JMH

Java Microbenchmark Harness

JAVA BENCHMARKING

- JIT Optimization
- Reproduction
- Concurrency

CONFIGURATION

at the example of maven (what else)

DEPENDENCIES

```
<dependencies>
    <dependency>
        <groupId>org.openjdk.jmh</groupId>
        <artifactId>jmh-core</artifactId>
        <version>1.21</version>
    </dependency>
    <dependency>
        <groupId>org.openjdk.jmh</groupId>
        <artifactId>jmh-generator-annprocess</artifactId>
        <version>1.21</version>
    </dependency>
</dependencies>
```

BUILD

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-shade-plugin</artifactId>
    <version>3.2.1</version>
    <executions>
        <execution>
            <phase>package</phase>
            <goals><goal>shade</goal></goals>
            <configuration>
                <finalName>benchmarks</finalName>
                <transformers>
                    <transformer implementation="org.apache.ma
                        <mainClass>org.openjdk.jmh.Main</mainC
                    </transformer>
                </transformers>
```

EXECUTION

```
java -jar target/benchmarks.jar
```

CREATING TESTS

SIMPLEST CASE

```java
import org.openjdk.jmh.annotations.Benchmark;

public class SomethingPerformanceCritical {
    @Benchmark
    public void testMePlenty() {
        // Doing heavy calculation stuff
    }
}
```

CONFIGURATION OVERVIEW

```
@Benchmark
@BenchmarkMode({Mode.Throughput, Mode.SingleShotTime, Mode.Sam
@Fork(warmups=5, value = 5)
@Measurement(iterations = 5, time = 5, timeUnit = TimeUnit.SEC
@OperationsPerInvocation(1)
@OutputTimeUnit(TimeUnit.SECONDS)
@Threads(5)
@Timeout(time = 100, timeUnit = TimeUnit.MILLISECONDS)
@Warmup(iterations = 5, time = 5, timeUnit = TimeUnit.SECONDS)
public void configure() {}
```

MODES

`@BenchmarkMode`

- Throughput → Ops per second
- Average Time → Avg duration
- Sample Time → Duration statistics (min, max…)
- Single Shot Time → Time for single, cold execution
- All

STATE

@State *(Thread, Group, Benchmark)*, @Setup, @TearDown

```
@State(Scope.Thread)
public static class MyState {
    public int a = 1;
    public int b = 2;
    public int sum ;
}

@Benchmark @BenchmarkMode(Mode.Throughput) @OutputTimeUnit(Tim
public void testMethod(MyState state) {
    state.sum = state.a + state.b;
}
```

REMARKS

- Loop Optimizations
- Dead Code Elimination

```
@Benchmark
public void testMethod(Blackhole blackhole) {
    // (...)
    blackhole.consume(computationResult);
}
```

OUTPUT

EXAMPLE

```
# JMH version: 1.21
# VM version: JDK 12, OpenJDK 64-Bit Server VM, 12+33
# VM invoker: C:\Java\jdk12\bin\java.exe
# VM options: --enable-preview
# Warmup: 1 iterations, 5 s each
# Measurement: 2 iterations, 5 s each
# Timeout: 10 min per iteration
# Threads: 1 thread, will synchronize iterations
# Benchmark mode: Average time, time/op
# Benchmark: bayern.jugin.jmh.LoopingComparison.imperativeLoop

# Run progress: 0,00% complete, ETA 00:02:15
# Warmup Fork: 1 of 1
# Warmup Iteration   1: 2,941 ms/op
Iteration   1: 2,206 ms/op
```