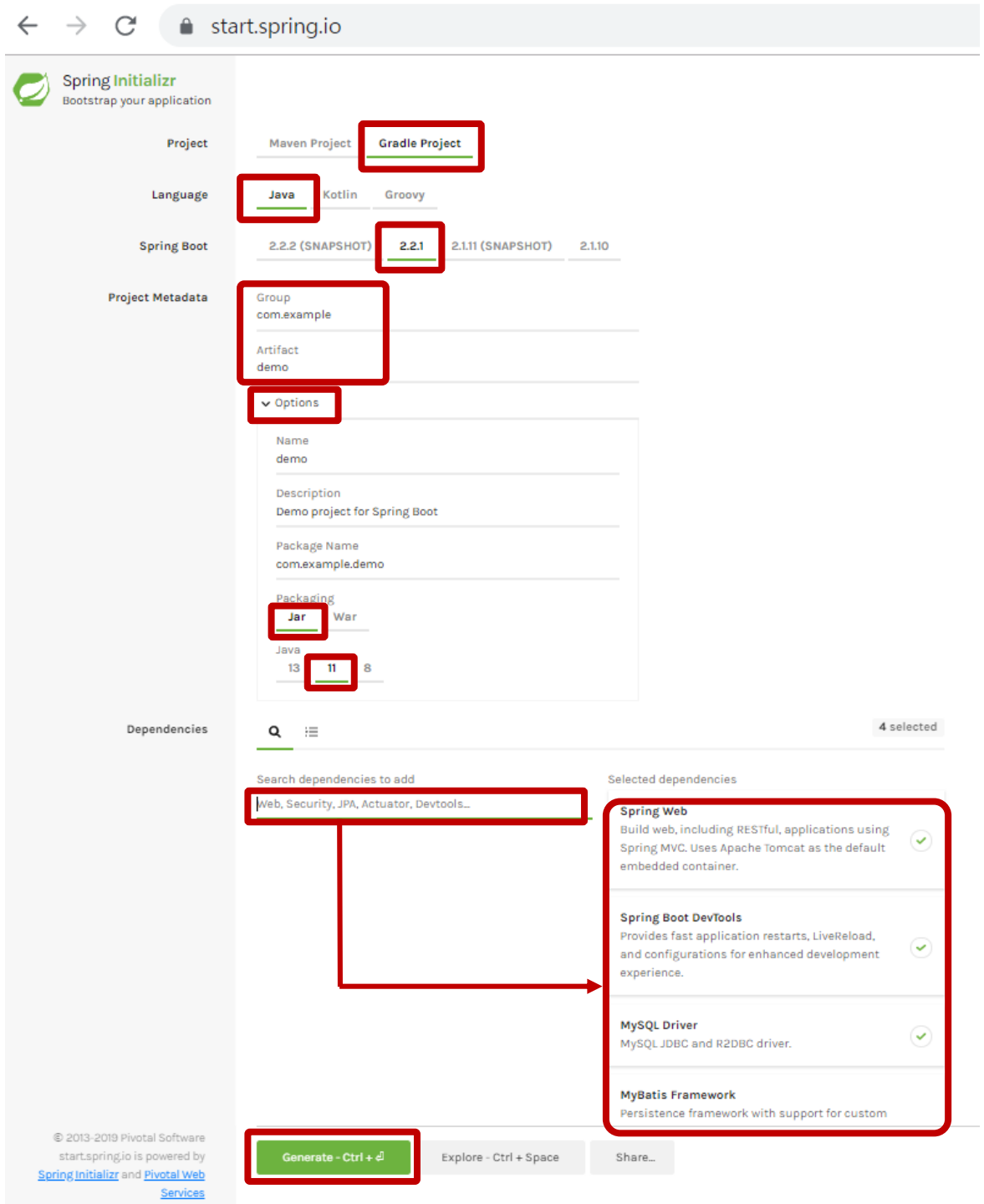
프로젝트 생성

Spring Initializr Link

https://start.spring.io

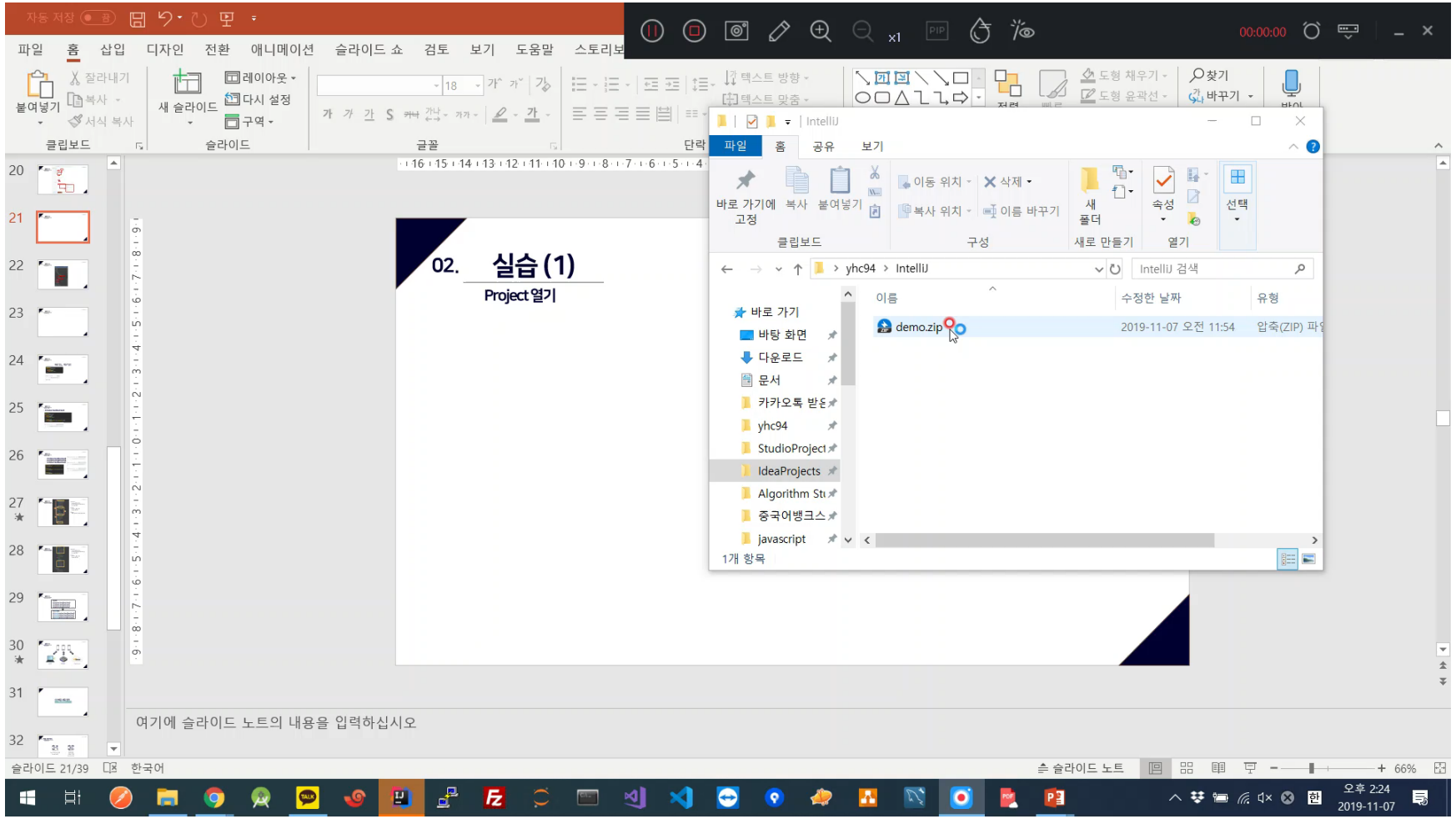
02. 실습 (1)

