자바 성능 튜닝 이야기

* 이상민 지음 / 인사이트

디자인 패턴(GOF 패턴, Gang Of Four)

Transfer Object 패턴

* Getter / Setter 의 경우 null check를 하지 않아도 길이가 0인 String 리턴
* Serializable은 왜 implements 했는가?
  + 객체를 직렬화 할 수 있음, 서버 사이의 데이터 전송이 가능해짐 / 원격지 서버에 데이터를 전송하거나 파일로 객체를 저장할 경우 이 인터페이스를 구현해야 함

Service Locator 패턴 ?

* EJB Home 객체나 유의 DataSource를 찾을 때 소요되는 응답속도를 감소시키기 위해 사용

APM 툴(Application Performance Monitoring)

* WebTune(미소 정보, 개발 장비에서 무료), Pharos(케이와이즈), Introscope(CA Wily)
* 애플리케이션의 장애 상황에 대한 모니터링 및 문제점 진단
* 서버의 사용자 수나 리소스에 대한 모니터링 가능
* 실시간 모니터링
* CPU수를 기반하여 가격 측정
* 자바 기반 클라이언트 프로그램은 분석이 불가능

프로파일링 툴(Profiling Tool, 시스템 문제 분석 툴)

* JProfiler(JProbe, ej-technologies), YourKit(YourKit)
* 소스 레벨의 분석을 위한 툴
* 애플리케이션의 세부 응답시간까지 분석 가능
* 메모리 사용량을 객체나 클래스, 소스의 라인 단위까지 분석 가능
* 사용자수 기반의 가격 측정
* 자바 기반의 클라이언트 프로그램 분석 가능

Library

* JMH, Caliper, JUnitPerf, JUnitBench, ContiPerf(그래프 결과 제공, ms단위 제공 X)
* Junit 자바 코드 테스트
* JMH JDK를 오픈소스로 제공하는 OpenJDK에서 만든 성능 측정용 Library

응답시간 & 메모리 Profiling / GC(Garbage Collection)

System Class

* Static void arraycopy(Object src, int srcPos, Object dest, int destPos, int length)

String / StringBuilder / StringBuffer

* StringBuffer(): 객체 생성
* StringBuffer(CharSequence seq): CharSequence를 매개변수로 받아 객체 생성
* StringBuffer(int capacirt): capacity용량의 객체 생성
* StringBuffer(String str): str을 갖는 객체 생성
* Append(): 기존 값의 맨 끝자리에 값들을 덧붙임
* Insert(): 지정된 위치 이후에 넘어온 값을 덧붙임
* 사용처
  + String: 짧은 문자열을 더할 때
  + StringBuffer: 스레드에 안전한 프로그램이 필요할 때나 개발중인 시스템의 부분이 스레드에 안전한지 모를 경우
  + StringBuilder: 스레드에 안전한지의 여부와 전혀 관계없는 프로그램을 계발하는 경우
* Java Version 별 차이( String str1 = “Here ” + “is ” + i + “ samples.”; )
  + JDK 1.4: 문자열을 미리 더해 줌   
    String str1 = “Here is” + i + “samples.”;  
     🡺 중간에 int와 같은 객체가 들어간다면 재생성한다는 한계가 있음
  + JDK 1.5이상: 자동으로 StringBuffer로 변환하여 append 한다.  
    String str1 = (new String(“Here is ”)).append(i).append(“ samples.”).toString;  
     🡺 객체를 계속 추가한다는 한계 존재

Calendar vs java Date

다른 날짜를 계산한다면 Calendar, Calendar는 다른 주 날짜 사이의 차이를 이용하여 계산

Date는 명확한 그 순간을 이용하여 계산