Seoul Software ACademy

웹 개발자 부트캠프 과정

SeSAC x CODINGOn

With. 팀 뤼쳐드

Spring

Spring이란?

- Java 웹 프레임워크로 Java 언어를 기반으로 함
- Java 기반의 웹 어플리케이션을 만들 수 있는 백엔드 프레임워크
- 수많은 국내 기업과 해외 기업에서 많이 사용하는 프레임워크

Spring이란?

엔터프라이즈용 Java 애플리케이션 개발을 편하게 할 수 있게 해주는
오픈소스 경량급 애플리케이션 프레임워크

Spring이란?

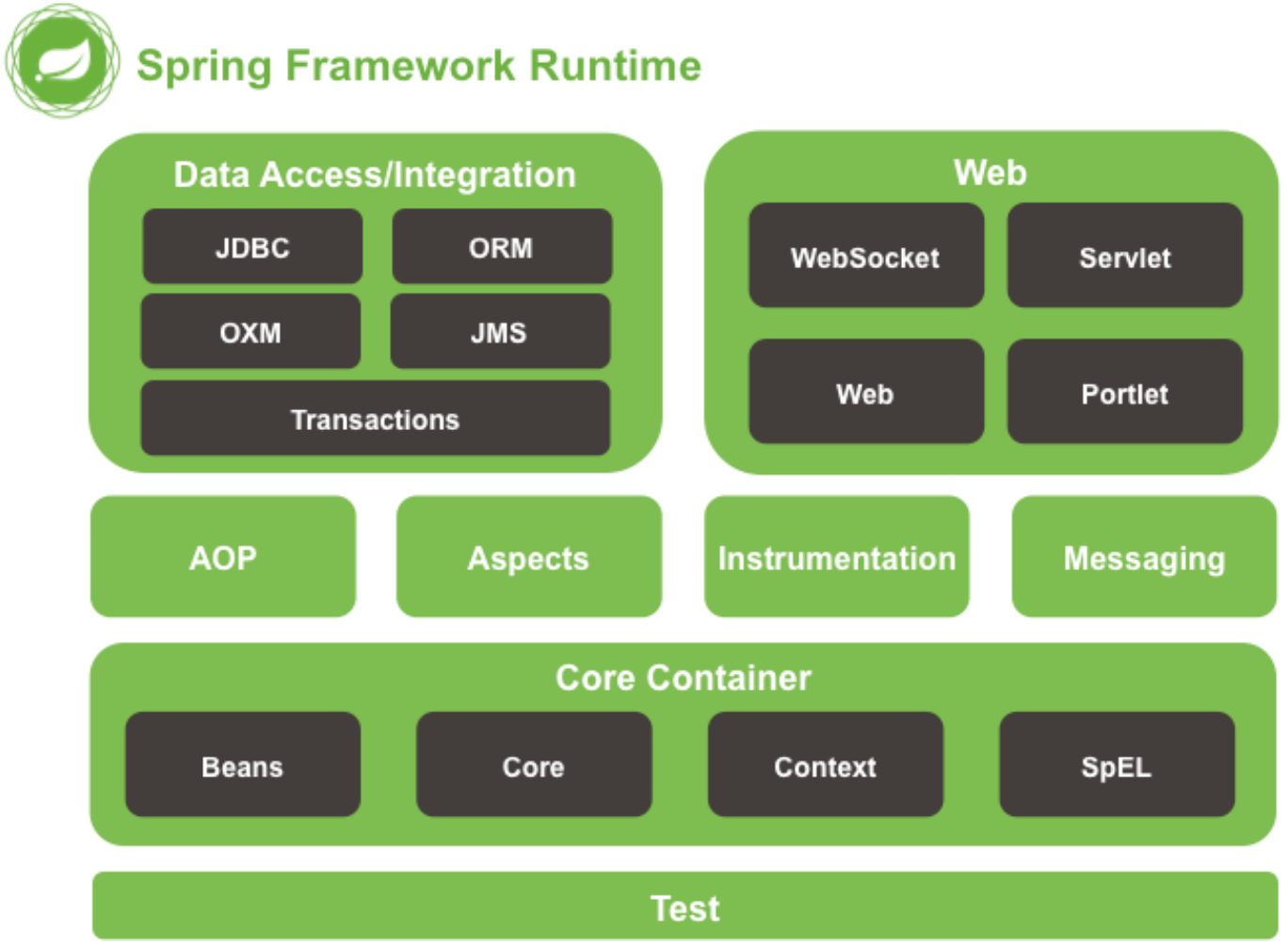
대규모의 복잡한 데이터를 관리하는 애플리케이션

엔터프라이즈용 Java 애플리케이션 개발을 편하게 할 수 있게 해주는
오픈소스 **경량급 애플리케이션 프레임워크**

기존 기술 대비 개발자가 작성할 코드가 상대적으로 단순

애플리케이션을 개발하는 데에 필요한 코드들의 뼈대를 제공

Spring Architecture



스프링 프레임워크 특징

1. **IoC** (Inversion of Control, 제어 반전)
2. **DI** (Dependency Injection, 의존성 주입)
3. **AOP** (Aspect Object Programming, 관점 지향 프로그래밍)
4. **POJO** (Pain Old Java Object 방식)

특징 1. IoC (Inversion of Control, 제어의 역전)

- 객체의 생성부터 소멸까지 개발자가 아닌 스프링 컨테이너가 대신해주는 것
- 제어권이 개발자가 아닌 IoC 에 있으며, IoC 가 개발자의 코드를 호출해 필요한 객체를 생성, 소멸해 생명주기를 관리한다

특징 1. IoC (Inversion of Control, 제어의 역전)

1. Java에서 객체 생성

   → Sample 클래스에서 new 키워드로 Apple 클래스의 객체 생성

```
class Sample {
  private Apple apple = new Apple();
}
```

2. 스프링 컨테이너가 객체 관리

   → Apple 객체의 제어권이 Sample 에 있는 것이 아닌, SampleTest 클래스에게 있음

