Artigo

Invista em você! Saiba como a DevMedia pode ajudar sua carreira.

Trabalhando com Threads em Java

Veja neste artigo como utilizar na prática e de forma simples Threads em Java para realizar processamentos paralelos em sua aplicação.

Anotar

Marcar como concluído

Threads em Java, com o objetivo de trabalhar de forma paralela, também conhecida por programação concorrente.

Threads

Antes de iniciarmos a **explicação prática de Thread, entenda que estas são subdivisões dos processos**, como assim? Cada processo possui diversas threads (linhas de instruções), assim nós podemos dividir partes do nosso processo (programa em Java) para trabalhar paralelamente.

As threads estão em nosso dia-a-dia, em todos os processamentos realizados em nosso computador. Quando estamos ouvindo uma música e olhando o Facebook ao mesmo tempo, estamos realizando um processamento paralelo, mesmo que de forma transparente ao usuário.

Em um **programa em Java podemos querer executar 2 ou mais threads ao mesmo tempo, ou seja, 2 ou mais procedimentos internos do programa ao mesmo tempo**. Para simplificar a explicação de threads em Java, vamos a um exemplo prático.

Imagine que você tem 1 procedimento que consome muito tempo do processador, vamos supor o exemplo abaixo de um cálculo que faz consultas a um Web Service (que pode demorar muito a responder):

```
1  public void calculaTotalRecebido(){
2      //Recebe aproximadamente 70mil registros.
3      List<HistoricoRecebimento> recebidos = getListRecebimentos
4      Integer soma = 0;
```

```
10      Integer porcentagemImposto = getReajusteAtualFromWebServic
11
12      soma = soma + ((porcentagemImposto/100)*soma);
13
14      retornaParaWebServiceValorTotal(soma);
15    }
```

Listagem 1. Procedimento comum sem thread

O procedimento acima parece simples (de fato é simples), recebe uma lista com mais ou menos 70 mil registros, depois pega o valor de cada registro e soma. Após isso, captura um valor para reajuste em porcentagem (capturado de um WebService que pode demorar a responder), e recalcula o valor somado. Por fim retorna o resultado via WebService.

Perceba aqui claramente que temos um procedimento que pode demorar minutos, e não vai retornar nada ao usuário, apenas fazer comunicação interna entre Web Services. Não podemos parar toda nossa aplicação para executar o procedimento acima. Pois imagina se o usuário está fazendo um cadastro simples e tem que esperar 4 minutos para terminar o processamento acima.

A solução então é fazer com que esse procedimento seja executado concorrentemente, ou seja, ao mesmo tempo em que o usuário está realizando o cadastro, o procedimento acima também é executado, e provavelmente quando ele terminar o cadastro, o procedimento acima também já terminou, de forma imperceptível a ele.

No código abaixo vamos usar o mesmo código, porém usando o conceito de Threads. Iremos criar uma Thread para um bloco específico de código, através da

paralelamente, sendo assim vamos colocar todo o código acima dentro de um método run. Como Runnable é apenas um contrato, precisamos de alguma classe que a implemente e faça o trabalho da "paralelização", que é a classe Thread.

```
1  public void calculaTotalRecebido(){
2    new Thread() {
3
4      @Override
5      public void run() {
6        //Recebe aproximadamente 70mil registros.
7        List<HistoricoRecebimento> recebidos = getListRecebiment
8        Integer soma = 0;
9
10       for(HistoricoRecebimento h1: recebidos){
11         soma = soma + recebidos.getValorRecebido();
12       }
13
14       Integer porcentagemImposto = getReajusteAtualFromWebServ
15
16       soma = soma + ((porcentagemImposto/100)*soma);
17
18       retornaParaWebServiceValorTotal(soma);
19
20     }
21   }.start();
22
23 }
```

Listagem 2. Usando Thread

Quando fazemos o ".start();" já estamos iniciando o processamento paralelo, e liberando o programa para executar qualquer outra thread. Então tenha a

Caso você deseja que sua classe seja processada paralelamente, porém ela já estende de outra, você poderá optar por implementar o Runnable, que é a interface padrão para Thread. Por boas práticas, geralmente implementamos Runnable em vez de estender de Thread.

Atente a outro ponto muito importante: O programador não tem nenhum controle sobre o escalonador do processador. Isso significa que sendo os processos executados paralelamente, você não tem como saber qual a ordem de execução destes. No exemplo abaixo vamos colocar 2 contadores para executar paralelamente, você irá perceber que o resultado pode diferir cada vez que você executar, ou até de computador para computador.

```
1  public class MyTest {
2      static int i = 0;
3      public static void main(String[] args) {
4          new Thread(t1).start();
5          new Thread(t2).start();
6      }
7
8      private static void countMe(String name){
9          i++;
10         System.out.println("Current Counter is: " + i + ", upd
11     }
12
13     private static Runnable t1 = new Runnable() {
14         public void run() {
15             try{
16                 for(int i=0; i<5; i++){
17                     countMe("t1");
18                 }
19             } catch (Exception e){}
```

```
25          public void run() {
26              try{
27                  for(int i=0; i<5; i++){
28                      countMe("t2");
29                  }
30              } catch (Exception e){}
31          }
32      };
33  }
```

Listagem 3. Teste de Contadores com Thread

Vou deixar por conta sua testar o código acima e ver qual o resultado, pois cada computador pode ter um ordem de execução diferente, isso é comum e muito normal.

Conclusão

Utilize os recursos da programação concorrente com cuidado e só nos momentos que de fato precisar. Não saia de forma alguma colocando Runnable em todos os códigos pensando que seu programa ficará mais rápido, pelo contrário, um processo pode precisar de recursos de outro que ainda não está disponível e assim por diante, analise com cuidado a necessidade do uso deste poderoso recurso.

Um exemplo mais comum ainda de ser visto é a barra de progresso presente na maioria dos softwares. Enquanto está carregando um documento X é mostrada uma barra de progresso para o usuário saber que o software não travou.