프로젝트 생성



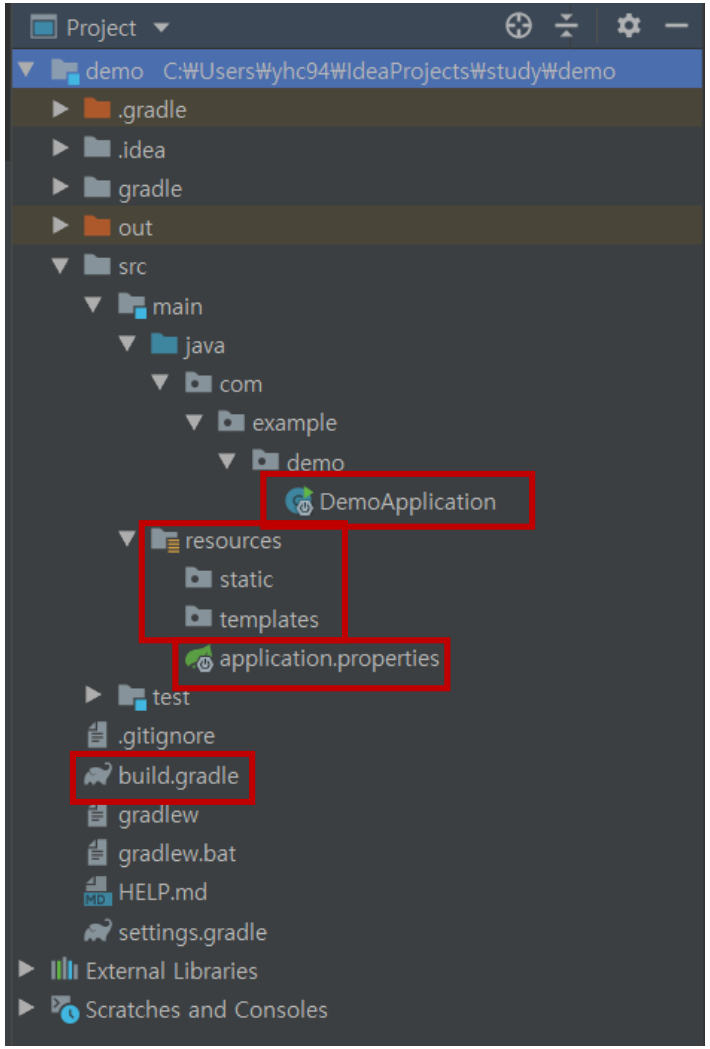
02. 실습 (1)

Project 열기



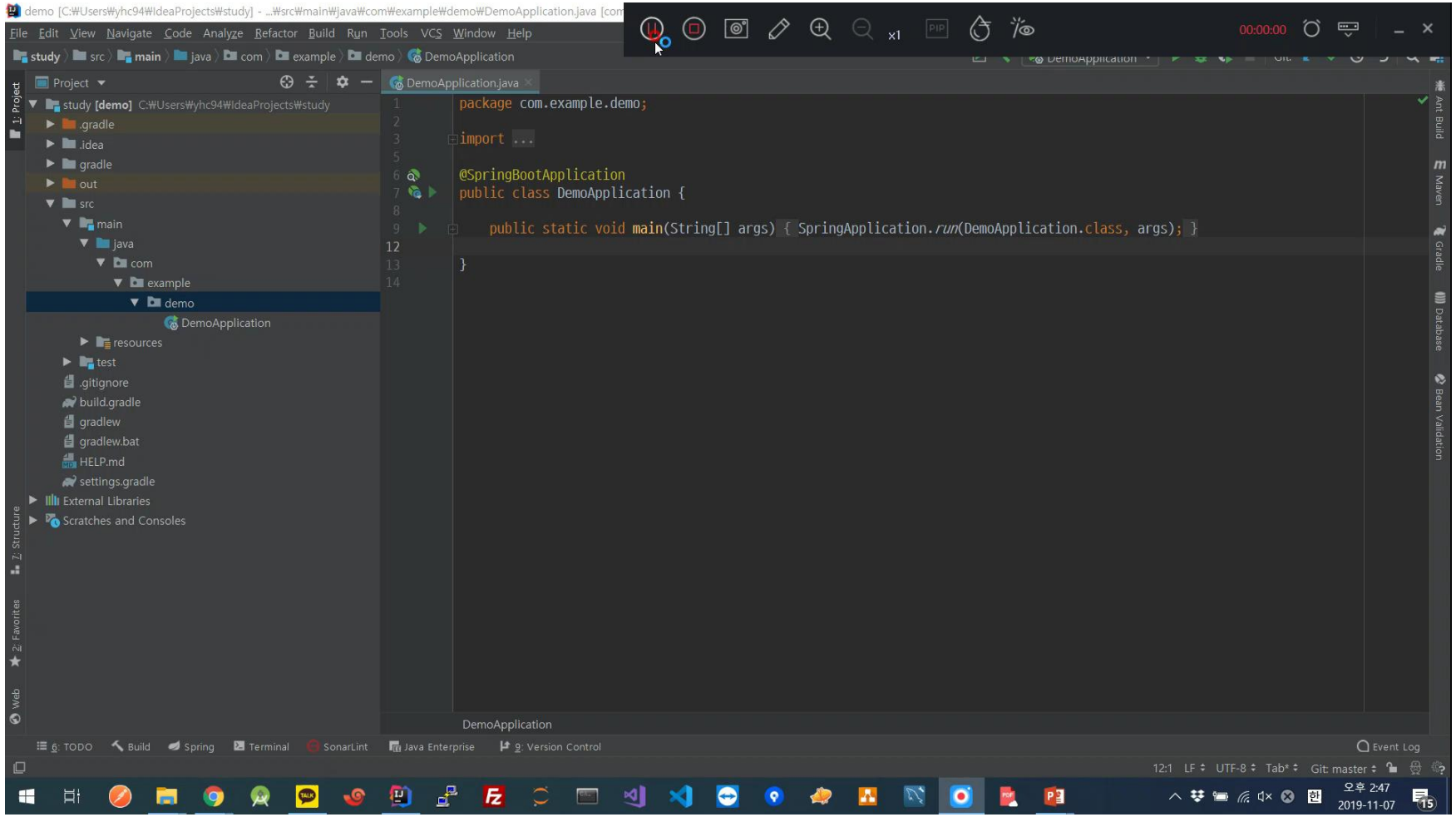
02. 실습 (1)

