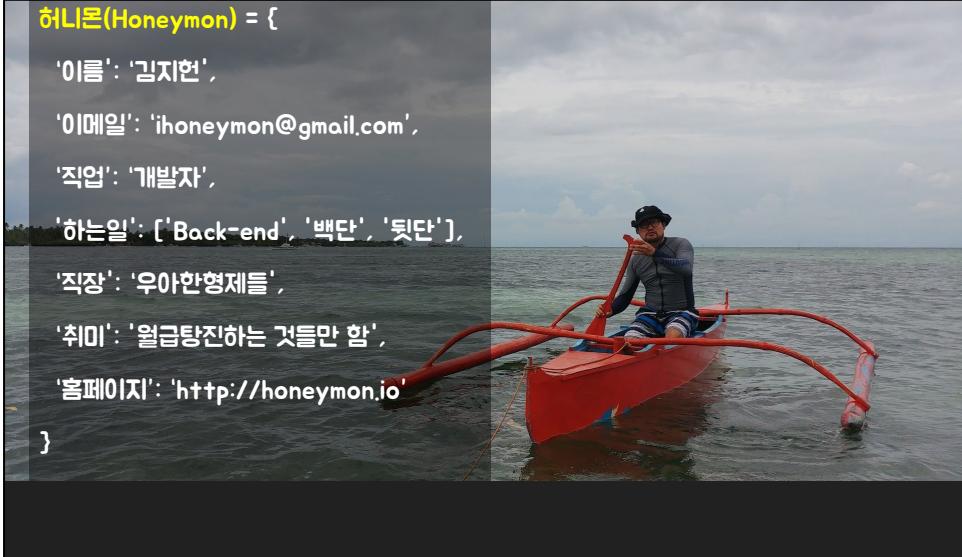


스프링 부트를 이용한 웹 서비스 개발하기

발표자 소개

```
허니몬(Honeymon) = {  
    '이름': '김지현',  
    '이메일': 'ihoneymon@gmail.com',  
    '직업': '개발자',  
    '하는일': ['Back-end', '백단', '뒷단'],  
    '직장': '우아한형제들',  
    '취미': '월급탕진하는 것들만 함',  
    '홈페이지': 'http://honeymon.io'  
}
```



제 소개를 먼저 하겠습니다.

제 이름은 김지현입니다. 이름보다는 허니몬이라는 활동명이 조금 더 유명합니다.
한동안 블로그가 뜼했는데, 다시 열심히 적도록 하겠습니다.
한동안 수습생활하면서 조용히 있었거든요. ^^;

진행순서

- 1. 스프링 부트 소개**
- 2. 스프링 부트 프로젝트 만들기**
- 3. 스프링 부트 웹 서비스 개발**

이 과정을 함께 통해서 스프링 부트를 기반으로 웹 서비스를 개발할 수 있는 기본지식을 갖출 수 있길 바랍니다.

첫시간에는 '스프링 부트'를 소개합니다. 스프링 부트는 빌드도구, 스프링 프레임워크, 스프링 부트 스타터를 기반으로 구성되어 있습니다. 스프링 부트의 기본적인 동작 원리를 이해하고 배우면서 익혀야할 것이 얼마나 많은지 경악(!?)하게 될 것입니다.

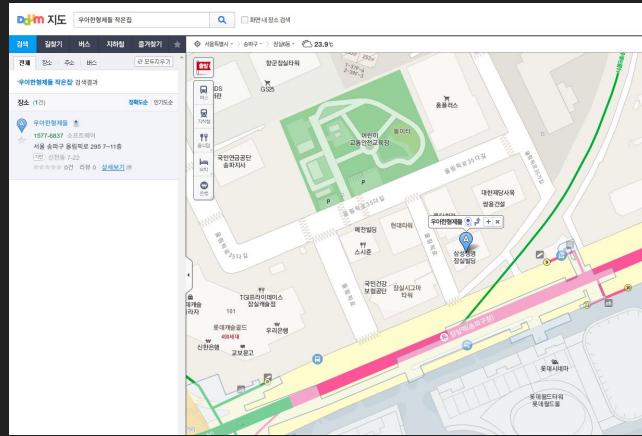
두번째시간에는 '스프링 부트 프로젝트'를 만들어봅니다. 빌드도구로 그레이들을 사용하여 스프링 부트 스타터를 추가하고 이를 통해서 스프링 부트 작동원리를 살펴본다. 간단하게 정보를 조회하고 등록할 수 있는 API를 만들어보겠습니다. CRUD 기능을 만들어보는 거죠. 이 과정에서 업무(=비즈니스 로직)구현에만 집중할 수 있도록 해주는 스프링 부트의 특징을 체험할 수 있다.

묻고 답하기

진행과정에서 궁금했던 점이나 묻고 싶은 것이 있으시면 서슴없이! 물어봐주세요 .^^

묻고 싶은 게 있으시다면

- 이메일: ihoneymon@gmail.com
- 트위터: [@ihoneymon](https://twitter.com/ihoneymon)
- 현피: 우아한형제들 작은집 7층에서 "빌링정산개발팀 김지현"을 찾아주세요.



꽃

스프링 부트 소개

Boot Spring Boot!

스프링 부트는 스프링 프레임워크를 기반한 개발 프레임워크다.

Spring Boot 1.0.0.GA

2014.04.01.

RELEASES

Spring Boot 1.0 GA Released

 RELEASES  PHIL WEBB  APRIL 01, 2014  10 COMMENTS

On behalf of the entire Spring Boot team, I am very pleased to announce the general availability of Spring Boot 1.0! You can download the 1.0.1 with an important security fix [here](#).

You'll find everything you need to get going at projects.spring.io/spring-boot, and from our ever-growing collection of "[Getting Started](#)" guides (most of which use Spring Boot).

It's been 18 months since the original request to "[improve containerless web application architectures](#)", that gave birth to Spring Boot, was raised. Since then we have seen 1720 commits by 54 different contributors, we've closed 549 issues, and have had the code forked 398 times. Thanks!

Why containerless? Today's PaaS environments provide much of the management, scale out, and reliability features already, so we focus on making spring boot an ultralight container, great for application or service deployment in the cloud. If you've not yet seen Spring Boot in action, here is a canonical "Hello World!" web application that you can actually run using the [CLI tool](#).

2014.04.01. 스프링 부트 1.0 출시

컨테이너 없는(컨테이너를 집어삼킨!) 웹 애플리케이션 아키텍처를 구현하는 것을 목표로 18개월만에 출시

Spring Boot 2.0.0.RELEASE

2018.03.01.



2018년 3월 10일 스프링 부트 2.0

Java 8 이상

스프링 프레임워크 5.0, 스프링 시큐리티 5.0, 스프링 Data JPA 2.0 적용



대 **스프링** 부트 입문(서) 시대

스프링 부트 입문자를 겨냥한 다양한 입문서 출시
 스프링 부트는 사용해보는 것은 쉽지만 활용하는 것은 쉽지 않다.
 스프링 프레임워크에 대한 이해가 없으면 한계에 부딪치게 된다.

그리고...



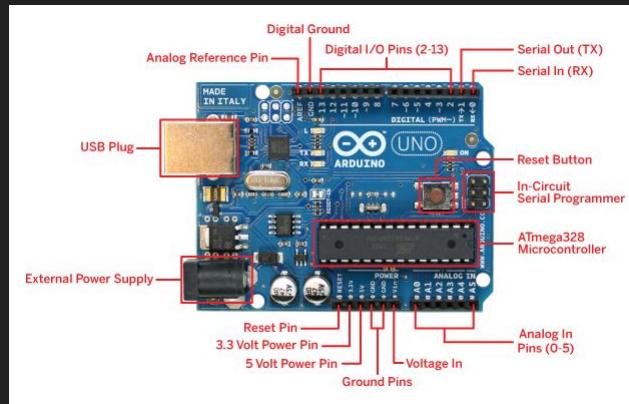
스프링 부트 참고서(Reference Documentation)를 표방하고 나온 책...

스프링 부트 기능정의

- 단독실행 가능한 스프링 애플리케이션 생성
- 내장 컨테이너로 톰캣, 제티 혹은 언더토우 중에서 선택가능
- 스타터(starter)를 통해 간결한 의존성 구성 지원
- 스프링에 대한 자동구성(Auto-Configuration) 제공
- 더이상 XML구성 필요없음
- 제품출시(release) 후
운영에 필요한 다양한 기능(상태점검, 모니터링 등) 제공

스프리 부트 기능적 정의 6가지

스프링 프레임워크를 기반으로 한 개발플랫폼



'개발플랫폼'이라 불리기 위해서는 개발에 필요한 라이브러리 의존성 관리, 빌드 및 배포, 나아가서 운영을 위한 편의기능까지 제공할 수 있어야 합니다. 스프링 부트는 개발 및 운영 전과정을 아우르는 기능을 제공합니다. 그래서 최근 스프링 부트 이용이 늘어나고 관련 서적의 출시도 늘어나고 있습니다. 자바 진영에서는 스프링 부트를 압도할만한 개발플랫폼이 당분간은 나올 것 같지 않습니다.

스프링 부트 구성요소

빌드도구(그레이이틀 vs 메이븐)

스프링 프레임워크(4.X vs 5.X)

스프링 부트(v1.5 vs v2.0)

스프링 부트 스타터(spring-boot-starter)

Build - Code - Deploy

프랑스 철학자 '장 폴 사르트르'가 '인생은 BCD이다'라고 이야기 했습니다. 태어나고 (Birth) - 죽는(Death) 사이에서 수많은 선택을 합니다.
‘개발자인생은 BCD이다.’라고 생각합니다. 구축(Build)하고 배포(Deploy)하는 사이에 수많은 기능을 구현(Code)합니다.-

Spring Boot Agenda

- **Build**
- **Code**
 - **spring-boot-starter**
 - **Auto-configuration**
 - **Programming in Spring Environment**
- **Deploy**

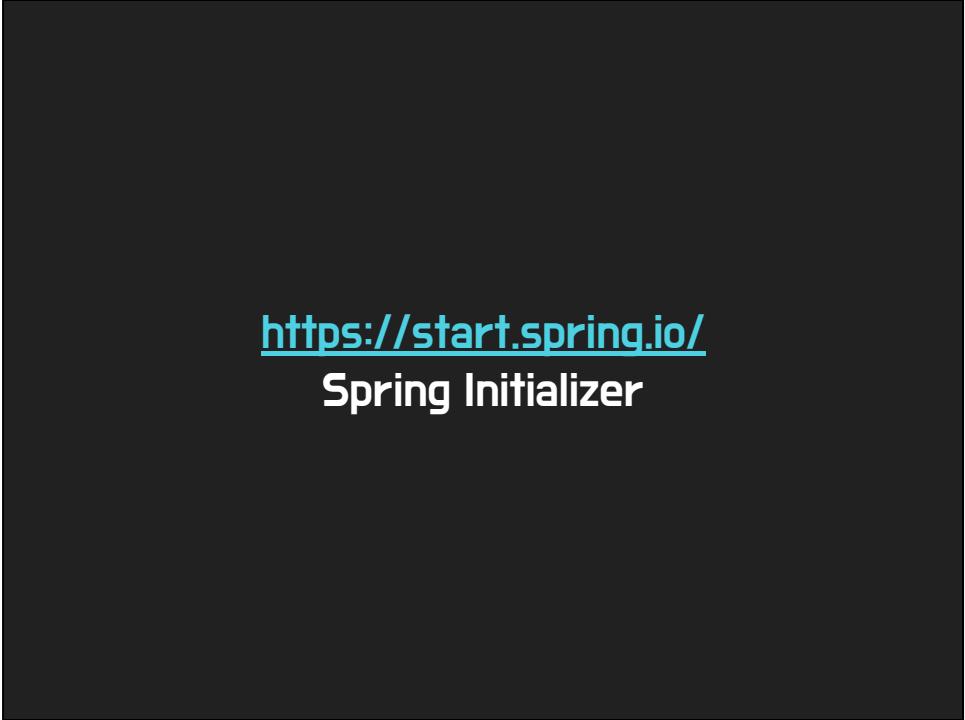
Spring Boot Starter: <https://goo.gl/MwZxZv>

BOM(Bill of Material)

spring-boot-starter-dependencies

빌드(Build)

스프링 부트 프로젝트를 시작할 때 우리는 선택해야 합니다.



