Chapter 05. Spring 을 조금 더 들여다 보기

스프링의 핵심

Spring

- Spring 1.0버전은 2004년 3월 출시
  지난 20년 가까지의 세월 동안 단 한번도 자바 엔터프라이즈 어플리케이션 개발의 최고의 자리를 차지

- 스프링 프레임워크의 구성은 20여가지로 구성 (https://spring.io/projects/spring-framework)
  이러한 모듈들은 스프링의 핵심기능 (DI, AOP, etc)을 제공해 주며, 필요한 모듈만 선택하여 사용 가능.

- 현재 단일 아키텍처(모놀리스) 마이크로서비스 아키텍처로 변환 중
  여기에 맞춰서 스프링도 진화하고 있는 상태.

- 여러 가지 모듈이 있지만 그 중에서 단연
  스프링 부트, 스프링 클라우드, 스프링 데이터, 스프링 배치, 스프링 시큐리티에 중점을 둔다.

Spring

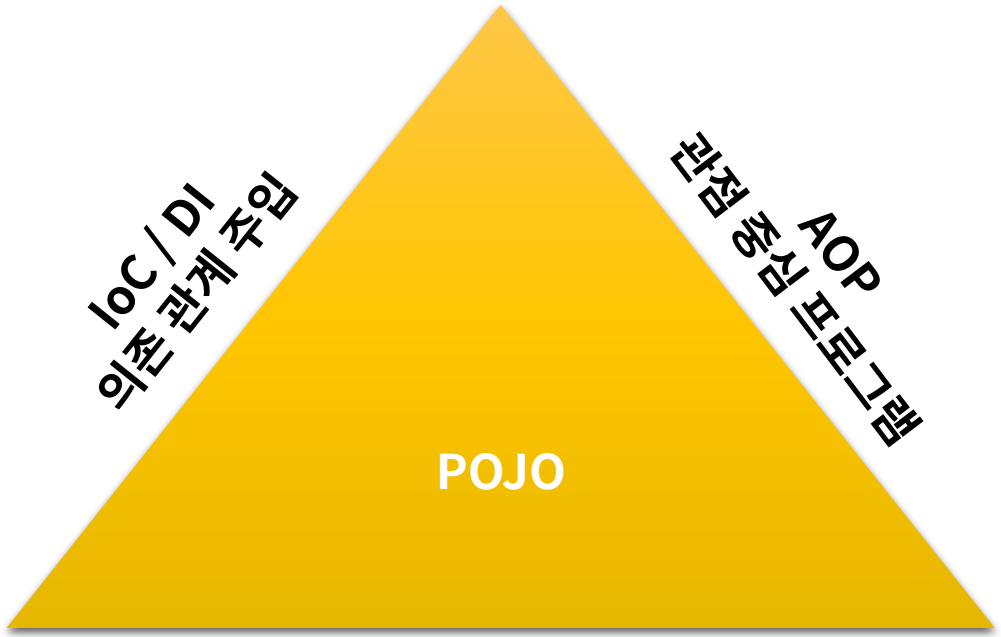
- Spring의 과제는 “테스트의 용이성” , “느슨한 결합”에 중점을 두고 개발

- 2000년대 초의 자바 EE 애플리케이션은 작성/테스트가 매우 어려웠으며, 한번 테스트 하기가 번거로웠다. 이로 인하여, 느슨한 결합이 된 애플리케이션 개발이 힘든 상태였으며, 특히 데이터베이스와 같이 외부에 의존성을 두는 경우 단위테스트가 불가능했다.

- IoC의 등장
  스프링이 다름 프레임워크와 가장 큰 차이점이 IoC를 통한 개발 진행

- AOP
  AOP를 사용하여, 로깅, 트랜잭션 관리, 시큐리티에서의 적용 등 AspectJ와 같이 완벽하게 구현된 AOP와 통합하여 사용 가능 하다

Spring



PSA
이식 가능한 추상화

Chapter 05. Spring 을 조금 더 들여다 보기

IoC / DI

IoC (Inversion of Control)

- **IoC (Inversion Of Control)**
  스프링에서는 일반적인 Java 객체를 new로 생성하여 개발자가 관리 하는 것이 아닌 Spring Container 에 모두 맡긴다.
  즉, 개발자에서 -> 프레임워크로 제어의 객체 관리의 권한이 넘어 갔음 으로 “제어의 역전” 이라고 합니다.

DI (Dependency Injection)

➢ DI 장점

- 의존성으로 부터 격리시켜 코드 테스트에 용이하다.

- DI를 통하여, 불가능한 상황을 Mock와 같은 기술을 통하여, 안정적으로 테스트 가능하다.

- 코드를 확장하거나 변경 할 때 영향을 최소화 한다 (추상화)

- 순환참조를 막을 수 있다.

Chapter 05. Spring 을 조금 더 들여다 보기

AOP

AOP (Aspect Oriented Programming)

관점지향 프로그램

스프링 어플리케이션은 대부분 특별한 경우를 제외 하고는 MVC 웹 어플리케이션에서는 Web Layer , Business Layer, Data Layer 로 정의.

