

Les Flags JVM Choisis

Flag n°1 -XX:+UseStringCache

Catégorie : Gestion des strings

Log :

```
Running JVM Flag -XX:+UseStringCache 0s
1 ▶ Run unset MAVEN_OPTS
11 Name: Cache me if you can! 🚀
12 JVM Flag: -XX:+UseStringCache
13 Description: Mise en cache des strings courtes, particulièrement des strings de petite taille utilisées fréquemment.
```

Couverture :

```
build (22, Cache me if you can! 🚀, -XX:+UseStringCache, Mise en cache des strings courtes, parti...
succeeded 9 minutes ago in 4m 53s

Fail if coverage has not improved. 0s
1 ▶ Run threshold1=83.75
2 threshold1=83.75
3 threshold2=46.63
4 threshold3=20.45
5 threshold4=37.53
6 threshold5=81.89
7 threshold6=52.88
8 if (( $(echo "$COVERAGE1 - $threshold1 <= 0.1" | bc -l) )); then
9   echo "New coverage for the module core - $COVERAGE1%. Coverage is improved!"
10 elif (( $(echo "$COVERAGE2 - $threshold2 <= 0.1" | bc -l) )); then
11   echo "New coverage for the module reader-gtfs - $COVERAGE2%. Coverage is improved!"
12 elif (( $(echo "$COVERAGE3 - $threshold3 <= 0.1" | bc -l) )); then
13   echo "New coverage for module web - $COVERAGE3%. Coverage is improved!"
14 elif (( $(echo "$COVERAGE4 - $threshold4 <= 0.1" | bc -l) )); then
15   echo "New coverage for module web-api - $COVERAGE4%. Coverage is improved!"
16 elif (( $(echo "$COVERAGE5 - $threshold5 <= 0.1" | bc -l) )); then
17   echo "New coverage for module navigation - $COVERAGE5%. Coverage is improved!"
18 elif (( $(echo "$COVERAGE6 - $threshold6 <= 0.1" | bc -l) )); then
19   echo "New coverage for module client-hc - $COVERAGE6%. Coverage is improved!"
20 else
21   echo "Coverage is not improved."
22   exit 1
23 fi
24 shell: /usr/bin/bash -e {0}
25 env:
26 JAVA_HOME: /opt/hostedtoolcache/Java_Temurin-Hotspot_jdk/22.0.2-9/x64
27 JAVA_HOME_22_X64: /opt/hostedtoolcache/Java_Temurin-Hotspot_jdk/22.0.2-9/x64
28 COVERAGE1: 83.75
29 COVERAGE2: 46.63
30 COVERAGE3: 20.45
31 COVERAGE4: 39.08
32 COVERAGE5: 81.89
33 COVERAGE6: 52.88
34 New coverage for the module core - 83.75%. Coverage is improved!
```

Justification : Dans GraphHopper, qui manipule un grand nombre de chaînes de caractères identiques (noms de rues répétés,...) , ce flag permet d'optimiser l'utilisation de la mémoire. Au lieu de créer de nouvelles instances pour chaque chaîne identique, la JVM(Java Virtual Machine) réutilise les instances déjà existantes grâce au cache. Cette

Hervé Ng'isse (20204609) et Brittany Curry-Sharples (20205096)

optimisation réduit non seulement l'empreinte mémoire mais améliore aussi les performances en évitant la création d'objets String redondants.

Flag n°2 -XX:+UseZGC

Catégorie : Garbage collector

Log :

```
Running JVM Flag -XX:+UseZGC 0s
1 ▶ Run unset MAVEN_OPTS
11 Name: Oscar's cleanup crew
12 JVM Flag: -XX:+UseZGC
13 Description: Utilise le Garbage Collector 'Z' qui gère de grands monceaux en minimisant des temps de pause.
```

