






Spring IDE 설치

Eclipse Marketplace

One solution selected for install

SearchRecentPopularFavoritesInstalledResearch at the Eclipse




Spring Tools 4 (aka Spring Tool Suite 4) 4.24.0.RELEASE

Spring Tools 4 is the next generation of Spring Boot tooling for your favorite coding environment. Largely rebuilt from scratch, it provides world-class support... [more info](#)

by Broadcom, EPL

[spring](#) [Spring IDE](#) [Cloud](#) [Spring Tool Suite](#) [STS](#)

★ 4245Installs: 2.93M (20,548 last month)Install Pending



Eclipse Enterprise Java and Web Developer Tools 3.33

Enables Enterprise Java Bean, Java Enterprise Application, Fragments, and Connector, Java Web Application, JavaServer Faces (JSF), Java Server Pages (JSP), Java... [more info](#)


by The Eclipse Foundation, EPL

[xml](#) [html](#) [CSS](#) [js](#) [jsp](#)

★ 1749Installs: 1.23M (14,353 last month)Installed

One solution selected | Deselect all

Marketplaces



?

< BackInstall Now >FinishCancel

Eclipse Marketplace

Confirm Selected Features

Press Confirm to continue with the installation. Or go back to choose more solutions to install.

☒ Spring Tools 4 (aka Spring Tool Suite 4) 4.24.0.RELEASE <https://cdn.spring.io>

☒ Spring Boot Language Server Feature (required)

☒ Spring Tool Suite 4 Main Feature (required)

☐ Cloud Foundry Manifest Language Server Feature

☐ Concourse Pipeline Language Server Feature

☒ Spring IDE Boot Microservices Dash

?

< Install MoreConfirm >FinishCancel



Spring Framework

- 기본적인 Java 웹 개발의 핵심은 자바와 서블릿이라 볼 수 있지만, 학습용이 아닌 실무(엔터프라이즈 애플리케이션)에서 개발하는 서비스는 고려할 사항이 많아진다.
- 엔터프라이즈 애플리케이션 : 대규모의 복잡한 데이터를 관리하는 애플리케이션. 많은 사용자의 요청을 동시에 처리해야 하기 때문에 서버의 성능과 안정성, 보안이 매우 중요하다.
- 하지만, 모든 것(성능, 안정성, 보안, 타 서비스와의 연계 등)을 신경쓰면서 사이트의 기능(비즈니스 로직)까지 개발하는 것은 매우 어렵다.
- Java EE(Enterprise Edition)가 생겨났지만 스펙의 복잡함과 구현의 어려움으로 인해 현업에 완전한 정착에 실패했다.
- 2003년 6월 스프링 프레임워크는 엔터프라이즈 애플리케이션을 위한 개발 환경을 제공해서 기능 개발에만 집중할 수 있도록 도와주는 도구이다.
- Spring Framework는 Java 기반의 OpenSource Framework로 단순하고 효율적인 해결이 가능하도록 설계되었다.
- 스프링 프레임워크는 현재 Java 웹 개발에서 빼놓을 수 없는, 수많은 기업과 개발자들이 애용하는 필수 도구 중 하나가 되었다.



Spring Framework

- **경량 프레임워크**
 - 스프링은 필요한 라이브러리만 선택적으로 사용할 수 있는 경량 프레임워크로 애플리케이션의 크기를 최소화하고, 실행 속도를 최적화할 수 있다.
- **제어의 역행(Inversion Of Control, IOC)**
 - 객체의 생성, 소멸과 같은 생명 주기를 스프링 컨테이너가 관리하게 된다. 이를 통해 코드의 유연성과 재사용성을 높일 수 있다.
- **의존성 주입(Dependency Injection, DI)**
 - 각 계층이나 서비스 간의 의존성이 존재하는 경우, 프레임워크가 연결해준다.
 - 객체 간의 의존성을 외부에서 주입하여 코드의 결합도는 낮추고 테스트와 유지보수를 쉽게 할 수 있다.
- **관점지향 프로그래밍(Asspect-Oriented Programming, AOP)**
 - AOP는 여러 모듈에서 공통적으로 사용하는 기능(예: 트랜잭션, 로깅, 보안)을 분리하여 관리할 수 있도록 한다.
 - AOP를 통해 관심사를 모듈화하고, 코드 중복을 줄이며 기능을 핵심 관점과 부가 관점으로 명확히 분리할 수 있다.
- **POJO(Plain Old Java Object) 기반 개발 지향**
 - 기존의 Java 객체를 그대로 사용 가능하다.



Spring Boot

- 스프링 프레임워크는 장점이 많은 개발도구이지만, 설정이 매우 복잡하다.
- 스프링 개발팀에서도 이런 단점을 인식하고 보완하고자 2013년 4월 스프링 부트를 출시했다.
- 스프링 부트는 스프링 프레임워크를 더 쉽고 빠르게 이용할 수 있도록 만들어주는 도구로, 빠르게 스프링 프로젝트를 설정할 수 있고, 의존성 세트라고 불리는 스타터(starter)를 사용해 간편하게 의존성을 사용하거나 관리할 수 있다.
- 스프링 부트가 갖는 주요 특징은 아래와 같다.
 - 웹 애플리케이션 서버(WAS)가 내장되어 있어서 따로 설치하지 않아도 독립적으로 실행할 수 있다. (바로 개발 가능)
 - 빌드 구성을 단순화하는 스프링 부트 스타터를 제공한다.
 - XML 설정 없이, Java 코드로 모두 작성할 수 있다.
 - JAR를 이용해서 자바 옵션만으로도 배포가 가능하다.



Spring 개념 (IoC, DI)

- Spring은 모든 기능의 기반을 제어의 역전(IoC)와 의존성 주입(DI)에 두고 있다.
- IoC(Inversion Of Control)
 - 객체를 직접 생성하거나 제어하는 것이 아니라 외부에서 관리하는 객체를 가져와 사용하는 것을 의미한다.
- DI(Dependency Injection)
 - IoC를 구현하기 위해 사용되는 방법으로, 특정 클래스가 다른 클래스에 의존하는 것을 의미한다.
 - @Autowired라는 애너테이션으로 스프링 컨테이너에 있는 빈(Beans)을 주입한다.



Spring 개념 (스프링 컨테이너, Bean)

- 스프링 컨테이너는 빈(Beans)을 생성하고 관리한다.
- 즉, 빈이 생성되고 소멸되기까지의 생명주기를 스프링 컨테이너가 관리한다.
- 빈(Beans)이란, 스프링 컨테이너가 생성하고 관리하는 객체이다.
- 빈을 스프링 컨테이너에 등록하기 위해서는 XML 파일을 설정하거나 애너테이션을 추가하는 방법을 사용한다.
- 빈의 이름은 클래스 이름의 첫 글자를 소문자로 바꿔 관리한다.