<https://start.spring.io/>
Spring Initializer

<https://start.spring.io/>

스프링 부트 프로젝트가 생성되는 곳입니다. 선택된 값을 가지고 스프링 부트 프로젝트를 생성하여 **zip** 파일로 다운로드 받을 수 있습니다.

IDE에서는 이렇게 내려받은 **zip** 파일 압축을 풀고 불러와서 **build.gradle** 혹은 **pom.xml** 을 읽어서 프로젝트 정의된 의존성 라이브러리를 추가하고 이용 가능한 상태로 만든다.

Java vs Kotlin vs Groovy

어떤 프로그래밍 언어를 사용할 것인가?

<https://www.technotification.com/2018/08/new-demanding-programming-languages.html>

자바와 코틀린 중 자신이 사용하는 프로그래밍 언어를 선택한다.

빌드도구(그레이이틀 vs 메이븐)



그레이들(Gradle) - build.gradle

```
buildscript { // 그레이들 바이너리 플러그인 버전 정의
    ext {
        springBootVersion = '2.0.4.RELEASE' // 스프링 부트 버전 정의
    }
    repositories {
        mavenCentral()
        maven { url "https://plugins.gradle.org/m2" } // 그레이들 플러그인 저장소
    }

    dependencies { // 바이너리 플러그인 의존성 정의
        classpath("org.springframework.boot:spring-boot-gradle-plugin:${springBootVersion}")
    }
}

apply plugin: 'java' // 언어 플러그인
apply plugin: 'eclipse' // eclipse 지원기능 제공
apply plugin: 'idea' // intelliJ 지원기능 제공(sourceSets)
apply plugin: 'org.springframework.boot' // 스프링 부트 그레이들 플러그인 정의

group = "io.honeymon.boot"
version = "1.0.0.RELEASE"

jar { // jar 패키징시 파일명 및 버전기준 설정
    baseName = "${project.name}"
    version = "${project.version}"
}
```

`buildscript` 는 그레이들 바이너리 플러그인 버전을 정의하는 부분입니다.

메이븐(Maven) - pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>io.honeymon.springboot.boot</groupId>
  <artifactId>boot-spring-boot</artifactId>
  <version>1.0.0.RELEASE</version>
  <packaging>jar</packaging>

  <name>boot-spring-boot</name>
  <description>Boot Spring Boot Project</description>
  <url>https://github.com/ihoneymon/boot-spring-boot</url>
  <organization>
    <name>honeymon.io</name>
    <url>http://honeymon.io</url>
  </organization>

  <parent> <!-- spring-boot-parent 정의 및 스프링 부트 버전 정의 -->
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.3.RELEASE</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>

  <!-- 생략 -->
```

프로젝트 기본 코드구조

```
.  
|   build.gradle (1)  
|   gradle  
|     wrapper  
|       |   gradle-wrapper.jar (2)  
|       |   gradle-wrapper.properties  
|   gradlew (3)  
|   gradlew.bat (4)  
|   settings.gradle (5)  
+-- src  
    |   main  
    |     java  
    |       |   io.honeymon  
    |       |         t  
    |       |           |   springboot  
    |       |               |   TSpringBootApplication.java (6)  
    |     resources  
    |       |   application.yml (7)  
    |       |   static  
    |       |   templates  
    |   test  
    |     java  
    |       |   io.honeymon  
    |       |         t  
    |       |           |   springboot  
    |       |               |   TSpringBootApplicationTests.java (8)
```

그레이들도 메이븐의 기본구조를 따르고 있습니다. `src/main/java`, `src/main/resources`

(Executable) JAR vs WAR

JAR냐 WAR냐 하는 패키징 방식은 내가 어디에 어떻게 배포하느냐에 따라 달라지게 됩니다.

스프링 부트는 기본적으로 실행가능한 JAR로 애플리케이션을 배포합니다. 생성된 JAR 파일을 JVM과 JRE가 있는 환경이면 어디서나 실행할 수 있도록 하고 있습니다. WAR의 경우에는 별도의 WAS에 배포한다는 가정을 했을 때 선택을 합니다.

스프링 부트는 빌드시 재포장 작업을 합니다. 애플리케이션과 그 실행에 필요한 의존성을 가지는 라이브러리를 한데 묶는 첫번째 단계 후에 내장컨테이너를 넣고 다시 한번 포장을 합니다.

실행가능한(Executable) JAR(or WAR)

고전적인 WAR

bootRepackage

APP



APP



스프링 부트에서 생성하는 실행가능한 JAR는 스프링 부트에서 제공하는 빌드 플러그인이 배포본 생성과정에서 재압축(리패키징) 과정을 통해 컨테이너를 내장시킨다.

스프링 부트 1.5.X vs 2.X

스프링 부트는 버전에 따라서 제공하는 라이브러리의 버전이 달라집니다.

- 스프링 부트 1.5 는 스프링 프레임워크 4.3 을 기반으로 하고 있음
- 스프링 부트 2.x 는 스프링 프레임워크 5.0 을 기반으로 하고 있음

스프링 부트는 2.0 부터!

 Spring Boot @springboot · 7월 30일

Important note for Spring Boot 1.x users. If you haven't upgraded to Spring Boot 2, now is a great time to do so!

트윗 번역하기



Spring Boot 1.x EOL Aug 1st 2019
All good things must come to an end, and for the 1.x line of Spring Boot that means we will need to cease maintenance twelve months from today, on Aug 1st 2...
spring.io

14 266 229

2018년 3월 10일 스프링 부트 2.0이 출시했습니다.

2019/08/01 스프링 부트 1.X에 대한 지원중단을 통해 사망처리 예정

스프링 부트는 애너테이션 기반 작동

스프링 프레임워크의 컴포넌트스캔 기능을 적극 활용

스프링 부트를 구성하는 애노테이션(Annotation)

- `@SpringBootApplication`(with `SpringApplication`)
- `@ComponentScan`
- `@EnableAutoConfiguration`
- `@Configuration`
- `@ConditionalOn~~`
- `@SpringBootConfiguration`(= `@Configuration`)
- `@EnableConfigurationProperties`
- `@ConfigurationProperties`

스프링 부트가 동작하는 애노테이션들이다 .

코드(Code)

```
/**  
 * 스프링 부트 애플리케이션이 시작되는 곳!  
 * {@link SpringApplication}을 살펴보세요. 스프링 부트의  
 * 마법이 시작되는 곳입니다.  
 *  
 * @author honeymon  
 */  
  
@SpringBootApplication  
public class BootSpringBootApplication {  
    public static void main(String[] args) {  
        SpringApplication.run(BootSpringBootApplication.class, args);  
    }  
}
```

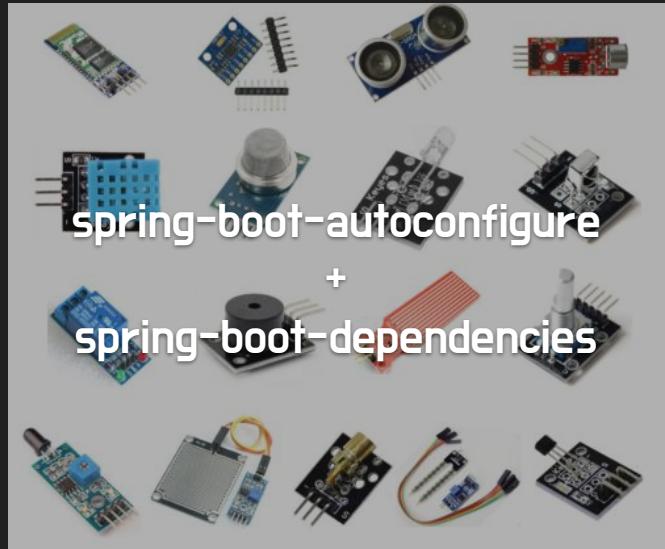
스프링 부트가 시작되는 곳 `@SpringBootApplication`

자바 애플리케이션이 시작되는 진입점 `public static void main()` 메서드 안에 `SpringApplication` 인스턴스를 생성하고 `run` 실행명령을 내린다.
`org.springframework.boot.SpringApplication` 을 살펴보면 스프링 부트가 동작하는 방식을 엿볼 수 있다.

스프링 부트는 애너테이션컨텐스트

(Spring Boot) Starter

- * 스프링 부트에서 제공하는 스타터는 기능과 관련된 의존성 선언뿐
- * 실제 스타터로 추가된 기능에 대한 구성은 `spring-boot-autoconfigure`
- * 스프링 부트 배포버전에 따라 지원 라이브러리 버전도 각기 다르다.



스터디를 통해 간결한 의존성 구성 지원

```
dependencies {  
    compile("javax.annotation:javax.annotation-api:${annotationApiVer}")  
    compile("org.springframework:spring-core")  
    compile("org.springframework:spring-web:${springFrameworkVer}")  
    compile("org.springframework:spring-webmvc:${springFrameworkVer}")  
    compile("org.hibernate.validator:hibernate-validator:${hibernateValidatorVer}")  
  
    //jackson  
    compile("com.fasterxml.jackson.core:jackson-databind:${jacksonVer}")  
    compile("com.fasterxml.jackson.datatype:jackson-datatype-jdk8:${jacksonVer}")  
    compile("com.fasterxml.jackson.datatype:jackson-datatype-jsr310:${jacksonVer}")  
    compile("com.fasterxml.jackson.module:jackson-module-parameter-names:${jacksonVer}")  
  
    compile("org.yaml:snakeyaml")  
  
    //logging  
    compile("ch.qos.logback:logback-classic:${logbackVer}")  
    compile("org.apache.logging.log4j:log4j-to-slf4j:${log4jVer}")  
    compile("org.slf4j:jul-to-slf4j:${slf4jVer}")  
}
```

스타터를 통해 간결한 의존성 구성 지원

