16 스레드 완전 정리

FinalMan 카페매니저
2016.06.15. 13:49 조회 53

멀티태스킹(멀티 스레드, 멀티 프로세스)

프로세스 : 운영체제에서 실행중인 하나의 애플리케이션
사용자가 애플리케이션을 실행하면 운영체제로 부터 실행에 필요한 메모리를 할당받아 애플리케이션의 코드를 실행하는데 이것이 프로세스이다.
하나의 애플리케이션은 다중 프로세스를 만들기도 한다.

멀티 태스킹 : 두가지 이상의 작업을 동시에 처리하는 것을 말한다.
운영체제는 멀티 태스킹을 할 수 있도록 CPU 및 메모리 자원을 프로세스 마다 적절히 할당 해 주고 병렬로 실행 시킨다.

멀티 태스킹은 꼭 멀티 프로세스를 뜻하지 않는다
대표적으로 미디어 플레이어와 메신저이다. 미디어 플레이어는 동영상 재생과 음악 재생이라는 두 작업을 동시에 처리한다. 이는 멀티스레드를 이용하는 방법이다.

하나의 스레드는 하나의 코드 실행흐름이다. 한 프로세스 내에 스레드가 두개라면 두개의코드 실행 흐름이 생긴다.

멀티 프로세스가 애플리케이션 단위인 멀티태스킹이라면
멀티 스레드는 애플리케이션 내부에서의 멀티 태스킹이다.

멀티 프로세스 : 운영체제에서 할당받은 자신의 메모리를 가지고 실행하기 때문에 서로 독립적
따라서 하나의 프로세스에서의 오류가 다른 프로세스에 영향을 미치지 x

멀티 스레드 : 하나의 프로세스 내부에 생성 되기 때문에 하나의 스레드가 예외를 발생시키면 프로세스 자체가 종료 될 수 있어 다른 스레드에게 영향을 미치게 한다. 예를 들어 멀티 프로세스인 워드 와 엑셀을 동시에 사용도중 워드에 오류가 생기면 엑셀은 여전히 사용 가능하다.
그러나 멀티 스레드로 동작하는 메신저의 경우 파일 전송도중 스레드에서 예외가 발생되면 메신저 프로세스 자체가 종료 되어 채팅 스레드도 같이 종료 된다.

싱글 스레드 애플리케이션: 메인 스레드 종료시 프로세스도 종료
멀티 스레드 애플리케이션: 실행 스레드가 하나라도 있으면, 프로세스가 종료 되지 않는다. 메인 스레드가 작업 스레드보다 먼저 종료 되더라도 작업 스레드가 계속 실행 중이라면 프로세스는 종료 되지 않는다.

<스레드 우선순위>
멀티스레드는 동시성, 병렬성으로 실행된다.

동시성 : 멀티 작업을 위해 하나의 코어에서 멀티스레드가 번갈아가며 실행하는 성질
병렬성 : 멀티 작업을 위해 멀티 코어에서 개별 스레드를 동시에 실행하는 성질

스레드 스케줄링 : 스레드의 갯수가 코어의 수보다 많을 경우, 스레드를 어떤 순서에 동시성을 실행할 것인가를 결정해야 한다.

자바 스레드 스케줄링
1. 우선순위 방식 : 우선순위가 높은 스레드가 실행 상태를 더 많이 가지도록 스케줄링 하는 것을 말함
-> 개발자에 의해 코드 제어 o
2. 순환 할당 방식 : 사간 할당량을 정해서 하나의 스레드를 정해진 시간 만큼 실행하고 다른 스레드를 실행하는 방식,
-> 개발자에 의해 코드제어 x

<동기화 메소드 및 동기화 블록>
임계영역 : 멀티 스레드 프로그램에서 단 하나의 스레드만 실행할 수 있도록 해주는 코드 영역
자바는 임계영역을 지정하기 위해 동기화 (synchronized) 메소드와 동기화 블록을 제공함
동기화 메소드 실행 방법 : synchronized키워드를 붙인다.
동기화 메소드는 메모리 전체 내용이 임계영역이므로 스레드가 동기화 메소드를 실행하는 즉시 객체에는 잠금이 일어나고, 스레드가 동기화 메소드를 실행 종료 하면, 잠금이 풀린다. 메소드 전체 내용이 아니라 일부 내용만 임계영역으로 만들고 싶으면 동기화 영역을 만들면 된다.
동기화 메소드는 1개의 스레드만 실행된다.