Spring Boot 프로젝트 생성

New Project

Select a wizard

Create new Spring Starter Project

Wizards:

type filter text

- > General
- > EJB
- > Gradle
- > Java
- > Java EE
- > Maven
- > Spring Boot
 - Import Spring Getting Started Content
 - Spring Starter Project
- > Web
- > Examples

< Back Next > Finish Cancel

New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location: Browse

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

Package:

Working sets

☐ Add project to working sets

Working sets: New... Select...

< Back Next > Finish Cancel



Spring Boot 프로젝트 생성

- 기존의 스프링 개발과는 달리 Spring Starter Project는 손쉽게 스프링 부트 기반의 프로젝트를 생성한다.
- 프로젝트 이름, 빌드 관리 도구 및 아티팩트, 자바 버전, 패키지 등 프로젝트의 기본적인 사항들을 설정은 물론, 필요한 의존성 라이브러리들을 선택하면 된다. (프로젝트 생성 후에 의존성을 추가할 수도 있다.)



Spring Boot 프로젝트 생성

New Project

Select a wizard

Create new Spring Starter Project

Wizards:

type filter text

- > General
- > EJB
- > Gradle
- > Java
- > Java EE
- > Maven
- > Spring Boot
 - Import Spring Getting Started Content
 - Spring Starter Project
- > Web
- > Examples

< Back Next > Finish Cancel

New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location: Browse

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

Package:

Working sets

☐ Add project to working sets

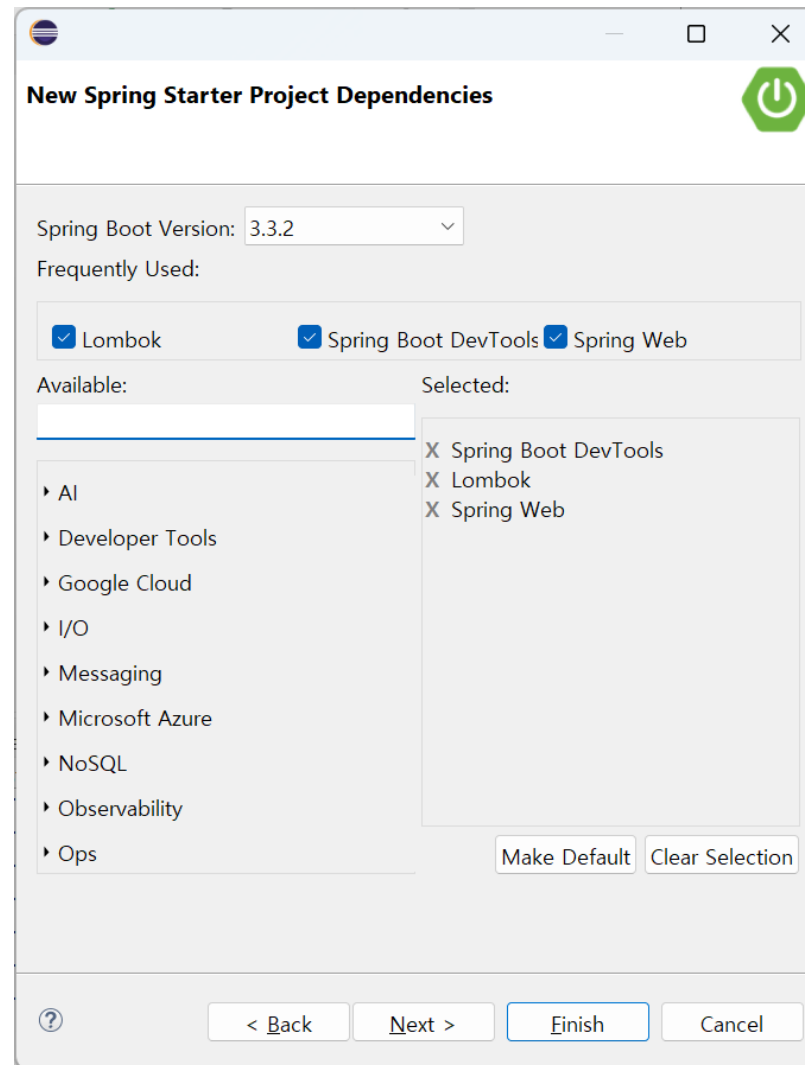
Working sets: New... Select...

< Back Next > Finish Cancel



Spring Boot 프로젝트 생성

- 웹 프로젝트 생성을 위해 Spring Web을 선택
- 개발 편의를 위해 DevTools를 선택
- Lombok도 선택





Spring Boot 프로젝트 실행

- Run as > Spring Boot App
- 콘솔 로그를 통해 프로젝트가 톰캣으로 실행되었으며, 8080 포트를 사용한다는 것을 알 수 있다.
- 웹 브라우저에서도 정상적으로 실행되는지 살펴보기 위해 localhost:8080을 입력해보자.

```
Starting FirstSpringApplication using Java 17.0.10 with PID 37872
No active profile set, falling back to 1 default profile: "default"
Devtools property defaults active! Set 'spring.devtools.add-properties' to 'false' to disable
For additional web related logging consider setting the 'logging.level.web' property to 'DEBUG'
Tomcat initialized with port 8080 (http)
Starting service [Tomcat]
Starting Servlet engine: [Apache Tomcat/10.1.26]
Initializing Spring embedded WebApplicationContext
Root WebApplicationContext: initialization completed in 2062 ms
LiveReload server is running on port 35729
Tomcat started on port 8080 (http) with context path '/'
Started FirstSpringApplication in 3.765 seconds (process running for 5.204)
Initializing Spring DispatcherServlet 'dispatcherServlet'
Initializing Servlet 'dispatcherServlet'
Completed initialization in 2 ms
```



Spring Boot 프로젝트 실행

- 404 에러 페이지가 나타나는 것이 정상적인 결과이다.

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Sat Aug 10 20:53:02 KST 2024

There was an unexpected error (type=Not Found, status=404).

No static resource .

- 스프링부트로 프로젝트를 생성하면 프로젝트의 실행에 관련된 기능이 자동으로 설정되지만, 화면에 보이는 부분은 자동으로 만들어지지 않기 때문이다.
- 이제 브라우저 화면에 눈에 보이는 결과를 만들어보자.



Spring Boot 프로젝트 실행

- com.sample.demo.controller 패키지를 생성하고, HelloController 클래스를 생성해보자.