```
dependencies {  
    compile("org.springframework.boot:spring-boot-starter-web")  
}
```

스타터를 통해 간결한 의존성 구성 지원

개발자가 신경써야 하는 것

- 스프링 부트 버전!
 - 스프링 부트 스타터는 스프링 부트 버전을 따른다.
- 사용하려는 라이브러리의 스프링 부트 스타터 지원여부!
- 지원하지 않는 경우 사용하려는 라이브러리 등록방법!
 - 그레이들에서 의존성 추가
 - 빈(Bean) 등록 방법 및 속성 정의방법
- 추가한 스타터가 동작하는데 사용하는 속성과 속성값

스프링 부트를 이용하면 스프링 부트의 버전에 따라 변경된다.

자동구성(Auto-Configuration)

자동구성에 대해 궁금하다면

<https://www.slideshare.net/sbcoba/2016-deep-dive-into-spring-boot-autoconfiguration-61584342> 를 살펴보자.

Auto-Configuration

- <http://bit.ly/2MCnqHV>
- 스프링 부트가 기술유행(trend)에 따라 제공하는 **관례(Convention)적인 구성**
- 봐야할 모듈: [spring-boot-autoconfigure](#)
- 동작선언
 - `@EnableAutoConfiguration`(in `@SpringBootApplication`)
 - `@Configuration`
- 사용 애너테이션
 - `@Configuration`
 - `@ConditionalOn`
 - 있거나 없거나
- `~AutoConfiguration` 접미사

spring-boot-starter-configure
spring-boot-starter-dependencies

외부구성(External Configuration)

외부구성(External Configuration)

적용 우선순위

1. 실행인자
2. SPRING_APPLICATION_JSON
3. 환경변수
4. 기타 등등
5. application.yml or application.properties
6. application-{default|profiles}.yml or application-{default|profiles}.properties

<https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-configuration.html>

보다 상세한 우선순위가 있지만 많이 활용되는 부분은 위에서 언급된 1, 3, 5, 6 조합이다.

Programming in Spring Environment

Programming in Spring Environment

- **@ComponentScan**을 통해 **ApplicationContext** 적재

- **@Repository**
- **@Component**
- **@Service**
- **@Controller & @RestController**
- **@Configuration**
 - **@Bean**
 - **@ConfigurationProperties**

개발자가 기능을 구현하면서
실제로 사용하고 작성하는
애너테이션과 코드

- **DI, IoC, @Autowired**
- **@Value Vs @ConfigurationProperties**
- **AOP**
- ... 등등등

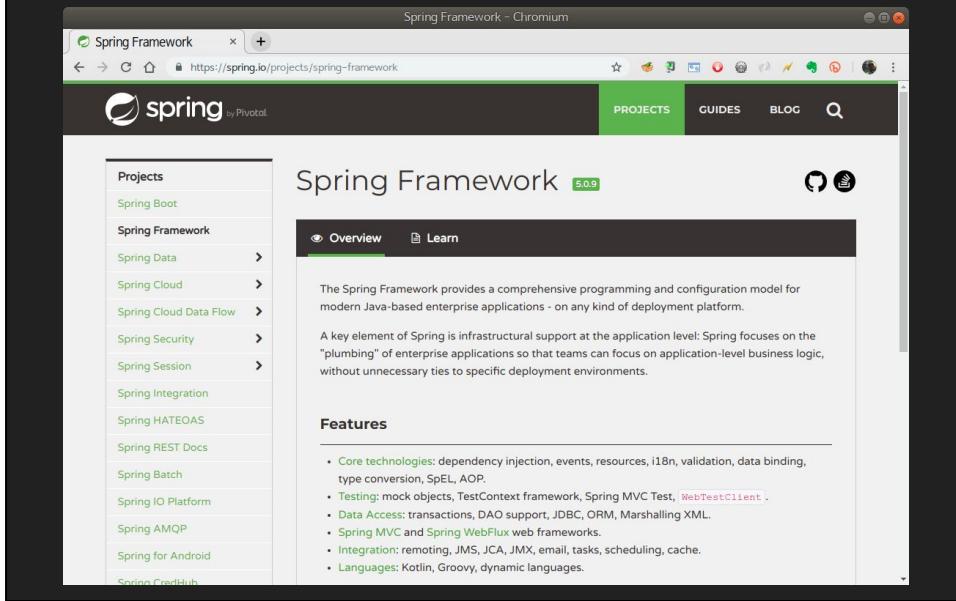
업무(비즈니스 로직) 구현!

업무에서 해결해야 할 문제는 정말 다양하다. 각 문제를 해결하는데 적절한 방안을 찾아내어 구현하는 것이 개발자의 몫이다.

스프링 프레임워크 및 부트는 개발자가 업무 구현에 집중할 수 있도록 한다.

스프링 부트는 스프링 프레임워크 기반 작동

Spring framework: <https://github.com/spring-projects/>



The screenshot shows a web browser window titled "Spring Framework - Chromium". The URL in the address bar is <https://spring.io/projects/spring-framework>. The page itself is the "Spring Framework" project page on the Spring website. On the left, there's a sidebar with a "Projects" heading and a list of various Spring projects: Spring Boot, Spring Framework, Spring Data, Spring Cloud, Spring Cloud Data Flow, Spring Security, Spring Session, Spring Integration, Spring HATEOAS, Spring REST Docs, Spring Batch, Spring IO Platform, Spring AMQP, Spring for Android, and Spring CreditHub. The main content area has a title "Spring Framework 5.0.9" and two tabs: "Overview" (which is active) and "Learn". Below these tabs, there's a brief introduction: "The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform. A key element of Spring is infrastructural support at the application level: Spring focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments." Under the "Features" section, there's a bulleted list: • Core technologies: dependency injection, events, resources, i18n, validation, data binding, type conversion, SpEL, AOP. • Testing: mock objects, TestContext framework, Spring MVC Test, WebTestClient. • Data Access: transactions, DAO support, JDBC, ORM, Marshalling XML. • Spring MVC and Spring WebFlux web frameworks. • Integration: remoting, JMS, JCA, JMX, email, tasks, scheduling, cache. • Languages: Kotlin, Groovy, dynamic languages.

Spring 1.0 RELEASE

<https://spring.io/blog/2004/03/24/spring-framework-1-0-final-released>



Spring Framework Runtime

Data Access/Integration

JDBC ORM

OXM JMS

Transactions

Web

WebSocket

Servlet

Web

Portlet

AOP

Aspects

Instrumentation

Messaging

Core Container

Beans

Core

Context

SpEL

Test

스프링 프레임워크를 공부하고 싶다면...



**스프링 부트는
스프링 프레임워크
애플리케이션 개발 플랫폼**

배포(Deploy)

다양한 환경에 다양한 방법으로!

스프링 부트 소개 끝!

참고문헌

- **Spring Boot 1.0. GA**
 - <https://spring.io/blog/2014/04/01/spring-boot-1-0-ga-released>
- **Spring Boot 2.0.0.RELEASE**
 - <https://twitter.com/springboot/status/1023951216810831877>
- **Boot Spring Boot!**
 - <http://www.yes24.com/24/goods/62112463?scode=029>
- **Spring Boot - spring.io**
 - <https://spring.io/projects/spring-boot>
- **Spring Boot Reference Documentation**
 - <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>
- **Spring initializer**
 - <https://start.spring.io/>
- **수요가 높은 프로그래밍 언어 5가지**
 - <https://www.technotivation.com/2018/08/new-demanding-programming-languages.html>

참고문헌

- OpenJDK
 - <http://openjdk.java.net/>
- Groovy
 - <http://groovy-lang.org/>
- Kotlin
 - <https://kotlinlang.org/>
- Gradle
 - <https://gradle.org/>
- Maven
 - <http://maven.apache.org/>
- Apache Tomcat
 - <http://tomcat.apache.org/>
- Spring Boot API Documentation
 - <https://docs.spring.io/spring-boot/docs/current/api/>

참고문헌

- Spring API Documentation
 - <https://docs.spring.io/spring-framework/docs/current/javadoc-api/>
- Deep dive into Spring Boot AutoConfiguration
 - <https://www.slideshare.net/sbcoba/2016-deep-dive-into-spring-boot-autocfg>
- Spring Framework 1.0 RELEASE
 - <https://spring.io/blog/2004/03/24/spring-framework-1-0-final-release>
- Sagan: the spring.io site and Reference application
 - <https://github.com/spring-io/sagan>

스프링 부트 프로젝트 만들기

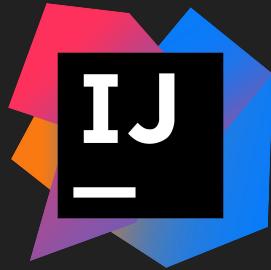
Getting Started - Spring Boot

IDE(Integration Development Environment)



STS(Spring Tool Suite, Eclipse)

VS



IDEA(IntelliJ)

자바에서 사용하는 통합개발환경 (IDE, Integration Development Envrionment)를 제공하는 대표적인 개발도구에는 STS와 IntelliJ가 있다.

STS는 이클립스에서 스프링 및 스프링 부트 개발을 지원하는 기능을 추가한 확장판이다.

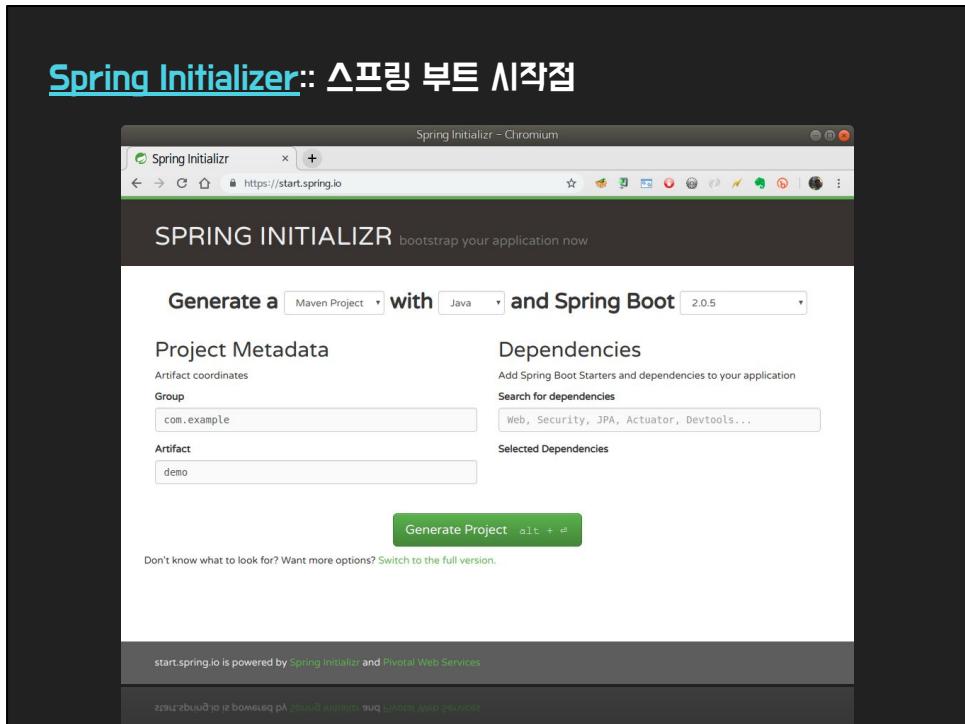
IntelliJ는 젯브레인에서 유료로 판매하고 있는 개발도구로 다양한 플로그인을 제공한다.

프로젝트 구성

Spring Initializer:: <https://start.spring.io>

스프링 부트 프로젝트 생성소

Spring Initializer:: 스프링 부트 시작점



앞서 설명했던 STS나 인텔리제이에서 스프링 부트 프로젝트를 만들 때 이 사이트에서 프로젝트 생성요청을 하여 생성된 압축파일을 내려받아 압축해제하고 불러오는 과정으로 프로젝트를 생성합니다. 그렇다면! 인터넷이 되지 않는 환경에서 스프링 부트 프로젝트를 생성할 수 있을까요? 당연히 안되겠죠? 이 점을 고려했을 때 스프링 부트 프로젝트를 생성하기 위해서는 온라인 상태여야 겠다는 생각을 하시는 게 자연스러운 겁니다.

Spring initializer:: Make spring-boot.zip

```
https://start.spring.io/starter.zip\
?name=spring-boot\
&groupId=io.honeymon.tacademy\
&artifactId=spring-boot\
&version=0.0.1-SNAPSHOT\
&description=Demo+project+for+Spring+Boot\
&packageName=io.honeymon.tacademy.springboot\
&type=gradle-project\
&packaging=jar\
&javaVersion=1.8\
&language=java\
&bootVersion=2.0.5.RELEASE\
&dependencies=lombok&dependencies=h2&dependencies=data-jpa&dependencies=web
```

이 URL을 브라우저 주소창에 입력하면 name과 같은 **spring-boot.zip** 파일을 내려받을 수 있습니다.

룸복(Lombok)

- 자바 프로젝트 필수 라이브러리
- 클래스에서 필수적으로 작성해야 하는 접근자/설정자
`(getter/setter), toString, equalsAndHashCode, 생성자
(Constructor)` 등을 작성하지 않아도 된다.
- 코드를 간결하게 사용할 수 있다.
- 배포버전을 확인하고 결합이 있는지 확인해야 한다.
- 사용시 주의사항

룸복(Lombok) 자바 프로젝트에서 필수적으로 필요한 라이브러리

H2Database

- 인-메모리(in-memory), 파일(file), TCP 지원 데이터베이스
- JDBC url 설정으로 데이터베이스 작동방식 지정가능
- 스프링 부트 자동구성으로 /h2-console h2 webConsole 제공
- 로컬 개발환경에서 별도의 DB설치없이 빠른 프로토타이핑 지원
- 필요에 따라 운영가능한 수준의 데이터베이스 활용가능

H2Database 로컬에서 손쉽게 사용가능한 Database

스프링 부트 프로젝트 기본구조