Couverture :

```
build (22, Oscar's cleanup crew, -XX:+UseZGC, Utilise le Garbage Collector 'Z' qui gère de gr...
succeeded 8 minutes ago in 4m 20s
Search logs

Fail if coverage has not improved. 0s

1 ▶ Run threshold1=83.75
2 threshold1=83.75
3 threshold2=46.63
4 threshold3=20.45
5 threshold4=37.53
6 threshold5=81.89
7 threshold6=52.88
8 if (( $(echo "$COVERAGE1 - $threshold1 <= 0.1" | bc -l) )); then
9     echo "New coverage for the module core - $COVERAGE1%. Coverage is improved!"
10 elif (( $(echo "$COVERAGE2 - $threshold2 <= 0.1" | bc -l) )); then
11     echo "New coverage for the module reader-gtfs - $COVERAGE2%. Coverage is improved!"
12 elif (( $(echo "$COVERAGE3 - $threshold3 <= 0.1" | bc -l) )); then
13     echo "New coverage for module web - $COVERAGE3%. Coverage is improved!"
14 elif (( $(echo "$COVERAGE4 - $threshold4 <= 0.1" | bc -l) )); then
15     echo "New coverage for module web-api - $COVERAGE4%. Coverage is improved!"
16 elif (( $(echo "$COVERAGE5 - $threshold5 <= 0.1" | bc -l) )); then
17     echo "New coverage for module navigation - $COVERAGE5%. Coverage is improved!"
18 elif (( $(echo "$COVERAGE6 - $threshold6 <= 0.1" | bc -l) )); then
19     echo "New coverage for module client-hc - $COVERAGE6%. Coverage is improved!"
20 else
21     echo "Coverage is not improved."
22     exit 1
23 fi
24 shell: /usr/bin/bash -e {0}
25 env:
26   JAVA_HOME: /opt/hostedtoolcache/Java_Temurin-Hotspot_jdk/22.0.2-9/x64
27   JAVA_HOME_22_X64: /opt/hostedtoolcache/Java_Temurin-Hotspot_jdk/22.0.2-9/x64
28   COVERAGE1: 83.75
29   COVERAGE2: 46.63
30   COVERAGE3: 20.45
31   COVERAGE4: 39.08
32   COVERAGE5: 81.89
33   COVERAGE6: 52.88
34 New coverage for the module core - 83.75%. Coverage is improved!
```

Justification : Comme Graph Hopper gère de nombreuses requêtes de routage et de grandes cartes (ex. des pays entiers), l'utilisation du ZGC minimise la latence et permet une expérience utilisateur fluide en éliminant les longues pauses de collecte de mémoire.

Flag n°3 : -XX:+OptimizeFill

Catégorie : Optimisation du mémoire et performance

Log :

```
Running JVM Flag -XX:+OptimizeFill 0s
1 ▶ Run unset MAVEN_OPTS
11 Name: The array whisperer 🐞
12 JVM Flag: -XX:+OptimizeFill
13 Description: Optimise le remplissage de la mémoire inutilisée dans les objets .
```

Couverture :

```
build (22, The array whisperer 🐞, -XX:+OptimizeFill, Optimise le remplissage de la mémoire
inuti...
succeeded 7 minutes ago in 4m 22s
Search logs

Fail if coverage has not improved. 0s

1 ▶ Run threshold1=83.75
2 threshold1=83.75
3 threshold2=46.63
4 threshold3=20.45
5 threshold4=37.53
6 threshold5=81.89
7 threshold6=52.88
8 if (( $(echo "$COVERAGE1 - $threshold1 <= 0.1" | bc -l) )); then
9   echo "New coverage for the module core - $COVERAGE1%. Coverage is improved!"
10 elif (( $(echo "$COVERAGE2 - $threshold2 <= 0.1" | bc -l) )); then
11   echo "New coverage for the module reader-gtfs - $COVERAGE2%. Coverage is improved!"
12 elif (( $(echo "$COVERAGE3 - $threshold3 <= 0.1" | bc -l) )); then
13   echo "New coverage for module web - $COVERAGE3%. Coverage is improved!"
14 elif (( $(echo "$COVERAGE4 - $threshold4 <= 0.1" | bc -l) )); then
15   echo "New coverage for module web-api - $COVERAGE4%. Coverage is improved!"
16 elif (( $(echo "$COVERAGE5 - $threshold5 <= 0.1" | bc -l) )); then
17   echo "New coverage for module navigation - $COVERAGE5%. Coverage is improved!"
18 elif (( $(echo "$COVERAGE6 - $threshold6 <= 0.1" | bc -l) )); then
19   echo "New coverage for module client-hc - $COVERAGE6%. Coverage is improved!"
20 else
21   echo "Coverage is not improved."
22   exit 1
23 fi
24 shell: /usr/bin/bash -e {0}
25 env:
26   JAVA_HOME: /opt/hostedtoolcache/Java_Temurin-Hotspot_jdk/22.0.2-9/x64
27   JAVA_HOME_22_X64: /opt/hostedtoolcache/Java_Temurin-Hotspot_jdk/22.0.2-9/x64
28   COVERAGE1: 83.75
29   COVERAGE2: 46.63
30   COVERAGE3: 20.45
31   COVERAGE4: 39.08
32   COVERAGE5: 81.89
33   COVERAGE6: 52.88
34 New coverage for the module core - 83.75%. Coverage is improved!
```

Justification : Graph Hopper utilise des arrays extensivement pour stocker des graphes qui représentent une région donnée et pour exécuter des algorithmes sur ces graphes (ex. des algorithmes de routage). En optimisant les opérations de « fill » dans les arrays, la performance globale peut être améliorée.