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

```
@RestController
```

```
public class HelloCon
```

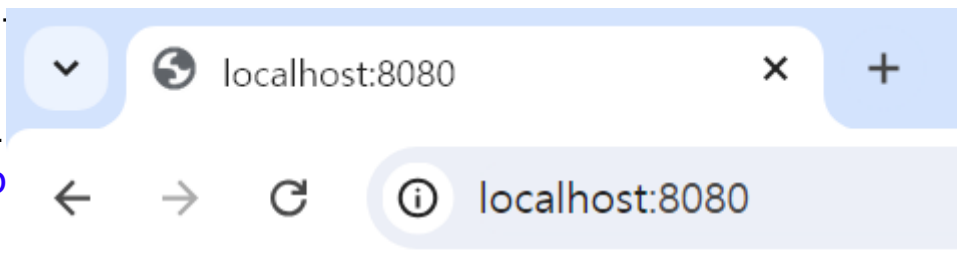
```
    @RequestMapping("
```

```
        public String hel
```

```
            return "Hello
```

```
        }
```

```
    }
```

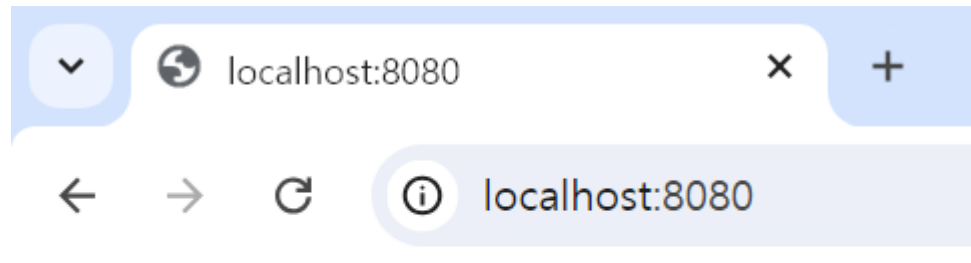


Hello World! 안녕하세요

- @RestController
해당 클래스가 REST Controller 기능을 수행하도록 한다.
- @RequestMapping
해당 메소드를 실행할 수 있는 주소를 설정한다.
- hello 메소드 :
반환되는 문자열을 화면에 전달해주는 역할을 한다.
@RequestMapping에 설정한 주소가 호출되면 메소드가 실행된다.



Spring Boot 프로젝트 실행

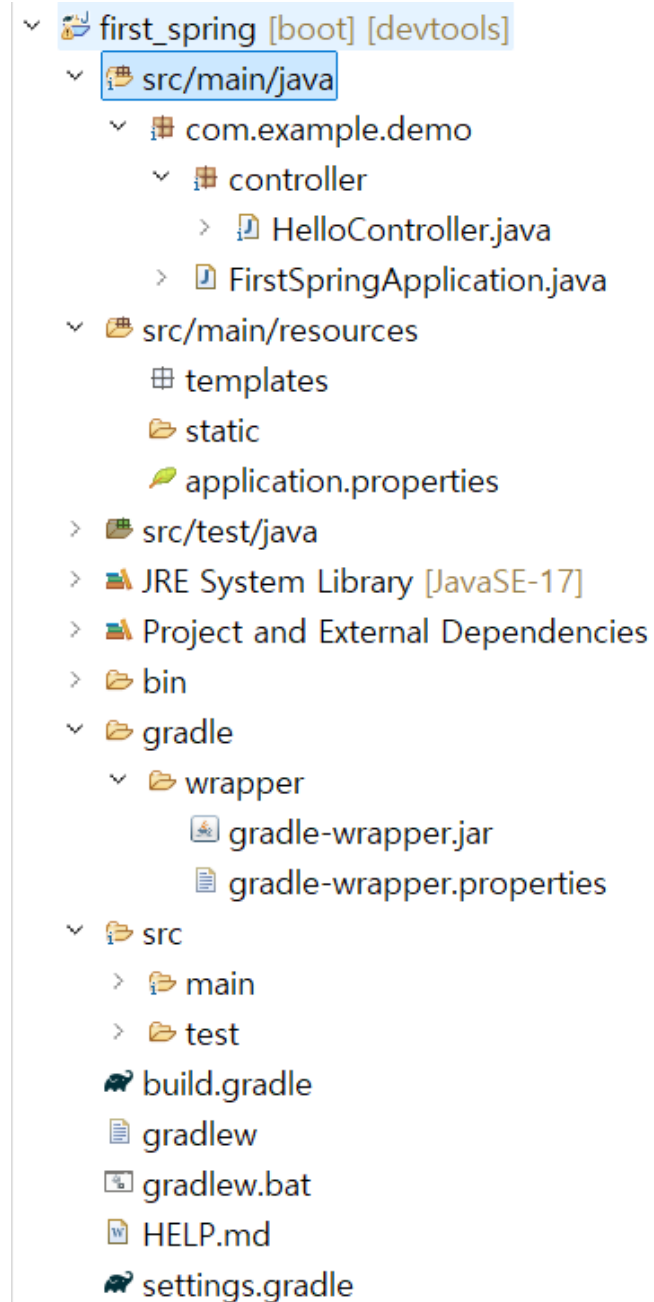


Hello World! 안녕~



Spring Boot 프로젝트 구조 파악

- src/main/java : 프로젝트의 Java 소스 코드가 위치하는 디렉토리
 - FirstSpringApplication : 애플리케이션을 시작할 수 있는 스프링 메인 클래스
- src/main/resources : 애플리케이션의 리소스 파일이 위치하는 디렉토리
 - static : CSS, JS, image 등 정적 리소스 디렉토리
 - templates : 스프링 부트에서 사용 가능한 뷰 템플릿 디렉토리
 - application.properties 또는 application.yml : 설정에 필요한 프로퍼티
- src/test/java : 테스트 소스 코드가 위치하는 디렉토리
- Project and External Dependency : Gradle에 명시한 프로젝트의 필수 라이브러리
- gradle/wrapper/ : 특정 버전 Gradle을 사용할 수 있게 하는 관련 파일
- gradlew, gradlew.bat : Gradle Wrapper 실행 스크립트
- build.gradle : Gradle 빌드 설정 파일