```
.  
|   └── build.gradle  
|   └── gradle  
|       └── wrapper  
|           ├── gradle-wrapper.jar  
|           └── gradle-wrapper.properties  
|   └── gradlew  
|   └── gradlew.bat  
|   └── settings.gradle  
// 다음
```

스프링 부트 프로젝트에는 빌드도구 래퍼(Wrapper)가 기본포함된다. 최근 CI/CD 도구에서 래퍼를 기본구성하는 것을 선호한다.
시스템에서 수동으로 설치할 필요가 없기 때문이다.

스프링 부트 프로젝트 기본구조

```
└── src
    ├── main
    │   ├── java
    │   │   └── io.honeymon.tacademy.springboot
    │   │       └── Application.java
    │   └── resources
    │       ├── application.yml
    │       ├── static
    │       └── templates
    └── test
        └── java
            └── io.honeymon.tacademy.springboot
                └── ApplicationTests.java
```

settings.gradle

```
rootProject.name = 'spring-boot'
```

- 멀티 프로젝트 구성시 사용
- 프로젝트 이름
- 하위 프로젝트 정의
- 하위 프로젝트 설명(제 개인적으로)

build.gradle

```
buildscript {
    ext {
        springBootVersion = '2.0.5.RELEASE'
    }
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath("org.springframework.boot:spring-boot-gradle-plugin:${springBootVersion}")
    }
}

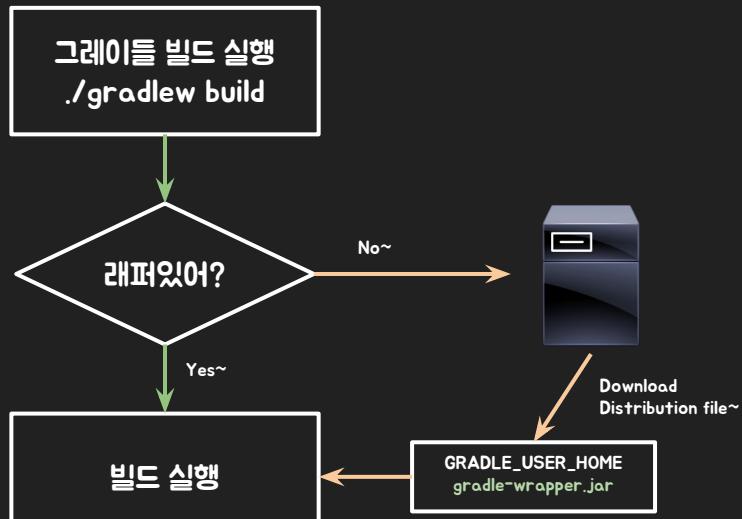
apply plugin: 'java'
apply plugin: 'eclipse'
apply plugin: 'org.springframework.boot'
apply plugin: 'io.spring.dependency-management'

group = 'io.honeymon.tacademy'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = 1.8

repositories {
    mavenCentral()
}

dependencies {
    // 생략
}
```

그레이디클 래퍼(Wrapper)



빌드도구 래퍼의 동작방식

GRADLE_USER_HOME에 래퍼 파일이 있는지 확인하고 없으면 배포서버에서 내려받아 압축해제한다.

그레이드 래퍼

```
.  
|   -- build.gradle      GRADLE_USER_HOME  
+-- gradle  
    +-- wrapper  
        +-- gradle-wrapper.jar  
        +-- gradle-wrapper.properties  
    +-- gradlew  
    +-- gradlew.bat  
    +-- settings.gradle  
// 다음
```

```
$ ./gradlew(.bat) clean build
```

스프링 부트 배포과정을 확인할 수 있다.

Spring Boot Repackaging

Spring Boot Repackaging

spring-boot.0.0.1.SNAPSHOT.jar

재압축
Repacking

spring-boot.0.0.1.SNAPSHOT.jar

BOOT-INF



```
$ unzip spring-boot-0.0.1-SNAPSHOT.jar
```

SpringApplication

SpringApplication:: 스프링 부트 시작점

```
/**  
 * 스프링 부트 애플리케이션이 시작되는 곳!  
 * {@Link SpringApplication}을 살펴보세요. 스프링 부트의  
 * 마법이 시작되는 곳입니다.  
 *  
 * @author honeymon  
 */  
@SpringBootApplication  
public class SpringBootApplication {  
    public static void main(String[] args) {  
        SpringApplication.run(SpringBootApplication.class, args);  
    }  
}
```

강조된 코드만 봤을 때는 순수한 자바 애플리케이션 코드

SpringApplication:: 스프링 부트 시작점

```
/**  
 * 스프링 부트 애플리케이션이 시작되는 곳!  
 * {@link SpringApplication}을 살펴보세요. 스프링 부트의  
 * 마법이 시작되는 곳입니다.  
 *  
 * @author honeymon  
 */  
@SpringBootApplication  
public class SpringBootApplication {  
    public static void main(String[] args) {  
        SpringApplication.run(SpringBootApplication.class, args);  
    }  
}
```

스프링 부트가 시작되는 곳 `@SpringBootApplication`

자바 애플리케이션이 시작되는 진입점 `public static void main()` 메서드 안에 `SpringApplication` 인스턴스를 생성하고 `run` 실행명령을 내린다.
`org.springframework.boot.SpringApplication` 을 살펴보면 스프링 부트가 동작하는 방식을 엿볼 수 있다.

스프링 부트는 애너테이션컨텐스트



스프링 부트 스터터

spring-boot-starter

- spring-boot-dependencies
- spring-boot-autoconfigure
- spring-boot-parent
- spring-boot-starters
- `spring-boot-starter-*/pom.xml`

스프링 부트 스타터는 위 4가지 모듈이 조합되어 동작한다.

- **spring-boot-dependencies**: 스프링 부트 배포버전에서 지원하는 각 라이브러리 모듈의 버전을 정의하고 있다.
- **spring-boot-autoconfigure**: 스프링 부트 스타터의 라이브러리 모듈에 대한 자동구성을 포함하고 있다.
- **spring-boot-parent**: 위에 언급된 두 모듈을 포함하고 있는 스프링 부트 최상위 모듈로써 스프링 부트 작동 핵심모듈이라고 볼 수 있다.

spring-boot-starter - ex) spring-boot-starter-data-jpa

```
dependencies {  
    compile("org.springframework.boot:spring-boot-starter-web")  
}  
  
//spring-boot-starter-data-jpa 추가  
  
dependencies {  
    compile("org.springframework.boot:spring-boot-starter-web")  
    compile("org.springframework.boot:spring-boot-starter-data-jpa")  
}  
  
// hibernate 의존성 추가, EntityManager 구성 활성화  
  
// JpaRepositoriesAutoConfiguration, HibernateJpaAutoConfiguration 함께 활성화
```

자동구성(Auto-Configuration)

```
@Configuration
@ConditionalOnWebApplication(type = Type.SERVLET)
@ConditionalOnClass({ Servlet.class, DispatcherServlet.class,
WebMvcConfigurer.class })
@ConditionalOnMissingBean(WebMvcConfigurationSupport.class)
@AutoConfigureOrder(Ordered.HIGHEST_PRECEDENCE + 10)
@AutoConfigureAfter({ DispatcherServletAutoConfiguration.class,
ValidationAutoConfiguration.class })
public class WebMvcAutoConfiguration {

    public static final String DEFAULT_PREFIX = "";

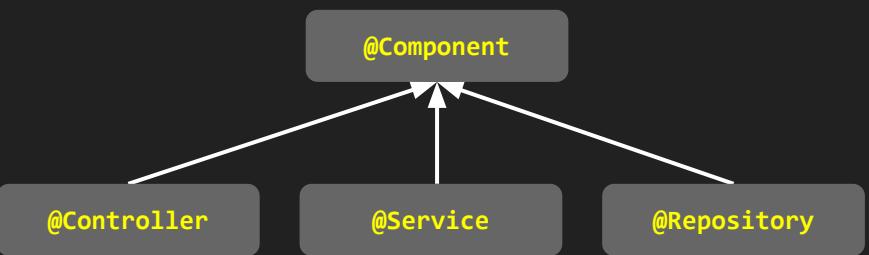
    public static final String DEFAULT_SUFFIX = "";

    private static final String[] SERVLET_LOCATIONS = { "/" };

    @Bean
    @ConditionalOnMissingBean(HiddenHttpMethodFilter.class)
    public OrderedHiddenHttpMethodFilter hiddenHttpMethodFilter() {
        return new OrderedHiddenHttpMethodFilter();
    }
}
```

Spring Framework

@Component 상속관계



@Bean vs @Component

@Bean: 남이 작성한 것

@Component: 내가 작성한 것

AnnotationConfigApplicationContext

@Bean vs 스프링 빈(Bean) 객체

스프링 빈(Bean) 객체는 '스프링 IoC 컨테이너'에서 생명주기를 관리하는 객체를 뜻한다.

의존성 주입(Dependency Injection, DI) - 없이 사용하는 경우

```
public class ObjectMapperTest {

    @Test
    public void test() throws JsonProcessingException {
        ObjectMapper objectMapper = new ObjectMapper();

        Book book =
            new Book("test-book", "test-isbn13", "test-isbn10");

        String strBook = objectMapper.writeValueAsString(book);
        // 검증 생략
    }
}
```

의존성 주입(Dependency Injection, DI)

```
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment=WebEnvironment.RANDOM_PORT)
public class ObjectMapperTest2 {
    @Autowired
    ObjectMapper objectMapper;

    @Test
    public void test() throws JsonProcessingException {
        Book book =
            new Book("test-book", "test-isbn13", "test-isbn10");

        String strBook = objectMapper.writeValueAsString(book);
        // 검증 생략
    }
}
```

의존성 주입 방법 - Note

- 생성자 주입(**권장**)
- 설정자 주입
- 필드 @**Autowired** 선언

의존성 주입 방법 - 생성자 주입(권장)

```
@Service
public class BookServiceImpl implements BookService {
    private final BookRepository repository;

    public BookServiceImpl(BookRepository repository) {
        this.repository = repository;
    }
    // 코드 생략
}
```

<http://www.mimul.com/pebble/default/2018/03/30/1522386129211.html>

왜 Constructor Injection을 권장하나?

- 1. 단일 책임의 원칙
생성자의 인자가 많을 경우 코드량도 많아지고, 의존관계도 많아져 단일 책임의 원칙에 위배된다. 그래서 Constructor Injection을 사용함으로써 의존관계, 복잡성을 쉽게 알 수 있어 리팩토링의 단초를 제공하게 된다.
- 2. 테스트 용이성
DI 컨테이너에서 관리되는 클래스는 특정 DI 컨테이너에 의존하지 않고 POJO여야 한다. DI 컨테이너를 사용하지 않고도 인스턴스화 할 수 있고, 단위 테스트도 가능하며, 다른 DI 프레임워크로 전환할 수도 있게 된다.
- 3. Immutability
Constructor Injection에서는 필드는 final로 선언할 수 있다. 불변 객체가 가능한데 비해 Field Injection은 final은 선언할 수 없기 때문에 객체가 변경 가능한 상태가 된다.
- 4. 순환 의존성
Constructor Injection에서는 멤버 객체가 순환 의존성을 가질 경우 BeanCurrentlyInCreationException이 발생해서 순환 의존성을 알 수 있게 된다.
- 5. 의존성 명시
의존 객체 중 필수는 Constructor Injection을 옵션인 경우는 Setter Injection을 활용할 수 있다.

의존성 주입 방법 - 설정자(Setter) 주입

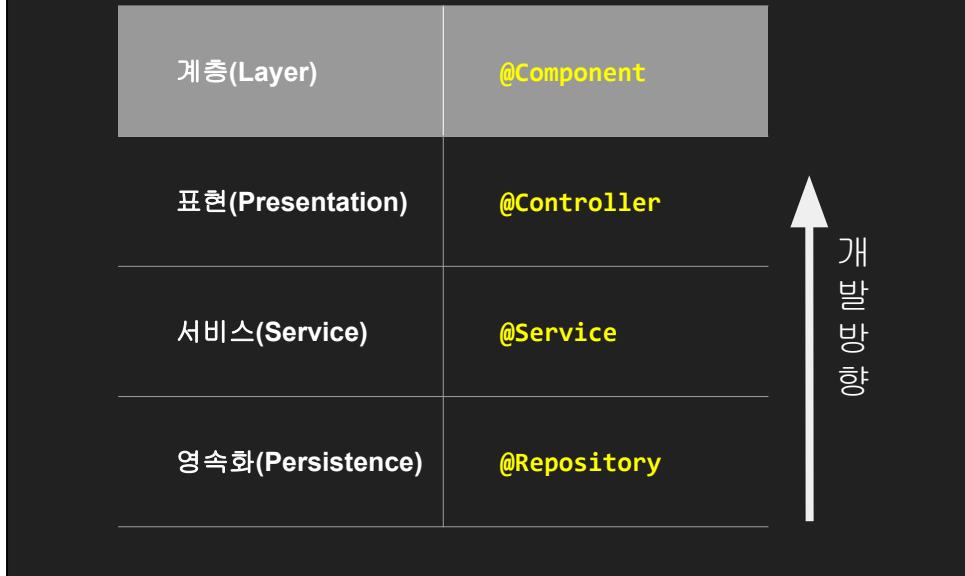
```
@Service
public class BookServiceImpl implements BookService {
    private BookRepository repository;