> Calculator
> User1
> User2
> MainThreadExample

<스레드 상태>
1 스레드 객체 생성 (NEW)
->start()
2 스레드 실행대기(RUNNABLE)
3 실행(2,3 반복) :run() 메소드를 실행한다.
4 종료(TERMINATED())
일시 정지 상태 :실행 상태에서 일시 정지 상태로 가기도 하는데 일시정지 상태는 스레드가 실행 할 수 없는 상태이다. (waiting, timed_waiting, blocked)
스레드가 다시 실행상태로 가기 위해서는 일시 정지 상태에서 실행 대기 상태로 가야 한다.

<다른 스레드에게 실행 양보 yield()>
스레드가 처리하는 작업이 반복적으로 실행하기 위해 포문이나 while 문을 포함한다.
가끔 이 반복문들이 무의미하게 반복한다.
run()이 실행되어 무한 반복 실행시 부의미한 반복이라면, 이것보다 다른 스레드에게 실행을 양보하고 자신은 실행 대기 상태로 가는 것이다. yield() 메소드를 호출한 스레드는 실행 대기 상태로 돌아가고 동일한 우선순위 또는 높은 우선 순위를 갖는 다른 스레드가 실행 기회를 가질 수 있도록 해준다.
> ThreadA
> ThreadB
> yieldExample
이 예제는 처음 실행후 1초동안 ab가 번갈아 가며 시행 된다 1초 후 메인 스레드가 a의 work를 false로 변경함으로써 a는 yield() 메소드를 호출 한다. 따라서 1초 이후, b가 더 많은 실행 기회를 얻는다 메인 스레드는 1초후 다시 threada의 work 필드를 true로 변경해서 threadA와 ThreadB가 번갈아 가며 실행하도록 한다. 마지막으로 메인 스레드는 1초 후에 threadA와 ThreadB가 스레드 반복 작업을 중지하고 종료하게 한다.

<다른 스레드의 종료를 기다림 (Join)>
스레드는 다른 스레드와는 독립적으로 실행하는 것이 기본이지만 다른 스레드가 종료 될 때까지 기다렸다가 실행해야 하는 경우도 있다.
예를 들어 계산 작업을 하는 스레드가 모든 계산작업을 마쳤을 때 계산 결과값을 받아 이용하는 경우이다.
> SumThread
> JoinExample

<스레드간의 협업 (wait(), notify(), notifyAll()) >
경우에 따라 두개의 스레드를 교대로 번갈아 가며 실행해야 한다.
정확한 교대작업이 필요할 경우 자신의 작업이 끝나면 상대의 스레드를 일시 정지 에서 풀고 자신을 일시 정지 상태로 만드는 것이다
이방법의 핵심은 공유객체이다.
공유 객체는 두 스레드가 작업할 내용을 각각 동기화 메소드로 구문하고, 한 스레드가 작업을 완료 하고 나면 notify()메소드를 호출해 일시 정지 상태의 다른 스레드를 실행상태로 만든후 자신은 두번 작업을 하지 않도록 wait를 호출하여 일시 정지를 만든다.
notifyALL() 메소드는 wait()에 의해 일시 정지된 모든 스레드들을 실행 대기 상태로 만든다.
주의! 이 메소드들은 동기화 메소드 혹은 동기화 블록 내에서만 사용 가능하다.
> ThreadA
> ThreadB
> WorkObject
> WorkNotifyExample

<스레드의 안전한 종료(stop 플래스, interrupt())>
스레드는 자신의 run()메소드가 모두 실행되면 자동적으로 종료된다. 경우에 따라서는 실행 중인 스레드를 즉시 종료 할 필요가 잇따.
1. Thread 는 스레드를 즉시 종료하기 위해 stop() 메소드를 제공하고 잇는데, 이 메소드는 deprecated 되었다. 그 이유는 stop() 메소드로 스레드를 갑자기 종료하게 되면 스레드가 사용중이던 자원들이 불안전한 상태로 남겨지기 때문이다.

stop 플래그 이용방법
> PrintThread
> StopFlagExample

위 코드는 stop 필드가 true일 경우 자동 종료 한다. 그리고 스레드가 사용한 자원을 정리하고 run 메서드가 끝나게 됨으로써 스레드는 안전하게 종료 된다.

