

TP3 – Analyse qualité de code Java

Brahim HAMDI

Objectifs :

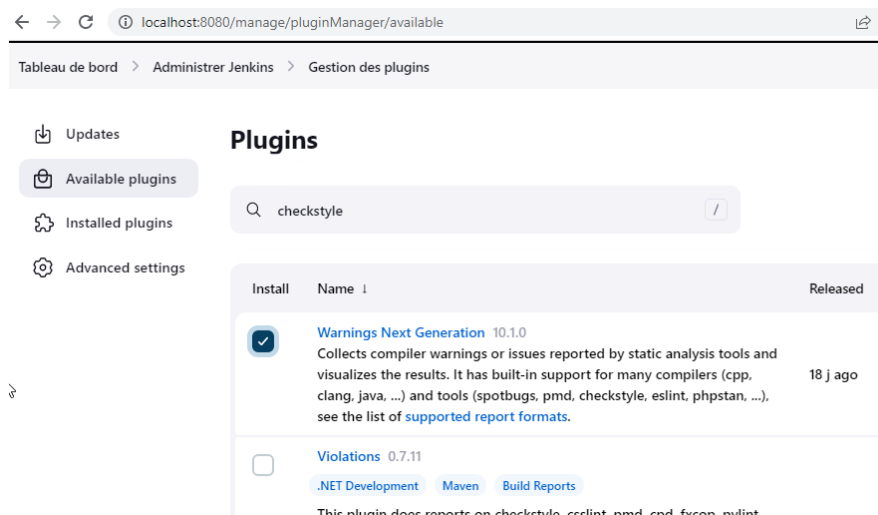
- Générer des rapports de tests et de qualité

Rapports Checkstyle et PMD

Revenons au projet free-style du TP2.

Vous pouvez générer des rapports sur les résultats des outils Checkstyle et PMD. Pour faire, vous devez tout d'abord installer le plugin *Warnings Next Generation*.

1. Installez le plugins *Warnings Next Generation* au travers du gestionnaire de plugin.



2. Une fois ces plugins installés, vous pouvez paramétrer la création des rapports dans l'interface de config de votre projet free-style *petclinic* (section « Actions à la suite du build »). Sélectionnez la case "Record Compiler warnings and static analysis results".

Configure

- General
- Gestion de code source
- Ce qui déclenche le build
- Environnements de Build
- Build Steps
- Actions à la suite du build**

Archiver des artefacts
Consolider les résultats des tests en aval
Construire d'autres projets (projets en aval)
Discover reference build
Enregistrer les empreintes numériques des fichiers pour en suivre l'utilisation
Mine SCM repository
Publier le rapport des résultats des tests JUnit
Publier les Javadocs
Record compiler warnings and static analysis results
Git Publisher
Editable Email Notification
Notifier par email
Set GitHub commit status (universal)
Set build status on GitHub commit [deprecated]
Delete workspace when build is done

Ajouter une action après le build ^

Sauver

Apply

3. Une fois l'interface de configuration affichée, ajoutez 3 outils : Checkstyle et PMD.

Configure

- General
- Gestion de code source
- Ce qui déclenche le build
- Environnements de Build
- Build Steps
- Actions à la suite du build**

Record compiler warnings and static analysis results

Static Analysis Tools

Tool ?

AcuCobol

Brakeman
Buckminster
CCM
CMake
CPD
CPPCheck
CSS-Lint
Cadence Incisive
Cargo Check
CheckStyle
Clair Scanner
Clang
Clang Analyzer
Clang-Tidy
Code Analysis
Code Generator Tool
CodeChecker
CodeNarc

Configure

- General
- Gestion de code source
- Ce qui déclenche le build
- Environnements de Build
- Build Steps
- Actions à la suite du build**

Tool ?

AcuCobol


OT Docker Linter
OWASP Dependency Check
Open Tasks Scanner
Oracle Invalids
PC-Lint Tool
PEP8
PHP Runtime
PHPStan
PHP_CodeSniffer
PIT
PMD
PREfast
PVS-Studio
Perforce Compiler
Perl: Critic
Polyspace Tool
ProtoLint
Puppet Lint
PyDocStyle
PyLint

Optional custom ID (URL) of this tool, overwrites the built-in ID 'acu-cobol'.

4. Modifiez le build *Invoquer les cibles Maven de haut niveau* avec la ligne suivante :

clean package -fn checkstyle:checkstyle pmd:pmd

Build Steps

 Invoquer les cibles Maven de haut niveau ?