Spring Boot 프로젝트 구조 파악

- FirstSpringApplication은 스프링 부트 애플리케이션의 구성과 실행을 담당하는 중요한 클래스이다.
- @SpringBootApplication은 SpringBoot 사용에 필요한 기본설정으로, 주요한 어노테이션 세 개로 구성되어 있다.
 - @EnableAutoConfiguration : 스프링의 다양한 설정을 자동으로 구성
 - @ComponentScan : Bean을 일일이 선언하지 않고, 자동으로 컴포넌트 클래스를 검색하고 컨텍스트에 등록
 - @SpringBootConfiguration : Java 기반 설정 (@Configuration이 내장되어 있음)

```
package com.example.demo;
```

```
@SpringBootApplication
```

```
public class FirstSpringApplication {
```

```
    // 프로젝트의 메인 메소드
```

```
    public static void main(String[] args) {
```

```
        // 애플리케이션 실행
```

```
        SpringApplication.run(FirstSpringApplication.class, args);
```

```
    }
```

```
}
```



build.gradle

- build.gradle은 생성된 프로젝트의 빌드를 관리하는 파일이다.

```
plugins {  
    id 'java' // Java 플러그인을 적용  
    id 'org.springframework.boot' version '3.3.2' // Spring Boot 플러그인을 적용  
    id 'io.spring.dependency-management' version '1.1.6' // Spring의 의존성 관리 플러그인 적용  
}
```

```
group = 'com.example.demo'  
version = '0.0.1-SNAPSHOT'
```

```
java {  
    toolchain {  
        languageVersion = JavaLanguageVersion.of(17) // Java 17 버전을 사용하도록 설정  
    }  
}
```



build.gradle

```
repositories {  
    mavenCentral() // 프로젝트의 의존성을 다운로드할 중앙 저장소 지정  
}  
  
dependencies {  
    // Spring Boot 웹 스타터 의존성 추가  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    // Lombok 라이브러리 추가  
    compileOnly 'org.projectlombok:Lombok'  
    annotationProcessor 'org.projectlombok:Lombok'  
    // Spring Boot DEV tool 추가 (서버 재시작 없이 수정 내용 반영)  
    developmentOnly 'org.springframework.boot:spring-boot-devtools'  
    // Spring Boot 테스트 스타터 의존성 추가  
    testImplementation 'org.springframework.boot:spring-boot-starter-test'  
    // Junit 테스트 실행을 위한 의존성 추가  
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'  
}  
  
tasks.named('test') { // 테스트 작업을 설정  
    useJUnitPlatform() // JUnit 플랫폼을 사용하여 테스트를 실행하도록 설정  
}
```



스프링 프레임워크

- 웹애플리케이션 개발을 위한 다양한 프레임워크 중 가장 많이 사용되는 프레임워크 중 하나는 스프링 프레임워크이다.
- 우리나라에서 전자정부 표준 프레임워크로 채택되면서 Java 개발자들이 알아야 할 사실상의 표준이 되어있다.
- 스프링 프레임워크에는 정말 많은 프로젝트들이 있으나 모든 프로젝트를 알아야 할 필요는 없으며, 상황에 따라 하나씩 추가해 나가면 된다.



Spring Boot

Takes an opinionated view of building Spring applications and gets you up and running as quickly as possible.



Spring Framework

Provides core support for dependency injection, transaction management, web apps, data access, messaging, and more.



Spring HATEOAS

Simplifies creating REST representations that follow the HATEOAS principle.



Spring Modulith

Spring Modulith allows developers to build well-structured Spring Boot applications and guides developers in finding and working with application modules driven by the domain.



Spring for Apache Pulsar

Provides Familiar Spring Abstractions for Apache Pulsar



Spring Data

Provides a consistent approach to data access – relational, non-relational, map-reduce, and beyond.



Spring Cloud

Provides a set of tools for common patterns in distributed systems. Useful for building and deploying microservices.



Spring REST Docs

Lets you document RESTful services by combining hand-written documentation with auto-generated snippets produced with Spring MVC Test or REST Assured.



Spring AI

Spring AI is an application framework for AI engineering.



Spring StateMachine

Provides a framework for application developers to use state machine concepts with Spring applications.



Spring Cloud Data Flow

Provides an orchestration service for composable data microservice applications on modern runtimes.



Spring Security

Protects your application with comprehensive and extensible authentication and authorization support.



Spring Batch

Simplifies and optimizes the work of processing high-volume batch operations.



Spring CLI

A CLI focused on developer productivity



Spring Web Services

Facilitates the development of contract-first SOAP web services.



Spring Authorization Server

Provides a secure, light-weight, and customizable foundation for building OpenID Connect 1.0 Identity Providers and OAuth2 Authorization Server products.



Spring for GraphQL

Spring for GraphQL provides support for Spring applications built on GraphQL Java.



Spring AMQP

Applies core Spring concepts to the development of AMQP-based messaging solutions.



Spring Flo

Provides a JavaScript library that offers a basic embeddable HTML5 visual builder for pipelines and simple graphs.



Spring Shell

Makes writing and testing RESTful applications easier with CLI-based resource discovery and interaction.



Spring Session

Provides an API and implementations for managing a user's session information.



Spring Integration

Supports the well-known Enterprise Integration Patterns through lightweight messaging and declarative adapters.



Spring for Apache Kafka

Provides Familiar Spring Abstractions for Apache Kafka.



Spring LDAP

Simplifies the development of applications that use LDAP by using Spring's familiar template-based approach.



Spring Web Flow

Supports building web applications that feature controlled navigation, such as checking in for a flight or applying for a loan.

