Java：多線程，java.util.concurrent.atomic包之AtomicInteger/AtomicLong用法

1. 背景

java.util.concurrent.atomic這個包是非常實用，解決了我們以前自己寫一個同步方法來實現類似於自增長字段的問題。

在Java語言中，增量操作符（++）不是原子的，也就是非線程安全的；在使用的時候，要保證數據同步，就需要使用類似於synchronized關鍵字等手段來保證數據正確。正因為如此，《Java：多線程，線程同步，synchronized關鍵字的用法（同步代碼塊、非靜態同步方法、靜態同步方法）》一文中用synchronized關鍵字來實現一個自增長的字段。

2. 實現代碼

如今使用java.util.concurrent.atomic包，問題簡單多了。示範代碼如下（沒有synchronized關鍵字，沒有Lock鎖）：

```
package com.clzhang.sample.thread;

import java.util.concurrent.atomic.AtomicInteger;
import java.util.concurrent.atomic.AtomicLong;

public class SimpleTest implements Runnable {
    // 創建AtomicInteger，初始值0（也可以指定初始值）
//    private static final AtomicInteger nextSerialNum = new AtomicInteger();
    private static final AtomicLong nextSerialNum = new AtomicLong();

    @Override
    public void run() {

        // 直接取得當前值並增長1
        System.out
                .println(Thread.currentThread().getName() + ":" +
nextSerialNum.getAndIncrement());
        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
        }
    }

    public static void main(String[] args) {
        SimpleTest st = new SimpleTest();

        for (int i = 0; i < 100; i++) {
            new Thread(st, "Thread" + i).start();
```

```
            }
        }
}
```

部分輸出：

Thread92:60
Thread94:59
Thread95:58
Thread90:57
Thread88:56
Thread93:55