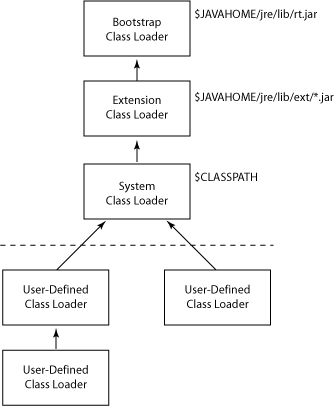
HW01. ClassLoader에 대해 조사하세요

자바 클래스로더(Java Classloader)는 자바 클래스를 자바 가상 머신(JVM)으로 동적 로드하는 자바 런타임 환경(JRE)의 일부이다. 일반적으로 클래스들은 요청 시 한 차례만 로드된다. 자바 런타임 시스템은 클래스로더 때문에 파일과 파일 시스템에 대해 알 필요가 없다. 위임(delegation)은 클래스로더에 대해 학습할 때 이해하는데 중요한 개념이다.

클래스는 코드의 명명 단위로 간주할 수 있다. 클래스 로더는 라이브러리를 위치시키고 내용물을 읽으며 라이브러리들 안에 포함된 클래스들을 읽는 역할을 한다. 로딩은 일반적으로 요청이 오면 이루어지는데, 이 말은 클래스가 프로그램에 의해 호출될 때까지 로드하지 않는다는 것을 뜻한다. 명명된 클래스는 주어진 클래스로더에 의해 한 번만 로드될 수 있다.

각 자바 클래스는 클래스 로더에 의해 로드되어야 한다.

HW02. ClassLoader의 계층구조



부트스트랩 클래스 로더

부트스트랩 클래스 로더는 <JAVA\_HOME>/jre/lib 디렉터리에 위치한 핵심 자바 라이브러리들을 불러들인다. 핵심 JVM의 일부분인 이 클래스 로더는 네이티브 코드로 작성되어 있다.

확장 클래스 로더

확장 클래스 로더는 확장 디렉터리(<JAVA\_HOME>/jre/lib/ext 또는 java.ext.dirs 시스템 속성에 지정된 기타 디렉터리)에 코드를 로드한다. sun.misc.Launcher$ExtClassLoader 클래스에 의해 구현되어 있다.

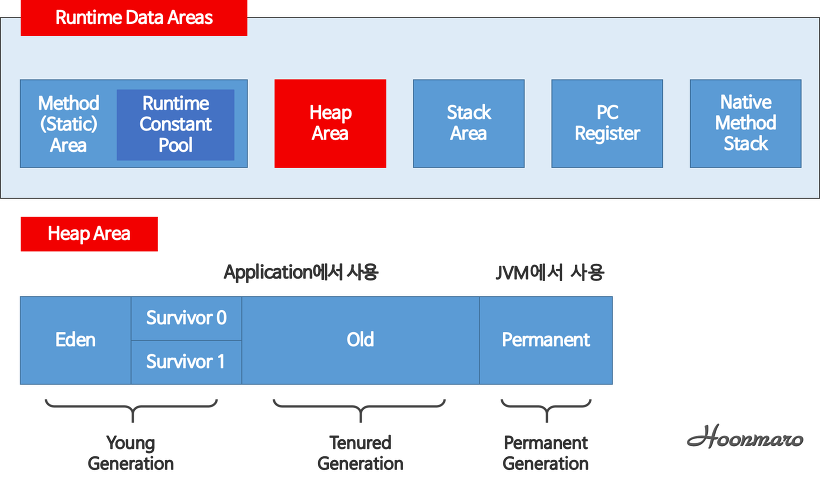
시스템 클래스 로더(일반적인 클래스로더)