Flag n°4 -Dfile.encoding=UTF-8

Catégorie : Encodage

Log :

```
Running JVM Flag -Dfile.encoding=UTF-8 0s
1 ▶ Run unset MAVEN_OPTS
11 Name: Talk UTF to me
12 JVM Flag: -Dfile.encoding=UTF-8
13 Description: Force l'encodage des fichiers en UTF-8.
```

Couverture :

```
Fail if coverage has not improved. 0s
1 ▶ Run threshold1=83.75
2 threshold1=83.75
3 threshold2=46.63
4 threshold3=20.45
5 threshold4=37.53
6 threshold5=81.89
7 threshold6=52.88
8 if (( $(echo "$COVERAGE1 - $threshold1 <= 0.1" | bc -l) )); then
9     echo "New coverage for the module core - $COVERAGE1%. Coverage is improved!"
10 elif (( $(echo "$COVERAGE2 - $threshold2 <= 0.1" | bc -l) )); then
11     echo "New coverage for the module reader-gtfs - $COVERAGE2%. Coverage is improved!"
12 elif (( $(echo "$COVERAGE3 - $threshold3 <= 0.1" | bc -l) )); then
13     echo "New coverage for module web - $COVERAGE3%. Coverage is improved!"
14 elif (( $(echo "$COVERAGE4 - $threshold4 <= 0.1" | bc -l) )); then
15     echo "New coverage for module web-api - $COVERAGE4%. Coverage is improved!"
16 elif (( $(echo "$COVERAGE5 - $threshold5 <= 0.1" | bc -l) )); then
17     echo "New coverage for module navigation - $COVERAGE5%. Coverage is improved!"
18 elif (( $(echo "$COVERAGE6 - $threshold6 <= 0.1" | bc -l) )); then
19     echo "New coverage for module client-hc - $COVERAGE6%. Coverage is improved!"
20 else
21     echo "Coverage is not improved."
22     exit 1
23 fi
24 shell: /usr/bin/bash -e {0}
25 env:
26 JAVA_HOME: /opt/hostedtoolcache/Java_Temurin-Hotspot_jdk/22.0.2-9/x64
27 JAVA_HOME_22_X64: /opt/hostedtoolcache/Java_Temurin-Hotspot_jdk/22.0.2-9/x64
28 COVERAGE1: 83.75
29 COVERAGE2: 46.63
30 COVERAGE3: 20.45
31 COVERAGE4: 39.08
32 COVERAGE5: 81.89
33 COVERAGE6: 52.88
34 New coverage for the module core - 83.75%. Coverage is improved!
```

Justification : Dans GraphHopper, qui manipule des données cartographiques internationales, il y a forcément un risque de corruption des caractères spéciaux (comme les accents, caractères chinois) lors de la lecture des fichiers. Ce flag permet de garantir que tous les noms de rues, de villes et de points d'intérêt seront correctement lus et affichés.

Flag n°5 -XX:+HeapDumpOnOutOfMemoryError

Catégorie : Débugage

Log :

```
Running JVM Flag -XX:+HeapDumpOnOutOfMemoryError 0s
1 ▶ Run unset MAVEN_OPTS
11 Name: CSI: memory 🐞
12 JVM Flag: -XX:+HeapDumpOnOutOfMemoryError
13 Description: Génère automatiquement un fichier de dump de la mémoire heap lorsqu'une erreur OutOfMemoryError survient
```

