

WA#LE
STUDIO



와플스튜디오 Backend Seminar

Instructor: 강지혁

2022.09.06.(화) 19:30

Session 0

Table of Contents

- 세미나장 소개
- Introducing Backend
- Framework 라는 것
- Spring Boot 란?
- Kotlin을 쓰는 이유

세미나장

- 19학번
- 17.5기 (2019년에 들어왔어요)
- 2021-2022년도 부와장
- 작년엔 Django 세미나장
- 산업기능요원 복무 중
 - Kotlin + Spring MVC

Introducing Backend

‘백엔드’ 세미나

백엔드 프로그래밍에 흥미가 있었습니다!

백엔드를 해보고싶어서

백엔드 개발자가 되고 싶습니다.

‘백엔드’ 개발자는 정확히 뭘 하는 사람?

프론트엔드랑 뭐가 다른가요?

The Internet

- [First WebSite Ever](#)

World Wide Web

The WorldWideWeb (W3) is a wide-area [hypermedia](#) information retrieval initiative aiming

Everything there is online about W3 is linked directly or indirectly to this document, includ

[What's out there?](#)

Pointers to the world's online information, [subjects](#) , [W3 servers](#), etc.

[Help](#)

on the browser you are using

[Software Products](#)

A list of W3 project components and their current state. (e.g. [Line Mode](#) ,X11 [Viola](#) , [N](#)

[Technical](#)

Details of protocols, formats, program internals etc

[Bibliography](#)

Paper documentation on W3 and references.

[People](#)

A list of some people involved in the project.

[History](#)

A summary of the history of the project.

[How can I help ?](#)

If you would like to support the web..

[Getting code](#)

Getting the code by [anonymous FTP](#) , etc.



- [요즘 웹 사이트](#)