   → **의존성을 역전시켜 제어권을 직접 갖지 않는 것이 IoC**

```
class Sample {
  private Apple apple;

  public Sample(Apple apple){
    this.apple = apple;
  }
}

class SampleTest {
  Apple apple = new Apple();
  Sample sample = new Sample(apple);
}
```

특징 2. DI (Dependency Injection, 의존성 주입)

- 구성 요소의 의존 관계가 소스코드 내부가 아닌 외부의 설정 파일을 통해 정의됨
- 즉, 외부에서 객체를 주입 받아 사용함
- DI (의존성 주입) 방법으로 IoC (제어의 역전) 를 구현

“이를 통해 코드 간의 재사용률을 높이고, 모듈 간의 결합도를 낮출 수 있음”

특징 2. DI (Dependency Injection, 의존성 주입)

방법 1. Field Injection (필드 주입)

```java
@Autowired
private FieldService fieldService;
```

방법 2. Setter Injection (수정자 주입)

```java
//Setter Injection
private SetterService setterService;

@Autowired
public void setSetterService(SetterService setterService) {
    this.setterService = setterService;
}
```

방법 3. Constructor Injection (생성자 주입)

```java
//Constructor Injection
private final ConstructorService constructorService;

@Autowired
public ExampleComponent(ConstructorService constructorService) {
    this.constructorService = constructorService;
}
```

특징 3. AOP (Aspect Object Programming, 관점 지향 프로그래밍)