- Web Layer : REST API를 제공하며, Client 중심의 로직 적용

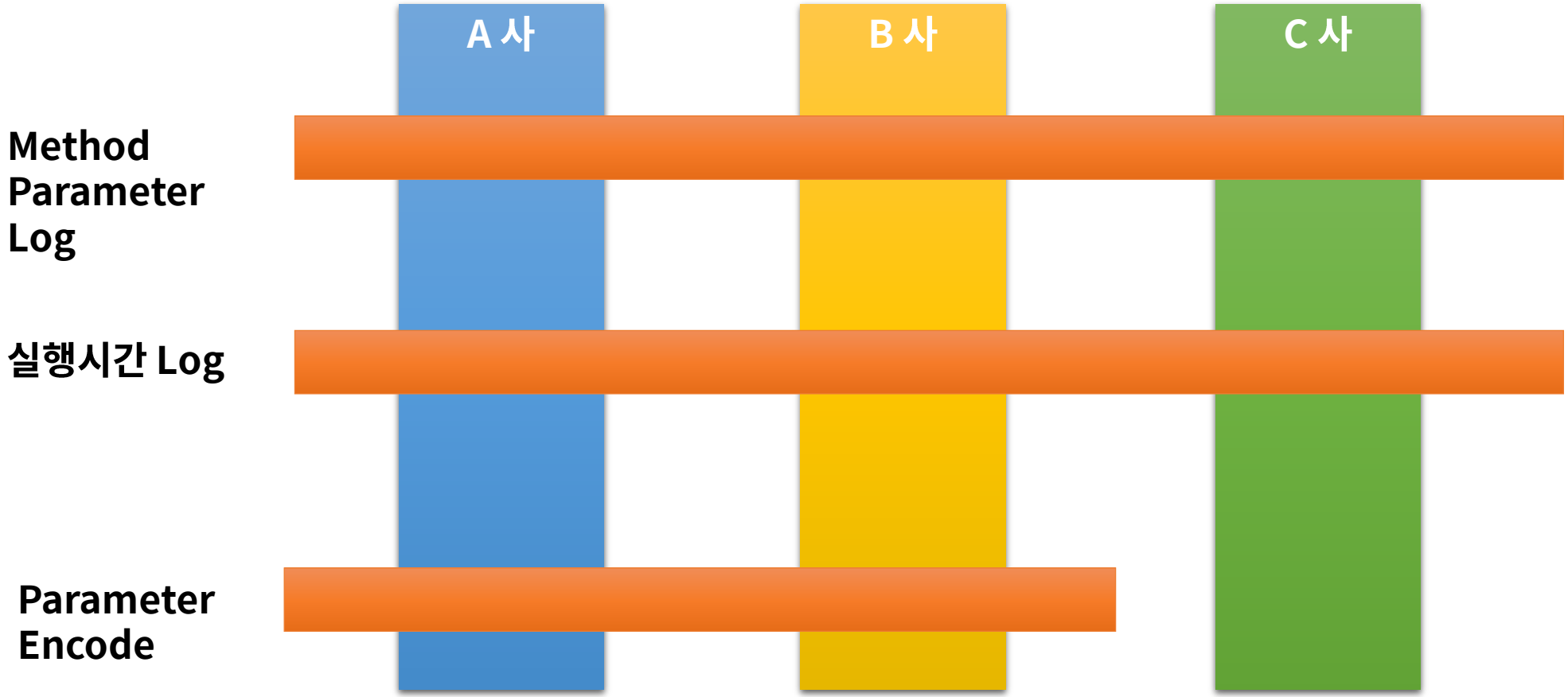
- Business Layer : 내부 정책에 따른 logic를 개발하며, 주로 해당 부분을 개발

- Data Layer : 데이터 베이스 및 외부와의 연동을 처리

주요 Annotation

Annotation	의미
@Aspect	자바에서 널리 사용하는 AOP 프레임워크에 포함되며, AOP를 정의하는 Class에 할당
@Pointcut	기능을 어디에 적용시킬지, 메소드? Annotation? 등 AOP를 적용 시킬 지점을 설정
@Before	메소드 실행하기 이전
@After	메소드가 성공적으로 실행 후, 예외가 발생 되더라도 실행
@AfterReturing	메소드 호출 성공 실행 시 (Not Throws)
@AfterThrowing	메소드 호출 실패 예외 발생 (Throws)
@Around	Before / after 모두 제어

횡단 관심



Chapter 05. Spring 을 조금 더 들여다 보기

여러 가지 Annotation

Spring Boot Annotations

Annotation	의미
@SpringBootApplication	Spring boot application 으로 설정
@Controller	View를 제공하는 controller로 설정
@RestController	REST API 를 제공하는 controller로 설정
@RequestMapping	URL 주소를 맵핑
@GetMapping	Http GetMethod URL 주소 맵핑
@PostMapping	Http PostMethod URL 주소 맵핑
@PutMapping	Http PutMethod URL 주소 맵핑
@DeleteMapping	Http DeleteMethod URL 주소 맵핑
@RequestParam	URL Query Parameter 맵핑
@RequestBody	Http Body를 Parsing 맵핑
@Valid	POJO Java class의 검증

Spring Boot Annotations

Annotation	의미
@Configration	1개 이상의 bean을 등록 할 때 설정
@Component	1개의 Class 단위로 등록 할 때 사용
@Bean	1개의 외부 library로부터 생성한 객체를 등록 시 사용
@Autowired	DI를 위한 곳에 사용
@Qualifier	@Autowired 사용시 bean이 2개 이상 일때 명시적 사용
@Resource	@Autowired + @Qualifier 의 개념으로 이해
@Aspect	AOP 적용시 사용
@Before	AOP 메소드 이전 호출 지정
@After	AOP 메소드 호출 이후 지정 예외 발생 포함
@Around	AOP 이전/이후 모두 포함 예외 발생 포함
@AfterReturning	AOP 메소드의 호출이 정상일 때 실행
@AfterThrowing	AOP시 해당 메소드가 예외 발생시 지정