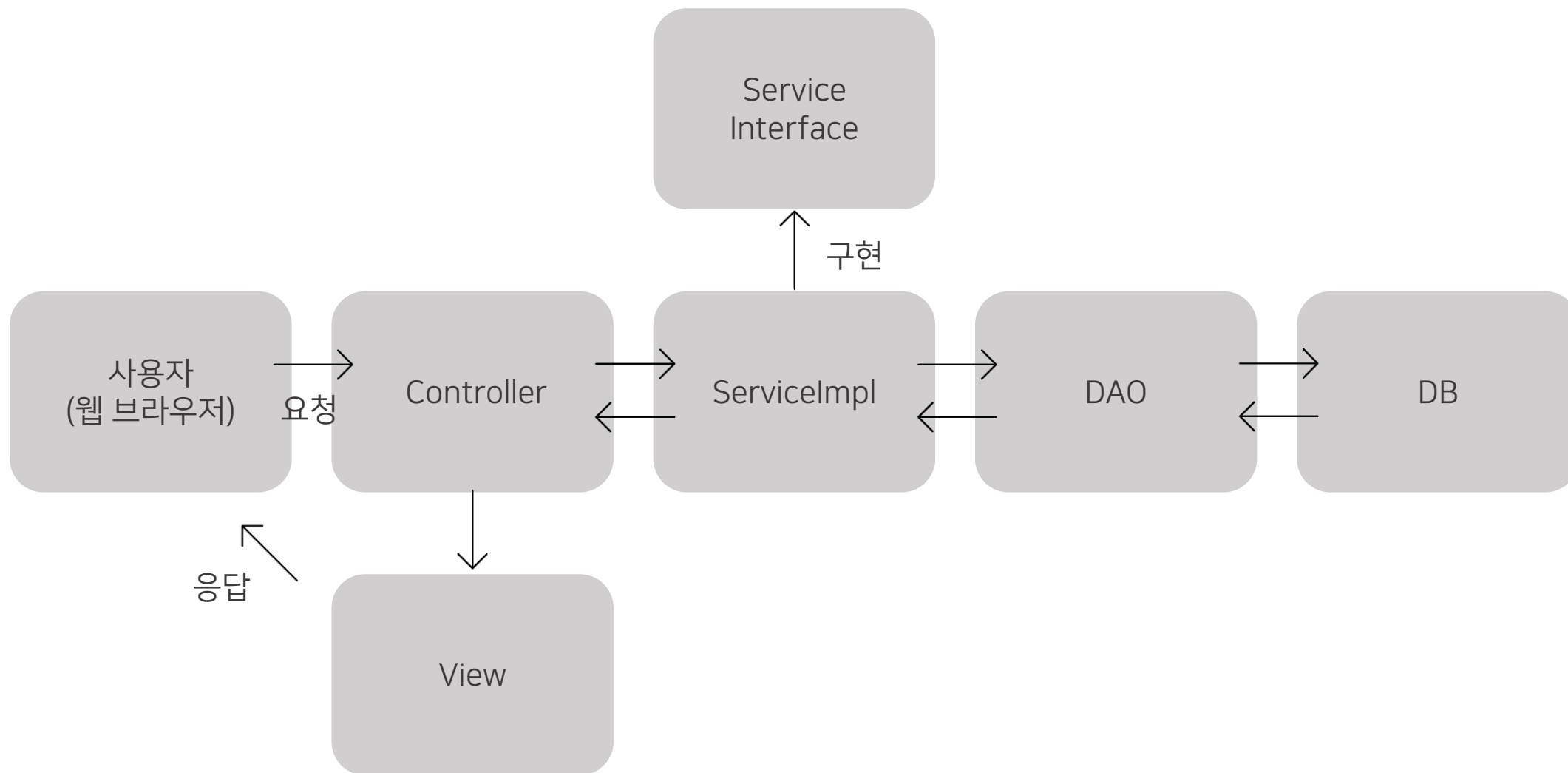


MVC 패턴

- 웹 애플리케이션의 아키텍처(구조)를 이야기하면 빠지지 않는 것이 MVC 패턴이다.
- MVC 패턴(Model, View, Controller 패턴)에는 모델1, 모델2가 있지만 최근 웹 개발은 대부분 모델 2를 사용한다.
- 따라서 MVC 패턴이라고 이야기하면 모델 2를 의미한다.
- MVC 패턴을 이용하면 사용자 인터페이스와 비즈니스 로직을 분리하여 개발할 수 있다.
- 즉, 서로의 영향을 최소화하여 개발하고 변경이 쉬운 애플리케이션을 만들 수 있다.



MVC 패턴





Spring boot로 게시판 개발하기

- board 프로젝트를 새롭게 생성해서 게시판 개발을 통해 MVC 패턴을 익혀보자.
- 게시판은 데이터의 조회, 입력, 수정, 삭제 뿐만 아니라 파일 업로드, 다운로드 등 웹 애플리케이션에서 필요한 기능들을 포함하고 있어서 적합한 예제가 된다.
- Spring Web
- Spring Boot DevTools
- Lombok
- MySQL Driver
- Thymeleaf
- MyBatis Framework

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:



Spring boot로 게시판 개발하기

- 스프링 부트 스타터를 이용해서 애플리케이션을 생성하면, 선택된 라이브러리들이 자동으로 다운로드된다.
- 만약 다운로드 중 에러가 발생하거나 선택한 라이브러리가 동작하지 않는 등 문제가 생기는 경우가 있다.
- 이런 경우에는 프로젝트를 선택하고, 마우스 우클릭 후에 Gradle > Refresh Gradle Project를 클릭하면 된다.
- Refresh Gradle Project를 클릭하면,
build.gradle에 정의되어 있는 라이브러리 중 새로 추가되거나 다운로드 도중 에러가 발생했던 라이브러리를 다시 다운로드한다.



Spring boot로 게시판 개발하기

- 웹 애플리케이션에 데이터베이스는 빠지지 않는 요소이다.
- 먼저 MySQL에 데이터베이스를 생성하고, board 테이블을 생성하자.

```
DROP DATABASE IF EXISTS `board_db`;  
CREATE DATABASE board_db DEFAULT CHARACTER SET = 'utf8mb4' COLLATE = 'utf8mb4_0900_ai_ci';  
  
USE board_db;  
  
CREATE TABLE board_tbl (  
    id INT PRIMARY KEY AUTO_INCREMENT COMMENT '글 번호',  
    title VARCHAR(300) NOT NULL COMMENT '글 제목',  
    content TEXT NOT NULL COMMENT '글 내용',  
    hit INT NOT NULL DEFAULT 0 COMMENT '조회 수',  
    created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '작성일',  
    creator VARCHAR(50) NOT NULL COMMENT '작성자',  
    updated_at DATETIME DEFAULT NULL ON UPDATE CURRENT_TIMESTAMP COMMENT '수정일',  
    is_deleted CHAR(1) NOT NULL DEFAULT 'n' COMMENT '삭제 여부'  
);
```



Spring boot로 게시판 개발하기

- 프로젝트에 데이터 소스(Driver, URL, UserName, PassWord)를 설정해보자.
- 데이터 소스는 application.properties 또는 application.yml 파일에 선언하면 된다.

```
spring.application.name=board
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/board_db
spring.datasource.username=root
spring.datasource.password=1234
```

```
[com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
[com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@2f51ba9c
[com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
[o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
[o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path '/'
[com.board.BoardApplication : Started BoardApplication in 5.328 seconds (process running for 7.077)
```



Spring boot로 게시판 개발하기

- DTO 생성
 - DTO(Data Transfer Object)는 애플리케이션 내에서 각 계층(뷰, 컨트롤러, 서비스, DAO, DB) 간 데이터를 주고 받을 때 사용되는 객체이다.

```
package com.board.dto;
```

```
@Data
```

```
public class BoardDTO {  
    private int id;  
    private String title, content;  
    private int hit;  
    private LocalDateTime createdAt;  
    private String creator;  
    private LocalDateTime updatedAt;  
}
```



Spring boot로 게시판 개발하기

- 마이바티스(MyBatis)는 쿼리 기반 웹 애플리케이션을 개발할 때 가장 많이 사용되는 SQL 매퍼 프레임워크이다.
- MyBatis는 SQL을 XML 파일에 분리하여 작성하기 때문에, SQL의 변환이 자유롭고 가독성이 좋다는 특징이 있다.
- 먼저 Java가 사용하는 카멜 표기법과 MySQL에서 사용하는 스네이크 케이스 표기법이 자동으로 일치되도록 설정하자.

```
mybatis.configuration.map-underscore-to-camel-case=true
```



Spring boot로 게시판 개발하기

- 클라이언트의 요청을 받아 해당 요청을 수행하는 컨트롤러를 생성해보자.
 - 컨트롤러는 요청에 필요한 비즈니스 로직을 호출하고, 그 결과를 포함한 응답을 해주는 Dispatcher 역할을 한다.
1. 컨트롤러는 클래스에 @Controller 어노테이션을 적용한다.
 2. @RequestMapping 어노테이션을 이용해 요청에 맞는 주소를 지정한다.
 3. 요청에 필요한 비즈니스 로직을 호출한다.
 4. 실행된 비즈니스 로직의 결과를 뷰로 반환한다.