    @Autowired
    public void setRepository(BookRepository repository) {
        this.repository = repository;
    }
    // 생략
}
```

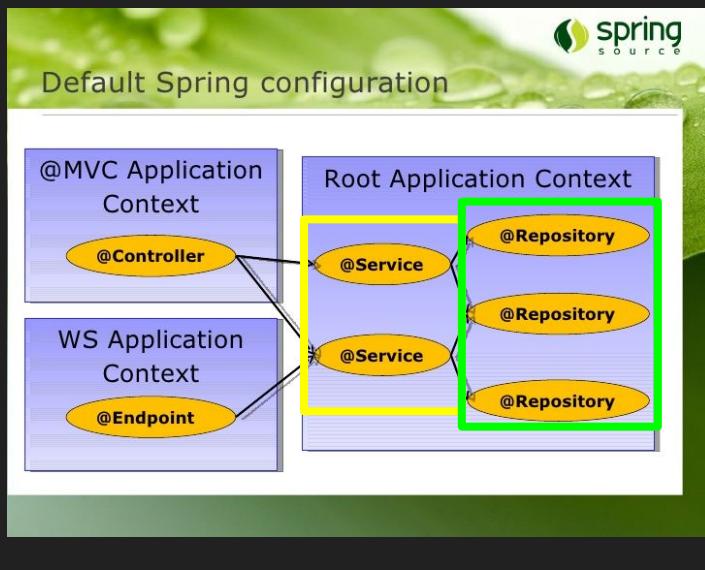
의존성 주입 방법 - 필드 @Autowired 선언

```
@Service  
public class BookServiceImpl implements BookService {  
    @Autowired  
    private BookRepository repository;  
    // 코드 생략  
}
```

애플리케이션 계층(layerd Spring Application)



기본적인 스프링 애플리케이션 계층 구성



@Repository

Spring Data JPA

- **ORM(Object-Relational Mapping)**
 - 대부분의 개발언어 플랫폼마다 제공
 - 객체(Object)로 관계형 데이터베이스(RDBS)를 관리
- **JPA(Java Persistence API)**
 - Java 객체 정보를 영속화하는 중간 과정을 처리한다.
 - 엔티티 객체를 저장하고 변경하고 삭제하면 그에 대응하는 쿼리(Query)를 생성하고 실행한다.
- **참고**
 - https://www.slideshare.net/zipkyh/ksug2015-jpa1-jpa-5_1213397
 - <https://www.slideshare.net/zipkyh/spring-datajpa>

객체 모델링 저장

```
@Entity  
class Book {  
    Long id;  
    String isbn13;  
    String isbn10;  
}  
  
insert into book(id, isbn13, isbn10) values(...);
```

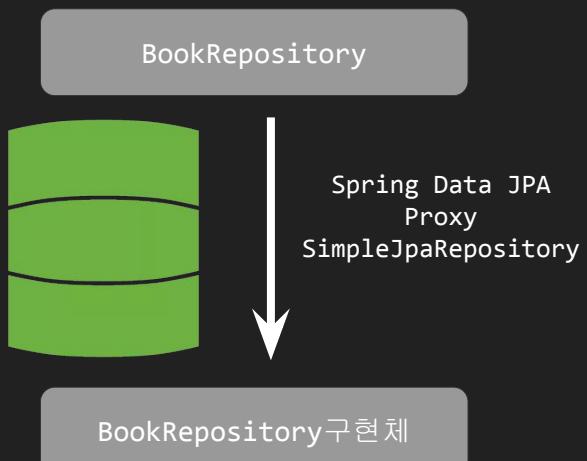
Spring Data JPA Repository interface

```
/**  
 *  
 * @author honeymon  
 */  
public interface BookRepository extends JpaRepository<Book, Long> {  
  
    List<Book> findByNameLike(String name);  
}
```

JpaRepository

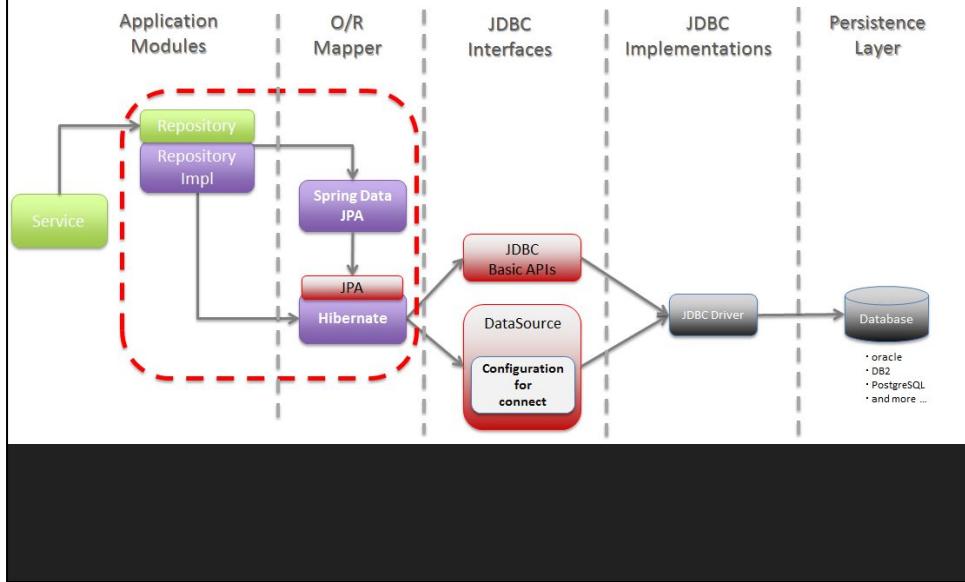
```
Optional<T> findById(ID id);
List<T> findAll();
List<T> findAll(Sort sort);
List<T> findAllById(Iterable<ID> ids);
<S extends T> List<S> saveAll(Iterable<S> entities);
void flush();
<S extends T> S saveAndFlush(S entity);
void deleteInBatch(Iterable<T> entities);
void deleteAllInBatch();
T getOne(ID id);
```

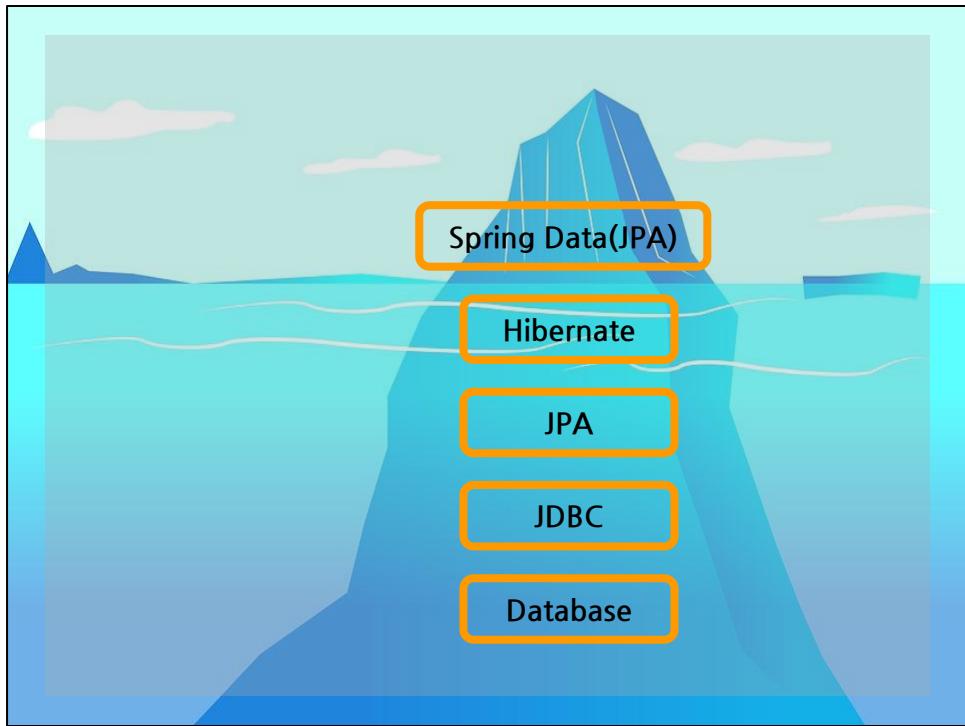
Spring Data JPA 작동원리



프록시에 대하여: <http://havivj.tistory.com/28>

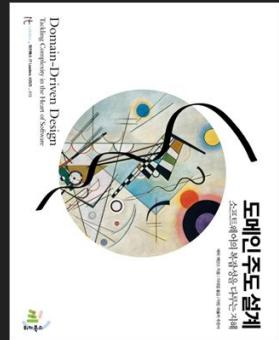
Spring Data JPA 작동기작





업무(=비즈니스 로직) 구현에만 집중해라!

- 영속화 계층(@Repository)에서는 엔티티 관리만
- 비즈니스 로직 구현은 도메인 영역에서
- 서로 다른 도메인 사이에 연계는 서비스 계층(@Service)에서
- 외부요청에 대한 처리는 컨트롤러 계층(@Controller)에서



@Service

@Service

```
@Service // <1>
@Transactional // <2>
public class BookServiceImpl implements BookService {
    private final BookRepository repository;

    public BookServiceImpl(BookRepository repository) { // <3>
        this.repository = repository;
    }

    @Override
    public Optional<Book> findById(Long id) {
        return repository.findById(id);
    }

    @Override
    public List<Book> findAll(OffsetPageRequest request) {
        return repository.findAll(request.getPageRequest()).getContent();
    }
}
```

@Service

- 트랜잭션(@Transactional) 관리 영역
- 서로 다른 도메인 연계(DI, @Autowired) 작업 영역
- @Controller 와 @Repository 사이의 중계

@Controller

@Controller - Note

```
@RestController // <1>
@RequestMapping(value = "/books") // <2>
public class BookController {

    private final BookService service;

    public BookController(BookService service) { // <3>
        this.service = service;
    }