Project 구조



02. 실습 (1)

Controller 생성



02. 실습 (1)

실행 준비

다음과 같이 `build.gradle`의 `dependency`에서

`MyBatis`와 `MySQL-Connector`관련 dependency를 주석처리해주자!

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
// implementation 'org.mybatis.spring.boot:mybatis-spring-boot-starter:2.1.1'
    developmentOnly 'org.springframework.boot:spring-boot-devtools'
// runtimeOnly 'mysql:mysql-connector-java'
    testImplementation('org.springframework.boot:spring-boot-starter-test') {
        exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
    }
}
```

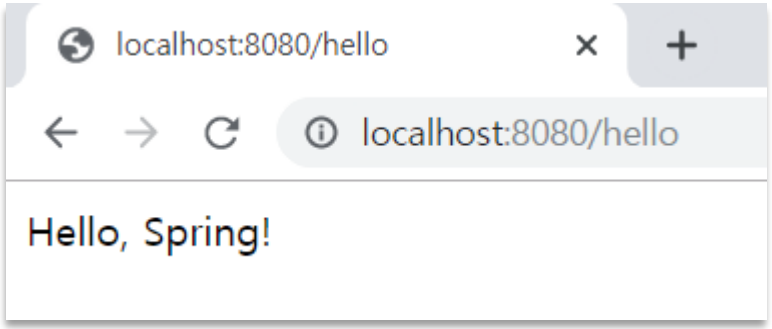
아직 사용하지 않으며, 그대로 실행 시 오류가 발생하므로 주석처리 해줘야 함

02. 실습 (1)

Hello, World!

Hello, World!

```
@RestController
public class HelloController {

    @GetMapping("/hello")
    public int hello() {
        return “Hello, Spring!";
    }
}
```

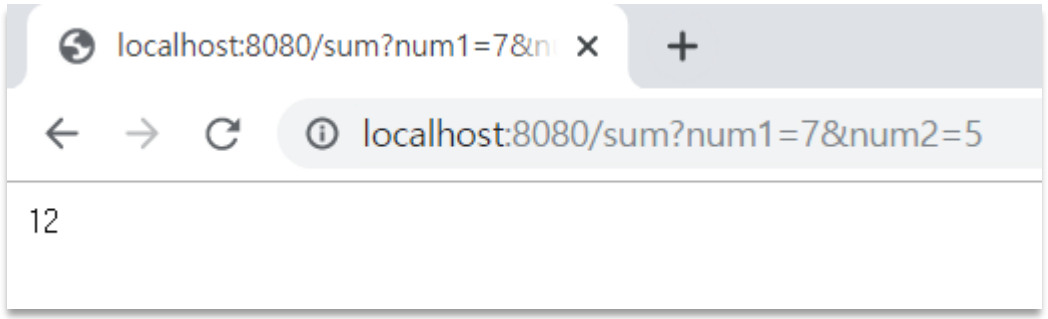


02. 실습 (1)

간단한 API 개발

- GET /sum?num1={num1}&num2={num2}

```java
@RestController
public class HelloController {

    @GetMapping("/hello")
    public int hello() {
        return “Hello, Spring!";
    }

