스프링 부트 3
백엔드 개발자 되기 (2판)

JPA+OAuth2+JWT+AWS와 배우는 스프링 부트 3
Java 백엔드 입문자를 위한 풀 패키지

02 스프링 부트 3 시작하기

2.1 스프링과 스프링 부트

- 스프링
  - 자바 기반의 엔터프라이즈 애플리케이션
  - 엔터프라이즈 애플리케이션
    - 대규모 서비스를 뜻함
  - 서버 성능, 안정성, 보안 등을 높은 수준으로 제공
- 스프링 부트
  - 스프링의 복잡한 설정을 쉽게 만들어 출시한 일종의 스핀오프
    - 빠르게 스프링 프로젝트 설정 가능
    - 스타터를 사용하면 의존성 사용, 관리 용이

스프링 부트의 주요 특징

· 톰캣, 제티, 언더토우 같은 웹 애플리케이션 서버(web application server, WAS)가 내장되어 있어서 따로 설치를 하지 않아도 독립적으로 실행할 수 있습니다.

· 빌드 구성을 단순화하는 스프링 부트 스타터를 제공합니다.

· XML 설정을 하지 않고 자바 코드로 모두 작성할 수 있습니다.

· JAR를 이용해서 자바 옵션만으로도 배포가 가능합니다.

· 애플리케이션의 모니터링 및 관리 도구인 스프링 액츄에이터(spring actuator)를 제공합니다.

2.1 스프링과 스프링 부트(cont.)

- 차이점 1
  - 구성의 차이 : 스프링은 개발에 필요한 환경을 수동 구성, 스프링 부트는 자동 구성
- 차이점 2
  - 내장 WAS의 유무 : 스프링 부트는 처음부터 WAS를 가지고 있음(tomcat)
    - WAS : 웹 애플리케이션을 실행하기 위한 장치

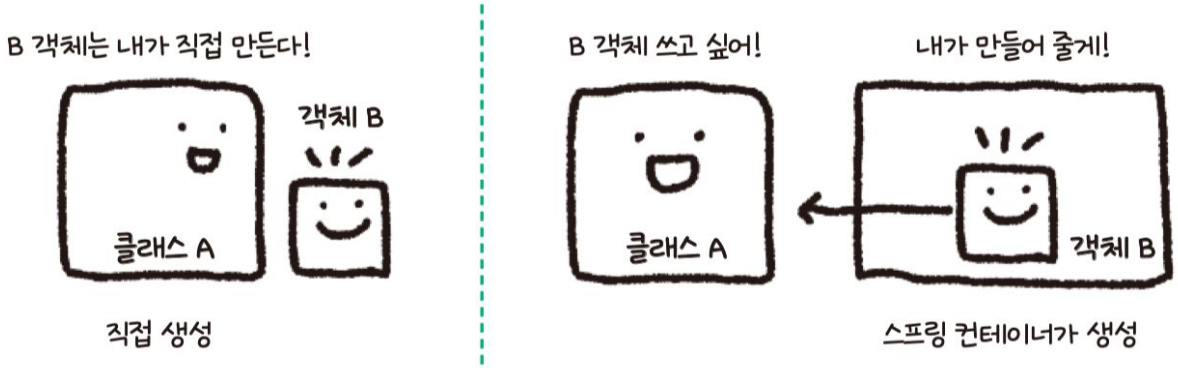
▼ 스프링과 스프링 부트 특징 비교

	스프링	스프링 부트
목적	엔터프라이즈 애플리케이션 개발을 더 쉽게 만들기	스프링의 개발을 더 빠르고 쉽게 하기
설정 파일	개발자가 수동으로 구성	자동 구성
XML	일부 파일은 XML로 직접 생성하고 관리	사용하지 않음
인메모리 데이터베이스 지원	지원하지 않음	인메모리 데이터베이스 자동 설정 지원
서버	프로젝트를 띄우는 서버(예 : 톰캣, 제티)를 별도로 수동 설정	내장형 서버를 제공해 별도의 설정이 필요 없음