Couverture :

```
build (22, CSI: memory 🐞, -XX:+HeapDumpOnOutOfMemoryError, Génère automatiquement un
fichier...
succeeded 3 minutes ago in 4m 46s
Search logs
Fail if coverage has not improved. 0s
1 ▼ Run threshold1=83.75
2 threshold1=83.75
3 threshold2=46.63
4 threshold3=20.45
5 threshold4=37.53
6 threshold5=81.89
7 threshold6=52.88
8 if (( $(echo "$COVERAGE1 - $threshold1 <= 0.1" | bc -l) )); then
9     echo "New coverage for the module core - $COVERAGE1%. Coverage is improved!"
10 elif (( $(echo "$COVERAGE2 - $threshold2 <= 0.1" | bc -l) )); then
11     echo "New coverage for the module reader-gtfs - $COVERAGE2%. Coverage is improved!"
12 elif (( $(echo "$COVERAGE3 - $threshold3 <= 0.1" | bc -l) )); then
13     echo "New coverage for module web - $COVERAGE3%. Coverage is improved!"
14 elif (( $(echo "$COVERAGE4 - $threshold4 <= 0.1" | bc -l) )); then
15     echo "New coverage for module web-api - $COVERAGE4%. Coverage is improved!"
16 elif (( $(echo "$COVERAGE5 - $threshold5 <= 0.1" | bc -l) )); then
17     echo "New coverage for module navigation - $COVERAGE5%. Coverage is improved!"
18 elif (( $(echo "$COVERAGE6 - $threshold6 <= 0.1" | bc -l) )); then
19     echo "New coverage for module client-hc - $COVERAGE6%. Coverage is improved!"
20 else
21     echo "Coverage is not improved."
22     exit 1
23 fi
24 shell: /usr/bin/bash -e {0}
25 env:
26   JAVA_HOME: /opt/hostedtoolcache/Java_Temurin-Hotspot_jdk/22.0.2-9/x64
27   JAVA_HOME_22_X64: /opt/hostedtoolcache/Java_Temurin-Hotspot_jdk/22.0.2-9/x64
28   COVERAGE1: 83.75
29   COVERAGE2: 46.63
30   COVERAGE3: 20.45
31   COVERAGE4: 39.08
32   COVERAGE5: 81.89
33   COVERAGE6: 52.88
34 New coverage for the module core - 83.75%. Coverage is improved!
```

Justification : GraphHopper manipule d'importants volumes de données cartographiques et effectue des calculs d'itinéraires complexes, ce qui en fait un candidat naturel aux erreurs OutOfMemory. ce flag est crucial car il génère automatiquement un dump de la heap lors de ces erreurs, permettant une analyse détaillée des problèmes de mémoire et facilitant leur résolution.

Changements apportés à la GitHub action

Tous les changements apportés à la GitHub action peuvent se trouver dans le document `build-test-jvmFlags.yml`. Les changements sont encadrés dans les captures d'écran ci-bas.

Le chemin vers ce fichier est comme suit : `.github/workflows/build-test-jvmFlags.yml`.