    @GetMapping("/sum")
    public int sum(@RequestParam int num1, @RequestParam int num2) {
        return num1+num2;
    }
}
```

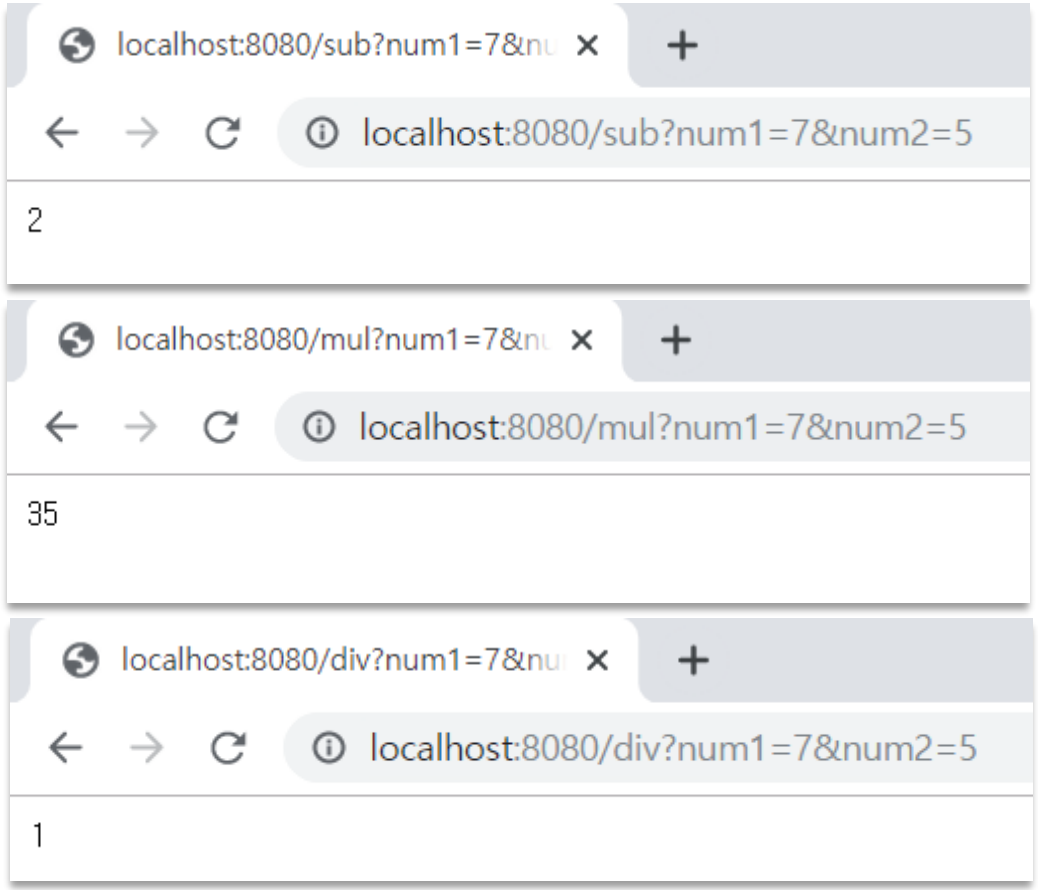


02. 실습 (1)

간단한 API 개발

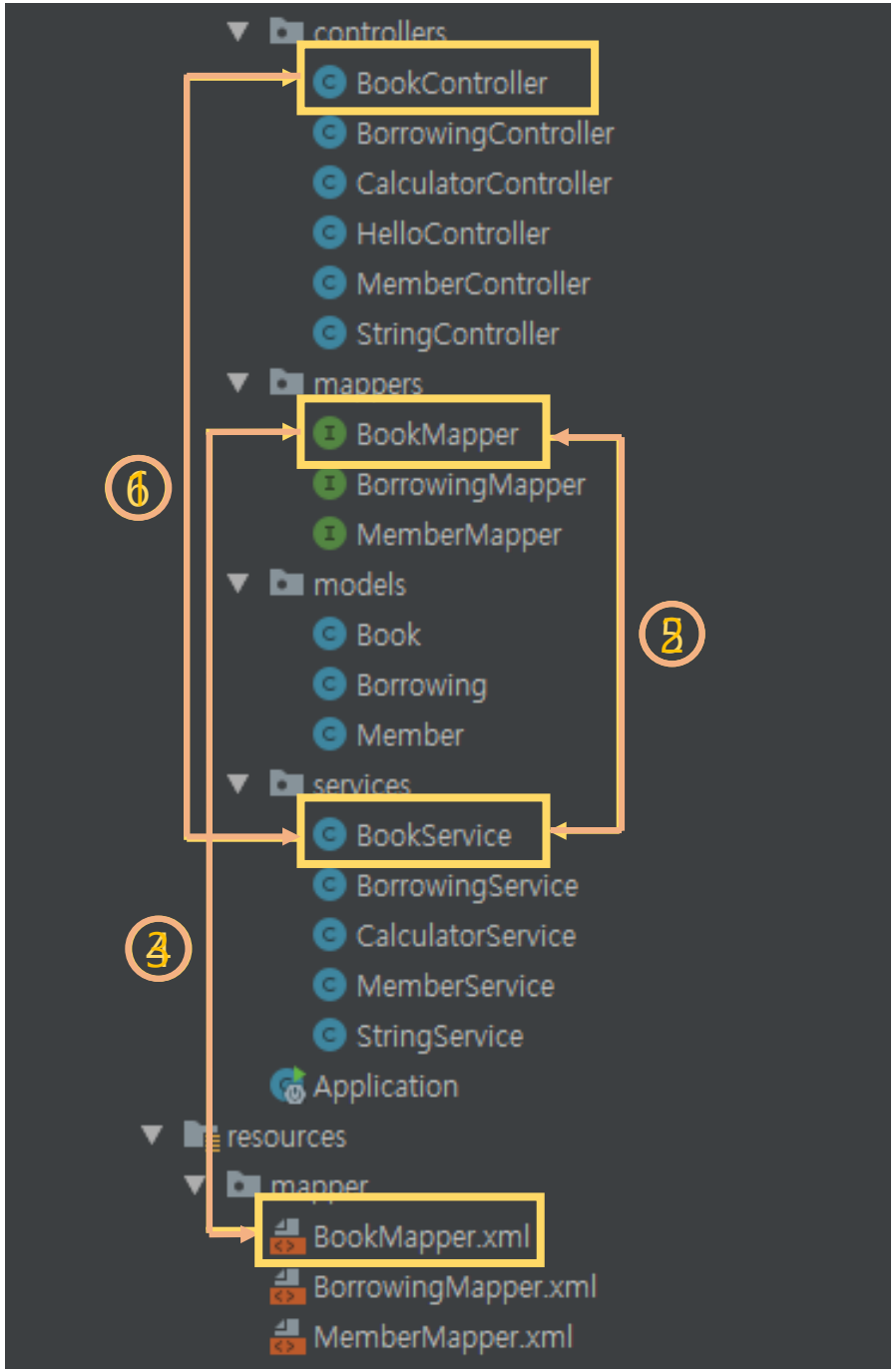
- GET /sub?num1={num1}&num2={num2}
- GET /mul?num1={num1}&num2={num2}
- GET /div?num1={num1}&num2={num2}

```java
@GetMapping("/sub")
public int sub(@RequestParam int num1, @RequestParam int num2) {
    return num1 - num2;
}

@GetMapping("/mul")
public int mul(@RequestParam int num1, @RequestParam int num2) {
    return num1 * num2;
}

