# YC Tech

## 웹 백엔드 실무 개발 프로젝트

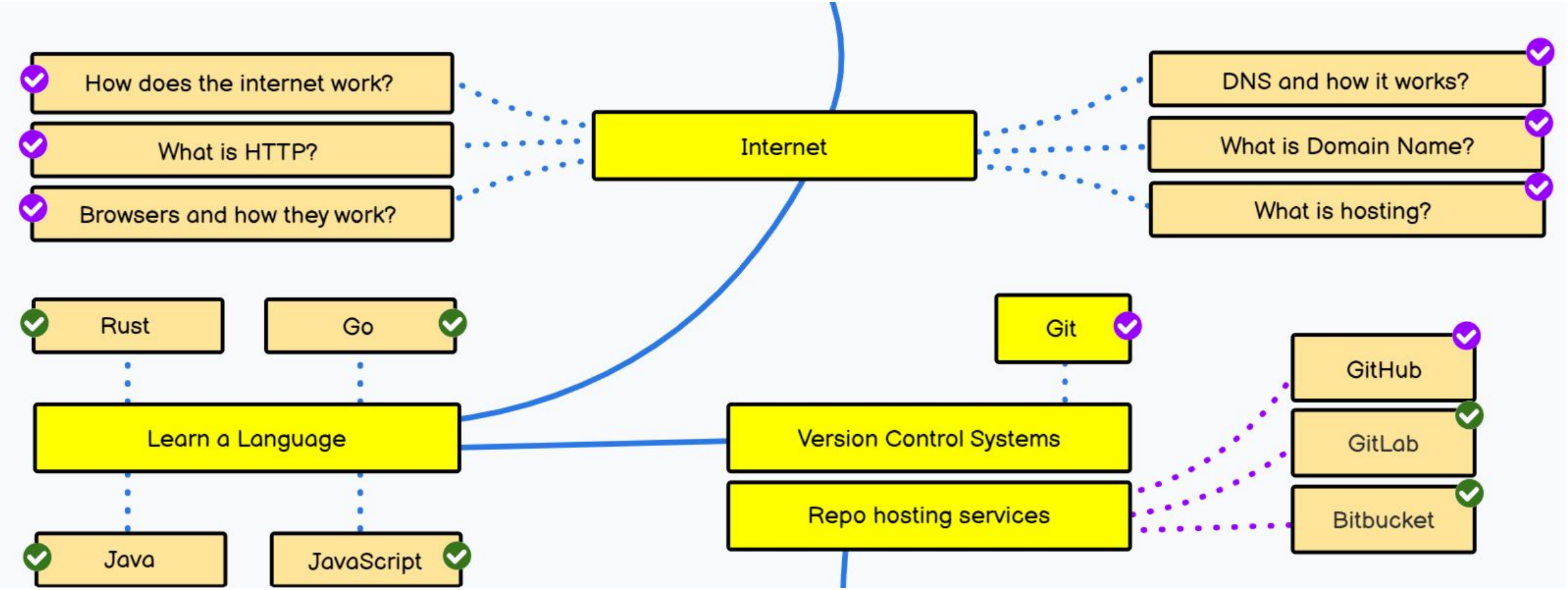

연세대학교 미래교육원

YC (Yonsei X Codepresso) Tech Academy

# Backend 개발

# Roadmap

- https://roadmap.sh/backend



# Backend 직무

- 요구사항 분석
- API 스펙 정의
- 인프라 구축 & 설계 & 운영
- Database 구축 & 설계 & 운영
- Backend 코드 구현
- 보안 체계 도입
- 내부 스크립트 도구 제작
- …

# 소프트웨어 개발 프로세스

소프트웨어 개발 프로세스(software development process)는 소프트웨어 제품을 개발하기 위해 필요한 과정 또는 구조이다.

소프트웨어 개발 프로세스에는 몇 가지 모델들이 존재하며, 이들 각각은 해당 단계별로 요구되는 활동이나 작업을 기술하고 있다.

ref)
https://en.wikipedia.org/wiki/Software_development_process


**배포**

각 환경 별로 버젼 배포

**테스트 & QA**

요구사항과 비교하여 테스트 & QA

**API 스펙 개발**

API 스펙 정의하여 frontend 에 공유 & 개발

**요구사항 분석**

요구사항 수집 & 분석

# REST API

# API (Application Programming Interface)


* API : 데이터베이스를 다루기 위한 코드

## 서버와 데이터베이스에 대한 출입구 역할

- 데이터베이스에는 중요한 정보들이 저장되기 때문에 접속을 통제해야 함
- API는 이를 방지하기 위해 여러분이 가진 서버와 데이터베이스에 대한 출입구 역할을 하며, 허용된 사람들에게만 접근성을 부여