2.2 스프링 콘셉트 공부하기

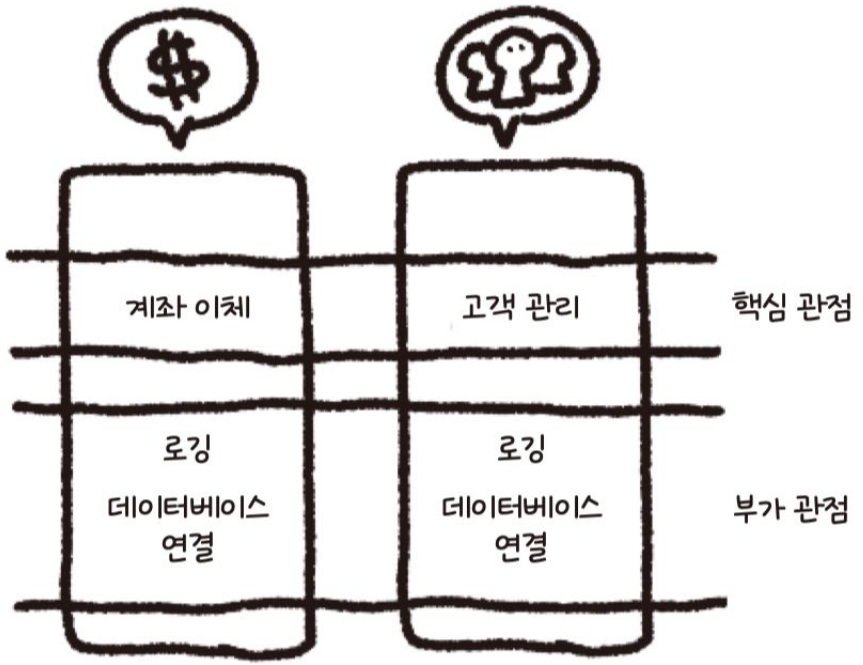
- 제어의 역전과 의존성 주입
  - 제어의 역전 = IoC = Inversion of Control
    - 객체를 직접 관리하는 것이 아닌 외부에서 관리하는 것
  - 의존성 주입 = DI = Dependency Injection
    - 빈 : 스프링 컨테이너가 관리하는 객체
    - 빈을 스프링 컨테이너로부터 주입 받아 사용(직접 객체를 생성하지 않음)

```
public class A {
  // A에서 B를 주입받음
  @Autowired
  B b;
}
```



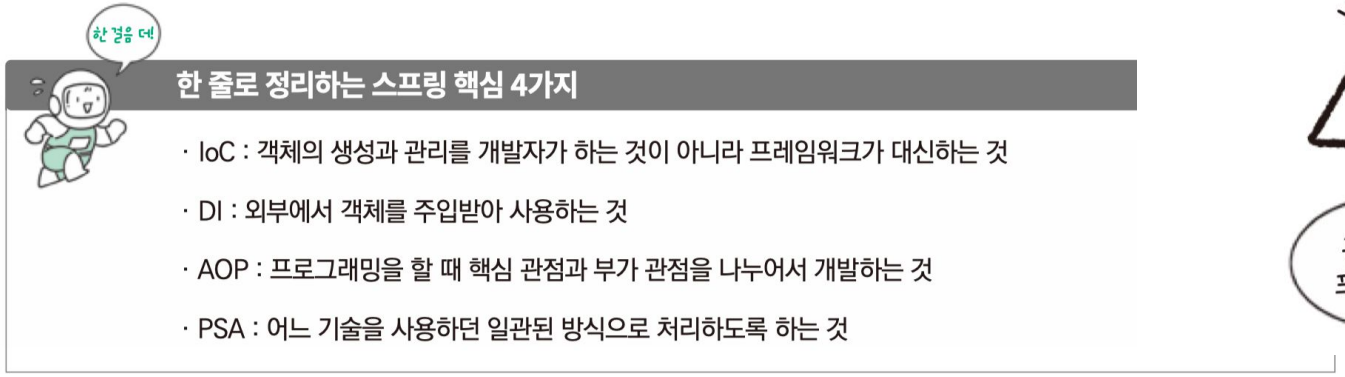
2.2 스프링 콘셉트 공부하기(cont.)