사람에게 보이는 것 = (거의) 데이터 원형

Mostly 정적인 데이터

사람에게 보이는 것 != 데이터 원형

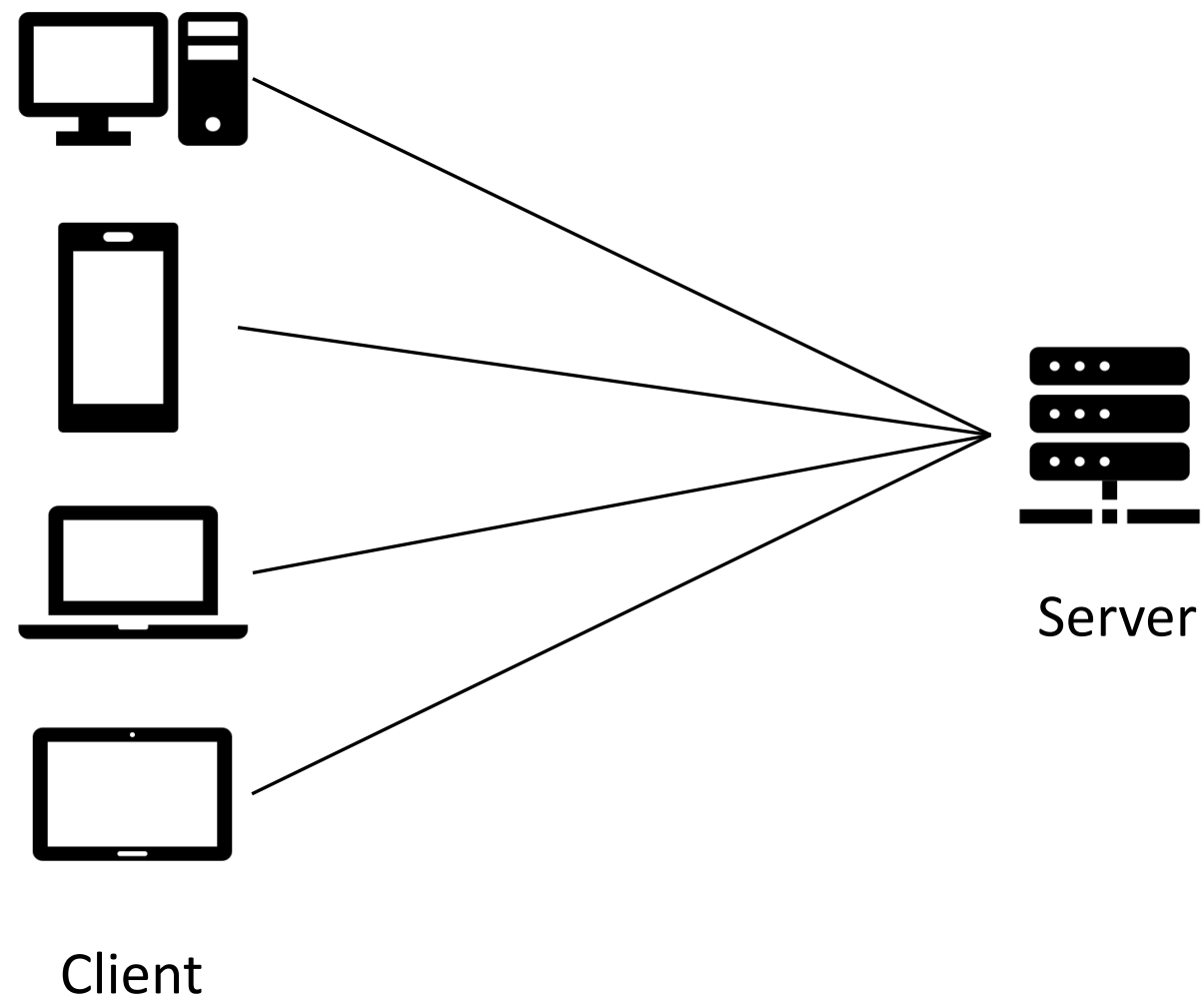
동적인 데이터가 대부분

Application Server

UI 고도화 & 복잡다양한 요구사항들의 등장

⇒ Frontend / Backend 분리

⇒ Backend : 요청에 따른 적절한 데이터 서빙



클라이언트 ⇄ 서버 :

클라이언트는 요청하고, 서버는 응답한다

브라우저 (클라이언트) ⇄ 302동 컴퓨터 (서버)

크롤링 앱 (클라이언트) ⇄ 실시간 주식 차트 (서버)

토스 앱 (클라이언트) ⇄ 은행 서버 (서버)

이들은 어떻게 통신을 주고받을까요?

⇒ HTTP, gRPC, WebSocket ...

HTTP

HTTP(Hypertext Transfer Protocol)