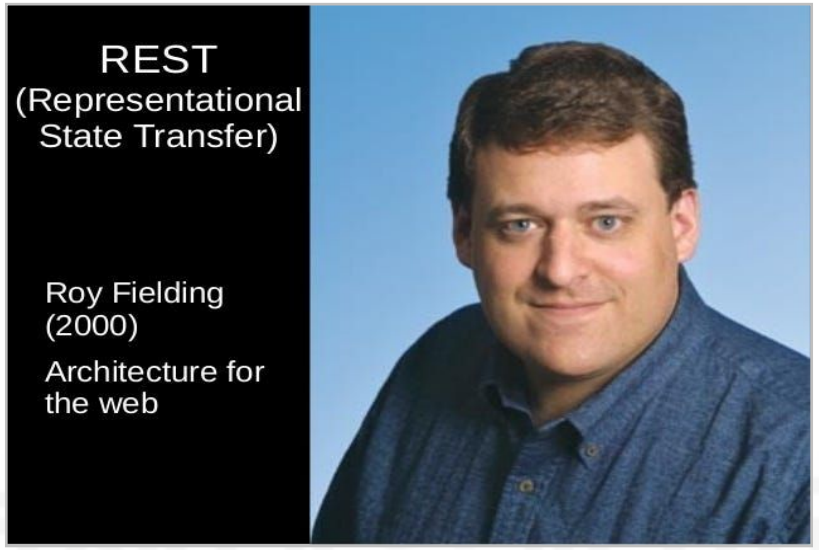
## 프로그램 끼리의 통신 가능

- 스마트폰 어플이나 프로그램 등의 통신 매개체 역할 수행
- API는 애플리케이션과 기기가 데이터를 원활히 주고받을 수 있도록 돕는 역할

## 모든 접속을 표준화

- 기계/ 운영체제 등과 상관없이 누구나 동일한 액세스를 얻을 수 있음
- USB-C 타입과 같이 API는 범용 플러그처럼 작동

# REST




https://miro.medium.com/v2/resize:fit:638/0*GN2n0rtrVwEN5XIM

## 탄생

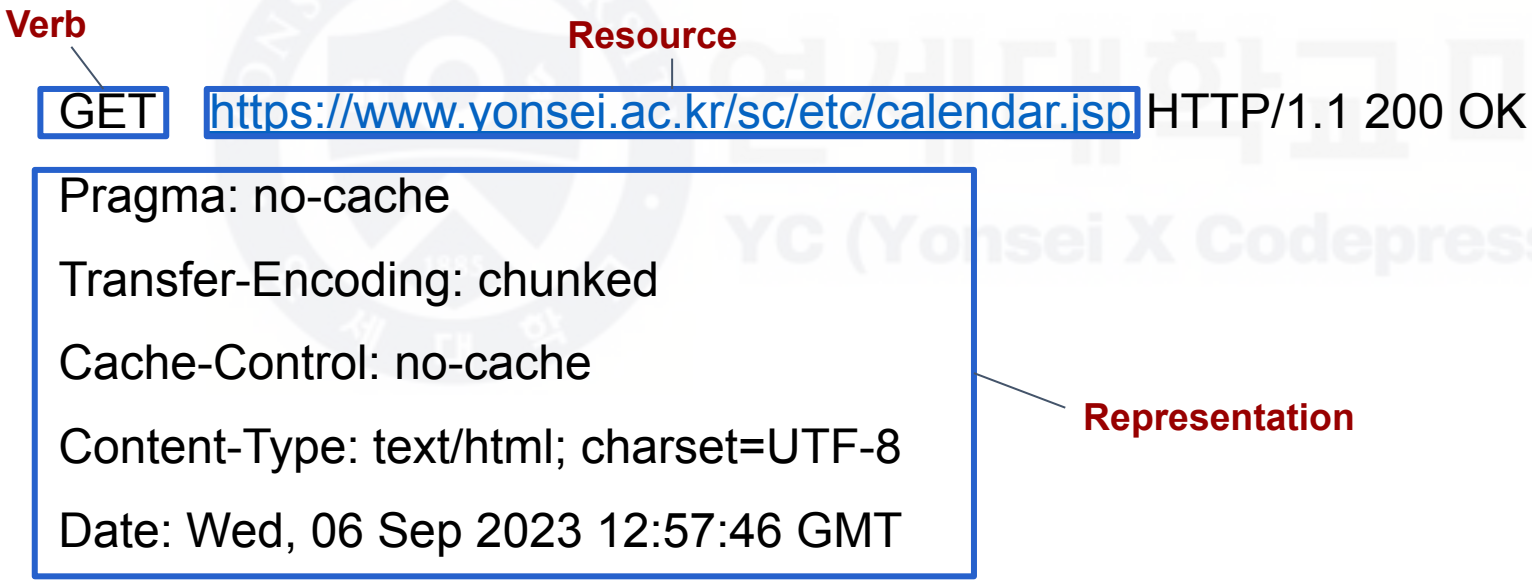
REST는 **Re**presentational **S**tate **T**ransfer라는 용어의 약자로서 2000년도에 로이 필딩 (Roy Fielding)의 박사학위 논문에서 최초로 소개되었습니다.

로이 필딩은 HTTP의 주요 저자 중 한 사람으로 그 당시 웹(HTTP) 설계의 우수성에 비해 제대로 사용되어지지 못하는 모습에 안타까워하며 웹의 장점을 최대한 활용할 수 있는 아키텍처로써 REST를 발표했다고 합니다.

# REST

## 구성

- 자원(Resource) - URI
- 행위(Verb) - HTTP METHOD
- 표현(Representations)



# REST

## 특징

- Uniform (유니폼 인터페이스) : Uniform Interface는 URI로 지정한 리소스에 대한 조작을 통일되고 한정적인 인터페이스로 수행하는 아키텍처 스타일을 말합니다.

- Stateless (무상태성) : 작업을 위한 상태정보를 따로 저장하고 관리하지 않습니다. 세션 정보나 쿠키정보를 별도로 저장하고 관리하지 않기 때문에 API 서버는 들어오는 요청만을 단순히 처리하면 됩니다. 때문에 서비스의 자유도가 높아지고 서버에서 불필요한 정보를 관리하지 않음으로써 구현이 단순해집니다.

- Cacheable (캐시 가능) : REST의 가장 큰 특징 중 하나는 HTTP라는 기존 웹표준을 그대로 사용하기 때문에, 웹에서 사용하는 기존 인프라를 그대로 활용이 가능합니다. * 네이버를 반복적으로 새로고침해도 기존 리소스를 사용하기 때문에 빠르게 불러오기 가능

- Self-descriptiveness (자체 표현 구조) : REST의 또 다른 큰 특징 중 하나는 REST API 메시지만 보고도 이를 쉽게 이해 할 수 있는 자체 표현 구조로 되어 있다는 것입니다.

- Client - Server 구조 REST 서버는 API 제공, 클라이언트는 사용자 인증이나 컨텍스트(세션, 로그인 정보)등을 직접 관리하는 구조로 각각의 역할이 확실히 구분되기 때문에 클라이언트와 서버에서 개발해야 할 내용이 명확해지고 서로간 의존성이 줄어들게 됩니다.

- 계층형 구조 : REST 서버는 다중 계층으로 구성될 수 있으며 보안, 로드 밸런싱, 암호화 계층을 추가해 구조상의 유연성을 둘 수 있고 PROXY, 게이트웨이 같은 네트워크 기반의 중간매체를 사용할 수 있게 합니다.