2. interrupt 메소드를 이용하는 방법
interrupt 메소드 : 스레드가 일시 정지 상태에 잇을때 InterruptedException 예외를 발생시키는 역할을 한다.
이것을 이용하면 run 메소드를 정상 종료 시킬 수 있다
ThreadA가 ThreadB의 intrerrupt() 메소드를 실행하게 되면 ThreadB() 가 sleep () 메소드로 일시 정지 상태가 될 때 ThreadB에서 InterruptedException이 발생하여 예외 처리 블록으로 이동한다. 결국 ThreadB는 while문을 빠져나오 run 메소드를 정상종료 하게 된다.
> PrntThread
> StopFlagExample

!! 스레드가 실행 대기 또는 실행 상태에 잇을 때 interrupt() 메소드가 실행되면 즉시 Interrupted 예외가 발생하지 않고, 스레드가 미래에 일시 정지가 되면, interrupt() 메소드 호출은 아무런 의미가 없다.
그래서 짧은 시간동안 sleep()를 호출 한 것이다.
일시 정지를 만들지 않고도 interrupt() 호출 여부를 알 수 잇는 방법이 있다. intterupt 메소드가 호출 되면 isInterrupted 를 true로 리턴 할 경우, 현재 스레드의 인터럽트 여부를 확인해준다.

< 데몬스레드 >
데몬스레드? 주스레드의 작업을 돕는 보조적 역할을 수행하는 스레드 이다. 주 스레드가 종료되면 데몬 스레드는 강제적으로 자동 종료 되는데, 그이유는 주스레드의 보조 역할을 수행하므로, 주스레드 종료시, 데몬스레드의 의미가 없어지기 때문이다.
예> 워드 프로세서의 자동저장, 미디어 플레이어의 동영상 및 음악 재생, 가비지 컬렉터 등, 이 기능들은 주스레드(워드 프로세스, 미디어 플레이어, JVM) 이 종료되면 같이 종료 된다.

스레드를 데몬으로만들기 위해서는 주스레드가 데몬이 될 스레드를 setDaemon(true)를 호출하면 된다.
주의 할 점은 start() 메소드가 호출되고 나서 setDaemon(true) 를 호출하면, IllegalThreadStateException이 발생하기 때문에 start() 메소드 호출 전에 setDaemone(true)를 호출 해야 한다.
현재 실행 중인 스레드가 데몬스레드가 아닌지 구별하는 방법 : inDaemone() 메소드의 리턴값 (데몬스레드일 경우 true를 리턴한다)
> PrintThread
> StopFlagExample

<스레드 그룹>
: 관련된 스레드를 묶어서 관리할 목적으로 이용된다.
JVM 이 실행되면 system 스레드 그룹을 만들고, JVM 운영에 필요한 스레드들을 생성해서 system 스레드 그룹에 포함 시킨다. 스레드는 반드시 하나의 스레드 그룹에 포함 되는데 명시적으로 스레드 그룹에 포함 시키지 않으면, 기본적으로 자신을 생성한 스레드와 같은 스레드 그룹에 속하게 된다.
우리가 생성하는 작업 스레드는 대부분 main 스레드가 생성하므로 기본적을로 main 스레드 그룹에 속하게 된다.

<스레드 풀>
: 병렬 작업 처리가 많아 지면 스레드 갯수가 증가 되고, 그에 따른 스레드 생성과 스케줄링으로 인해 CPU가 바빠져 메모리 사용량이 늘어난다
따라서 애플리케이션 성능이 저하도니다.
갑작스런 병렬 작업의 폭증이로 인한 스레드 폭증을 막으로면 스레드 풀을사용해야 한다.

스레드 풀은 작업 처리에 사용되는 스레드를 제한된 개수만큼 정해 놓고 작업 큐에 들어오면, 작업들을 하나씩 스레드가 맡아 처리한다.
작업 처리가 끝난 스레드는 다시 작업 큐에서 새로운 작업을 가져와 처리한다
그렇기 대문에 작업 처리 요청이 폭증 되어도 스레드의 전체 갯수가 늘어나지 않으므로, 애플리케이션의 성능이 급격히 저하되지 않는다.
자바는 스레드 풀을 생성하고 사용할 수 있도록 java.util.concurrent 패키지에서
1, ExecutorService 인터페이스 2 Excutores 클래스를 제공한다 .
Excutors 의 다양한 정적 메소드를 이용해서 ExcutorService 구현 객체를 만들 수 있는데, 이것이 바로 스레드 풀이다.
- 초기 스레드 수 : ExecutorService 객체가 생성될 때 기본적으로 생성되는 스레드 수이다.
- 코어 스레드 수 : 스레드수가 증가된 후 사용되지 않는 스레드를 스레드 풀에서 제거할 때 최소

댓글

댓글알림

Developer Shin

댓글을 남겨보세요

등록