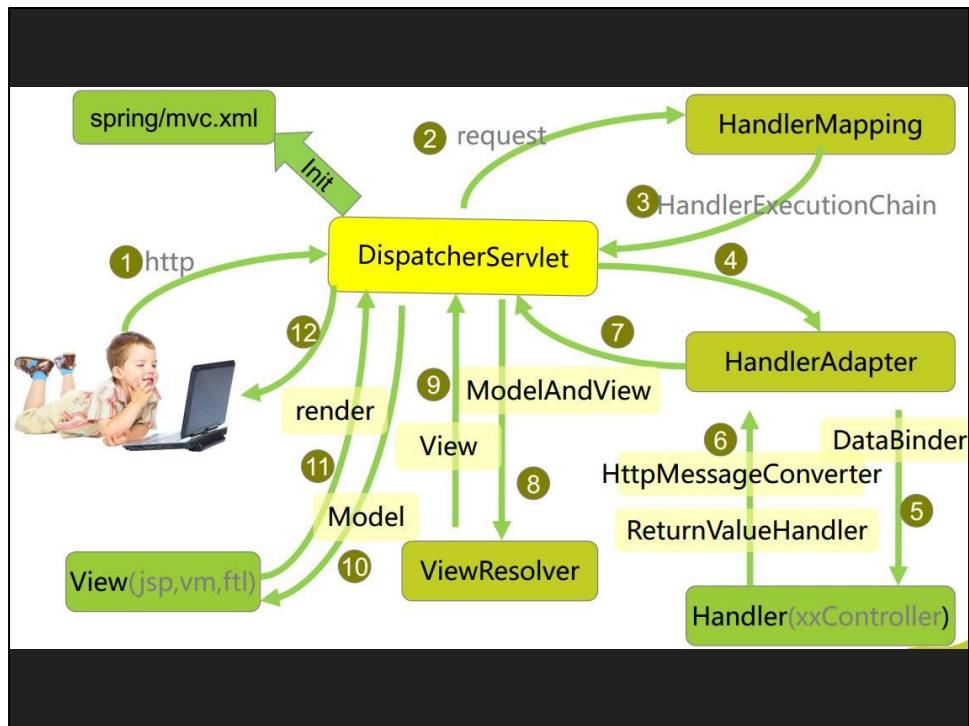
    /**
     * API 지원에서도 유형 반환
     *
     * @return HttpMethod types
     */
    @RequestMapping(method = RequestMethod.OPTIONS) // <4>
    public ResponseEntity<List<RequestMethod>> options() { // <5>
        return ResponseEntity.ok()
            .allow(HttpMethod.OPTIONS, HttpMethod.HEAD, HttpMethod.GET).build();
    }
    // 생략
}
```

@Controller

- DispatcherServlet에 등록된 @RequestMapping 호출됨
- 템플릿 엔진이 랜더링할 뷰 페이지를 지정
- 호출된 API에서 처리한 응답을 반환

```
@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Component
public @interface Controller {

    /**
     * The value may indicate a suggestion for a Logical component name,
     * to be turned into a Spring bean in case of an autodetected component.
     * @return the suggested component name, if any (or empty String otherwise)
     */
    @AliasFor(annotation = Component.class)
    String value() default "";
}
```



@Controller 예외처리(ExceptionHandler)

- @ControllerAdvice 를 이용한 처리

```
@ControllerAdvice(annotations = {RestController.class})
@ResponseBody
public class GlobalRestControllerAdvice {

    @ExceptionHandler(BookNotFoundException.class)
    public ApiResponse<Void> handleException(Exception e) {
        log.error("Occurred Exception: {}", e);
        return ApiResponse.error(e.getMessage());
    }
}
```

REST API

- <https://www.restapitutorial.com/index.html>
- 시스템의 자원(Resource)에 대한 접근 및 제어를 제공하는 API
- 자원(ex: Book)에 대한 접근 및 제어
 - GET /books
 - GET /books/{bookId}
 - POST /books
 - PUT /books/{bookId}
 - DELETE /books/{bookId}
- 스프링에서는 요청에 따라 등록되어 있는 적절한 [HttpMessageConverter](#)를 통해서 응답데이터를 반환한다.

Spring REST DOCs

- <https://spring.io/projects/spring-restdocs>
- Spring MVC test와 Asciidoctor 조합을 통해서
RESTful 서비스에 대한 문서화 지원
- 작성된 테스트코드에 대한 아스키독 조각 생성
- 개발자가 아스키독 조각을 모아 아스키독 문서를 작성한다.
- 코드에 침투적이지 않은 노력에 따라 고품질의 코드가 될 수
있음
- <https://youtu.be/A3WDAVQP32k>

REST Client - cURL

- <https://curl.haxx.se/>
- 커맨드라인에서 실행할 수 있는 REST Client

```
curl 'http://localhost:8080/books/1' -i -X GET \
-H 'Content-Type: application/json; charset=UTF-8'
```

REST Client - httpie

- <https://httpie.org/>
- 커맨드라인에서 실행할 수 있는 REST Client

```
http GET 'http://localhost:8080/books/1' \
'Content-Type:application/json; charset=UTF-8'
```

애플리케이션 동작작동 정의

프로파일(=실행환경정의. @Profile)

여기서 말하는 프로파일(Profile)이란 스프링에서 제공하는 프로파일 애너테이션을 제공합니다. 프로파일은 컴포넌트를 활성화할 실행환경을 정의한다. 스프링 부트는 하나의 패키징된 배포폰을 가지고 다양한 실행환경에서 이용할 수 있도록 외부구성을 통해 애플리케이션 속성을 정의할 수 있으며, 이런 실행환경을 명시적으로 선언하는 애너테이션 @Profile이 있다.

- `@Profile("dev"), @Profile("beta"), @Profile("prod")`

스프링 부트 테스트 실행시 실행환경을 명시할 수 있는 `@ActiveProfile("{profile}")` 이 있다.

```
@Profile -TYPE
```

```
@Profile("local")
@Configuration
public class LocalApiConfig {

    @Autowired
    private RestTemplateBuilder restTemplateBuilder;

    @Bean
    public RestTemplate restTemplate() {
        return restTemplateBuilder.build();
    }
}
```

@Profile - METHOD

```
@Configuration
public class LocalApiConfig {

    @Autowired
    private RestTemplateBuilder restTemplateBuilder;

    @Profile("local")
    @Bean
    public RestTemplate restTemplate() {
        return restTemplateBuilder.build();
    }

    @Profile("!local")
    @Bean
    public RestTemplate restTemplate() {
        return restTemplateBuilder.rootUri("http://test.honeymon.io")
            .build();
    }
}
```

```
@Profile - application-{profile}.yml
```

- `application-datasource.yml = @Profile("datasource")`
- `application-api.yml = @Profile("api")`
- `application.yml`

애플리케이션 속성(properties) 정의

- * 코드로 명시적으로 작성하는 방법
- * 애플리케이션 속성파일 정의
- * 애플리케이션 속성을 외부에서 구성하는 방법
 - ** 환경변수로 지정하는 방법
 - ** 실행인자로 지정하는 방법

스프링 부트 애플리케이션 속성정의

1. 테스트 속성정의
2. 실행인자 지정
3. 운영체제 환경변수 지정
4. 속성파일(application.yml or application.properties) 지정
 - a. @EnableConfigurationProperties
 - b. @ConfigurationProperties
5. 프로그래밍적 코드 구현

애플리케이션 속성정의 - 테스트 속성정의

```
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment = DEFINED_PORT, properties = {"server.port=9090"})
public class BookControllerTest {
```

애플리케이션 속성정의 - 실행인자 속성정의

```
$ java -jar spring-boot.0.0.1.SNAPSHOT.jar \
--server.port=9000
```

애플리케이션 속성정의 - 환경변수 정의