프로토콜이란 상호 간에 정의한 규칙

Server와 client 간에 데이터를 주고받기 위해 정의

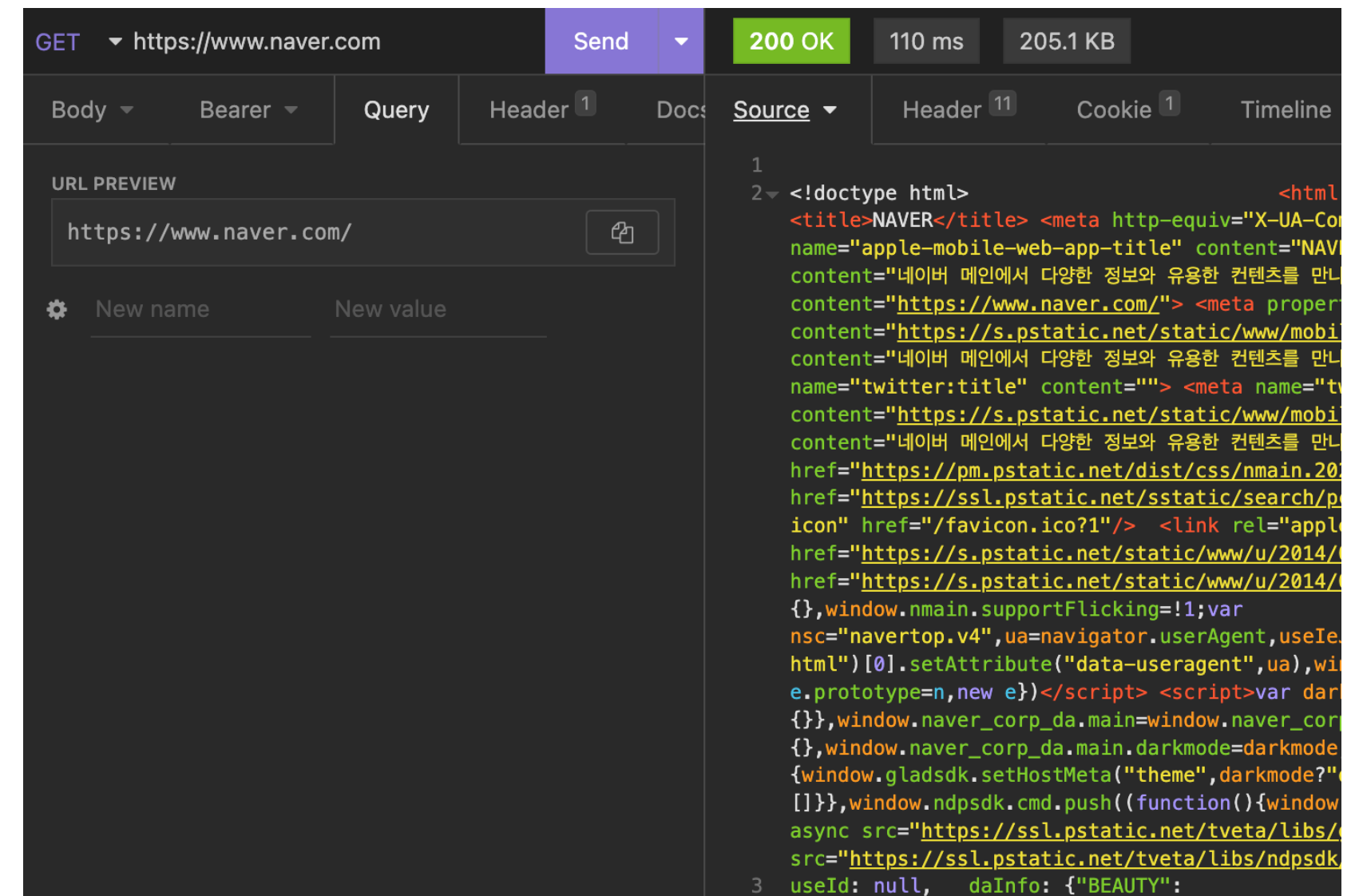
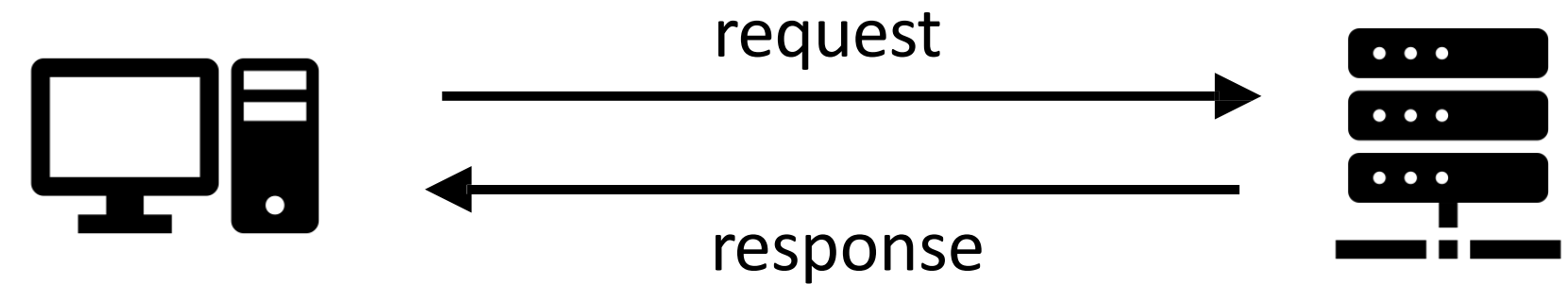
Http에 의해 서버는 항상 client가 요청할 때에만 데이터를 전송