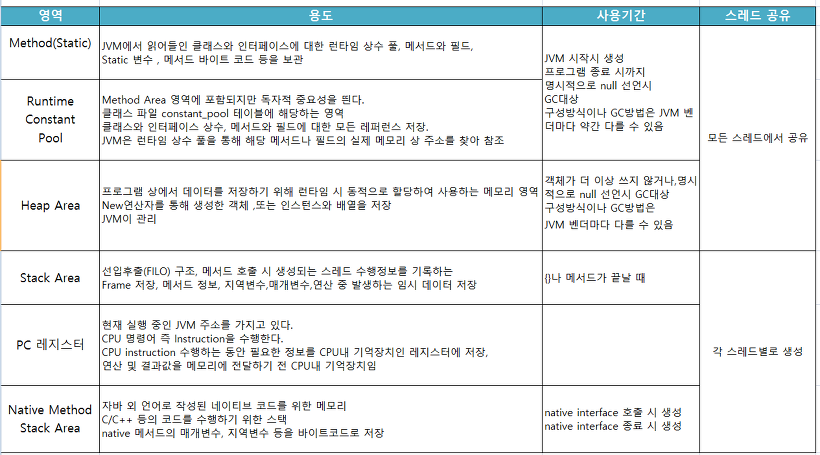
java.class.path에서 볼 수 있는 시스템 클래스 로더는 CLASSPATH 환경 변수에 매핑된다. sun.misc.Launcher$AppClassLoader 클래스에 의해 구현되어 있다.

사용자정의 클래스로더

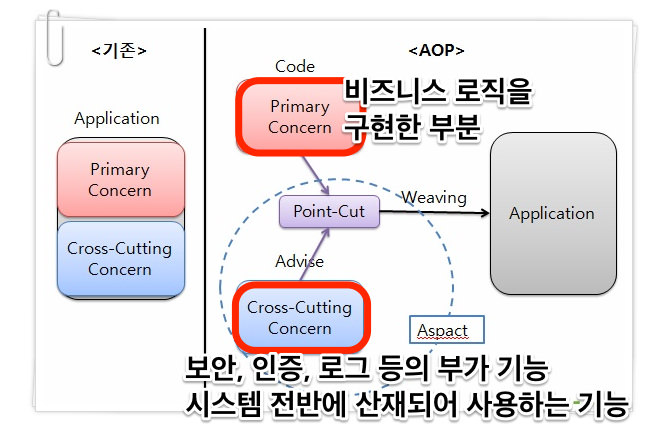
애플리케이션 사용자가 코드상에서 직접 생성하는 클래스로더 이다.

HW03. JVM 메모리 구조





HW04. AOP의 용어 정리(그림있으면 좋음)



Aspect - 공통 관심사에 대한 추상적인 명칭. 예를 들어 로깅이나 보안, 트랜잭션과 같은 기능 자체에 대한 용어

Advice - 실제로 기능을 구현한 객체

Join points - 공통 관심사를 적용할 수 있는 대상. Spring AOP에서는 각 객체의 메소드가 이에 해당

Pointcuts - 여러 메소드 중 실제 Advice가 적용될 대상 메소드

target - 대상 메소드를 가지는 객체

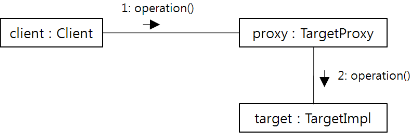
Proxy - Advice가 적용되었을 때 만들어지는 객체

Introduction - target에는 없는 새로운 메소드나 인스턴스 변수를 추가하는 기능

Weavcing - Advice와 target이 결합되어서 프록시 객체를 만드는 과정

HW05. Proxy

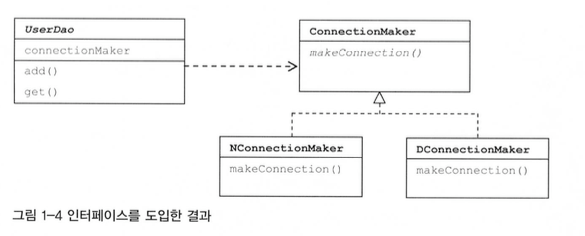
스프링은 Aspect의 적용대상이 되는 객체에 대한 프록시를 만들어 제공합니다. 비지니스 로직에 접근할 때 대상 객체로 바로 접근하는 게 아닌 프록시를 통해서 간접적으로 접근하게 됩니다. 이 과정에서 프록시는 공통 기능을 실행한 뒤 대상 객체의 실제 메서드를 호출하거나 또는 대상객체의 실제 메소드를 호출한 후에 공통기능을 실행합니다.