# REST API Best Practice

https://www.freecodecamp.org/news/rest-api-best-practices-rest-endpoint-design-examples/

< REST API를 잘 짜는 법 >
1. JSON 이용하기
2. URI에는 verb 보다 noun 사용하기
3. URI에 나타내는 자원 명사는 복수형으로 나타내기
4. 실제 서버의 상태를 error response에 잘 담아서 보내기 (error handling)
5. nesting을 사용하여 관계를 나타내기
6. filtering, sorting, pagination을 사용하여 방대한 데이터 request 처리하기
7. 보안을 위해 SSL 사용하기
8. 어떤 버전을 사용하는지 명확히 명시하기
9. 정확한 API 문서를 제공하기

## Quiz

mysite.com 에서

1. 전체 사용자를 조회하는 API URI 를 정의해보시오.
2. 특정 사용자의 인적 사항(profile)중 이름(name)으로 조회하는 API URI 를 정의해보시오.
   - ex) 사용자 중 'John' 이라는 이름을 조회하는 API URI
3. 특정 사용자의 인적 사항(profile)을 삭제하는 API URI 를 정의해보시오.

https://forms.gle/4PBw1BXjruSoUMQg8

# Spring Framework

## 탄생

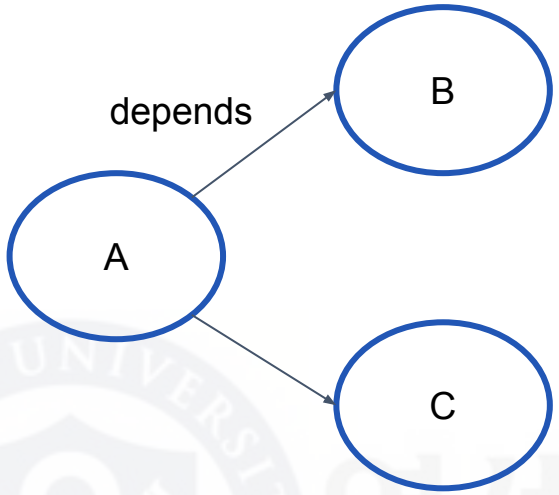
* 스프링에 비해 빠르게 설정 가능

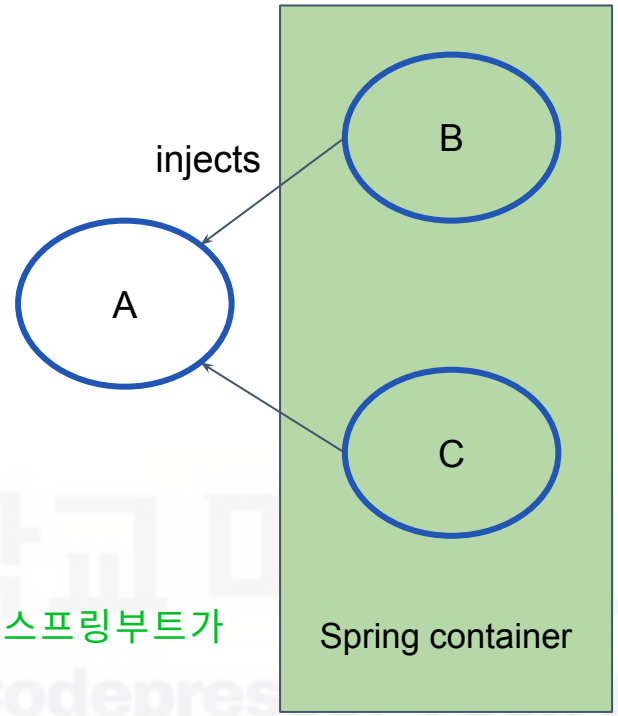
스프링 부트(Spring Boot)는 스프링의 문제점을 해결해 주기 위해 개발된 스프링의 프레임워크로 개발자들이 더 쉽고 빠르게 스프링 애플리케이션을 개발하도록 도와주기 위해 개발되었습니다.

개발 초기에 "스프링 부트 스타터"라는 프로젝트명으로 시작되었는데요. 이름에서도 느껴지듯이 간단한 설정과 구성을 통해 스프링 애플리케이션의 개발을 빠르게 시작할 수 있도록 도와주는 프로젝트였습니다.

시간이 흘러 프로젝트명은 "스프링 부트"로 변경되었고, 2014년 4월에 공식적으로 스프링 부트1.0이 출시되었습니다. 스프링 부트는 기업용 애플리케이션 개발을 더 쉽고 빠르게 하도록 도와주며, 모니터링, 건강 상태 확인(Health Check), 로깅, 설정관리 등 운영에 필요한 필수 기능을 내장하고 있습니다.

# Spring Boot 특징



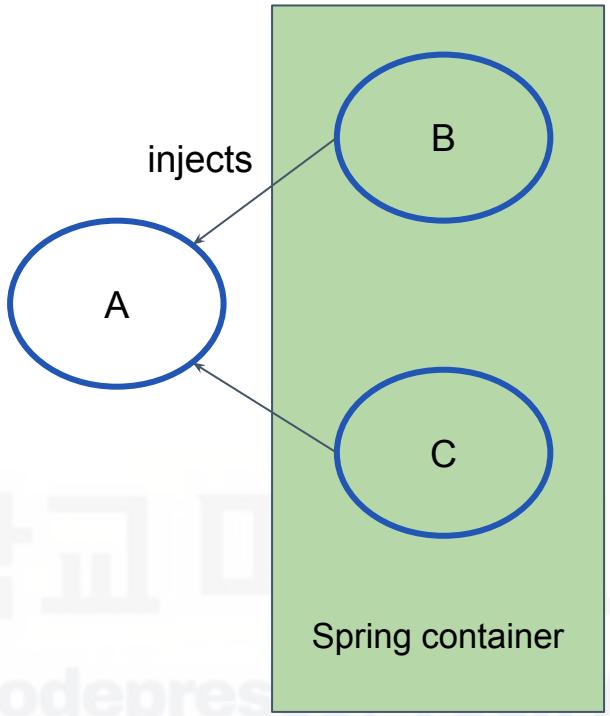

* 개발자가 추가적인 설정을 해주지 않아도 어노테이션을 삽입하면 스프링부트가 이를 인식하여 자동으로 설정을 해줌

## 제어 역전 (Inversion of Control)

- 스프링은 객체의 생명 주기 및 의존성 관리를 담당하는 IoC 컨테이너를 제공합니다. 개발자는 객체의 생성과 관계 설정을 스프링에 위임할 수 있으며, 스프링 컨테이너가 객체의 생명 주기를 관리하고 필요한 의존성을 주입합니다.