Spring boot로 게시판 개발하기

- ModelAndView는 호출된 요청의 결과를 보여줄 뷰를 지정한다.
- 여기서 /board/boardList는 templates 폴더 아래에 있는 /board/boardList.html을 의미한다.
- 해당 위치에 boardList.html을 생성해주자.

```
package com.board.controller;
```

```
@Controller
```

```
public class BoardController {  
    @RequestMapping("/board/list")  
    public ModelAndView boardList() {  
        ModelAndView mv = new ModelAndView("/board/boardList");  
        // 비즈니스 로직 수행  
        return mv;  
    }  
}
```



Spring boot로 게시판 개발하기

- 서비스 영역은 일반적으로 Service 인터페이스와 Service 인터페이스를 구현한 Impl 클래스로 구성된다.
- 이렇게 인터페이스와 인터페이스의 구현 클래스로 분리하여 얻는 장점은 아래와 같다.
 - 느슨한 결합을 유지하여 각 기능 간의 의존관계를 최소화한다.(기능의 변화에도 최소한의 수정으로 개발할 수 있는 유연함을 갖는다.)
 - 모듈화를 통해 어디서든 사용할 수 있어, 재사용성이 높아진다.
 - 스프링의 IoC, DI을 이용하여 관리할 수 있다.

```
package com.board.service;
```

```
public interface BoardService {  
    List<BoardDTO> selectBoardList() throws Exception;  
}
```



Spring boot로 게시판 개발하기

- Service 인터페이스 구현 클래스는 비즈니스 로직을 처리하는 서비스 클래스임을 나타내는 어노테이션 @Service를 명시한다.
- 인터페이스 구현을 나타내는 implements도 명시하여, 추상 메소드를 Override해준다.
- 메소드 내부에서 사용자 요청을 처리하는 비즈니스 로직을 수행하게 된다.

```
package com.board.service;
```

```
@Service
```

```
public class BoardServiceImpl implements BoardService {
```

```
    @Override
```

```
    public List<BoardDTO> selectBoardList() throws Exception {
```

```
        // 사용자 요청을 처리하기 위한 비즈니스 로직(리스트 가져오기) 구현
```

```
        return null;
```

```
    }
```

```
}
```




Spring boot로 게시판 개발하기

- Service 인터페이스 구현 클래스는 비즈니스 로직을 처리하는 서비스 클래스임을 나타내는 어노테이션 @Service를 명시한다.
- 인터페이스 구현을 나타내는 implements도 명시하여, 추상 메소드를 Override해준다.
- 메소드 내부에서 사용자 요청을 처리하는 비즈니스 로직을 수행하게 된다.

```
package com.board.service;
```

```
@Service
```

```
public class BoardServiceImpl implements BoardService {
```

```
    @Override
```

```
    public List<BoardDTO> selectBoardList() throws Exception {
```

```
        // 사용자 요청을 처리하기 위한 비즈니스 로직(리스트 가져오기) 구현
```

```
        return null;
```

```
    }
```

```
}
```



Spring boot로 게시판 개발하기

- BoardController에서 서비스 빈(Beans)을 연결해주어야 하는데, @Autowired 어노테이션이 필요한 의존 객체의 타입에 해당하는 빈을 찾아 자동으로 주입해준다.

```
package com.board.controller;
```

```
@Controller
```

```
public class BoardController {
```

```
    @Autowired
```

```
    private BoardService bs;
```

```
    @RequestMapping("/board/list")
```

```
    public ModelAndView boardList() throws Exception {
```

```
        ModelAndView mv = new ModelAndView("/board/boardList");
```

```
        // 비즈니스 로직 수행
```

```
        List<BoardDTO> boardList = bs.selectBoardList();
```

```
        mv.addObject("list", boardList);
```

```
        return mv;
```

```
    }
```

```
}
```



Spring boot로 게시판 개발하기

- Spring boot에서는 일일이 DAO를 만들지 않고, Mapper 인터페이스를 만들기만 하면 된다.
- @Mapper 어노테이션을 붙여 Mybatis의 매퍼 인터페이스임을 선언한다.
- Mapper 인터페이스에서 작성된 추상 메소드의 이름이 자동으로 XML과 매핑된다.

```
package com.board.mapper;
```

```
@Mapper
```

```
public interface BoardMapper {  
    List<BoardDTO> selectBoardList() throws Exception;  
}
```



Spring boot로 게시판 개발하기

- Service에서 BoardMapper를 통해 데이터베이스에서 접근할 수 있도록 @Autowired 어노테이션으로 자동 주입을 해주어야 한다.

```
package com.board.service;
```

```
@Service
```

```
public class BoardServiceImpl implements BoardService {
```

```
    @Autowired
```

```
    private BoardMapper boardMapper;
```

```
    @Override
```

```
    public List<BoardDTO> selectBoardList() throws Exception {
```

```
        // 사용자 요청을 처리하기 위한 비즈니스 로직(리스트 가져오기) 구현
```

```
        List<BoardDTO> boardList = boardMapper.selectBoardList();
```

```
        return boardList;
```

```
    }
```

```
}
```



Spring boot로 게시판 개발하기

- Service에서 BoardMapper를 통해 데이터베이스에서 접근할 수 있도록 @Autowired 어노테이션으로 자동 주입을 해주어야 한다.

```
package com.board.service;
```

```
@Service
```

```
public class BoardServiceImpl implements BoardService {
```

```
    @Autowired
```

```
    private BoardMapper boardMapper;
```

```
    @Override
```

```
    public List<BoardDTO> selectBoardList() throws Exception {
```

```
        // 사용자 요청을 처리하기 위한 비즈니스 로직(리스트 가져오기) 구현
```

```
        List<BoardDTO> boardList = boardMapper.selectBoardList();
```

```
        return boardList;
```

```
    }
```

```
}
```



Spring boot로 게시판 개발하기

- Service에서 BoardMapper를 통해 데이터베이스에서 접근할 수 있도록 @Autowired 어노테이션으로 자동 주입을 해주어야 한다.

```
package com.board.service;
```

```
@Service
```

```
public class BoardServiceImpl implements BoardService {
```

```
    @Autowired
```

```
    private BoardMapper boardMapper;
```

```
    @Override
```

```
    public List<BoardDTO> selectBoardList() throws Exception {
```

```
        // 사용자 요청을 처리하기 위한 비즈니스 로직(리스트 가져오기) 구현
```

```
        List<BoardDTO> boardList = boardMapper.selectBoardList();
```

```
        return boardList;
```

```
    }
```

```
}
```