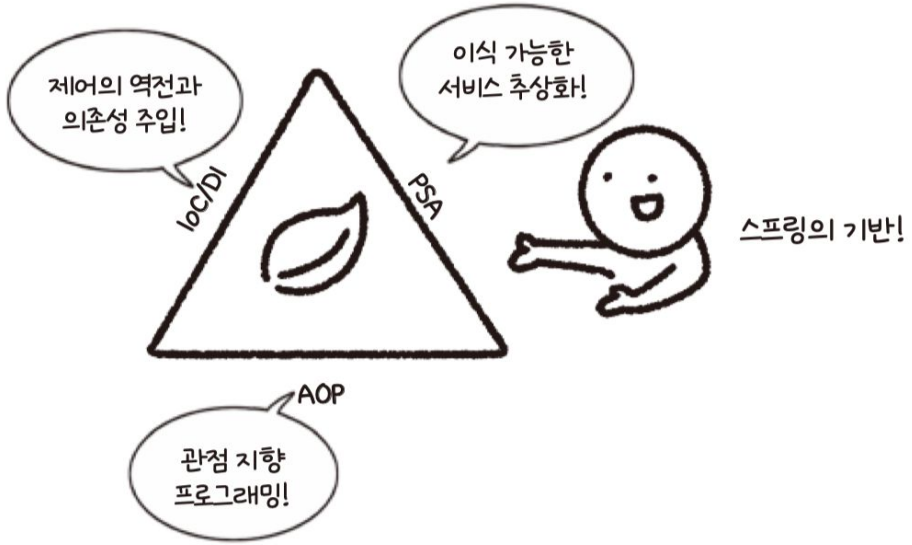
- 스프링 컨테이너
  - 스프링 안에서 동작하는 빈 생성 및 관리 주체
- 빈
  - 스프링 컨테이너가 생성하고 관리하는 객체
  - 빈 등록 방법은 여러가지(51쪽 참고)
- 관점 지향 프로그래밍
  - AOP = Aspect Oriented Programming
  - 핵심 관점, 부가 관점으로 나누어 프로그래밍하는 것



2.2 스프링 콘셉트 공부하기(cont.)