@GetMapping("/div")
public int div(@RequestParam int num1, @RequestParam int num2) {
    return num1 / num2;
}
```



02. 실습 (1)

프로젝트 구조 살펴보기



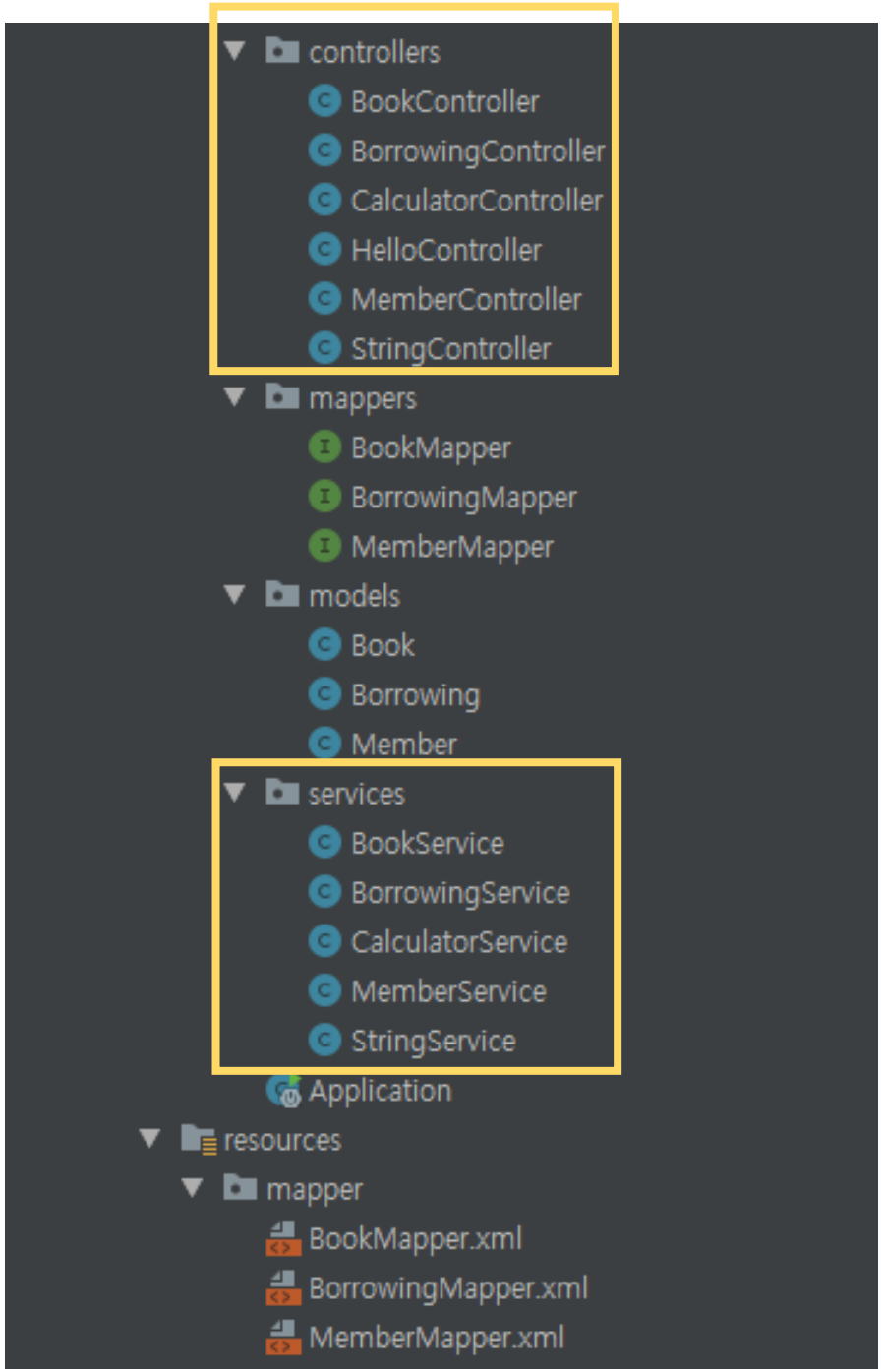
Controller
- Routing 해주는 역할
- URL과 method를 이어주기만 함
- Service를 호출

Service
- 비즈니스 로직 수행
- Mapper를 호출

Mapper
- DB에 접근하여 데이터 저장/처리

02. 실습 (1)

프로젝트 구조 살펴보기



Controller
- Routing 해주는 역할
- URL과 method를 이어주기만 함
- Service를 호출

Service
- 비즈니스 로직 수행
- Mapper를 호출

02. 실습 (1)

Controller 나누기

- GET /hello
- GET /sum?num1={num1}&num2={num2}
- GET /sub?num1={num1}&num2={num2}
- GET /mul?num1={num1}&num2={num2}
- GET /div?num1={num1}&num2={num2}

▼

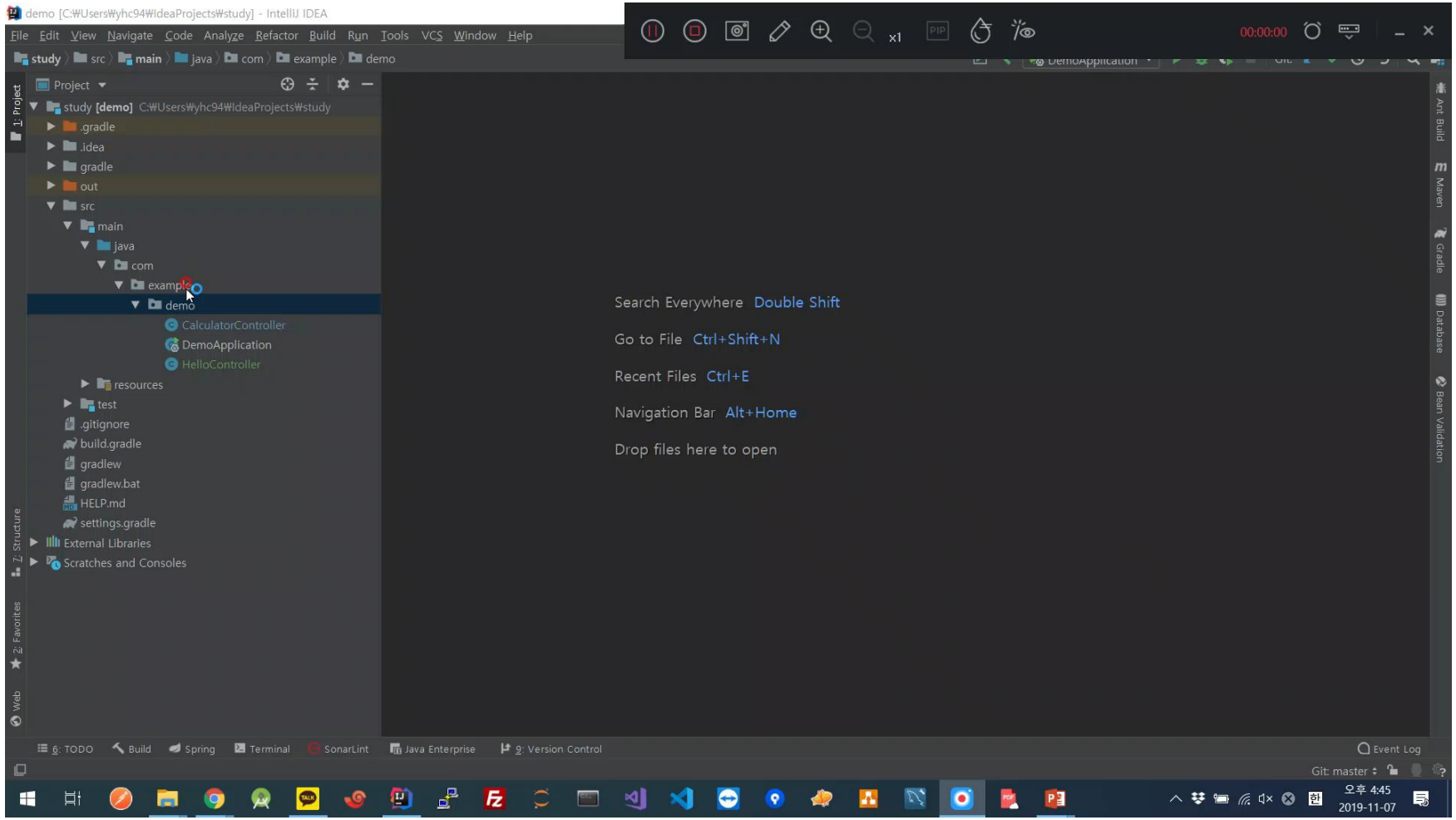
- GET /hello/hello
- GET /calculator/sum?num1={num1}&num2={num2}
- GET /calculator/sub?num1={num1}&num2={num2}
- GET /calculator/mul?num1={num1}&num2={num2}
- GET /calculator/div?num1={num1}&num2={num2}