Request : request method + URL + header + body

Response: Status code + header + body

Request method: GET,POST,PUT,DELETE,PATCH

Status code: 200(ok), 401(unauthorized), 404(not found) ...



머릿속에 떠오르는 서비스에 대해 한번 GET 요청을 날려볼까요?

HTTP Status code

10X - Informational

100 Continue

20X - Success

200 OK, 201 CREATED, 202 ACCEPTED

30X - Redirection

301 Moved Permanently , 303 See other, 307 Temporary Redirect

40X – Client Error

400 Bad Request, 401 Unauthorized, 404 Not Found, 409 Conflict

50X – Server Error

500 Internal Server Error : 대충 서버 개발자가 잘못했다는 뜻

REST API

앞서 살펴본 Web의 발전

⇒ “서비스 간 통신” (= API) 호출이 많아짐

⇒ 어떤 문제가 있을까?

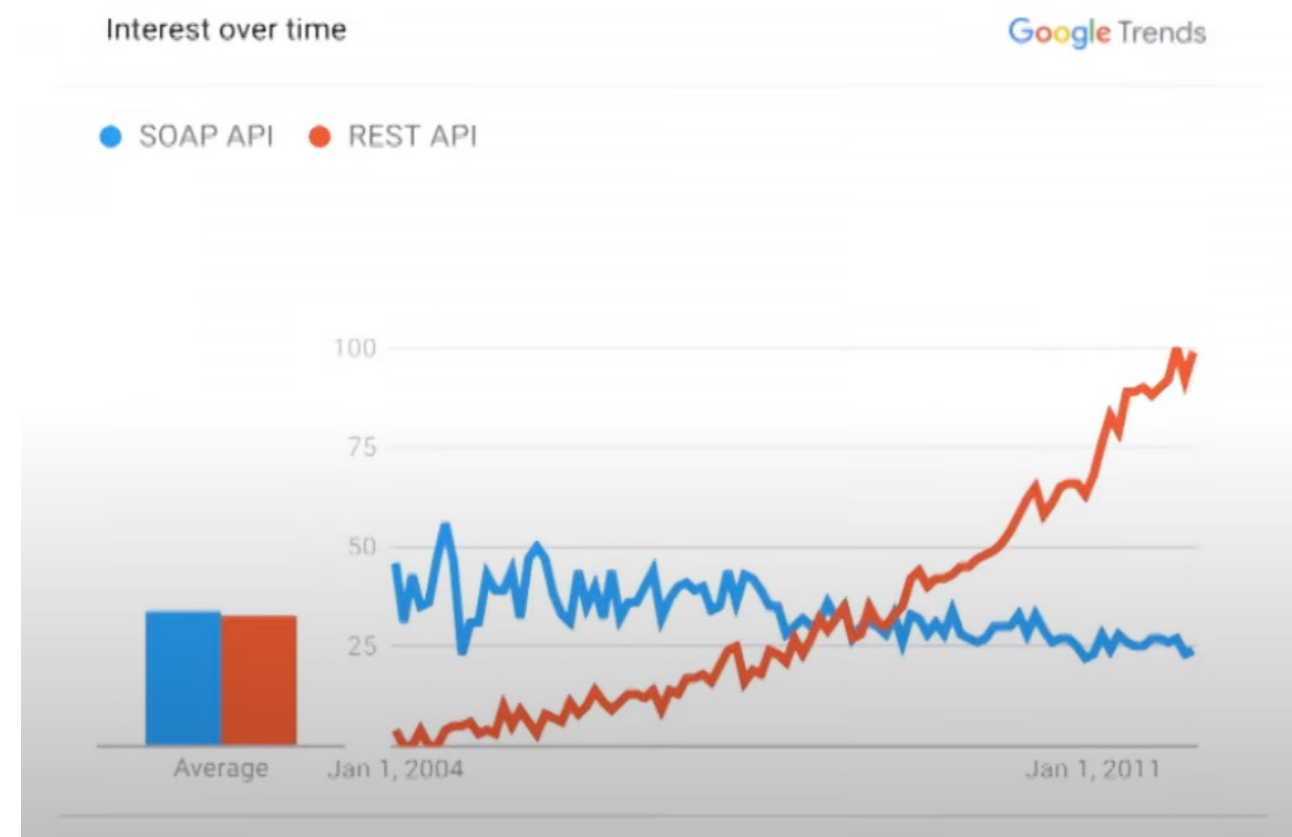
⇒ HTTP 위에서 어떻게 소통해야 하지?