Spring boot로 게시판 개발하기

- src/main/resources에 XML파일을 생성해서 BoardMapper 인터페이스와 XML 파일 내 쿼리가 매칭되어 사용될 수 있도록 한다.
- src/main/resources 디렉토리에 mapper 폴더를 두고, 그 밑에 board 폴더를 생성한다. 그리고 그 하위에 BoardMapper.xml 파일을 생성하자.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
```

```
<mapper namespace="com.board.mapper.BoardMapper">
  <select id="selectBoardList" resultType="com.board.dto.BoardDTO">
    SELECT
      id, title, hit, created_at
    FROM board_tbl
    WHERE is_deleted = "N"
  </select>
</mapper>
```



Spring boot로 게시판 개발하기

- 매퍼 파일(XML)의 namespace는 BoardMapper인터페이스의 경로를 명시해주어야한다.
- SELECT 태그의 id 값은 BoardMapper 메소드의 이름과 동일해야한다.
- resultType의 값은 해당 쿼리의 실행 결과의 형식을 의미한다.
- 만약 메소드에 매개변수가 있다면, 입력되는 파라미터가 형식을 parameterType으로 지정해주면 된다.
- application.properties 또는 application.yml에 매퍼 파일(XML) 경로를 알려주기만 하면 이제 자동으로 매칭이 된다.

`mybatis.mapper-locations=mapper/**/*.xml`



Spring boot로 게시판 개발하기

- src/test/java테스트 코드 작성해보기

```
package com.board.service;
```

```
@MybatisTest
```

```
@AutoConfigureTestDatabase(replace = AutoConfigureTestDatabase.Replace.NONE)
```

```
public class BoardServiceTest {
```

```
    @Autowired
```

```
    private BoardMapper boardMapper;
```

```
    @Test
```

```
    @DisplayName("Select All Board List Test")
```

```
    public void selectMyBatisTest() throws Exception {
```

```
        List<BoardDTO> boardList = boardMapper.selectBoardList();
```

```
        assertThat(boardList.size()).isEqualTo(0);
```

```
    }
```

```
}
```



타임리프

- 타임리프는 템플릿 엔진(HTML 상에 데이터를 넣어 보여주는 도구)이다.
- 템플릿 엔진은 각각 문법이 미묘하게 달라, 템플릿 엔진마다 문법을 새로 배워야 하는데, 대부분의 구조는 비슷하다.
- 대표적인 템플릿 엔진은 JSP, 타임리프, 프리마커 등이 있는데, 스프링에서 공식 지원하고 있는 타임리프를 사용하겠다.
- 타임리프의 문법은 직관적이므로 쉽게 배울 수 있다.
- 타임리프의 표현식은 아래와 같다.

표현식	설명
<code>\${ }</code>	변수의 값 표현
<code>#{ }</code>	속성 파일 값 표현
<code>@{ }, </code>	URL 표현
<code>*{ }</code>	선택한 변수의 표현 (th:object)



타임리프

표현식	설명	예제
th:text	텍스트를 표현할 때 사용	th:text="\${person.name}"
th:utext	텍스트를 표현할 때 사용 (HTML 태그 사용 가능)	th:utext="\${person.name}"
th:속성	속성(src, value, ...) 표현할 때 사용	th:속성명="\${person.img}"
th:each	컬렉션을 반복할 때 사용	th:each="person : \${persons}"
th:switch, th:case	switch 조건문 작성	th:switch="\${persons.age}" th:case="10"
th:if	조건이 true인 경우 표시	th:if="\${person.age} >= 20"
th:unless	조건이 false인 경우 표시	th:if="\${person.age} >= 20"
th:href	이동 경로	th:href="@{/persons(id=\${person.id})}"
th:with	변수값으로 지정	th:with="name=\${person.name}"
th:object	선택한 객체로 지정	th:object="\${person}"
th:classappend	클래스 동적 추가	th:classappend="\${person.age == 20} ? 'on'"



Spring boot로 게시판 개발하기

- boardList.html에서 게시물 목록을 디자인해보자.

```
<table class="table table-striped">
  <thead>
    <tr>
      <th scope="col" class="text-center">글번호</th>
      <th scope="col">제목</th>
      <th scope="col" class="text-center">조회수</th>
      <th scope="col" class="text-center">작성일</th>
    </tr>
  </thead>
  <tbody>
    <tr th:if="${list.size() > 0}" th:each="item : ${list}">
      <td class="text-center" th:text="${item.id}"></td>
      <td th:text="${item.title}"></td>
      <td class="text-center" th:text="${item.hit}"></td>
      <td class="text-center" th:text="${#temporals.format(item.createdAt, 'yyyy-MM-dd')}"></td>
    </tr>
    <tr th:unless="${list.size() > 0}">
      <td colspan="4" class="text-center">게시물 없음</td>
    </tr>
  </tbody>
</table>
```



Spring boot로 게시판 개발하기

- 우리가 사용하는 프로그램들은 다양한 로그를 남기고 있다.
- 개발을 할때 로깅은 개발자에게 여러 정보를 전달해주는 역할을 한다.
- `System.out.println()`으로 로그를 찍게 되면, 상세한 로그를 찍을 수 없고 로그 관리도 어려워진다.
- 스프링 부트에서는 Slf4j를 통해 log4j2나 Logback와 같은 로깅 프레임워크를 사용한다.



Spring boot로 게시판 개발하기

- 우리가 사용하는 프로그램들은 다양한 로그를 남기고 있다.
- 개발을 할때 로깅은 개발자에게 여러 정보를 전달해주는 역할을 한다.
- `System.out.println()`으로 로그를 찍게 되면, 상세한 로그를 찍을 수 없고 로그 관리도 어려워진다.
- 스프링 부트에서는 Slf4j를 통해 log4j2나 Logback와 같은 로깅 프레임워크를 사용한다.
- 로그 레벨은 아래와 같이 나뉘 수 있다.
 1. FATAL: 심각한 에러
 2. ERROR: 에러가 일어났을 때 사용, 개발자가 의도하지 않은 에러
 3. WARN: 에러는 아니지만 주의할 필요한 로그에 사용
 4. INFO: 운영에 참고할만한 사항 또는 중요 정보를 나타낼 때 사용
 5. DEBUG: 개발 단계에서 사용하며, 일반 정보를 나타낼 때 사용
 6. TRACE: 모든 레벨에 대한 로깅이 추적



Spring boot로 게시판 개발하기

```
@Slf4j
@Controller
public class BoardController {
    @Autowired
    private BoardService bs;

    @RequestMapping("/board/list")
    public ModelAndView boardList() throws Exception {
        String data = "로그 출력 연습";
        Log.trace("trace: {}", data);
        Log.debug("debug: {}", data);
        Log.info("info: {}", data);
        Log.warn("warn: {}", data);
        Log.error("error: {}", data);

        ModelAndView mv = new ModelAndView("board/boardList");
        // 비즈니스 로직 수행
        List<BoardDTO> boardList = bs.selectBoardList();
        mv.addObject("list", boardList);
        return mv;
    }
}
```



Spring boot로 게시판 개발하기

- 간단한 설정은 application.properties 또는 application.yml에서 할 수 있지만 상세한 설정을 위해서는 logback-spring.xml을 작성해야 한다.