```java
@RestController
@RequestMapping("/hello")
public class HelloController {
    @GetMapping("/hello")
    public String hello() {
        return "Hello, Spring!";
    }
}
```

```java
@RestController
@RequestMapping("/calculator")
public class CalculatorController {

    @GetMapping("/sum")
    public int sum(@RequestParam int num1, @RequestParam int num2) {
        return num1 + num2;
    }

    @GetMapping("/sub")
    public int sub(@RequestParam int num1, @RequestParam int num2) {
        return num1 - num2;
    }

    @GetMapping("/mul")
    public int mul(@RequestParam int num1, @RequestParam int num2) {
        return num1 * num2;
    }

    @GetMapping("/div")
    public int div(@RequestParam int num1, @RequestParam int num2) {
        return num1 / num2;
    }
}
```

02. 실습 (1)

비즈니스 로직 분리하기



+ sum 메서드 뿐만 아니라 다른 메서드들도 똑같이 Service에 구현한 후 호출해주세요!

※주의 ※ package의 위치를 잘 확인해주세요!

02. 실습 (1)

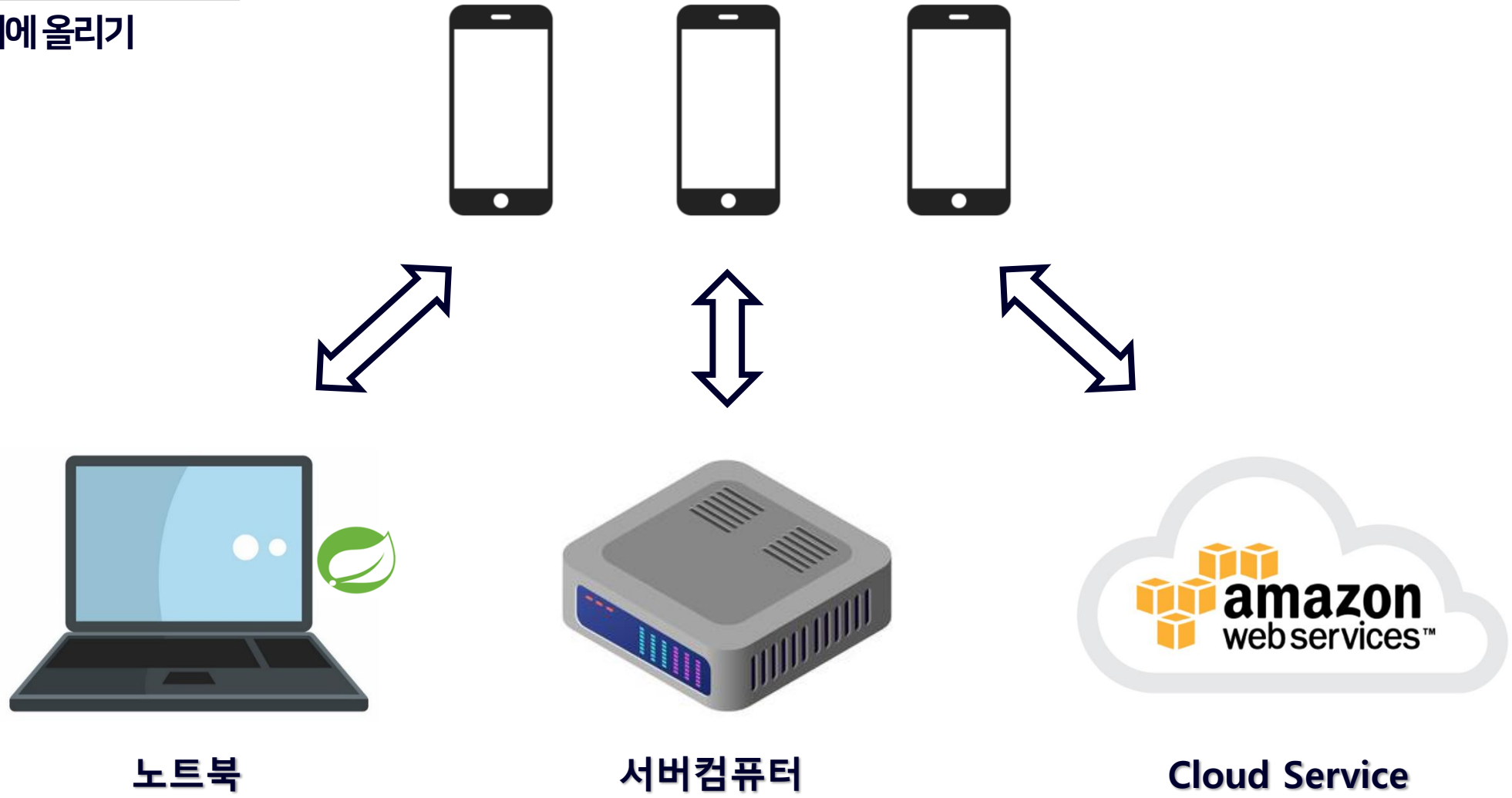
비즈니스 로직 분리하기 - 전체코드