대상 타겟은 결국 빈 객체가 생성 되는데, 런타임시에 오브젝트 생성 설정에 따라서 스프링 컨테이너가 지정한 빈 객체 대한 프록시 객체를 생성하고, 원본 빈 객체 대신에 프록시 객체를 사용하게 합니다.

프록시 객체를 생성하는 방식은 대상 객체가 인터페이스를 구현하고 있느냐 없느냐 여부에 따라 달라집니다. 2가지 방식이 존재하는데 JDK Dynamic Proxy, CGLIB를 이용하여 프록시를 생성하는 방식이 있습니다.

JDK Dynamic Proxy

스프링은 자바 리플렉션 API가 제공하는 java.lang.reflect.Proxy를 이용하여 프록시 객체를 생성합니다. 아래 그림과 NConnectionMaker,DConnectionMaker와 같이 동일한 인터페이스를 구현하게되면 UserDao와 같은 클라이언트는 인터페이스를 통해서 필요한 메서드를 호출하게 됩니다. 하지만, *인터페이스를 기반으로 프록시 객체를 생성하기 때문에 인터페이스에 정의되어 있지 않은 메서드에 대해서는 AOP가 적용되지 않는 점에 유의해야합니다.*  


CGLIB

대상객체가 인터페이스를 구현하고 있지 않고 바로 클래스를 사용한다면, 스프링은 CGLIB를 이용하여 클래스에 대한 프록시 객체를 생성합니다. CGLIB는 대상 클래스를 상속 받아 프록시를 구현합니다. 따라서 클래스가 final인경우에는 프록시를 생성할 수 없습니다.

강제로 CGLIB를 이용하여 Proxy를 생성하는 방법

@EnableAspectJAutoProxy 어노테이션 속성으로 proxyTaretClass란 것이 있습니다. 위에서 말한대로 Spirng AOP는 Proxy class를 생성함으로 AOP를 구현 합니다. AOP의 대상이 되는 클래스가 Interface를 구현하고 있을 경우 Spring은 Interface를 이용하는 JDK Dynamic Proxy를 만들어서 AOP를 구현하고 Interface를 구현하지 않고 클래스에 대해서는 CG-LIB를 이용한 대상클래스의 상속을받아 서브클래스를 만들어 이를 Proxy삼아 AOP 구현하게 됩니다. 이 속성 값을 주지 않으면 기본값은 false이기 때문에 대상이 되는 클래스가 interface를 이용한것은 자동으로 JDK Dynamic Proxy를 이용해서 AOP를 생성하고 아닌경우에는 CG-LIB를 이용해 클래스 Proxy로 AOP를 구현하게 됩니다. 이 속성을 true로 주게되면 인터페이스 또한 CG-LIB를 이용하여 서브클래스 생성하는 방식으로 AOP의 프록시를 구현하게 됩니다.

HW06. Thread와 공용객체

쓰레드

프로세스의 자원으로 실제 작업을 실행하는 것이 쓰레드이다.

한 프로세스 내에서 동작되는 여러 실행의 흐름으로, 프로세스 내의 주소 공간이나 자원들을 대부분 공유하면서 실행된다.

Thread 사용의 장점

스레드는 프로세스내에서 각각의 스택 공간을 제외한 나머지 공간과 시스템 자원을 공유한다.

그러므로 프로세스를 이용하여 동시에 처리하던 일을 스레드로 구현할 경우 메모리 공간은 물론 시스템 자원 소모도 현격히 줄어든다.