# Spring Boot 특징

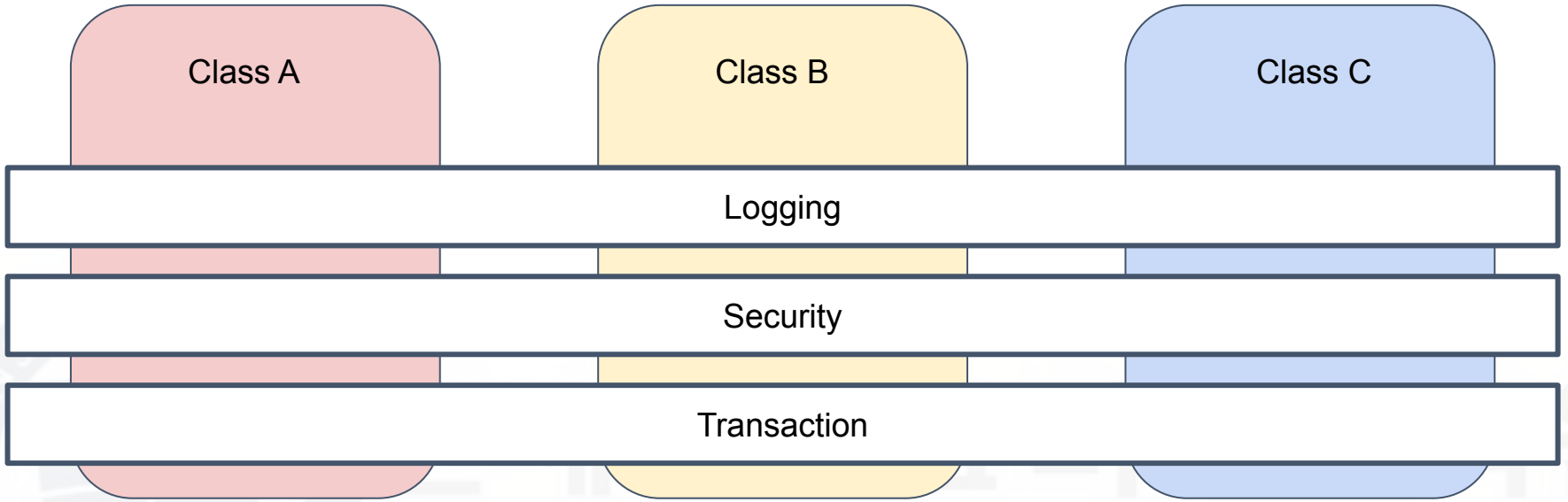
```
public class MainApp {
    public static void main(String[] args) {
        // Load the Spring container
        ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
        // Retrieve the Person bean from the container
        Person person = (Person) context.getBean("person");
        // Use the Person object
        person.displayInfo();
    }
}
```




## 의존성 주입(Dependency Injection)

- 스프링은 의존성 주입을 통해 객체 간의 관계를 설정합니다. 의존성 주입은 애플리케이션의 결합도를 낮추고 유연성과 테스트 용이성을 향상시킵니다.

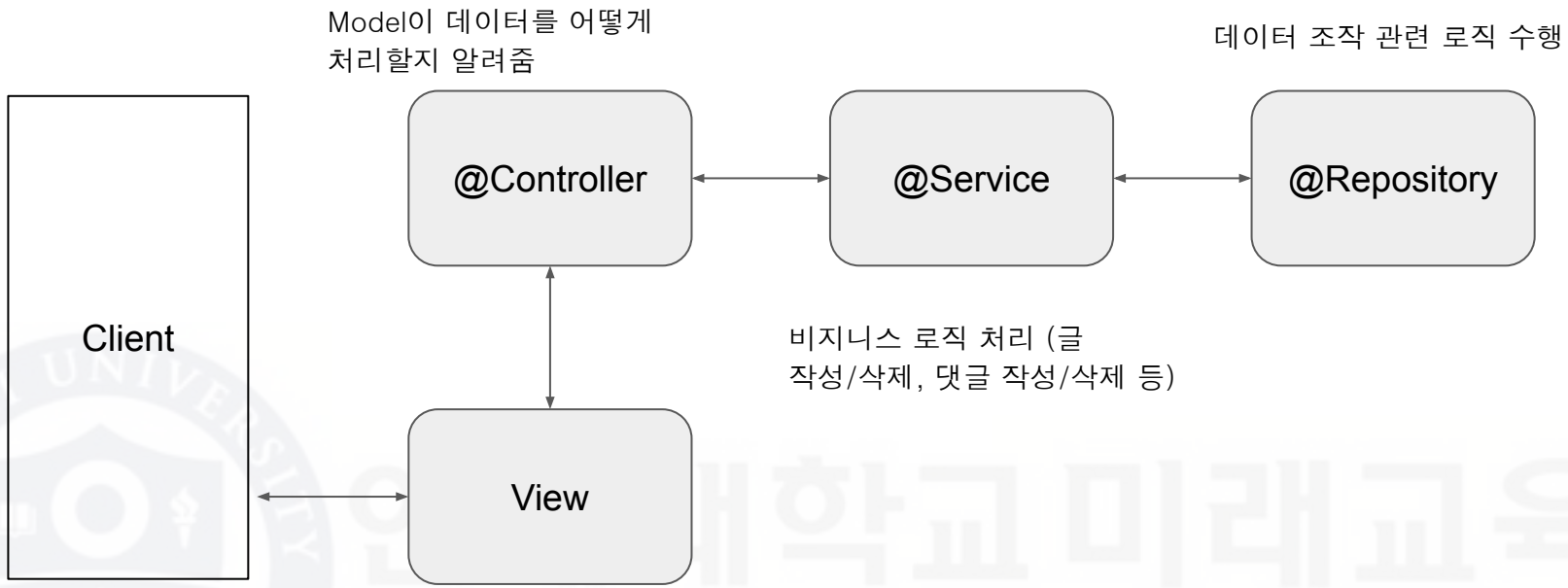
# Spring Boot 특징




## AOP지원(관점 지향 프로그래밍)

- 스프링은 AOP를 지원하여 애플리케이션의 핵심 비즈니스 로직과 부가적인 기능(로깅, 트랜잭션 관리 등)을 분리하여 모듈화할 수 있습니다.