Version de Maven

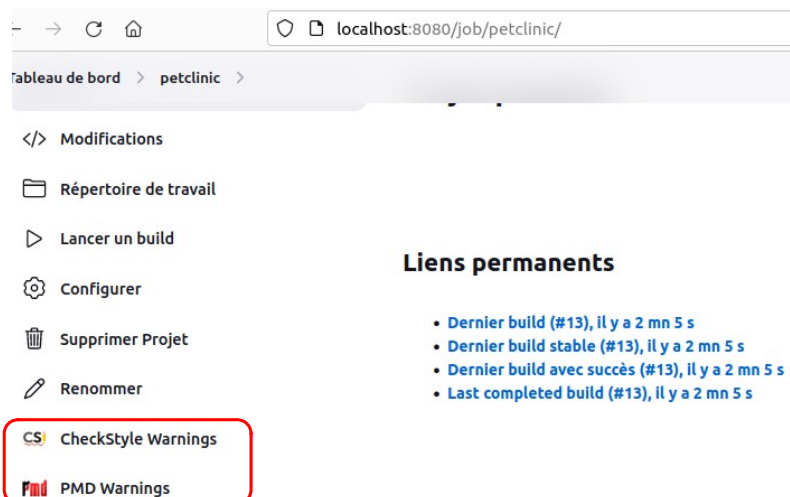
M3

Cibles Maven

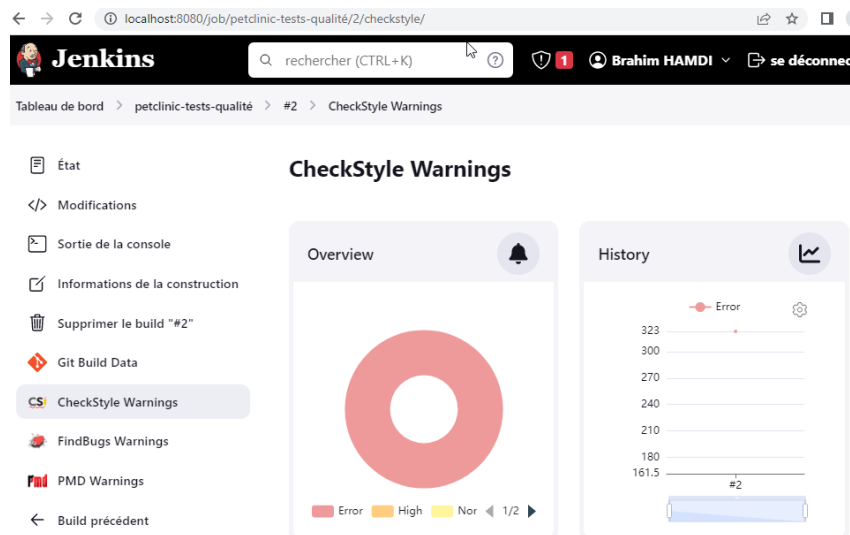
clean package -fn checkstyle:checkstyle pmd:pmd

5. Sauver et lancer le build du projet.

Après l'exécution, les rapports seront affichés sur l'interface de Jenkins.



- Vous pouvez visualiser les détails en cliquant sur les icônes.



6. Reprenez le projet de pipeline et ajoutez les 2 outils à la ligne « mvn clean package ».

```
Script ?
11 }
12 stage('Build') {
13   steps {
14     script
15     {
16       if (isUnix())
17       {
18         sh 'mvn -fn clean package checkstyle:checkstyle pmd:pmd'
19       }
20       else
21       {
22         bat 'mvn -fn clean package checkstyle:checkstyle pmd:pmd'
23       }
24     }
25   }
26 }
27 stage('QA checks') {
28   steps {
29     recordIssues(tools: [checkStyle(), pmdParser()])
30   }
31 }
32 }
```

7. Ensuite, ajoutez le stage de QA et lancez le build pour voir les résultats .

```
26 }
27 stage('QA checks') {
28   steps {
29     recordIssues(tools: [checkStyle(), pmdParser()])
30   }
31 }
32 }
```

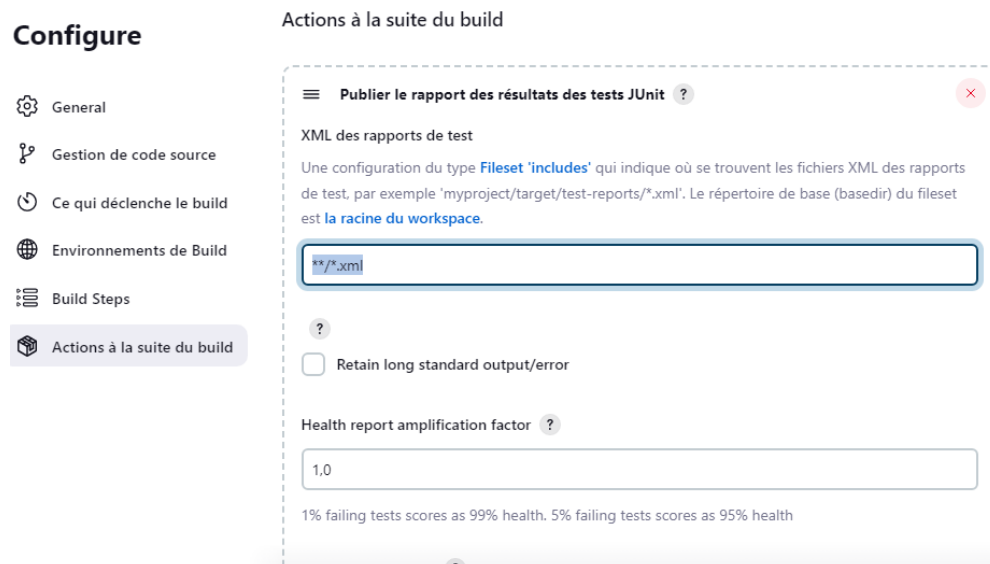
Rapport de tests et archivage des artifacts