“표준 형식”에 대한 필요성 

SOAP vs REST API

REST API : HTTP 기반의 아키텍처

높은 자유도 & 간단한 형식



REST API

“Representational State Transfer” By Roy Fielding (Http Author)

소프트웨어 아키텍처의 일종

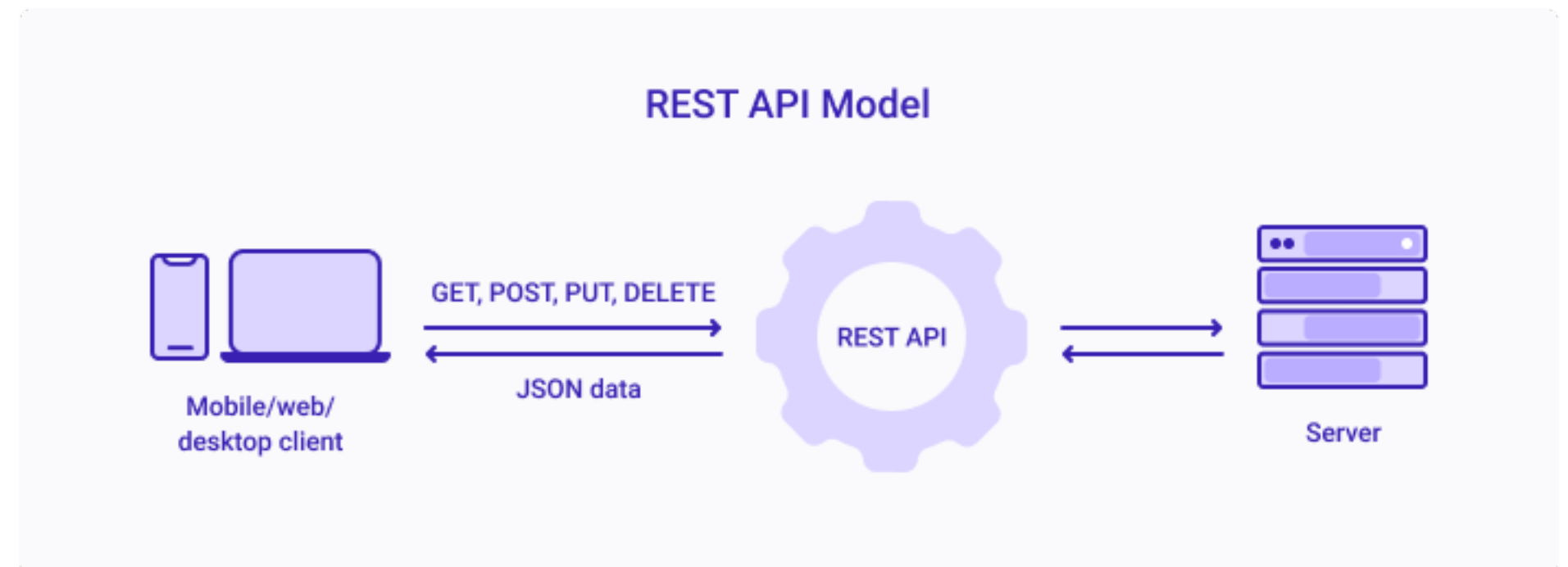
- 네트워킹을 더 잘 수행하기 위한 설계

Uniform Interface

REST의 핵심 원칙

- 요청은 자원을 식별할 수 있어야 한다.
- 요청은 자원에 대한 접근을 나타낼 수 있어야 한다.