# Spring Boot 특징




## 웹 개발 지원

- 스프링은 웹 애플리케이션 개발을 위한 다양한 기능과 웹 프레젠테이션 계층을 제공합니다. 스프링 MVC는 유연하고 확장 가능한 웹 애플리케이션을 개발할 수 있는 MVC(Model-View-Controller) 아키텍처를 지원합니다.

  * 관점과 기능을 분리함으로써 수정 및 변경에 용이

# MVC

## Model

- Data와 애플리케이션이 무엇을 할 것인지를 정의하는 부분으로 내부 비즈니스 로직을 처리하기 위한 역할을 합니다. 즉, 모델은 컨트롤러가 호출을 하면 DB와 연동하여 사용자의 입출력 데이터를 다루는 일과 같은 데이터와 연관된 비즈니스 로직을 처리하는 역할을 합니다. 데이터 추출, 저장, 삭제, 업데이트 등의 역할을 수행합니다.

## View

- 사용자에게 보여주는 화면(UI)이 해당됩니다. 사용자와 상호작용을 하며 컨트롤러로부터 받은 모델의 결과값을 사용자에게 화면으로 출력하는 일을 합니다. MVC에서는 여러개의 View가 존재할 수 있습니다. Model에서 받은 데이터는 별도로 저장하지 않습니다.

## Controller

- Model과 View 사이를 이어주는 인터페이스 역할을 합니다. 즉, Model이 데이터를 어떻게 처리할지 알려주는 역할을 합니다. 사용자로부터 View에 요청이 있으면 Controller는 해당 업무를 수행하는 Model을 호출하고 Model이 업무를 모두 수행하면 다시 결과를 View에 전달하는 역할을 합니다.

# Spring Boot Component

```java
@RestController
@RequestMapping("/api")
public class SampleRestController {

    @GetMapping("/hello")
    public String sayHello() {
        return "Hello, World!";
    }

    @PostMapping("/greet")
    public String greet(@RequestBody String name) {
        return "Hello, " + name + "!";
    }
}
```

## Controller

- Model과 View 사이를 이어주는 인터페이스 역할을 합니다. 즉, Model이 데이터를 어떻게 처리할지 알려주는 역할을 합니다. 사용자로부터 View에 요청이 있으면 Controller는 해당 업무를 수행하는 Model을 호출하고 Model이 업무를 모두 수행하면 다시 결과를 View에 전달하는 역할을 합니다.
- @Controller vs @RestController
  https://www.geeksforgeeks.org/difference-between-controller-and-restcontroller-annotation-in-spring/
  * @Controller: view를 반환 / @RestController: REST API를 처리하기 위함, 객체 데이터를 반환

# Spring Boot Component

```java
@Service
public class SampleService {

    public String generateGreeting(String name) {
        return "Hello, " + name + "!";
    }
}
```

## @Service

- Repository에서 얻어온 정보를 바탕으로 가공 후 다시 Controller에게 정보를 보내는 곳 Controller는 클라이언트에, Repository는 데이터에 맞닿아서 정보를 주고받는 부분으로 여길 수 있으나 실질적으로 중요한 작동이 많이 일어나는 부분 애플리케이션의 비즈니스 로직 처리와 비즈니스와 관련된 도메인 모델의 적합성을 검증하고, 트랜잭션을 처리한다.

# Spring Boot Component

```
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
    User findByUsername(String username);
    List<User> findByAgeGreaterThan(int age);
}
```

## @Repository

- @Repository는 해당 클래스가 DB에 접근하는 클래스임을 명시한다.
- 스프링 데이터 접근 계층으로 인식하고 데이터 계층의 예외를 스프링 예외로 변환해준다.
- JPA를 사용하면 보통 JpaRepository를 상속받는다.

# 프로젝트 환경 구성

# 준비물

**Student 프로그램을 운영중인 서비스도 있기 때문에, 대학 이메일 계정으로 가입&설치하는 것을 추천**

## 설치해야될 프로그램

- Source code
- IntelliJ CE / Ultimate ( https://www.jetbrains.com/ko-kr/idea/download )
- OpenJDK ( https://openjdk.org/install )

## 가입해야될 서비스

- Github ( https://www.github.com )

# 프로젝트 구성

- Java 17
- Spring Boot
- JPA, MySQL
- Freemarker
- Gradle
- AWS

# Version Control System


https://i.pinimg.com/originals/4f/17/dc/4f17dce8ccf24c32f5083e47c401f46b.jpg

- 버전관리 시스템(VCS, Version Control System)이란 파일 변화를 시간에 따라 기록했다가 나중에 특정 시점의 버전을 다시 불러올 수 있는 시스템을 의미합니다.

- VCS를 사용하면 선택한 파일을 이전 상태로 되돌릴 수 있고, 변경 사항을 비교하고, 변경한 사람 및 변경시기를 추적할 수 있습니다. 또한, 파일을 잃어버리거나 잘못 고쳤을 때도 쉽게 복구할 수 있습니다.

- 협업 도구로써 사용되기도 합니다.
- https://www.github.com

# Build tool

## 탄생

- 웹, 앱 프로그래밍 개발이 발전하며 어플을 개발함에 있어 필요한 라이브러리도 많아지게 되었다 이 많은 라이브러리를 직접 다운받아서 추가하여 사용하는 방법도 있지만 많은 번거로움이 이따른다 이때문에 발생한것이 빌드 도구이다.

## 특징

- 소스 코드를 컴파일, 테스트, 정적분석 등을 실행하여 실행 가능한 애플리케이션으로 자동 생성하는 프로그램
- 계속해서 늘어나는 라이브러리 자동 추가 및 관리
- 프로젝트를 진행하며 시간이 지남에 따라 라이브러리의 버전을 동기화