```java
package com.example.demo.service;

import org.springframework.stereotype.Service;

@Service
public class CalculatorService {

    public int sum(int num1, int num2) {
        return num1 + num2;
    }

    public int sub(int num1, int num2) {
        return num1 - num2;
    }

    public int mul(int num1, int num2) {
        return num1 * num2;
    }

    public int div(int num1, int num2) {
        return num1 / num2;
    }

}
```

```java
package com.example.demo.controller;

import com.example.demo.service.CalculatorService;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/calculator")
public class CalculatorController {

    private final CalculatorService calculatorService;

    public CalculatorController(CalculatorService calculatorService) {
        this.calculatorService = calculatorService;
    }

    @GetMapping("/sum")
    public int sum(@RequestParam int num1, @RequestParam int num2) {
        return calculatorService.sum(num1, num2);
    }

    @GetMapping("/sub")
    public int sub(@RequestParam int num1, @RequestParam int num2) {
        return calculatorService.sub(num1, num2);
    }

    @GetMapping("/mul")
    public int mul(@RequestParam int num1, @RequestParam int num2) {
        return calculatorService.mul(num1, num2);
    }

    @GetMapping("/div")
    public int div(@RequestParam int num1, @RequestParam int num2) {
        return calculatorService.div(num1, num2);
    }
}
```

02. 실습 (1)

서버에 올리기



노트북 서버컴퓨터 Cloud Service

03 배운 내용 정리

#왜 기억을 못하니 ㅠ.ㅠ

03. 배운 내용 정리

오늘 배운 내용

01 Server란?

Server와 Client

Spring Framework 구조

Spring Boot란?

02 실습 (1)

프로젝트 구조

프로젝트 생성

간단한 API 개발

Controller 나누기

03. 배운 내용 정리

추가로 알아보면 좋을 것들

HTTP (status code, http method, headers, body, etc..)
AOP(Aspect Oriented Programming)