```
$ SERVER_PORT=9000 \
SPRING_PROFILES_ACTIVE=local \
java -jar api-0.0.1-SNAPSHOT.jar
```

계층화된 테스트

계층화된 테스트

- @SpringBootTest
- @WebMvcTest
- @WebFluxTest
- @DataJpaTest
- @JdbcTest
- ...

영어공부를 하고 싶어진다....

다음은 웹 서비스 만들기

참고문헌

- **STS(Spring Tool Suite)**
 - <https://spring.io/tools/sts>
- **IntelliJ**
 - <https://www.jetbrains.com/idea/>
- **Lombok**
 - <https://projectlombok.org/>
- **H2Database**
 - <http://www.h2database.com/html/main.html>
- **Gradle Wrapper**
 - https://docs.gradle.org/current/userguide/gradle_wrapper.html
- **SpringApplication API**
 - <https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/SpringApplication.html>
- **spring-boot-starter**
 - <https://github.com/spring-projects/spring-boot/tree/master/spring-boot-project/spring-boot-starters>
 -

참고문헌

- DI(Dependency Injection)
 - <https://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-spring-beans-and-dependency-injection.html>
- Developing Modular Java Applications
 - <https://www.slideshare.net/julien.dubois/developing-modular-java-applications>
- JPA(Java Persistence API)
 - <https://github.com/eclipse-ee4j/jpa-api>
- JPA 프로그래밍
 - <https://www.slideshare.net/zipkyh/ksug2015-jpa1-jpa-51213397>
 - <https://www.slideshare.net/zipkyh/spring-data-jpa>
 - <http://bit.ly/2CJWuq9> – 백기선님의 스프링 DATA JPA(인프린 강좌)
- Proxy mechanism
 - <https://docs.spring.io/spring/docs/3.0.0.M3/reference/html/ch08s06.html>

참고문헌

- Spring Data JPA Reference Document
 - <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/>
 - 번역: http://arahansa.github.io/docs_spring/jpa.html
- Spring MVC 전체흐름 알아보기
 - <https://www.slideshare.net/hanmomhanda/spring-mvc-fullflow>
- DispatcherServlet
 - <https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/servlet/DispatcherServlet.html>
- Learn REST: RESTful tutorial
 - <https://www.restapitutorial.com/index.html>
- Spring REST DOCs
 - <https://spring.io/projects/spring-restdocs>
- cURL
 - <https://curl.haxx.se/>
- httpie
 - <https://httpie.org/>

참고문헌

- **@Profile**
 - <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/context/annotation/Profile.html>
- **Spring Boot - common application properties**
 - <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#common-application-properties>
- **Spring Boot - Testing**
 - <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#boot-features-testing>

스프링 부트 웹 서비스 개발

BookStore24 서비스 만들기

[https://github.com/ihoneymon
/tacademy-spring-boot](https://github.com/ihoneymon/tacademy-spring-boot)

BookStore24

- 서비스 기획
- 기능분석 및 설계
- 구현
- 빌드
- 배포
- 운영

서비스 시발(始發)

네, 알겠습니다.
기획안 올리도록 하겠습니다.

온라인 도서 판매사업을 하고
싶습니다(3개월 뒤부터...).



대표



기획자



개발자

기획...

도서관리 절차

회원가입절차

1. 회원가입신청
2. 가입신청 메일인증
3. 개인정보 이용 동의
4. 개인정보 등록
5. 본인인증(휴대폰)
6. 회원가입완료
7. 회원가입완료 안내메일 발송

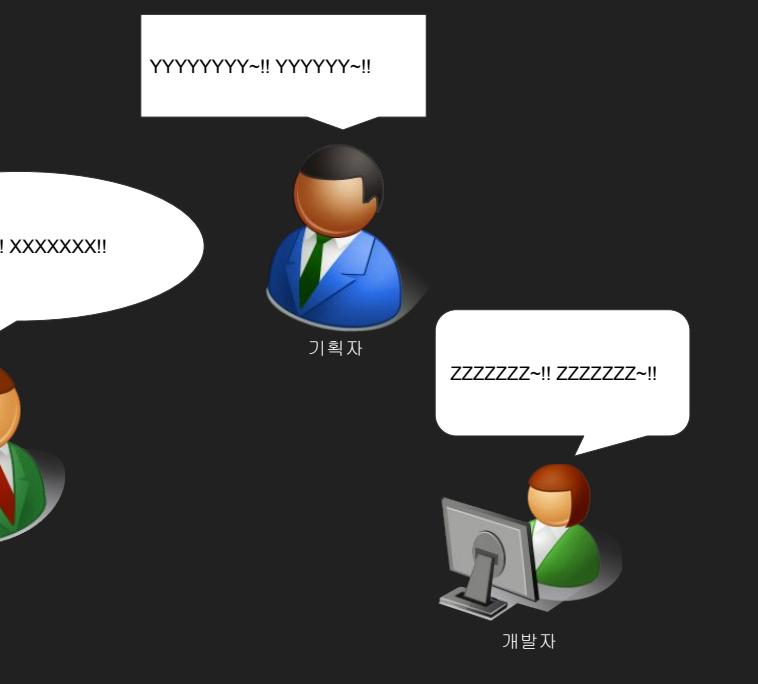
1. 도서정보를 등록한다.
2. 도서는 출간전 예약판매를 한다.
3. 예판(예약판매) 도서는 출간일 전까지 배송할 수 없다.
4. 도서의 출간일에 배송가능여부를 결정한다.
5. 오전 11시 전까지 결제완료된 도서는 당일배송된다 (.).
6. 오후 5시 전까지 결제완료된 도서에 대한 배송을 진행한다.

배송관리 절차

1. 도서에 대한 결제완료된 순간 알림톡 발송한다.
2. 오후 0 시 이후에는...
3. 오전 00시 이전에는...



회의



회의...

YYYYYYYYYY~!! YYYYYYYY~!!

XXXXXXX~!! XXXXXX!!

기획자

ZZZZZZ~!! ZZZZZZ~!!

대표

개발자

서비스 기획

온라인 도서 판매서비스를 만들 겁니다.

- 이용 플랫폼은 모바일웹, 아이폰, 안드로이드 지원해야 합니다.
- 판매상품은 책이다.
- 책은 유형별로 관리되어야 한다.
- 책은 작가가 출판사를 통해 출간한다.
- 책은 서비스 관리자가 등록한다.
- 고객이 책을 주문한다.
- ...



기획자

네,
알겠습니다
.....
.....



개발자

PPT 장인



기획자의 목업(Mock-Up)도구로 각광받는 파워포인트 ... @_@);;
개발자가 파워포인트를 사용하는 경우는 거의 없는데 기획자의 문서를 보기 위해서 설치하기도 한다.
목업도구 (<http://ylab.kr/94>)

기능분석 및 설계

- 지원플랫폼: 웹, [아이폰](#), [안드로이드](#)
- 개발언어: [Java](#)
- 개발도구: [STS\(IntelliJ 쓰고 싶...\)](#)
- 빌드도구: [Gradle or Maven](#)
- 개발플랫폼: [스프링 부트](#)
- 운영플랫폼: [AWS](#)(예만 하고 싶지만 [개인정보 보호법](#))
- 필요한 기능
 - 플랫폼에서 호출할 REST API
 - 관리시스템(백오피스)
 - 고객관리
 - 도서유형관리
 - 도서관리
 - 매일발송 시스템
 - (나중에...) 알림푸시 시스템
 - (나중에...) 포인트시스템
 - (나중에...) 정산시스템
 - (나중에...) 통계 및 매출차트 지원
 - (나중에...) ...



기능분석 및 설계 초기에 해야 할 일 중 하나는 프로젝트를 구성하는 것이다.
기능분석을 통해 필요한 기술적인 요구사항을 어떻게 충족할지 고민하는 과정이
필요한데

스프링 부트를 이용하면 그런 과정을 많이 간소화할 수 있다.

기술트렌드에 따라 많이 사용되는 기술을 스타터로 제공하고 있기 때문에 자신의
기술셋에 따라 스프링 부트 스타터만 살펴봐도 충분히 기술적인 요구사항을 수용할
수 있다.

기능분석 및 설계

- 소스버전관리 시스템: [Git](#) with [Github](#)
- 빌드배포 시스템: [jenkins](#)(나중에 [AWS CodeDeploy](#)... 써보고 싶다)
- 로그수집...
- 메일발송([Mail Chimp](#), [SendGrid](#), ... 등)
- 앱푸시([AWS SNS](#), [Firebase](#), ...)
- 문자발송(SMS? [알림톡](#))
- 개발환경은...
 - local
 - test
 - dev
 - beta(경우에 따라서 생략했다가 나중에...)
 - prod
- DB(데이터베이스)는... [MariaDB](#)...



소스버전을 관리하고

그외에 필요한 기능(메일발송, SMS발송, 앱푸시 등)을 제공하는 서비스를 탐색하는 과정도 개발자의 뛰....

사람이 많은 곳에서야 그런 기술탐색을 해주는 업무수행자도 있겠지만
스타트업에서는 개발자가 알아서 해야 한다.

도메인(or 모델) 설계

- 판매상품은 책이다.
- 책은 유형별로 관리되어야 한다.
- 책은 작가가 출판사를 통해 출간한다.
- 책은 서비스 관리자가 등록한다.
- 고객이 책을 주문한다.

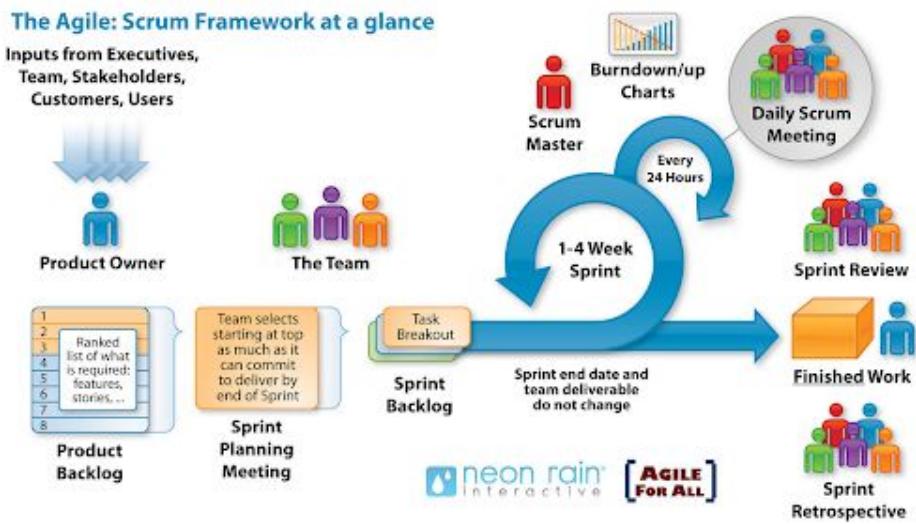


- 판매상품(Item)은 책(Book)이다.
- 책은 유형(Category)별로 관리되어야 한다.
- 책은 작가(Author)가 출판사(Publisher)를 통해 출간한다.
- 책은 서비스 관리자/Administrator가 등록한다.
- 고객(Customer)이 책을 주문(Order)한다.
- ...



개발자

개발방법론 - 애자일(Agile) - Note



잦은 출시(Release)를 해야하는 이유



요구사항이 쏟아져 나온다.



프로토타이핑이라도 동작하는 것을 보여주어야, 기획자와 사용자가 무엇을 원하는지
조금 더 명확하게 확인하고 이를 시기에 수정하며 개발비용을 줄일 수 있다.
다 개발한 상태에서 수정하는 것은 정말 어렵다.

개발(develop) 시작

The screenshot shows the Spring Tool Suite interface with the following details:

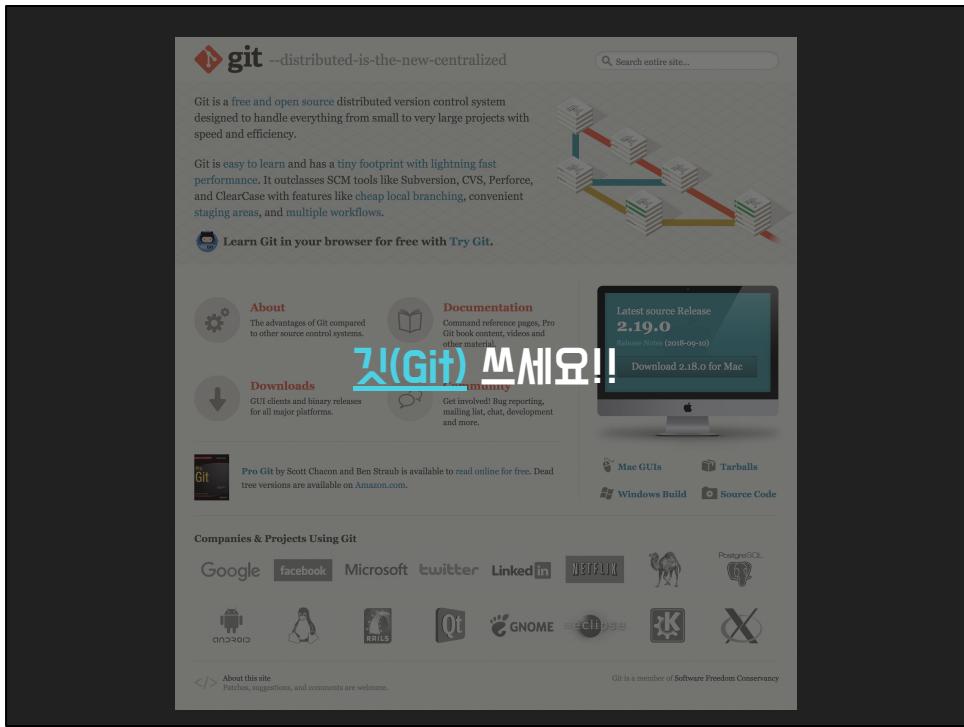
- Title Bar:** t-book-store-24 - tacademy-book-store/settings.gradle - Spring Tool Suite
- File Menu:** File Edit Navigate Search Project Run Window Help
- Packaging Explorer:** Shows the project structure:
 - tacademy-book-store [root]
 - src/main/java
 - BookControllerT
 - ApiController
 - build.gradle
 - settings.gradle
 - WebMvcConfig
 - src/main/resources
 - src/test/java
 - src/test/resources
 - Integration-test/java
 - JRE System Library [JavaSE-1.8]
 - Project and External Dependencies
 - out
 - src
- Editor:** The 'settings.gradle' file is open, showing the following Groovy code:

