Programação Concorrente em Java

Aula 10

Java Thread Local

Introdução

- A classe **ThreadLocal** possibilita a criação de variáveis que só podem ser lidas e escritas pela mesma Thread.
- Então, mesmo se duas threads estão executando o mesmo código e o código tem uma referência para uma variável ThreadLocal, então as duas threads não podem ver as variáveis ThreadLocal umas das outras.

Criando uma ThreadLocal

- Criando uma variável ThreadLocal

```
private ThreadLocal myThreadLocal = new ThreadLocal();
```

- Um objeto do tipo ThreadLocal é instanciado
- Só é necessário fazer isso apenas umas vez por thread
- Mesmo que duas threads executem o mesmo código o qual acessa a variável ThreadLocal, cada thread verá apenas a sua instância
- Mesmo que duas threads coloquem valores diferentes no mesmo objeto ThreadLocal, elas não podem ver os valores uma da outra.

Acessando uma ThreadLocal

- Uma vez criada, pode-se armazenar valores da seguinte forma:

```
myThreadLocal.set("A thread local value");
```

- E lê-se da seguinte forma:

```
String threadLocalValue = (String) myThreadLocal.get();
```

Valor inicial

- Sobrescrever o método initialValue()

```
private ThreadLocal myThreadLocal = new ThreadLocal<String>()
{
    @Override protected String initialValue() {
        return "This is the initial value";
    }
};
```

Exemplo Completo

```
public class ThreadLocalExample {
    public static class MyRunnable implements Runnable {
        private ThreadLocal<Integer> threadLocal =
               new ThreadLocal<Integer>();
        @Override
        public void run() {
            threadLocal.set( (int) (Math.random() * 100D) );

            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
            }

            System.out.println(threadLocal.get());
        }
    }

...
```

```
...
public static void main(String[] args) {
        MyRunnable sharedRunnableInstance = new MyRunnable();
        Thread thread1 = new Thread(sharedRunnableInstance);
        Thread thread2 = new Thread(sharedRunnableInstance);
        thread1.start();
        thread2.start();
        thread1.join(); //wait for thread 1 to terminate
        thread2.join(); //wait for thread 2 to terminate
    }
}
```

Referência

- http://tutorials.jenkov.com/java-concurrency/threadlocal.html