Dans cette partie, nous allons prendre soin des rapports et artifacts générés par le build dans les étapes de post-construction.

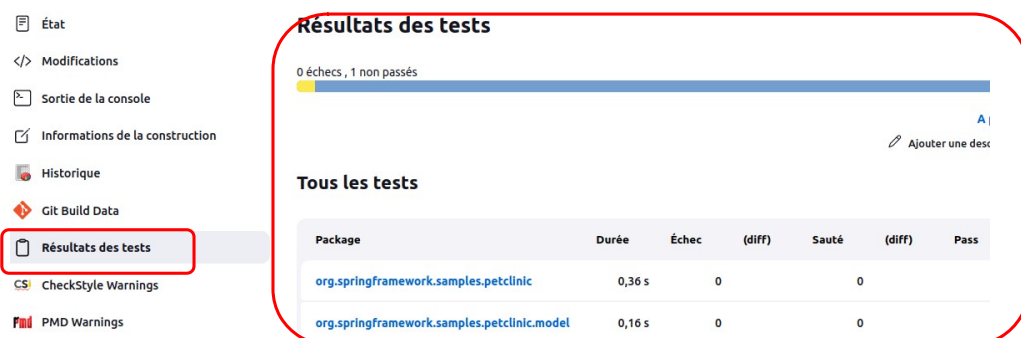
8. Nous allons ajouter une action qui se déroulera après les étapes de build et publiera les tests Maven sorties (unitaire, intégration, etc.). Il définira également si l'état de la compilation est instable ou a échoué en fonction des résultats des tests.

Pour revenir à l'interface de configuration du projet, cliquez sur «*Configurer*» (à droite sur l'interface principale du projet).

- Cliquez sur «*Ajouter une action après le build*», et sélectionnez «*Publier un rapport de résultat de test Junit*»
- Définir le champ XML du rapport de test sur «***/*.xml*», puis cliquez sur «*Apply*».



- Une fois le build passé avec succès, les résultats de tests seront affichés.



9. Ajoutez une autre action après le build : « **Archiver les artefacts** »

- Définir le champ Fichier à archiver sur:

target/**/*.*jar

Actions à la suite du build

Archiver des artefacts ?

Fichiers à archiver ?

target/**/*.*jar

Avancé ▾

Publier le rapport des résultats des tests JUnit ?

XMI des rapports de test

10. Cliquez sur « **Sauver** » et vérifiez les résultats de tests et l’archivage de l’artefact.

État

Modifications

Sortie de la console

Informations de la construction

Supprimer le build "#19"

Git Build Data

Résultats des tests

CheckStyle Warnings

PMD Warnings

✓ Construction #19 (23 avr. 2023, 12:52:06)

Conserver cette construction sans limite de

Démarrée il y a 1 mn 17 s.

A duré 38 s

Ajouter une description

Artefacts du build

spring-petclinic-3.0.0-SNAPSHOT.jar 52,55 MB view

No changes.

Lancé par l'utilisateur brahim

Revision: a613fe0aeb8fb77acda09b5c10e78bede8f88d01

Repository: <https://github.com/brahimhamdi/petclinic>

• refs/remotes/origin/main

11. Reprenez le projet de type pipeline et ajoutez les lignes suivantes pour archiver l’artefacts jar et publier les rapports de tests:

Script ?

```
52 stage('Archivage artifacts') {
53   steps {
54     script
55     {
56       if (isUnix())
57       {
58         sh 'echo Archivage des artifacts'
59       }
60       else
61       {
62         bat 'echo Archivage des artifacts'
63       }
64     }
65   }
66   post {
67     success {
68       archiveArtifacts 'target/*.jar'
69     }
70   }
```

Script ?

```
32 stage('Rapports de Tests') {
33   steps {
34     script
35     {
36       if (isUnix())
37       {
38         sh 'echo Rapports de tests'
39       }
40       else
41       {
42         bat 'echo Rapports de tests'
43       }
44     }
45   }
46   post {
47     success {
48       junit '**/target/surefire-reports/TEST-*.xml'
49     }
50   }
```

- Vérifiez les résultats une fois le build terminé avec succès.