⇒ 예시와 함께 살펴보기 ([kakao REST API](#))



[Reference: More About REST API](#)

중간 정리

여러분은 앞으로..

- 데이터를 동적으로 처리하는 Application Server를 작성합니다.
- REST API를 통해 클라이언트와 소통하는 법을 배웁니다.
 - 이따 한번 같이 만들어봐요 ! 😊
- 데이터를 어딘가에 저장하는 법을 배웁니다.
- 위와 같은 일들을 Spring Boot 프레임워크를 통해 구현할 수 있습니다.

[Application Server에 대해 더 알아보기](#)

Framework

HTTP 요청을 처리하기 위해서는..

1. HTTP 커넥션을 열고
2. HTTP요청 내용을 파싱 => URL + Parameter 획득
3. 요청 처리
4. HTTP 응답 생성
5. 클라이언트에게 전달 후 커넥션 종료

1, 2, 4, 5의 무한 반복

ANNOYING

Business Logic에만
집중하고 싶다.

Framework to Rescue

- @RestController

=> 요청 핸들링 담당

- @RequestMapping & @RequestHeader

=> 요청 파싱 담당

- 우리가 해야 할 것

=> 3번, 즉 비즈니스 로직 뿐!

```
package com.wafflestudio.seminar.controllers

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

@RestController
class SeminarController {

    @GetMapping("/user/me")
    fun userMe(
        @RequestHeader("MY-NAME") name: String,
    ): String {
        return "안녕하세요 $name 님 :)"
    }
}
```

Spring Framework : 탄생

Spring 이전의 Java 웹 개발

=> Java Enterprise Edition (J2EE) : Java 진영의 웹 개발 표준 명세

복잡하고, 제약이 많고,, 실무에 불편한 점이 너무 많았다

Whatever happened next, the framework needed a name. In the book it was referred to as the “Interface21 framework” (at that point it used com.interface21 package names), but that was not a name to inspire a community. Fortunately Yann stepped up with a suggestion: “Spring”. His reasoning was association with nature (having noticed that I'd trekked to Everest Base Camp in 2000); and the fact that Spring represented a fresh start after the “winter” of traditional J2EE. We recognized the simplicity and elegance of this name, and quickly agreed on it.



자바 진영의 웹 개발에 봄을 가져다 줄 프레임워크

[Why is Spring Framework called Spring](#)

[StackOverflow : J2EE vs Spring](#)

Spring Framework : 핵심

Java의 장점을 최대한 살려서 개발할 수 있도록 하고 싶다.