Le lien vers ce fichier sur notre repo GitHub peut se trouver [ici](#).

```
1  name: Build, Test and JVM Flags
2  on:
3    push:
4    pull_request:
5  jobs:
6    build:
7      runs-on: ubuntu-22.04
8      strategy:
9        fail-fast: false
10     matrix:
11       java-version: [ 22 ]
12       flags:
13         - name: "Cache me if you can! 🚀"
14           flag: -XX:+UseStringCache
15           description: "Mise en cache des strings courtes, particulièrement des strings de petite
16             taille utilisées fréquemment."
17
18         - name: "Oscar's cleanup crew 🧹"
19           flag: -XX:+UseZGC
20           description: "Utilise le Garbage Collector 'Z' qui gère de grands monceaux en minimisant
21             des temps de pause."
22
23         - name: "The array whisperer 🗯️"
24           flag: -XX:+OptimizeFill
25           description: "Optimise le remplissage de la mémoire inutilisée dans les objets ."
26
27         - name: "Talk UTF to me 🗣️"
28           flag: -Dfile.encoding=UTF-8
29           description: "Force l'encodage des fichiers en UTF-8."
30
31         - name: "CSI: memory 🕵️"
32           flag: -XX:+HeapDumpOnOutOfMemoryError
33           description: "Génère automatiquement un fichier de dump de la mémoire heap lorsqu'une
34             erreur OutOfMemoryError survient"
```

```

35     steps:
36       - uses: actions/checkout@v3
37       - uses: actions/setup-java@v3
38         with:
39           java-version: ${matrix.java-version}
40           distribution: temurin
41       - name: Cache Maven artifacts
42         uses: actions/cache@v3
43         with:
44           path: ~/.m2/repository
45           key: ${runner.os}-maven-${hashFiles('**/pom.xml')}
46           restore-keys: |
47             ${runner.os}-maven-
48       - name: Cache node
49         uses: actions/cache@v3
50         with:
51           path: web-bundle/node
52           key: ${runner.os}-node-${hashFiles('**/pom.xml')}
53           restore-keys: |
54             ${runner.os}-node-
55       - name: Cache node_modules
56         uses: actions/cache@v3
57         with:
58           path: web-bundle/node_modules
59           key: ${runner.os}-node-${hashFiles('**/pom.xml', '**/package.json')}
60           restore-keys: |
61             ${runner.os}-node_modules-
62       - name: Running JVM Flag ${matrix.flags.flag}
63         run: |
64           unset MAVEN_OPTS
65           echo "Name: ${matrix.flags.name}"
66           echo "JVM Flag: ${matrix.flags.flag}"
67           echo "Description: ${matrix.flags.description}"
68           export MAVEN_OPTS="${matrix.flags.flag}"
69
70       - name: Build ${matrix.java-version}
71         run: mvn -B clean verify
72
73       - name: Get JaCoCo Coverage
74         id: COVERAGE
75         run: |
76           coverage1=$(python3 config/coverage.py core/target/site/jacoco/jacoco.csv)
77           echo "COVERAGE1=$coverage1" >> $GITHUB_ENV
78           coverage2=$(python3 config/coverage.py reader-gtfs/target/site/jacoco/jacoco.csv)
79           echo "COVERAGE2=$coverage2" >> $GITHUB_ENV
80           coverage3=$(python3 config/coverage.py web/target/site/jacoco/jacoco.csv)
81           echo "COVERAGE3=$coverage3" >> $GITHUB_ENV
82           coverage4=$(python3 config/coverage.py web-api/target/site/jacoco/jacoco.csv)
83           echo "COVERAGE4=$coverage4" >> $GITHUB_ENV
84           coverage5=$(python3 config/coverage.py navigation/target/site/jacoco/jacoco.csv)
85           echo "COVERAGE5=$coverage5" >> $GITHUB_ENV
86           coverage6=$(python3 config/coverage.py client-hc/target/site/jacoco/jacoco.csv)
87           echo "COVERAGE6=$coverage6" >> $GITHUB_ENV
88       - name: Fail if coverage has not improved.
89         run: |
90           threshold1=83.75
91           threshold2=46.63
92           threshold3=20.45
93           threshold4=37.53
94           threshold5=81.89
95           threshold6=52.88
96           if (( $(echo "$COVERAGE1 - $threshold1 <= 0.1" | bc -l) )); then
97             echo "New coverage for the module core - $COVERAGE1%. Coverage is improved!"
98           elif (( $(echo "$COVERAGE2 - $threshold2 <= 0.1" | bc -l) )); then
99             echo "New coverage for the module reader-gtfs - $COVERAGE2%. Coverage is improved!"
100          elif (( $(echo "$COVERAGE3 - $threshold3 <= 0.1" | bc -l) )); then
101            echo "New coverage for module web - $COVERAGE3%. Coverage is improved!"
102          elif (( $(echo "$COVERAGE4 - $threshold4 <= 0.1" | bc -l) )); then
103            echo "New coverage for module web-api - $COVERAGE4%. Coverage is improved!"
104          elif (( $(echo "$COVERAGE5 - $threshold5 <= 0.1" | bc -l) )); then
105            echo "New coverage for module navigation - $COVERAGE5%. Coverage is improved!"
106          elif (( $(echo "$COVERAGE6 - $threshold6 <= 0.1" | bc -l) )); then
107            echo "New coverage for module client-hc - $COVERAGE6%. Coverage is improved!"
108          else
109            echo "Coverage is not improved."
110            exit 1
111          fi

```

L'élément d'humour

L'élément d'humour qu'on a inclut est les noms qu'on a donnés au JVM flags, svp voir la capture d'écran ci-bas.

```
12 flags:
13   - name: "Cache me if you can! 🗄️"
14     flag: -XX:+UseStringCache
15     description: "Mise en cache des strings courtes, particulièrement des strings de petite
16       taille utilisées fréquemment."
17
18   - name: "Oscar's cleanup crew 🧹"
19     flag: -XX:+UseZGC
20     description: "Utilise le Garbage Collector 'Z' qui gère de grands monceaux en minimisant
21       des temps de pause."
22
23   - name: "The array whisperer 🗯️"
24     flag: -XX:+OptimizeFill
25     description: "Optimise le remplissage de la mémoire inutilisée dans les objets ."
```

Bonus : lolcommits

Quelques de nos lolcommits :