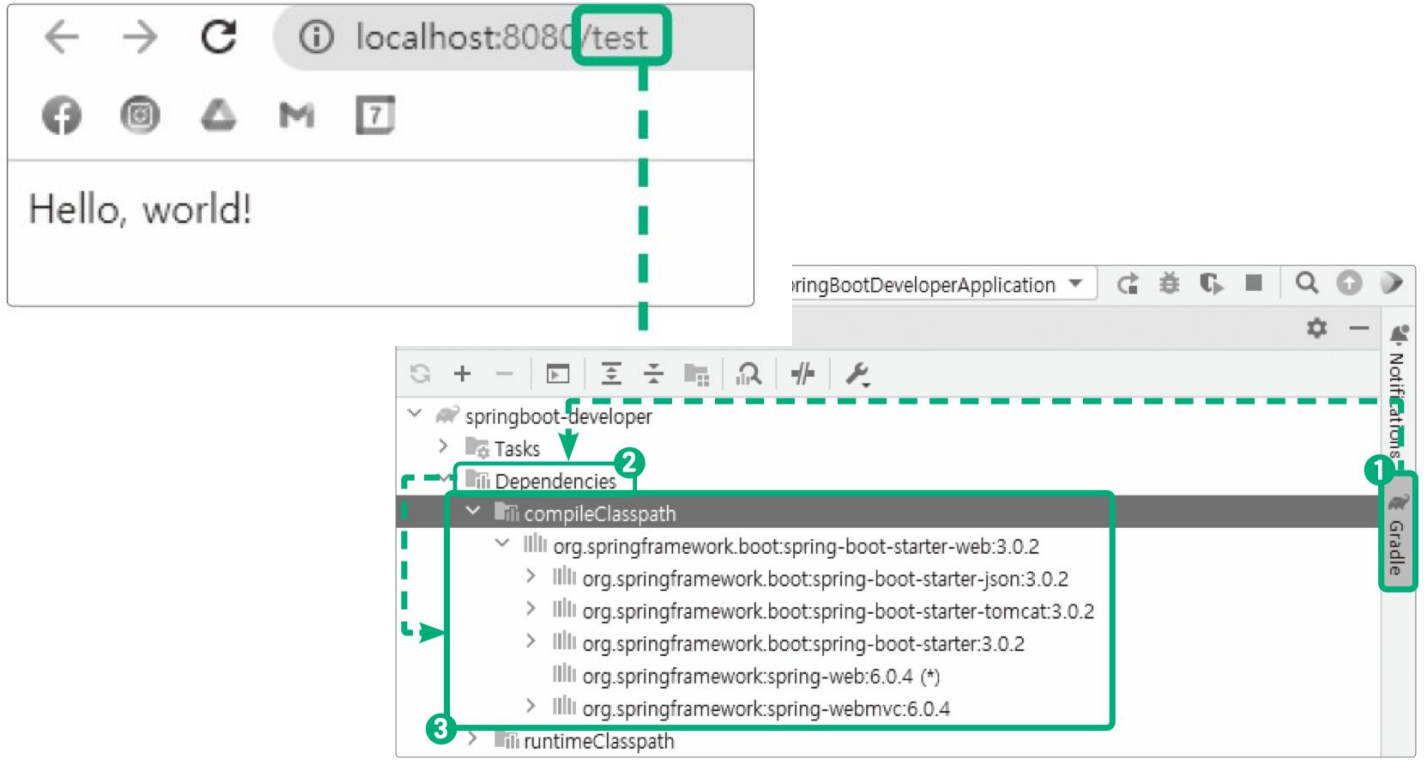
- 이식 가능한 서비스 추상화
  - PSA = Portable Service Abstraction
  - 스프링에서 제공하는 다양한 기술을 추상화해 개발자가 쉽게 사용하는 인터페이스
    - 데이터베이스를 위한 PSA : JPA, MyBatis, JDBC
    - 실행을 위한 PSA : WAS





2.3 스프링 부트 3 둘러보기

- 54쪽부터 참고
  - TestController.java
  - 서버 실행 및 확인
  - 스프링 부트 스타터 살펴보기
  - 자동 구성 항목 살펴보기



2.4 스프링 부트 3 코드 이해하기

- SpringBootDeveloperApplication.java
  - 자바의 main( ) 메서드와 같은 역할
  - 내부에 @SpringBootConfiguration, @ComponentScan, @Enabl AutoConfiguration 구성
  - 63쪽부터 참고

SpringBootDeveloperApplication.java

```
@SpringBootApplication
public class SpringBootDeveloperApplication {
  public static void main(String[] args) {
    SpringApplication.run(SpringBootDeveloperApplication.class, args);
  }
}
```

2.4 스프링 부트 3 코드 이해하기(cont.)

- TestController.java
  - @RestController : 클래스가 라우터 역할을 할 수 있게 해주는 애너테이션
  - @GetMapping(/test) : /test GET 요청을 처리
  - @RestController의 근본은 @Component 애너테이션

TestController.java

```
@RestController
public class TestController {
  @GetMapping("/test")
  public String test() {
    return "Hello, world!";
  }
}
```

/test GET 요청이 오면 test( ) 메서드 실행