스프링부트란?

목차

---

1. 스프링 부트란?

- 자바 기반의 애플리케이션 프레임워크
- 목적에 따라 다양한 프로젝트 제공 → 이 중 하나가 스프링 부트!

1.1 스프링 프레임워크

- 스프링 프레임워크는 자바에서 가장 많이 사용하는 프레임워크
- 오픈 소스 경량급 애플리케이션 프레임워크
- 스프링의 핵심 가치

💡 애플리케이션 개발에 필요한 기반을 제공해서 개발자가 비즈니스 로직 구현에만 집중할 수 있게끔 하는 것

1.1.1 제어 역전(Inversion of Control, IoC)

- 일반적인 자바 개발 = 사용하려는 객체를 선언하고 해당 객체의 의존성을 생성한 후 객체에서 제공하는 기능 사용
- 스프링은 IoC를 특징으로 함
- IoC = 사용할 객체를 직접 생성하지 않고 객체의 생명주기 관리를 외부(스프링 컨테이너 or IoC 컨테이너)에 위임
- 이를 통해 의존성 주입(DI), 관점 지향 프로그래밍(AOP)이 가능해짐

1.1.2 의존성 주입(Dependency Injection)

- 제어 역전 방법 중 하나
- 사용할 객체를 직접 생성하지 않고 외부 컨테이너가 생성한 객체를 주입받아 사용하는 방식
- 스프링에서 의존성 주입 받는 방법
  - **생성자를 통한 의존성 주입**

```
@RestController
public class DIController {

  MyService myService;

  @Autowired
  public DIController(MyService myService) {
    this.myService = myService;
  }

  @GetMapping("/di/hello")
  public String getHello() {
    return myService.getHello();
  }

}
```

    - 가장 권장하는 방식

        - 레퍼런스 객체 없이 객체를 초기화할 수 없게 설계할 수 있기 때문
    - 필드 객체 선언을 통한 의존성 주입

```
@RestController
public class FieldInjectionController {

  @Autowired
  private MyService myService;

}
```

    - setter 메서드를 통한 의존성 주입

```
@RestController
public class SetterInjectionController {

  Myservice myService;

  @Autowired
  public void setMyService(MyService myService) {
    this.myService = myService;
  }

}
```

1.1.3 관점 지향 프로그래밍 (Aspect-Oriented Programming)

- 스프링의 매우 중요한 특징
- OOP(추상화, 캡슐화, 상속, 다형성)를 돕는 개념
- 관점을 기준으로 묶어 개발하는 방식 → 핵심 기능과 부가 기능으로 구분
    - 핵심 기능 : 비즈니스 로직을 구현하는 과정에서 비즈니스 로직이 처리하려는 목적 기능
    - 부가 기능 : 핵심 기능 로직 사이에 로깅 처리, 트랜잭션 처리
- 즉 여러 비즈니스 로직에서 반복되는 부가 기능을 하나의 공통 로직으로 처리하도록 모듈화해서 삽입하는 방식
- AOP 구현 방법
    - 컴파일 과정에 삽입하는 방식
    - 바이트코드를 메모리에 로드하는 과정에 삽입하는 방식

    - **프락시 패턴을 이용한 방식**
        - 스프링이 제공하는 방식

1.1.4 스프링 프레임워크의 다양한 모듈

1.2 스프링 프레임 워크 vs. 스프링 부트

- 스프링에서 하이버네이트를 사용하기 위해 작성하는 설정 파일 → 필요한 모듈을 추가하면 설정이 복잡해지는 문제
- 이를 해결하기 위해 나타난 스프링 부트
- 스프링 부트를 이용하면 단독으로 실행 가능한 상용 수준의 스프링 기반 애플리케이션을 손쉽게 만들 수 있다.

1.2.1 의존성 관리

- 스프링 프레임워크에서는
    - 개발에 필요한 각 모듈의 의존성을 직접 설정해야 했음
    - 호환되는 버전을 명시해야 정상 동작함
    - 연관된 다른 라이브러리의 버전도 고려해야 함
- 스프링 부트에서는
    - spring-boot-starter 이라는 의존성 제공
    - 여러 종류가 있으며, 각 라이브러리의 기능과 관련해서 자주 사용되고 서로 호환되는 버전의 모듈 조합을 제공
    - 많이 사용되는 라이브러리 여러가지를 소개하고 있네용

1.2.2 자동 설정

- 스프링 부트는 스프링 프레임워크의 기능을 사용하기 위한 자동 설정 Auto Configuration을 지원
- 자동 설정은 애플리케이션에 추가된 라이브러리를 실행하는 데 필요한 환경 설정을 알아서 찾음

```
@SpringBootApplication
public class SpringBootApplication {

  public static void main(String[] args) {
    SpringApplication.run(SpringBotApplication.class, args);
  }

}
```

- 여기에서 @ SpringBootApplication 어노테이션은
  - @ SpringBootConfiguration, @ EnableAutoConfiguration, @ ComponentScan을 합친 구성

1.2.3 내장 WAS

- 스프링 부트의 각 웹 애플리케이션에는 내장 WAS(Web Application Server)가 존재한다
- 특히 웹 애플리케이션을 개발할 때 가장 기본이 되는 의존성인 'spring-boot-starter-web'의 경우 톰캣을 내장하고 있다

1.2.4 모니터링

- 개발이 끝나고 서비스를 웅영하는 시기에는 해당 시스템이 사용하는 스레드, 메모리, 세션 등의 요소를 모니터링해야함
- 스프링 부트에는 Spring Boot Actuator라는 자체 모니터링 도구가 있음

📢 **개발에 앞서 알면 좋은 기초 지식**

🥏 **데이터베이스 연동**