JSON(JavaScript Object Notation) vs XML(eXtensible Markup Language)
WAS(Web Application Server)
IoC(Inversion of Control)
DI(Dependency Injection)
MVC Model(Model-View-Controller)
Bean
기타 디자인 패턴(Singleton, Factory, Proxy, etc..)
RESTful API
Java Annotation
Project Build Tools (Gradle, Maven, Ant, etc...)
AWS (Amazon Web Service)
Persistence Framework (JPA, MyBatis)
RESTful API

03. 배운 내용 정리

어떻게 공부할까?

백엔드 개발자를 꿈꾸는 학생개발자에게 - NAVER D2 - 네이버

2018년 웹 개발자가 되기 위한 로드맵 (번역) - OKKY

인터넷 강의

- Inflearn / Udemy / Cosera / 생활코딩 / Youtube

Github

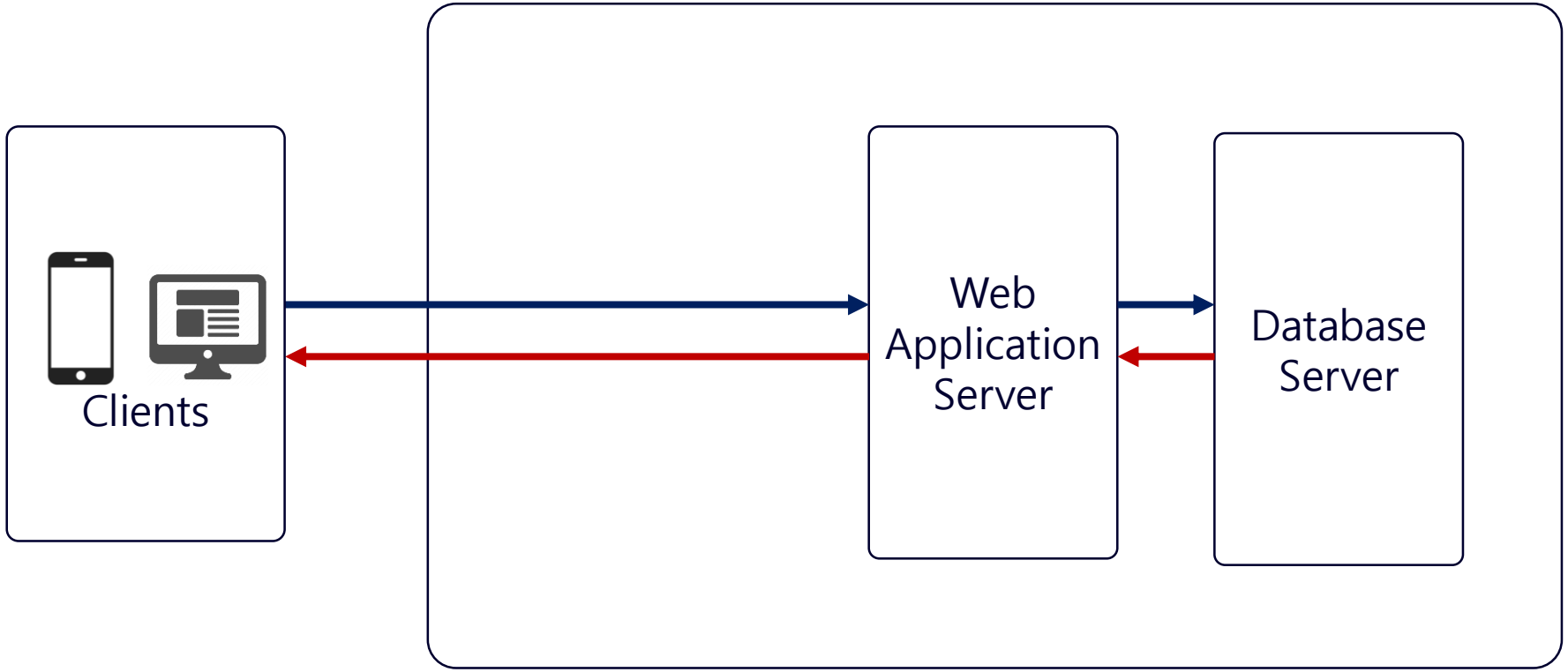
- 다른 사람들의 Github 구경

대외활동

- Naver D2 백엔드 밋업 – 현업 개발자들과 자신과 비슷한 사람들을 만날 수 있음
- Spring Camp – Spring을 사용하는 기업들이 와서 관련 경험과 팁을 발표
- SOPT – 대학생 연합 동아리. 개발, 디자인, 기획 파트로 구성되어 실제 창업 프로젝트 진행
- Nexters – SOPT와 비슷한 연합동아리. 개발자, 디자이너들로 구성
- 인턴 실습
- 공모전
- 등

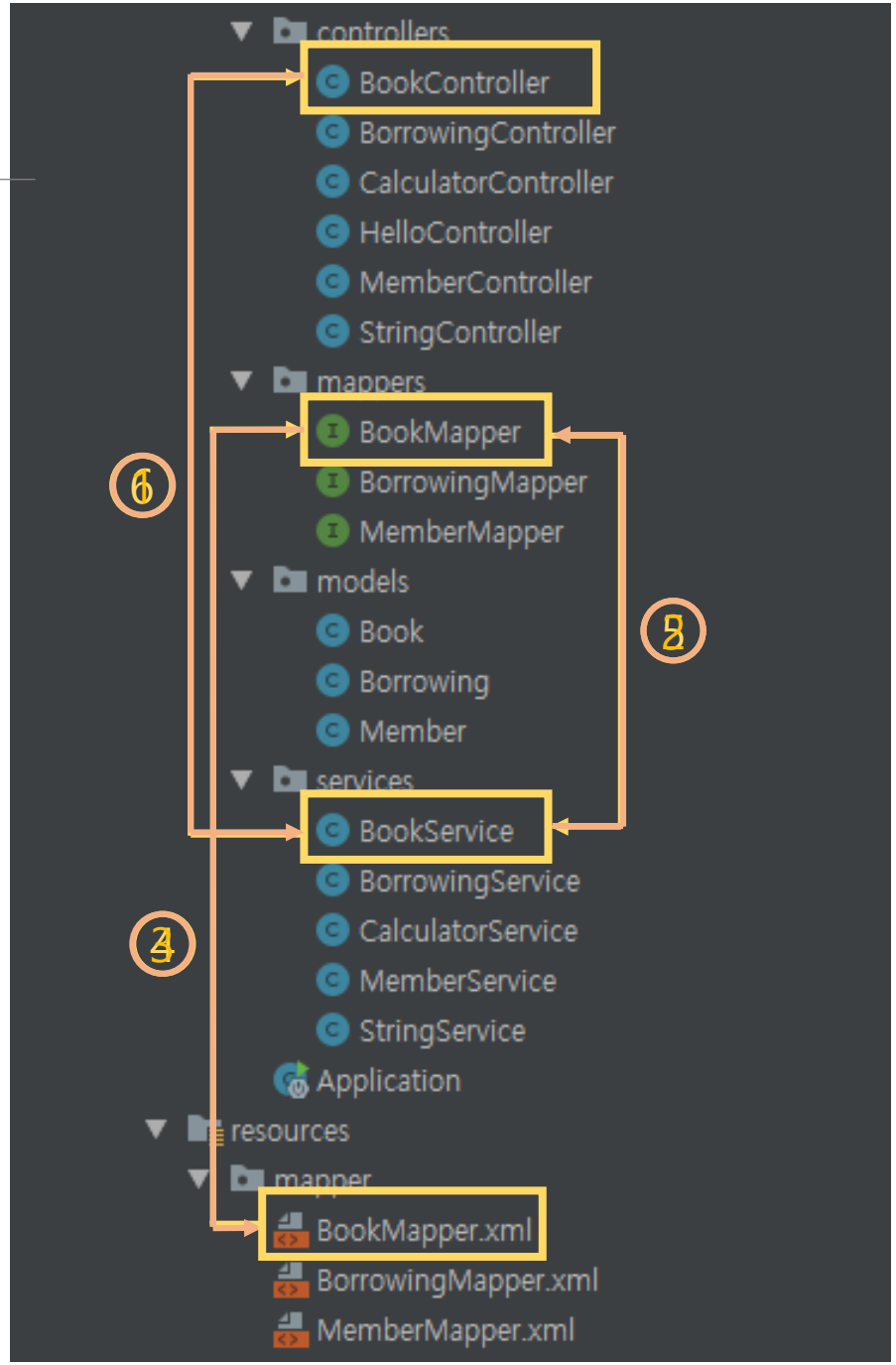
03. 배운 내용 정리

다음 실습 내용



03. 배운 내용 정리

다음 실습 내용



Controller
- Routing 해주는 역할
- URL과 method를 이어주기만 함
- Service를 호출

Service
- 비즈니스 로직 수행
- Mapper를 호출

Mapper
- DB에 접근하여 데이터 저장/처리

과제

#해주실거죠..?

과제

1. 실습한 내용을 Github Repository에 업로드
2. 해당 프로젝트에서 추가로 아래 API를 구현한 후 업로드

※ 힌트 ※ StringController를 생성해야합니다

- GET /calculator/mod?num1={num1}&num2={num2}
  - num1을 num2로 나눈 나머지
- GET /calculator/min?num1={num1}&num2={num2}
  - 둘 중 작은 값
- GET /calculator/max?num1={num1}&num2={num2}
  - 둘 중 큰 값
- GET /calculator/pow?num1={num1}&num2={num2}
  - num1의 num2제곱 ($num1^{num2}$)

- GET /string/append?str1={str1}&str2={str2}
  - str1과 str2를 합친 문자열
- GET /string/contains?str1={str1}&str2={str2}
  - str1이 str2를 포함하고 있는지 여부
- GET /string/len?str1={str1}
  - str1의 길이
- GET /string/equals?str1={str1}&str2={str2}
  - str1과 str2가 같은 문자열인지 여부

과제를 마친 분께서는 Github Repository URL을 보내주시면 코드리뷰 해드리겠습니다!

Thank you

감사합니다