JMH :

Benchmark Mode Cnt Score Error Units

ApplicationJMHTests.addProduct ss 20 1,607 ± 0,194 ms/op

ApplicationJMHTests.deleteProduct ss 20 1,112 ± 0,165 ms/op

ApplicationJMHTests.getProduct ss 20 0,594 ± 0,049 ms/op

ApplicationJMHTests.getProducts ss 20 2,102 ± 0,777 ms/op

ApplicationJMHTests.setProductAvailable ss 20 0,905 ± 0,092 ms/op

Intelij Profiler

Method samples (around +-1ms) Cnt Average

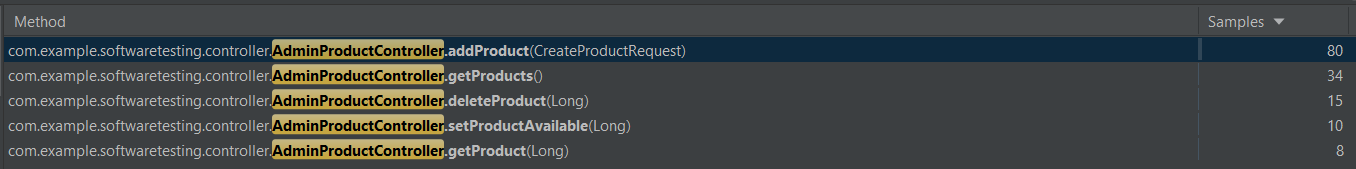
addProduct 80 50+10 1.33 ms

deleteProduct 15 10 1.5 ms

getProduct 8 10 0.8 ms

getProducts 35 10 3.5 ms

setProductAvailable 10 10 1 ms



|  |  |  |
| --- | --- | --- |
| Metoda | jmh [ms] | Intelij Profiler [ms] |
| addProduct | 1,607 | 1.33 |
| deleteProduct | 1,112 | 1.5 |
| getProduct | 0,594 | 0.8 |
| getProducts | 2,102 | 3.5 |
| setProductAvailable | 0,905 | 1 |

Ocena skuteczności JMH

Java Microbenchmark Harness jest przyjemnym narzędziem ułatwiającym testowanie wydajności poszczególnych funkcji. Głównymi zaletami jest możliwość testowania w wielu trybach pomiarów (pomiar średni dla wielu pomiarów, pomiar dla pojedynczego wykonania) oraz obliczanie błędów pomiarowych. Korzystanie z wbudowanego w środowisko programistyczne profilera nie daje takich możliwości. Można osiągnąć podobny efekt poprzez korzystanie z funkcji System.nanoTime() i skomplikowanych funkcji matematycznych, jednak jest to dużo bardziej czasochłonne. JMH posiada wiele opcji konfiguracji, które mogą ułatwić i przyspieszyć testowanie takie jak np. wielowątkowość, koniec testów przy napotkanym wyjątku czy też iteracje rozruchowe(niewliczane do testów).

Podsumowując JMH jest prostym w użyciu narzędziem które w znaczący sposób ułatwia testowanie aplikacji pod względem wydajności.