```
rootProject.name = "tacademy-book-store"
/*
 * 스템리티 등 모듈 공통 기능 제공
 */
include "common"
/*
 * JPA를 기반으로 구조된 도메인(@Entity, @Repository) 및 데이터베이스 구성
 */
include "core"
/*
 * 워치 툴
 */
include "api"
/*
 * 서비스 관련 시스템(백오피스)
 */
include "admin"
/*
 * TODO 항목, 통계 등의 데이터 처리작업 담당
 */
include "batch"

```
- Console:** 0 errors, 31 warnings, 0 others
- Problems:** Description, Resource, Path, Location, Type
- Servers:** Boot Dashboard
- Bottom Status Bar:** Runs: 0/0, Errors: 0, Failures: 0
- Failure Trace:** (empty)

소스코드관리

(SCM, Source Code Management)



깃(Git)

- 리눅스 토발즈가 리눅스 소스코드 관리를 위해 만듦!
- Git != GitHub
- Git은 소스코드 버전관리 시스템
- Github은 원격저장소 및 개발지원플랫폼
 - 5명 이하의 작은 규모인 경우: BitBucket
- 잘 모르는 이들을 위해!
 - 생활코딩 - [지옥에서 온 Git](#)
 - [OSS개발자포럼&국민대학교]Git/GitHub 입문하기 Hands on Lab: 2018/11/10 예정
 - [Resource to learn git](#)
 - 찾아보면 무료 영상자료나 교육자료 많음

bookstore24 프로젝트 모듈 구성

- **common**: 프로젝트에서 공통으로 사용하는 유틸리티, 예외 (Exception) 등
- **core**: 프로젝트 도메인(@Entity, @Repository)
- **api**: 외부에 정보를 제공하는 REST API 모듈
- **admin**: 서비스를 관리하기 위한 백오피스
- **batch**: 정기적으로 실행될 배치 프로그램 모음
- **message**: 알림톡, SMS, 메일 발송 등 담당

프로젝트의 소스를 활용하기 위해 멀티모듈로 구성하게 된다.
그레이들이 멀티모듈을 구성하고 빌드하는데 편하기에 애용하고 있다.

gitignore.io

gitignore.io

Create useful .gitignore files for your project

Search Operating Systems, IDEs, or Programming Languages [Create](#)

[Source Code](#) | [Command Line Docs](#) | [Watch Video Tutorial](#)

깃에서 저장하지 않을 파일을 지정(.gitignore)한다.

java, gradle, idea, eclipse, 정도면 적당하다.

프로젝트 구성시 필수조건

- **README를 작성하라.**
 - 실행절차를 설명하라.
 - 실행절차에 따라 빌드하고 실행되도록 하라.
- **커밋 및 푸시는 테스트 및 빌드가 성공되었을때 하라.**
 - 자신이 작성한 코드가 충돌이 난다면 충돌을 해결하고(지우지 말고!) 올리자.
- **코딩 컨벤션(Convention, 관례)은 팀원들과 함께 만들자.**
 - 이미 만들어져 있다면 코드리뷰(Code Review)를 통해 공유하고 적용해보자.

One Source Multi Use

하나의 배포본 파일을 가지고 여러 곳에서 사용할 수 있어야 한다.

스프링 부트는 외부구성 기능을 통해 이런 방향성을 제공한다.

프로파일 구성

- **local**: 개발자 로컬 실행환경
 - 개발자가 자유롭게 초기화 및 구성을 수행할 수 있다.
- **test**: 통합테스트 환경(주로 빌드 전 실행된다)
 - 테스트 실행때마다 초기화된다.
- **dev**: 개발서버 실행환경
 - 운영서버와 동일한 환경을 가지며 개발기능을 확인하는 용도로 사용된다.
- **beta**: (준)운영서버 실행환경
 - 운영서버와 동일한 환경으로 큰 배포에 앞서서 운영서버의 데이터를 복제하여 정상동작확인
- **prod**: 운영서버 실행환경
 - (가급적) 손대지 않아야 할 환경

실행환경별 프로파일 구성

프로파일 Profile	spring.profile	Datasource application-datasource.yml	구글 맵 API KEY application-api.yml
로컬환경 local	spring.profile: local	jdbc:h2:file:~/tacademy;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE;MODE=mysql	무료 설정 Authorization: Bearer 00000-0000-00000
자동테스트 test	spring.profile: test	jdbc:h2:mem:tacademy;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE;MODE=mysql	무료 설정 Authorization: Bearer 00000-0000-00000
개발 내용 확인 dev	spring.profile: dev	jdbc:mysql://hostname:3306/bookstore?user=userName&password=password	무료 설정 Authorization: Bearer 12341-23412-34123
관제자 확인 beta	spring.profile: beta	jdbc:mysql://hostname:3306/bookstorebeta?user=userName&password=password	유료 설정 Authorization: Bearer 12412-23141-23142
운영 prod	spring.profile: prod	jdbc:mysql://hostname-prod:3306/bookstore?user=userName&password=password	유료 설정 Authorization: Bearer 12412-23141-23142

application-api.yml

```
google:
  map:
    api-key: 00000-0000-00000
    --- 문서 구분자
    spring.profile: dev
    google:
      map:
        api-key: 12341-23412-34123
        --- 문서 구분자
        spring.profile: prod
        google:
          map:
            api-key: 12412-23141-23142
```

application-datasource.yml

```
spring:
flyway:
  enabled: false
jpa:
  hibernate:
    ddl-auto: validate
    show-sql: false
---
spring:
  profiles: local
  datasource:
    url:
      jdbc:h2:file:~/bookstore24;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE;MODE=mysql
    h2:
      console:
        enabled: true
      flyway:
        enabled: true
        locations: classpath:db/migration/{vendor}
        baseline-on-migrate: true
    jpa:
      hibernate:
        ddl-auto: update
      show-sql: true
---
spring:
  profiles: test
  datasource:
    url:
      jdbc:h2:mem:bookstore24;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE;MODE=mysql
    jpa:
      hibernate:
        ddl-auto: create
        show-sql: true
---
spring:
  profiles: dev
  datasource:
    url:
      jdbc:h2:mem:bookstore24;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE;MODE=mysql
    jpa:
      hibernate:
        ddl-auto: create
        show-sql: true
---
spring:
  profiles: beta
  datasource:
    url:
      jdbc:h2:mem:bookstore24;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE;MODE=mysql
    jpa:
      hibernate:
        ddl-auto: create
        show-sql: true
---
spring:
  profiles: prod
  datasource:
    url:
      jdbc:h2:mem:bookstore24;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE;MODE=mysql
    jpa:
      hibernate:
        ddl-auto: create
        show-sql: true
```

```
application.yml
```

```
spring:  
  profiles:  
    include:  
      - datasource  
      - api
```

CI/CD: 코드를 푸시하면 배포가 일어난다!

1. 기능을 정의하고 이를 관리하기 위한 이슈 발급
2. 기능(feature) 개발하고 리뷰 받고 OK하면 develop 브랜치에 푸시
3. 개발서버에 배포
4. 개발서버에서 기능확인
5. 베타서버 배포
6. 베타서버에서 기능관련자(기획자)의 기능확인
7. develop 브랜치 코드를 master에 머지하고 푸시
8. 운영서버 배포
9. 운영서버 적용

어디서 배포하고 운영할까?

어디서 배포하고 운영할까?



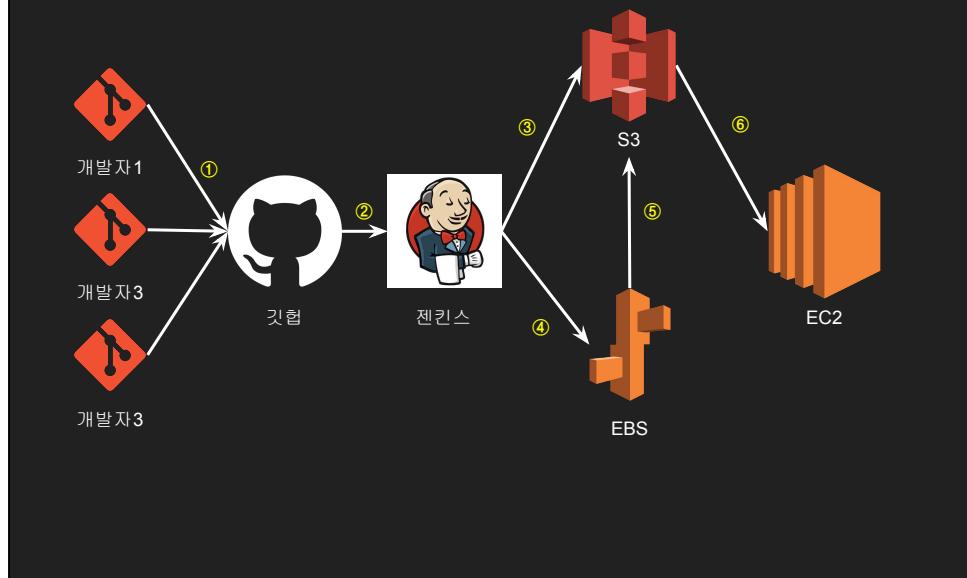
Cloud Platform

VS

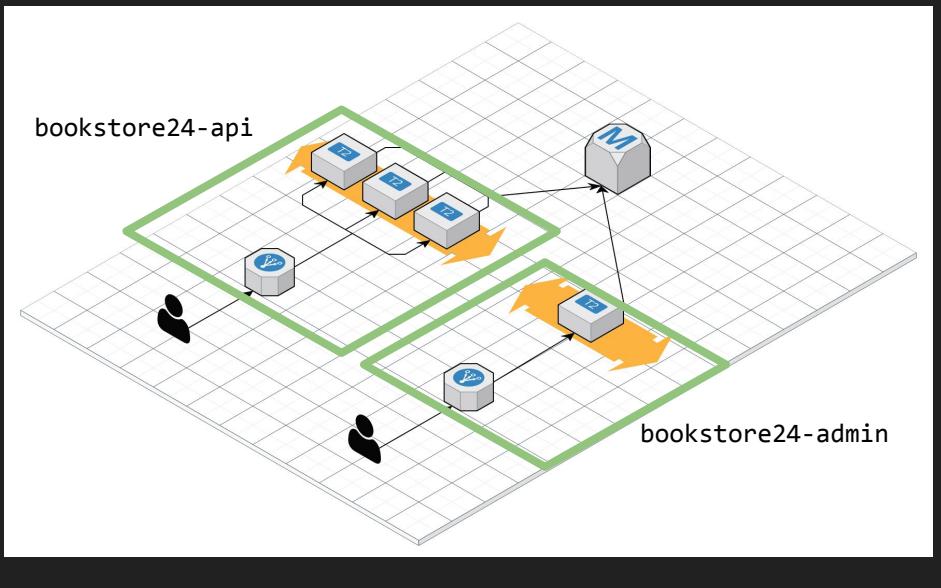


IDE(Internet Data Center)

지속적 통합과 지속적 배달(CI&CD)



기본적인 AWS 구성



CI/CD 구축하기

- **창천향로 - Jenkins**

- [1. Jenkins로 Beanstalk + Multi Module 배포하기 - Jenkins 설치](#)
- [2. Jenkins로 Beanstalk + Multi Module 배포하기 - Jenkins와 Github 연동하기](#)
- [3. Jenkins로 Beanstalk + Multi Module 배포하기 - Beanstalk 연동하기](#)

- **창천향로 - AWS**

- [1\) AWS로 배포하기 시리즈 - 1. Code Deploy 사용하기](#)
- [2\) AWS로 배포하기 시리즈 - 2. AWS Code Build로 빌드하기](#)
- [3\) AWS로 배포하기 시리즈 - 3. AWS Code Pipeline으로 배포하기](#)

클라우드를 믿지 말아요.

클라우드가 안정성을 보장해주지 않는다.

사용자(개발자)가 클라우드 플랫폼에서 사용해도 안정적으로 운영될 수 있도록 운영해야 한다(니취팔로마!! 비싼돈 내고 그래야 하니?).

이제 코딩을 해봅시다.

기본적인 개발방식은 도메인 계층을 먼저 개발한다.

- `@Entity`
- `@Repository`

그리고 도메인간 서로 연계되는 부분이 있는 경우 서비스 계층에서 개발한다.

- `@Service`
- `@Transaction` 트랜잭션 관리도 이 영역에서!

외부에 노출되는 부분에서는 표현 계층에서 개발한다.

- `@Controller`, `@RestController` + `@ViewController`
- `@RequestMapping`
- `ModelAndView`

꼿!!
이 아닌 시작!!

참고문헌

- Development - 허니몬 컬렉션
 - <https://plus.google.com/collection/4d04Y>
- 깃(Git)
 - <https://git-scm.com/>
- .gitignore
 - <http://gitignore.io/>
- 생활코딩
 - <https://opentutorials.org/course/1>
- README
 - <https://help.github.com/articles/about-readmes/>
- Azure(Microsoft)
 - <https://azure.microsoft.com/ko-kr>
- AWS(Amazon Web Service)
 - <https://aws.amazon.com/ko/>

참고문헌

- Development - 허니몬 컬렉션
 - <https://plus.google.com/collection/4d04Y>
- 깃(Git)
 - <https://git-scm.com/>
- .gitignore
 - <http://gitignore.io/>
- 생활코딩
 - <https://opentutorials.org/course/1>
- README
 - <https://help.github.com/articles/about-readmes/>
- Azure(Microsoft)
 - <https://azure.microsoft.com/ko-kr>
- AWS(Amazon Web Service)
 - <https://aws.amazon.com/ko/>

참고문헌

- Google Cloud Platform
 - <https://cloud.google.com/gcp/>
- Github
 - <http://github.com>
- Jenkins
 - <https://jenkins.io/>
- CodeDeploy
 - <https://aws.amazon.com/ko/codedeploy/>
- Travis CI
 - <https://travis-ci.org/>
- Cloudcraft
 - <https://cloudcraft.co/>
- 창천향로
 - <https://jojoldu.tistory.com/>

참고문헌

- **Http Headers: Authorization**
 - <https://developer.mozilla.org/ko/docs/Web/HTTP/Headers/Authorization>
- **HTTP authentication**
 - https://developer.mozilla.org/ko/docs/Web/HTTP/Authentication#%EC%9D%B8%EC%A6%9D_%EC%8A%A4%ED%82%B4