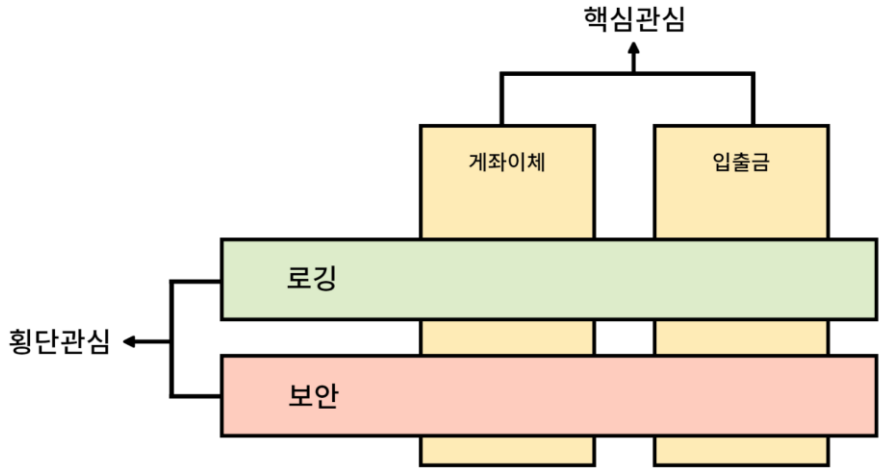
- 어떤 로직을 기준으로 “핵심 관점”, “부가 관점”으로 나누어서 보고 그 관점을 기준으로 각각 모듈화
- 기능을 비즈니스 로직과 공통 모듈로 구분한 후, 개발자의 코드 밖에서 필요한 시점에 비즈니스 로직을 삽입하여 실행되도록 함

“여러 객체에 공통으로 적용할 수 있는 기능을 구분함으로써 재사용성을 높여주는 기법”

특징 3. AOP (Aspect Object Programming, 관점 지향 프로그래밍)

- 계좌 이체와 입출금 로직을 처리할 때 공통적으로 로깅과 보안 작업을 수행해야 한다
- 일반적으로 이 경우, 공통의 코드를 두 개의 로직에 모두 넣고 사용한다
- AOP는 공통 관심(로깅, 보안)을 따로 빼내어 객체 별로 처리하는 것이 아닌 관점 별로 외부에서 접근해 사용하도록 만든다

"따라서 개발자는 핵심 관점 코드에만 집중할 수 있다!"



특징 4. POJO (Plain Old Java Object, 단순한 자바 오브젝트)

- 다른 기술을 사용하지 않고 순수 Java만을 통해서 생성한 객체
- ex. 필드, getter, setter만 존재하는 기본적인 Java 오브젝트

```
class Person {
    private String name;

    public String getName(){
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

특징 4. POJO (Plain Old Java Object, 단순한 자바 오브젝트)

- 왜 중요할까?
  - 외부 라이브러리를 import 해서 특정 객체를 사용한다면? (즉, POJO 가 아닌 객체)
  - 해당 기술이 deprecated (더 이상 사용되지 않음) 되거나 새로운 기술로 변경되면 관련 코드를 모두 변경해야 하는 번거로움
  - 해당 객체가 외부 라이브러리에 의존적인 상황 -> 좋지 않음

"객체 지향적인 원리에 충실하면서
환경과 기술에 종속되지 않고 필요에 따라 재활용될 수 있는 방식으로 설계된 오브젝트"

Spring Boot란?

Spring Boot란?

- Spring 은 필요한 여러 설정 (ex 톰캣 서버 설정, XML 설정, …) 이 복잡하다는 단점!
- Spring Boot 는 쉽고 빠르게 스프링 프레임워크를 사용할 수 있는 도구

“Spring Boot 는 Spring 에 속해 있다!!”

Spring Boot 주요 특징

- WAS 내장 되어 있어 독립적으로 실행 가능
  - WAS (Web Application Server): 웹 애플리케이션 실행 장치
  - 내장된 WAS 는 톰캣, 제티 등 여러 옵션 중 선택 가능
- 스프링 부트 스타터
  - 개발에 필요한 빌드 구성을 단순화하는 스프링 부트 스타터 제공
- 애플리케이션 설정을 XML 이 아닌 Java 코드 작성 가능
  - 개발자가 더 직관적이고 유연하게 애플리케이션 설정 가능
- JAR 이용해 자바 옵션 만으로도 배포 가능
  - JAR (Java Archive) 파일: 애플리케이션과 의존 라이브러리들을 하나의 파일로 묶은 형태로 배포 및 실행 환경에서 별도의 설정이 필요 없다는 장점 (애플리케이션 이식성과 배포 과정 단순화)
  - JAR 파일로 패키징해 배포 가능

IntelliJ 설치하기

제공된 IntelliJ 설치 PDF를 참고해 IntelliJ 설치 완료하기