⇒ POJO (Plain Old Java Object) 기반의 “순수한” 객체지향

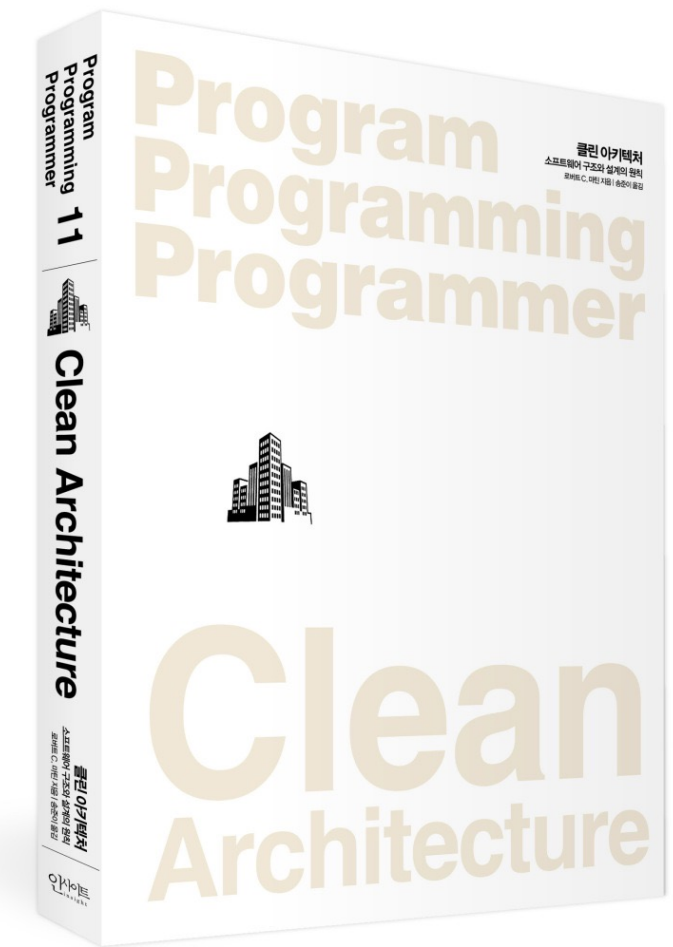
⇒ IoC Container, DI, AOP, PSA 등의 기법을 통해 그것이 가능하도록 했다

