스프링/스프링부트의 Concept

1. **Bean**

   스프링 애플리케이션 컨텍스트(ApplicationContext)에 의해 생성, 관리 및 조립되는 객체를 의미함. 이 객체들은 스프링 컨테이너에 의해 생성되고, 필요한 곳에서 주입되어 사용됨. (new를 이용하여 인스턴화할 필요 없음) 스프링에서는 개발자가 클래스를 작성하고, 해당 클래스를 스프링 컨테이너에 등록하여 Bean으로 관리할 수 있음.

   `@Component` 어노테이션은 스프링 프레임워크에서 Bean으로 등록할 클래스를 표시하는 데 사용.

   `@Component` 어노테이션은 스프링의 컴포넌트 스캔(`@ComponentScan`) 기능과 함께 사용되는데, `@ComponentScan`은 클래스 경로에서 `@Component` 어노테이션이 붙은 클래스를 찾아서 자동으로 빈으로 등록함.

   `@Component` 어노테이션은 스프링의 다른 어노테이션들의 상위 어노테이션으로서, `@Repository`, `@Service`, `@Controller` 등의 어노테이션들도 모두 `@Component` 어노테이션을 확장하여 사용.

2. IoC (=Inversion of Control) 제어의 역전

   특정 클래스가 내부에 외부 클래스를 명시적으로 사용할 경우, 의존성 관계가 발생할 수 있음. 그때 그 의존성 객체의 제어(=생성과 소멸)을 그 객체를 사용하는 클래스가 가지지 않고 애플리케이션 컨텍스트(ApplicationContext)가 가지는 개념을 말함.

   애플리케이션 컨텍스트(ApplicationContext)는 의존성 주입(Dependency Injection, DI)을 통해 제어의 역전을 수행함.

3. DI (=Dependency Injection) 의존성 주입

   객체가 직접 필요로 하는 의존 객체를 직접 생성하는 것이 아니라, 외부에서 의존 객체를 주입받는 것을 말함. 의존성 주입을 하는 방법은 **@Autowired**를 아래 3가지 방식으로 사용하는 것.

   - 생성자 주입 : 클래스의 생성자를 통해 주입하는 방식. (가장 권장되는 방식)
   - 세터 주입 : 클래스의 setter 메서드를 통해 주입하는 방식
   - 필드 주입 : 직접 클래스의 필드로 주입하는 방식. 필드 주입은 클래스의 내부 구현에 의존성을 노출시키고, 테스트하기 어렵게 만들 수 있음

4. AOP (=Aspect Oriented Programming) 관점 지향 프로그래밍

   프로그램을 설계할 때 각 기능을 모듈화하고 관점에 따라 나누어 프로그래밍하는 프로그래밍 패러다임임. OOP(Object-Oriented Programming)를 보완하여 코드의 재사용성과 모듈성을 향상시키는 데 도움을 줄 수 있음.

   특히 기능을 핵심관점, 부가관점, 공통관점등으로 나누고 부가, 공통 관심 사항을 애플리케이션의 핵심 비즈니스 로직과 분리함. 예를 들어, 로깅, 트랜잭션 관리, 보안, 성능 측정 등의 여러 곳에서 반복적으로 사용되는 코드들을 AOP를 사용하여 한 곳에서 관리할 수 있도록 함.

5. PSA (=Portable Service Abstraction) 이식가능한 서비스 추상화

   다양한 환경에서 동일한 코드를 실행할 수 있도록 해주는 기능. 이식 가능한 추상화는 특정 환경에 종속되지 않고 동일한 인터페이스를 통해 다양한 환경에서 작동할 수 있도록 설계됨. 이를 통해 개발자는 코드를 작성할 때 특정 환경에 의존하지 않고 환경에 관계없이 일관된 방식으로 작업할 수 있음.

   예를 들어, 스프링 프레임워크를 사용하여 JPA를 구성하고 애플리케이션을 개발할 때, 애플리케이션 코드는 JPA의 Entity 클래스와 Repository 인터페이스를 사용하여 데이터베이스와 상호 작용함. 이때 개발자는 실제 데이터베이스에 대한 세부 사항(예: SQL 문법, 데이터베이스 제품의 특정 기능)을 신경 쓰지 않고도 애플리케이션을 개발할 수 있음. 만약 데이터베이스를 변경해야 하는 경우, 코드를 수정하지 않고도 JPA 설정만 변경하여 새로운 데이터베이스에 대해 동일한 애플리케이션을 실행할 수 있음.