# Build tool

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>info.solidsoft.rnd</groupId>
    <artifactId>spock-10-groovy-24-gradle-maven</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <surefire.version>2.18.1</surefire.version>
    </properties>
    <build>
        <plugins>
            <plugin>
                <groupId>org.codehaus.gmavenplus</groupId>
                <artifactId>gmavenplus-plugin</artifactId>
                <version>1.4</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>compile</goal>
                            <goal>testCompile</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
            <plugin>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>${surefire.version}</version>
                <configuration>
                    <includes>
                        <include>**/*Spec.java</include> <!-- Yes, .java extension -->
                        <include>**/*Test.java</include> <!-- Just in case having "normal" JUnit tests -->
                    </includes>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <dependencies>
        <dependency>
            <groupId>org.codehaus.groovy</groupId>
            <artifactId>groovy-all</artifactId>
            <version>2.4.1</version>
        </dependency>
        <dependency>
            <groupId>org.spockframework</groupId>
            <artifactId>spock-core</artifactId>
            <version>1.0-groovy-2.4</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

pom.xml

maven

```
apply plugin: 'groovy'

group = "info.solidsoft.rnd"
version = "0.0.1-SNAPSHOT"

repositories {
    mavenCentral()
}

dependencies {
    compile 'org.codehaus.groovy:groovy-all:2.4.1'

    testCompile 'org.spockframework:spock-core:1.0-groovy-2.4'
}
```

build.gradle

```
rootProject.name = 'spock-10-groovy-24-gradle-maven'
```

settings.xml

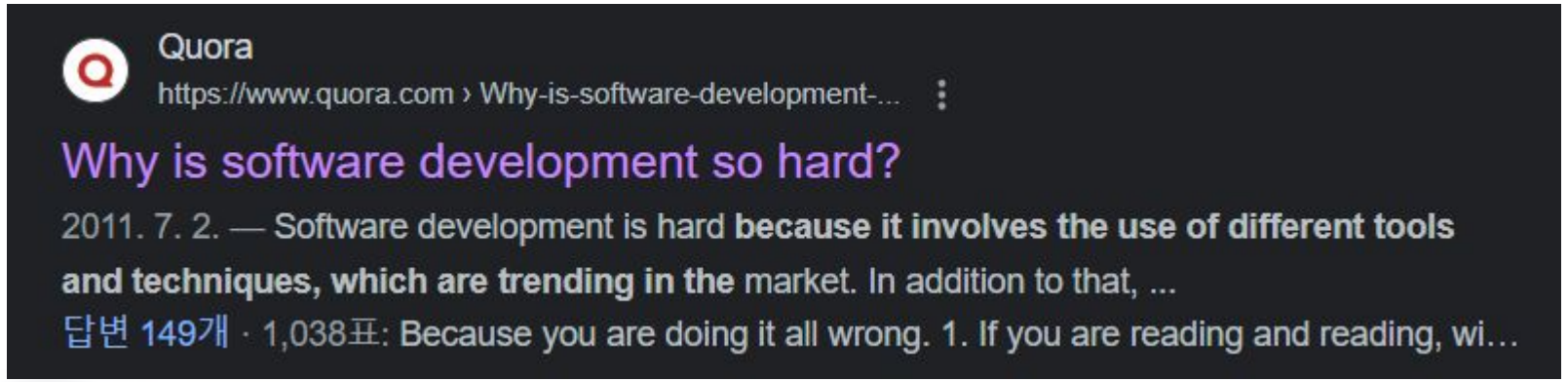
gradle

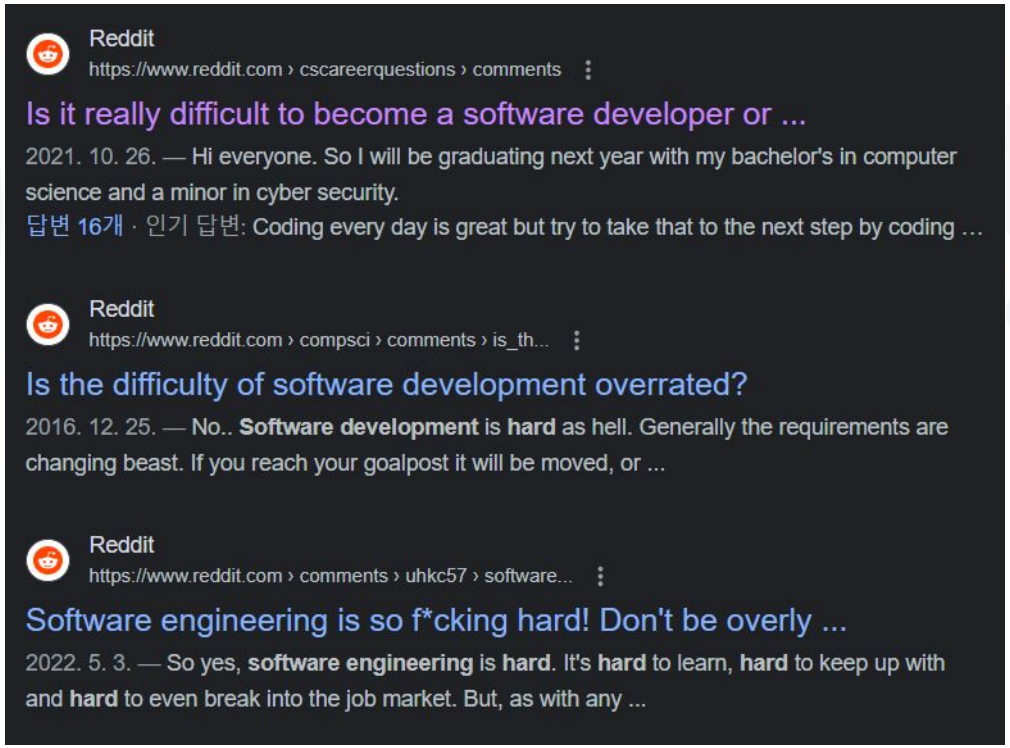
https://miro.medium.com/v2/resize:fit:1200/1*JFMnZ7bLx94LIZ6p-29PbA.png