Spring은 J2EE에 대한 대안으로 출발했다

⇒ Enterprise Application 개발을 위한 프레임워크

⇒ 유지 보수, 테스트 등 “장기적으로 관리할 수 있는” 앱을 만드는 프레임워크

⇒ 객체 지향을 적절히 활용해야 그러한 목표를 달성할 수 있다 🙄

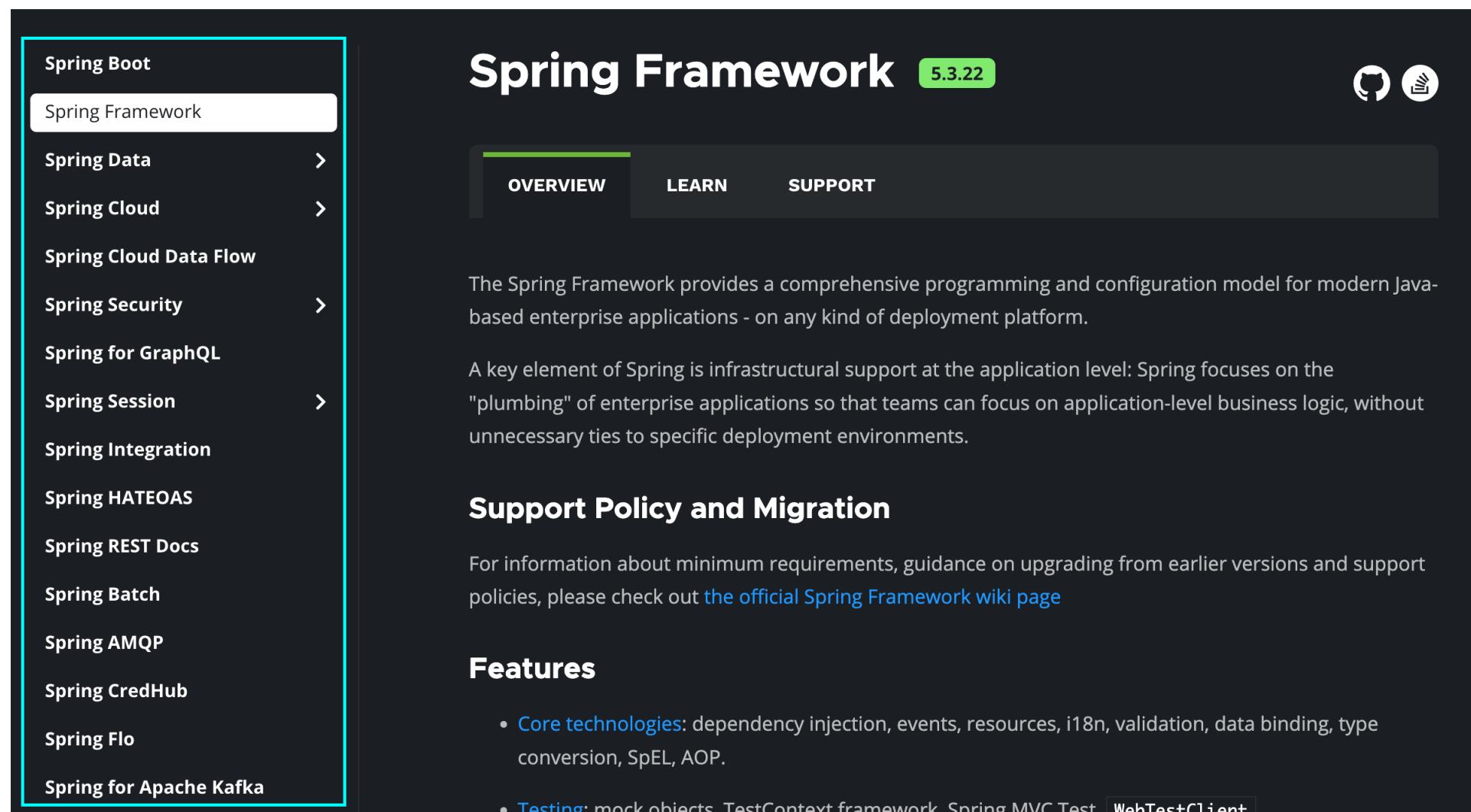


Spring Framework : 생태계

[공식 문서](#)를 들어가보면 ..

Spring, Spring Data, Spring Batch ...

스프링은 거대한 생태계, 스프링 기반 위에서 필요한 기능을 쉽게 가져다 쓸 수 있음



Spring Framework : Spring **Boot**

- 우리가 사용하는 것은 **Spring Boot** ([테코톡 영상 참고](#))
 - 설정 및 의존성 관리가 더욱 더 간편해짐
 - 내장 서버를 사용해 간단하게 빌드 / 배포 가능
 - 많은 레퍼런스 / 사용자 풀
- 분명 편리하지만,,
 - DI, IoC 등 아키텍처를 잘 이해하고 써야 함
 - Django에 비해 DB 접근에 대한 더 깊은 이해를 요구함

NAVER

kakao

배달의민족

LINE

Why Use Kotlin?

- 높은 생산성
 - 강력한 표준 라이브러리
 - 함수형 프로그래밍 ([관련 좋은 글](#))
 - 간결한 문법
- 자바와 높은 호환성을 보이면서도, 자바의 단점을 보완
 - Null type 핸들링
 - Type Inference
 - Data Class



[스프링 부트 Supports Kotlin](#)

Hello Spring !

- Kotlin + Spring 을 통해 간단한 Web 서버를 만들어보아요
 - REST API URL을 만들어봅시다
- 아키텍처, DB 등에 대해서는 다음 시간에 더 자세히 다루어 봅니다.
- 준비할 것
 - IntelliJ IDEA
 - <https://start.spring.io/>
 - 간단한 유저 관리 API
 - NO DB

REST API URL을 만들어봅시다

1. User 정보 전체 불러오기
2. 특정 User 정보만 불러오기
3. User 생성하기
4. User 수정하기
5. User 삭제하기

과제

1. Kotlin Tutorial

- <https://play.kotlinlang.org/byExample/overview>

2. Kotlin 사용해서 Recruit 문제 3번 다시 풀어 보기

- Kotlin과 OOP를 적절히 고려하며 풀어보아요

3. 간단한 REST API 구현해보기

- 자세한 내용은 [GitHub](#)에서

Q&A