```
logging.level.com.board.controller=TRACE
```




Spring boot로 게시판 개발하기

- build.gradle에 아래 라이브러리를 설치하자

```
implementation 'org.bgee.log4jdbc-log4j2:log4jdbc-log4j2-jdbc4.1:1.16'
```

```
spring.application.name=board  
spring.datasource.driver-class-name=net.sf.log4jdbc.sql.jdbcapi.DriverSpy  
spring.datasource.url=jdbc:log4jdbc:mysql://localhost:3306/board_db  
spring.datasource.username=root  
spring.datasource.password=1313
```

```
mybatis.configuration.map-underscore-to-camel-case=true  
mybatis.mapper-locations=mapper/**/*.xml
```

```
logging.level.jdbc.sqlonly=INFO  
logging.level.jdbc.resultsettable=INFO  
logging.level.jdbc.sqltiming=off  
logging.level.jdbc.resultset=off  
logging.level.jdbc.audit=off  
logging.level.jdbc.connection=off
```



Spring boot로 게시판 개발하기

- MySQL드라이버 경로 문제가 출력되는 경우에는 src/main/resources에 log4jdbc.log4j2.properties를 생성해서 아래와 같이 작성한다.

```
log4jdbc.auto.load.popular.drivers=false  
log4jdbc.drivers=com.mysql.cj.jdbc.Driver
```



Spring boot로 게시판 개발하기

- [실습] 게시글 등록 기능을 만들어보자.

```
<div class="container mt-5">
  <h2 class="mb-4 text-center">게시글 등록</h2>
  <!-- 게시글 등록 폼 -->
  <form action="/board/insert" method="post">
    <div class="mb-3">
      <label for="title" class="form-label">제목</label>
      <input type="text" class="form-control" id="title" name="title" required>
    </div>
    <div class="mb-3">
      <label for="content" class="form-label">내용</label>
      <textarea class="form-control" id="content" name="content" rows="10" required></textarea>
    </div>
    <div class="text-center">
      <button type="submit" class="btn btn-primary">저장</button>
    </div>
  </form>
</div>
```



Spring boot로 게시판 개발하기

- [실습] 게시글 등록 기능을 만들어보자.

```
<div class="container mt-5">
  <h2 class="mb-4 text-center">게시글 등록</h2>
  <!-- 게시글 등록 폼 -->
  <form action="/board/insert" method="post">
    <div class="mb-3">
      <label for="title" class="form-label">제목</label>
      <input type="text" class="form-control" id="title" name="title" required>
    </div>
    <div class="mb-3">
      <label for="content" class="form-label">내용</label>
      <textarea class="form-control" id="content" name="content" rows="10" required></textarea>
    </div>
    <div class="text-center">
      <button type="submit" class="btn btn-primary">저장</button>
    </div>
  </form>
</div>
```



Spring boot로 게시판 개발하기

- [실습] 게시글 등록 기능을 만들어보자.

```
@RequestMapping("/board/insert")
public String boardInsert(BoardDTO boardDTO) throws Exception {
    // 비즈니스 로직 수행
    System.out.println(boardDTO);
    return "redirect:/board/list";
}
```



Spring boot로 게시판 개발하기

- [실습] 게시글 상세보기 기능을 만들어보자.

```
<tr th:if="${list.size() > 0}"
    th:each="item : ${list}"
    th:onclick="/location.href='/board/detail?id=${item.id}'/"
    role="button">

    <td class="text-center" th:text="${item.id}"></td>
    <td th:text="${item.title}"></td>
    <td class="text-center" th:text="${item.hit}"></td>
    <td class="text-center" th:text="${#temporals.format(item.createdAt, 'yyyy-MM-dd')}"></td>

</tr>
```



Spring boot로 게시판 개발하기

- [실습] 게시글 상세보기기능을 만들어보자.

```
@RequestMapping("/board/detail")
    public ModelAndView boardDetail(@RequestParam("id") int id) throws Exception {
        ModelAndView mv = new ModelAndView("board/boardDetail");
        // 비즈니스 로직 수행
        return mv;
    }
```



Spring boot로 게시판 개발하기

```
<div class="container mt-5">
  <h2 class="mb-4 text-center">게시물 상세보기</h2>
  <div class="mb-3">
    <label class="form-label"><strong>글번호:</strong></label> <p th:text="${board.id}"></p>
  </div>
  <div class="mb-3">
    <label class="form-label"><strong>조회수:</strong></label> <p th:text="${board.hit}"></p>
  </div>
  <div class="mb-3">
    <label class="form-label"><strong>작성자:</strong></label> <p th:text="${board.creator}"></p>
  </div>
  <div class="mb-3">
    <label class="form-label"><strong>작성일:</strong></label>
    <p th:text="${#temporals.format(board.createdAt, 'yyyy-MM-dd')}"></p>
  </div>
  <div class="mb-3">
    <label class="form-label"><strong>제목:</strong></label>
    <div>
      <input type="text" name="title" class="form-control" th:value="${board.title}" required />
    </div>
  </div>
  <div class="mb-3">
    <label class="form-label"><strong>내용:</strong></label>
    <div>
      <textarea title="내용" name="contents" class="form-control" rows="10" th:text="${board.contents}"></textarea>
    </div>
  </div>
  <input type="hidden" name="id" th:value="${board.id}" />
</div>
```




Spring boot로 게시판 개발하기

- [실습] 게시글 수정하기 및 삭제하기 기능을 만들어보자.

```
<div class="container mt-5">
  <form action="/board/update" method="post">
    <h2 class="mb-4 text-center">게시물 상세보기</h2>

    [중간 생략]

    <input type="hidden" name="id" th:value="${board.id}" />
    <div class="text-center">
      <a href="/board/list" class="btn btn-primary">목록으로 돌아가기</a>
      <button type="submit" class="btn btn-primary">수정하기</button>
      <button type="button" class="btn btn-danger"
        onclick="if(confirm('정말로 삭제하시겠습니까?')) {
          document.getElementById('deleteForm').submit();
        }">삭제하기</button>
    </div>
  </form>
  <!-- 삭제를 위한 폼 -->
  <form id="deleteForm" action="/board/delete" method="post" style="display: none;">
    <input type="hidden" name="id" th:value="${board.id}" />
  </form>
</div>
```