* Gradle의 경우 방금 전 빌드한 내용을 어느 정도 캐싱하고 있어 빌드 속도가 더 빠름

# REST API 문서 설계

# 어려운 소프트웨어 개발



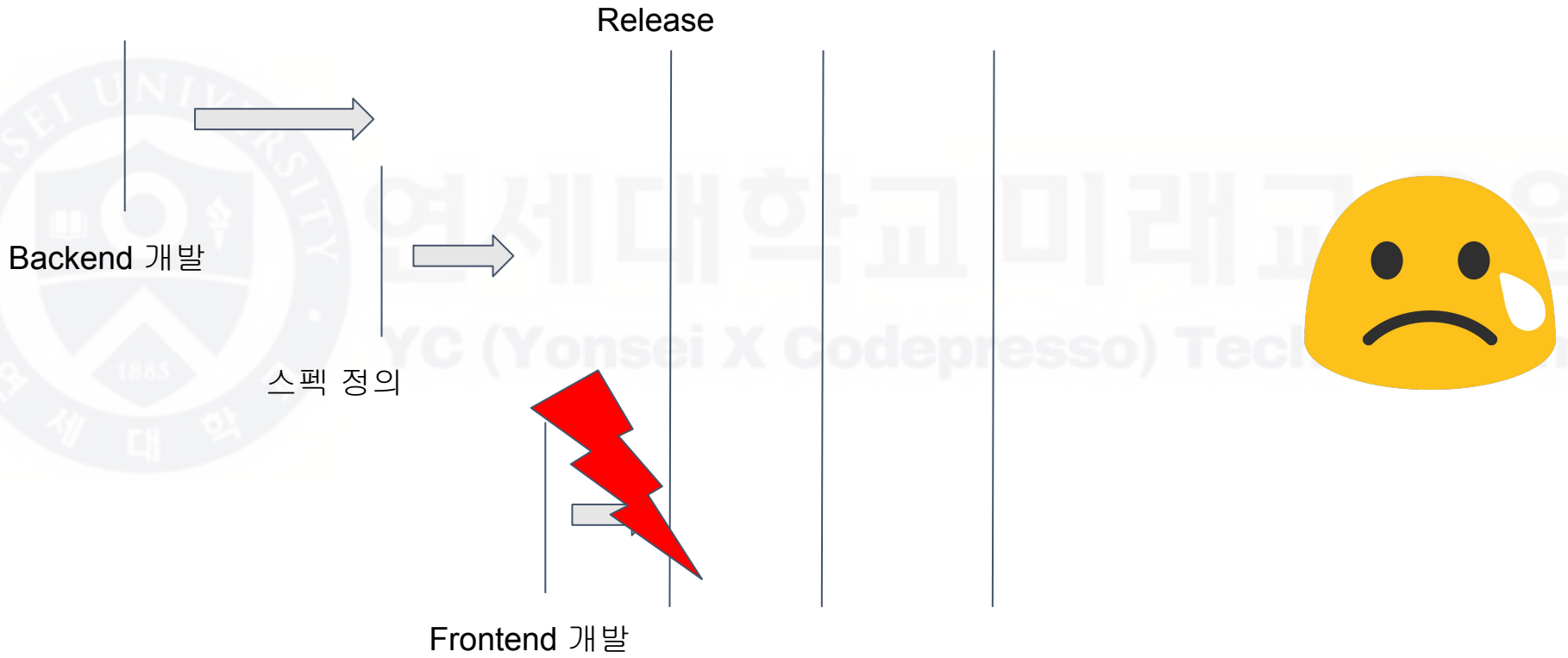
# 예측을 하기 쉽도록 빠른 피드백

# 예시

당신은 연말을 맞아 새로운 기능을 개발해야 합니다. 우리 Backend 팀의 고객은 iOS, Android, Web Frontend 개발 팀입니다. API 개발을 하던 중 간헐적으로 발생하는 특정 이슈가 생겨 구현한 로직을 변경해야 합니다.

# 예시

당신은 연말을 맞아 새로운 기능을 개발해야 합니다. 우리 Backend 팀의 고객은 iOS, Android, Web Frontend 개발 팀입니다. API 개발을 하던 중 간헐적으로 발생하는 특정 이슈가 생겨 구현한 로직을 변경해야 합니다.



# OpenAPI 3

- OpenAPI specification(이전의 Swagger specification)은 API를 설명하고 문서화하기 위한 오픈 소스 포맷입니다.

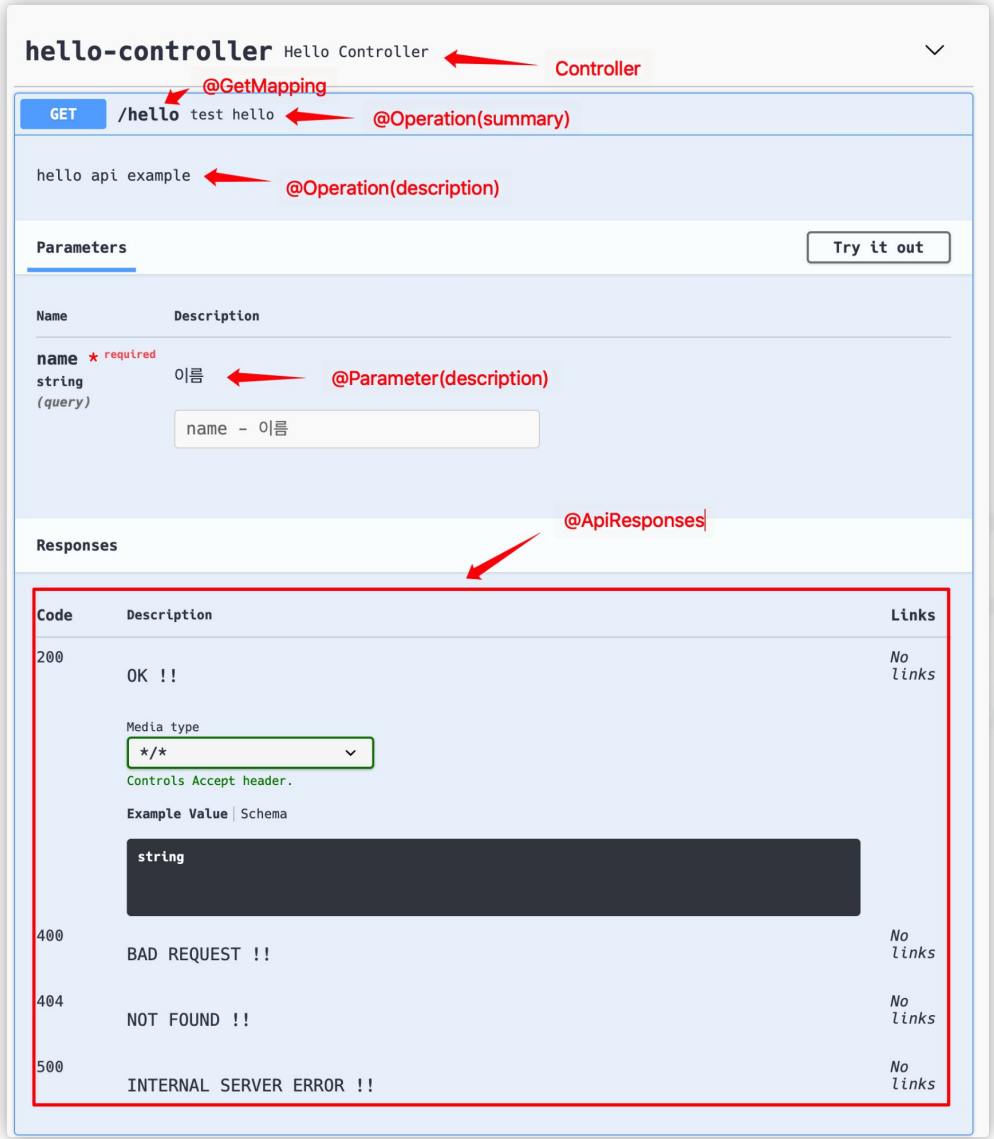
- RESTful API를 설계하고 설명하기 위한 사실상의 표준입니다. OpenAPI의 최신 버전은 3.0이며, OpenAPI 정의는 JSON 또는 YAML로 작성할 수 있습니다. 읽고 쓰기 더 쉽기 때문에 YAML을 권장합니다.