스레드 간의 통신이 필요할 경우 별도의 자원을 이용하는 것이 아니라 전역 변수의 공간을 이용하여 데이터를 주고 받을 수 있다.

- 시스템의 처리량이 향상된다.

- 시스템의 자원 소모가 줄어든다.

- 프로그램의 응답 시간이 단축된다.

- 프로세스 간 통신 방법에 비해 스레드 간의 통신 방법이 훨씬 간단하다.

( 스레드 간의 통신 시 데이터를 주고받는 방법은 메모리 공간을 공유하므로 데이터 세그먼트, 즉 전역 변수를 이용하여 구현한다. )

(공유하는 전역 변수를 여러 스레드가 함께 사용하기 위해서는 충돌 문제가 발생할 수 있는데 이런 문제가 발생하지 않도록 동기화 문제를 해결해야 한다.)

Thread 사용의 단점

여러 개의 스레드를 이용하는 프로그램을 작성하는 경우 주의 깊게 설계해야 한다.

미묘한 시간 차나 잘못된 변수를 공유함으로써 오류가 발생할 수 있다.

싱글스레드의 경우 프로세스 내 단 하나의 스레드만 작업하기 때문에 자원을 이용하는 데에 문제가 없지만 멀티 스레드일 경우 같은 자원을 공유할 때 문제가 생길 수 있다.

그렇기 때문에 현재 작업 시 lock을 걸어 다른 스레드가 접근하지 못하게 하는 작업이 필요하다. 이 작업을 '동기화(synchronized)'라고 한다.

HW07. Gradle이 무엇인가. Maven이 무엇인가.

Gradle

Gradle은 빌드 시스템입니다. 구글이 만든 Android Studio에서 기본적인 빌드 시스템으로 사용됩니다.

▶Gradle은 진화된 빌드툴로 빌드,테스트, 배포, 개발등을 자동화 할수있다.

▶Ant의 유연성과 효과적인 빌드 툴인 Maven의 편리성을 조합하여 많은 오픈소스 프로젝트의 빌드 시스템으로 채택되고 있다.

▶Ant의 가장 큰 장점은 개발자가 자유롭게 빌드 단위(target)을 지정하고 빌드 단위간의 의존관계를 자유롭게 설정할 수 있다는 것이다.

▶Gradle은 Ant의 자유도와 Maven의 관례의 장점을 모두 흡수했다. 그리고 Ant와 Maven빌드 툴 모두 가지고 있었던 가장 큰 단점인 XML에 대한 이슈도 Groovy언어를 사용해 해결하고 있다.

Maven

주로 java 진영에서 프로젝트 빌드, 관리에 사용되는 도구이다.

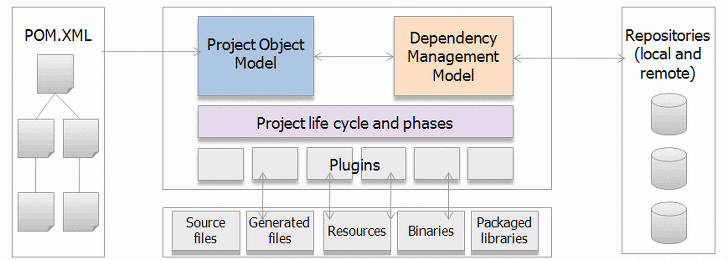
특징

빌드 절차 간소화

동일한 빌드 시스템 제공

프로젝트 정보 제공

구조



ant와의 차이점

Ant가 비교적 자유도가 높다. 전처리, 컴파일, 패키징, 테스팅, 배포 가능

Maven은 정해진 라이프사이클에 의하여 작업 수행하며, 전반적인 프로젝트 관리 기능까지 포함하고 있음. (Build Tool + Project Management)

gradle과의 차이점

XML 대신 groovy 스크립트를 사용하여 동적인 빌드 가능.

maven은 멀티프로젝트에서 상속구조인데, gradle은 주입 방식이다. 멀티프로젝트에서 gradle이 더 적합하다.