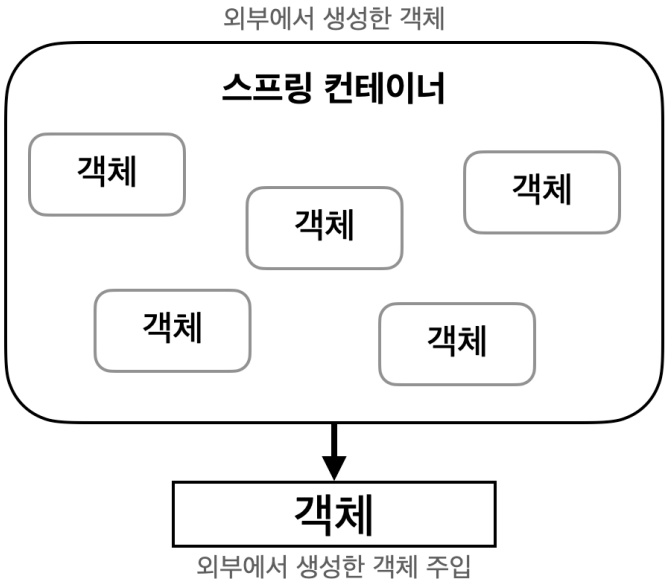
2. 스프링프레임워크기본개념1



스프링 컨테이너와 의존성 주입 (DI)

1. 스프링 컨테이너란?

스프링 프레임워크의 핵심 요소 중 하나인 **스프링 컨테이너**는 자바 객체를 관리하는 "컨테이너"입니다. 자바 애플리케이션에서 객체를 생성하고 관리하며, **빈(Bean)** 이라고 불리는 객체들을 스프링 컨테이너가 관리합니다. 모든 스프링 애플리케이션은 스프링 컨테이너 안에서 동작하며, 이 컨테이너 안에는 자바 객체들이 생성되고 관리됩니다.

2. 스프링 빈(Bean)이란?

스프링 컨테이너에서 관리되는 자바 객체를 **빈(Bean)** 이라고 부릅니다. 개발자가 직접 객체를 생성하는 것이 아니라, 스프링 컨테이너가 객체를 대신 생성하고 관리합니다. 이렇게 스프링 컨테이너가 관리하는 자바 객체를 빈이라고 하며, 빈을 통해 자바 객체의 생성을 제어하고 관리하는 **제어의 역전(IOC: Inversion of Control)** 이 발생합니다.

3. 의존성 주입 (DI: Dependency Injection)

스프링 컨테이너는 의존성을 주입하는 역할을 합니다. **의존성 주입(DI)** 은 객체를 생성할 때 필요한 의존 객체를 스프링 컨테이너가 자동으로 주입해주는 기능입니다. 예를 들어, `Car` 객체가 `Engine` 객체를 필요로 할 때, 개발자가 직접 `Engine` 객체를 생성하여 `Car`에 주입하지 않고, 스프링 컨테이너가 `Engine` 객체를 자동으로 생성하여 `Car`에 주입해줍니다.

의존성 주입 방법은 다음과 같습니다:

1. **생성자 주입**: 객체 생성 시 필요한 의존 객체를 생성자에서 주입합니다. 유지보수가 쉽고, 코드가 간결해지는 장점이 있습니다.
2. **필드 주입**: 클래스의 필드에 직접 의존 객체를 주입하는 방식입니다. 간단하게 적용할 수 있으나, 유지보수 측면에서 생성자 주입보다 불리할 수 있습니다.

4. 싱글톤 패턴

스프링 컨테이너는 기본적으로 **싱글톤 패턴**을 사용하여 빈을 관리합니다. 즉, 동일한 빈 객체는 한 번만 생성되며, 여러 곳에서 동일한 객체를 재사용합니다. 이는 메모리 효율을 높이고, 성능을 최적화하는 데 도움이 됩니다. 싱글톤 패턴 덕분에 애플리케이션 내에서 한 번 생성된 객체를 여러 클래스에서 재사용할 수 있습니다.

5. 프로토타입 빈

싱글톤 패턴과 반대로, **프로토타입(Prototype)** 빈은 매번 새로운 객체를 생성합니다. 스프링 컨테이너에 요청할 때마다 새로운 인스턴스를 반환하며, 매번 다른 객체를 생성해야 할 때 유용합니다.

예시 코드

다음은 스프링 의존성 주입을 통한 객체 생성과 싱글톤 패턴을 활용한 예시 코드입니다:

```
@Component
public class Car {
    private final Engine engine;

    @Autowired
    public Car(Engine engine) {
        this.engine = engine;
    }

    public void drive() {
        engine.start();
        System.out.println("Driving the car!");
```

```
    }
}

@Component
public class Engine {
    public void start() {
        System.out.println("Engine started!");
    }
}
```

위 코드에서는 `Car` 클래스가 `Engine` 클래스에 의존하고 있으며, 의존성 주입을 통해 `Engine` 객체가 `Car` 클래스에 주입됩니다. 스프링 컨테이너는 이 빈 객체들을 관리하며, 싱글톤 패턴을 통해 동일한 `Engine` 객체를 재사용합니다.