ref) https://swagger.io/specification/

# Swagger

- Swagger 는 REST API를 설계, 빌드, 문서화 및 사용하는 데 도움이되는 OpenAPI 사양을 중심으로 구축 된 오픈 소스 도구 세트입니다. - About Swagger Specification
- Swagger는 annotation을 통해 문서를 작성할 수 있다. 테스트 할 수 있는 UI를 제공한다.
- 문서 화면에서 API를 바로 테스트 할 수 있다.
- https://petstore.swagger.io

# Swagger




- https://springdoc.org

# Spring Initializr

https://start.spring.io/

- 스프링 프로젝트 시작하기

https://start.spring.io/#!type=gradle-project&language=java&platformVersion=2.7.15&packaging=jar&jvmVersion=17&groupId=com.example&artifactId=sns&name=sns&description=&packageName=com.example.sns&dependencies=devtools,lombok,freemarker,web,data-jpa

# Coding 을 시작하기 전에

- Rule 1: Follow a consistent coding standard
- Rule 2: Name things properly, long variable and function names are allowed
- Rule 3: Be expressive, write code as you speak, and be optimally verbose
- Rule 4: Max indent per method should be 2, in case of exceptions 3
- Rule 5: Avoid creating god object and long methods
- Rule 6: Keep the method in one place, inject the class and call it, DRY
- Rule 7: Avoid in-line comments (comment with code), put comments in the method doc

ref) https://blog.yipl.com.np/7-golden-rules-of-simple-and-clean-code-and-some-more-considerations-slides-e66662af2daf

# 요구사항 수집

회원 관리

- 회원가입
- 로그인
- 로그아웃
- 프로필 보기

게시글 관리

- 게시글 등록/삭제/수정
- 댓글 등록/삭제

...

| RQ-ID | 화면명 | 요구사항명 | 요구사항 내용 | 날짜 | 작성자 | 진행사항 | 버전명 |
|---|---|---|---|---|---|---|---|
| RQ-0001 | 관리자 | 통계 | 매물의 [조회수], [장비등록일], [장비판매일]을 기록한다. | 2/3 | 송미경 | 반영 | 0.2.0 |
| RQ-0002 | 공통 | 오른쪽 영역 | 동급 모델 표시를 해준다. | 12/28 | 송미경 | 다음 버전 | - |
| RQ-0003 | 공통 | 오른쪽 영역 | 최근 본 매물이 화면 오른쪽에 있다. (아래)<br>- 3개씩 5페이지 - 총 15개 | 12/28 | 송미경 | 반영 | 0.5.5 |
| RQ-0004 | 공통 | 오른쪽 영역 | 많이 본 매물이 화면 오른쪽에 있다. (위)<br>- 3개씩 1페이지 - 총 3개<br>- 판매완료된 매물은 제외한다.<br>- 7일 기준이며, 월요일 날 기록을 리셋시킨다.<br>- 기록이 없는 경우, 랜덤으로 3개 띄운다.<br>- 많이 본 매물에서 선택시 카운트 제거<br>- 조회수 기반이다. | 12/28 | 송미경 | 반영 | 0.5.5 |
| RQ-0005 | 공통 | 상단 메뉴 구성 | 딜러와 관련된 메뉴를 구성한다.<br><br>방법1) 상단을 사용자를 위한 메뉴로만 구성, 푸터에 딜러관련 메뉴 배치 -> 웹사이트 이용자 중에서 사용자가 다수일 때 선택<br>방법2) 상단에 매물등록 등을 배치 -> 사용자에게 여기 딜러가 있다는것을 암시 | 12/28 | 송미경 | 미반영 | - |
| RQ-0006 | 공통 | 명칭 변경 | 추천 매물 -> 스페셜 매물, 일반 매물 -> 추천 매물로 변경한다. | 1/2 | 송미경 | 반영 | 0.4.0 |
| RQ-0007 | 공통 | 명칭 변경 | 딜러 -> 판매자<br>제휴딜러 -> 제휴업체 | 1/9 | 송미경 | 반영 | 0.5.0 |
| RQ-0008 | 공통 | 마이페이지 | 사용자 마이페이지<br>- 판매자 마이페이지와 동일 | 1/9 | 송미경 | 반영 | 0.5.0 |
| RQ-0101 | 인트로 | 화면 개설 | [인트로] 화면이 필요하다.<br>- 중고장비매매서비스, 장치매칭서비스<br>- 예시 : http://line25.com/articles/interesting-web-design-trend-vertical-split-layouts | 12/16 | 송미경 | 반영 | 0.2.0 |
| RQ-0102 | 인트로 | 언어 | 추후 다국어를 지원할 계획이다. | 12/16 | 송미경 | 다음 버전 | - |

https://mklab-co.medium.com/%EC%9E%91%EC%84%B1%EB%B2%95-%EC%9A%94%EA%B5%AC%EC%82%AC%ED%95%AD-%EB%AA%85%EC%84%B8%EC%84%9C-requirements-specification-ad3533d6d5b8

# To DO

- ❑ 개발 환경 설정
- ❑ Github 저장소 생성 & push
- ❑ 프로젝트 구동
- ❑ 테스트 controller 작성 & swagger 문서 작성
- ❑ 프로젝트 주제 선정
- ❑ 요구사항 추출

# EOD