AOP

| 1 | 다음 설명 중 맞는 것을 고르면?  (1) DI와 AOP는 스프링 프레임웍을 지배하는 주요한 개발 방법이다.  (2) AOP란 Aspect Oriented Programming , 즉 관점 지향 프로그래밍을 의미한다  (3) DI가 객체 간 결합도를 낮추기 위한 개발 방법이라면, AOP는 기능 간 결합도를 낮추기 위한 방법이다  (4) AOP가 결합도를 낮추려는 대상이 되는 기능은 어플리케이션내에서 사용빈도가 낮은 일회성을 대상으로  한다  (5) AOP는 스프링을 창조한 로드 존슨이 고안한 개발 방법이다 |
| --- | --- |
| 2 | 다음 설명 중 틀린 것을 고르면?  (1) 어플리케이션내의 여러 부분에서 공통적으로 사용되는 기능을 횡단적 관심사(**Cross-Cutting Concern**)라 한다  (2) 횡단적 관심사를 사용하려는 객체는, 직접 횡단적 관심사 코드를 객체 내에 포함시키기 보다는, 별도로 분리시켜 놓는 것이 유지 보수하기 좋다  (3) 별도로 분리된 횡단적 관심사는 마치 외부에서 동작을 관찰하다가, 특정 시점에 관여하기 때문에 이를 관점 지향 프로그래밍(AOP)라 한다  (4)AOP를 적용하기에 적절한 공통 업무는 로깅, 트랜잭션, 보안 등 어플리케이션에 전반적으로 사용 되는  공통 기능이다.  (5)OOP는 앞으로 AOP로 대체될 것이다. |
| 4 | 알맞는 말을 채워넣으세요   | 명칭 | 설명 | | --- | --- | |  | 여러 객체에 공통으로 적용하려는 기능 | |  | 횡단적 관심사를 메서드 호출 전, 후 등 어느 시점에 관여 시킬지를 결정하는 내용을 갖는 단위 | |  | 멤버 변수, 메서드 등 Advice를 적용할 수 있는 지점 | |  | JoinPoint 후보 들 중 Adivce 적용이 결정된 JoinPoint ,즉 현재 적용하고 있는 부분이 메서드 JoinPoint 인지, 멤버변수 JoinPoint 인지 결정된 JoinPoint | |
| 5 | 다음 설명 중 틀린 것은?  (1) 스프링의 AOP는 스프링에서만 사용되는 개발 방법이 아니라, 이미 예전부터 AOP프로그래밍을 위해 사용되던 AspectJ 라는 유명한 프로젝트를 스프링이 도입한 것이다.  (2) 특정 객체의 메서드 호출 시 Aspet가 적용될 수 있는 이유는 객체와 Advice 코드가 Weaving 될 수 있기  때문인데 이러한 Weaving 방법에는 컴파일 시 엮는 방법, 클래스 로딩시에 엮는 방법, 런타임시에 엮는  방법의 총 3가지 유형이 있다.  (3)컴파일 시 엮는 방법은 AspectJ 에서 지원하므로 스프링에서는 사용할 수 없다.  (4)클래스 로딩시 엮는 방법을 이용하면 java코드는 변경사항이 없으며 단지 이 클래스 로딩 시 공통 코드가  삽입 된 새로운 바이트 코드가 생성되는 방식이다.  (5)런타임 시에 엮는 방법은 실행 시 클래스 바이트 코드에 공통 코드를 삽입하여 새로운 바이트 코드를 생성  하는 방식이다. |
| 6 | 다음 설명 중 틀린 것은?  (1)객체와 공통 코드를 Weaving 하기 위해 스프링이 지원하는 방식은 런타임시 Weaving 하는 방식이다.  (2)런타임 시 Weaving하는 방식은 Aspect 적용 대상 객체와 공통 코드의 중간에 Proxy라 불리는 대리 객체를  이용한다.  (3)런타임 Weaving 방식은 오직 메서드가 호출될 때만 공통 코드를 적용할 수 있다.  (4)프록시 객체를 생성할 때는 대상 객체가 인터페이스를 구현했는지 여부에 따라 2가지 방법으로 프록시를  생성 한다. |
| 7 | 다음 설명 중 틀린 것은?  (1)결국, 보다 정교한 AOP 기능을 사용하려면 ApectJ를 직접 사용해야 되며 스프링에서는 AspectJ의 기능  중 메서드 차원의 JoinPoin 만을 지원하는 한계가 있다.즉 메서드 호출시에만 Aspect 을 관여시킬 수 있다  (2)특정 객체의 메서드 호출 시 Aspect 를 적용시킬 수 있는 이유는 , 타겟이 되는 객체가 구현하고 있는인터  페이스를 스프링의 프록시(Proxy) 객체가 내부적으로 똑같이 구현하고 있기 때문이다.  (3)타겟 객체가 인터페이스를 구현하지 않으면 Aspect 가 적용될 방법은 전혀 없다.  (4)타겟 객체가 인터페이스를 구현하지 않을 경우, 스프링은 CGLIB 을 이용하여 프록시 객체를 생성한다. |
| 8 | 빈 칸을 채우세요  프록시는 대상 객체와 동일한 ( A )를 구현하므로, 만일 ( A )에 정의되어 있지 않은 ( B )를  호출할 경우엔 Aspect가 적용될 수 없다. |
| 9 | 다음 설명 중 틀린 것은?  (1)AOP를 구현하는 방법에는 선언적 방법과 어노테이션을 이용하는 방법이 있다  (2)선언적 방법이라 하면 XML과 같은 설정 파일에 의한 구현 방법을 의미한다.  (3)어노테이션을 이용하면, 자바 코드내에서 AOP 설정을 명시할 수 있다.  (4)어노테이션 방법을 사용하면, AOP가 코드에 종속되지 않는 장점이 있다. |
| 10 | 어드바이스의 유형에 대해 빈칸을 채우세요   | 유형 | 설명 | | --- | --- | |  | 대상 객체의 메서드 호출 전에 Aspect 를 실행 | |  | 대상 객체의 메서드 호출 후에 Aspect 를 실행.  이때 예외가 발생하더라도 Aspect 를 무조건 실행 ( finally 처럼) | |  | 대상 객체의 메서드 호출 전, 후에 Aspect 를 실행.  이때, 예외가 발생해도 해당 예외 전, 후에 Aspect를 실행 | |  | 대상 객체의 메서드가 예외 발생 없이 무사히 실행된 이후에 Aspect 를 실행 | | After Throwing Advice | 대상 객체의 메서드 실행 시 예외가 발생하면 Aspect 를 실행 | |