Documentation Tâches 3

Farah Mohamed, Salim Fathya

Introduction

Ce document présente les modifications apportées au workflow CI pour un projet Java. Le workflow initial a été enrichi pour permettre l'exécution des tests sous différentes configurations de flags JVM, afin d'améliorer la qualité, la performance, et l'observabilité des tests.

Description du Workflow

Modifications Apportées

Ajout de Flags JVM

Dans la nouvelle version, une **stratégie de matrix** a été ajoutée pour exécuter les tests avec cinq configurations de flags JVM différentes. Voici les flags ajoutés et leur utilité :

- `-XX:+ShowCodeDetailsInExceptionMessages`: Active l'affichage de détails supplémentaires sur le code dans les messages d'exception. Comme des informations spécifiques sur la méthode ou la classe où une exception s'est produite. Ce flag est utile pour améliorer la lisibilité des messages d'erreur lors du débogage.
- `-XX:+UseG1GC` : Active le Garbage Collector G1, optimisé pour les applicationsnécessitant une gestion fine de la mémoire. Aussi, permet une meilleure gestion de la mémoire et garantit la stabilité de la JVM même sous forte charge.
- `-XX:+PrintFlagsFinal` : Affiche tous les flags JVM actifs en fin d'exécution pour une meilleure observabilité. Ce qui facilite le diagnostic et l'optimisation des configurations.

- `-XX:CompileThreshold=1500` : Abaisse le seuil de compilation JIT (Just-In-Time), optimisant le temps de compilation pour les méthodes souvent invoquées en réduisant le nombre de compilations nécessaires pour le code fréquemment exécuté, ce qui peut accélérer les tests.
- `-XX:+UseStringDeduplication` : Active la déduplication des chaînes pour réduire l'utilisation de la mémoire lorsque de nombreuses chaînes identiques sont créées. Cela peut être utile pour une application comme Makelangelo.

Exécution des Tests avec Différentes Configurations JVM

La commande Maven a été modifiée pour inclure chaque configuration de flags JVM définie dans le `matrix`. L'option `-DargLine="${{ matrix.flags }}"` permet de transmettre ces flags à chaque exécution des tests, fournissant ainsi des informations sur le comportement de l'application sous différentes conditions.

Vérification de la Couverture de Code

Le calcul de la couverture de code avec JaCoCo reste inchangé, mais les différentes configurations JVM permettent de détecter des cas de test supplémentaires, améliorant potentiellement la couverture globale.

Message TRÈS important !!

Bon, je ne sais pas si c'est votre premier devoir ou votre trentième devoir que vous corrigez, mais je m'excuse en avance de ce que vous allez lire. En même temps, vous l'avez un peu cherché en demandant à des informaticiens de faire l'humour. Toutefois, je ne me priverais pas de faire mes blagues personnelles (oui, oui MES blagues).

Pourquoi les bélugas sont en voie d'extinction ? Parce qu'ils n'ont pas de bélufilles !

Généralement quand on parle de Mozart, il n'est pas là. Mais quand on ouvre le réfrigérateur, Mozzarella !

Qui va avoir besoin de sortir prendre l'air après cette session de correction ? Vous (vous en aurez besoin je crois).

Conclusion

Cette nouvelle configuration de workflow CI optimise le processus de tests pour le projet Java avec Maven. En explorant différentes configurations JVM, elle permet d'évaluer et d'améliorer la stabilité, la performance et l'efficacité mémoire des tests, tout en offrant une meilleure observabilité.