Université De Montpellier
Faculté Des Sciences

Niveau : Master 2

Module : Évolution et restructuration des logiciels

HAI913I

Rapport de TP N°5 : Introduction à la traçabilité logiciel avec Spoon

---

Supervisé par :
Mr. Abdelhak-Djamel Seriai
Mr. Bachar Rima

Réalisé par :
KACI Ahmed
YANIS Allouch

2021/2022

Table des matières

1 Question 1

Pour répondre à cette question, nous avons réalisé une application qui permet à un utilisateur de s'authentifier en fournissant sur la console, un ensemble d'informations le concernant.
Une fois, l'utilisateur authentifié en lui fournit un menu lui permettant d'effectuer les opérations suivantes :

- Afficher la liste des produits présents dans un dépôt,
- Sélectionné un produit par son identifiant.(si aucun produit avec l'ID fourni n'existe, une exception est levée),
- Ajouter un nouveau produit (si un produit avec le même identifiant existe déjà, une exception est levée),
- Supprimer un produit en fournissant son id (si aucun produit avec l'ID fourni n'existe, une exception est levée),
- Mettre à jour les informations d'un produit (si aucun produit avec l'ID fourni n'existe, une exception est levée).

Afin de réaliser la gestion des exceptions, nous avons implémenté deux classes, intitulées *ProductNotFoundException* et *ProductAlreadyExistException*, qui étendent la classe Exception prédéfinie en java. Ces deux classes prennent un message personnalisé dans leur constructeur, qu'on affiche à l'utilisateur lors de la gestion des exceptions de ces types.
Le code cette application est dans la branche main du dépôt git suivant :
https://gitlab.com/yanisallouch/hai913i-tp5-projet-de-test

2 Question 2

Pour répondre à cette question, nous avons choisi d'utiliser *Java Logging API*.
Afin de comprendre la méthode d'utilisation de cette API, nous avons inséré, dans un premier temps, manuellement l'ensemble des instructions de log permettant de tracer les opérations qu'effectue un utilisateur en utilisant l'application présentée ci-dessus. Nous nous sommes fondés sur les bonnes pratiques indiquées dans l'article [1] ainsi que la documentation Java.
Le code cette application est dans la branche logging-exemple du dépôt git suivant :
https://gitlab.com/yanisallouch/hai913i-tp5-projet-de-test/-/tree/logging-exemple

3 Question 3

Pour répondre à cette question, nous avons utilisé Java Logging API. De par l'instrumentation manuelle effectuée dans la question 2, nous avons repéré trois lieux de transformation de code à effectuer via Spoon. Ces trois points d'injections sont les suivants :

1. Dans les attributs d'une classe
   - Nous devons injecter la déclaration et l'initialisation du Logger en tant qu'attribut d'une classe,
   - Nous devons injecter le FileHandler associé au Logger d'une classe, en tant qu'attribut de cette classe.
2. Dans les constructeurs d'une classe
   - Nous devons injecter l'initialisation du FileHandler dans le constructeur. Parce que nous devons gérer les exceptions levées par l'ouverture d'un fichier.
3. Dans les méthodes d'une classe
   - Nous devons injecter les instructions de logging dans chaque méthode, afin de tracer les actions de l'utilisateur.

L'application de test, ainsi instrumenté via Spoon permet de satisfaire l'objectif de traçage suivant :

1. Un profil pour ceux qui ont principalement effectué des opérations de lecture sur la base de données,
2. Un profil pour ceux qui ont principalement effectué des opérations d'écriture sur la base de données,
3. Un profil pour ceux qui recherchent les produits les plus chers sur la base de données.

On observe que l'instrumentation peut se décomposer en trois temps. Tout d'abord, on crée un processeur permettant de transformer les attributs de classe. Puis un second processeur, pour les constructeurs de classe. Enfin, on crée un processeur ajoutant les LPS dans les méthodes.

3.1 Auto-import des librairies

Pour que l'instrumentation importe les libraires des objets Java injecté dans la classe en cours de transformation, il faut configurer le launcher Spoon de la façon suivante :

```
1 Launcher spoon = new Launcher();
2 ...
3 spoon.getEnvironment().setAutoImports(true);
```

Listing 1 – Configuration de l'importation automatique via Spoon

Puis, il est important de créer la déclaration d'un attribut de la façon suivante :

```
final CtTypeReference<Logger> loggerRef = getFactory().Code().
    createCtTypeReference(Logger.class);
// Create static field LOGGER.
final CtField<Logger> loggerField = getFactory().Core().<Logger>
    createField();
loggerField.<CtField>setType(loggerRef);
loggerField.<CtField>addModifier(ModifierKind.STATIC);
loggerField.<CtField>addModifier(ModifierKind.PRIVATE);
loggerField.setSimpleName("LOGGER");
// Create default statement value to initialize the LOGGER
String expression = "Logger.getLogger(" + elementClass.
    getSimpleName() + ".class.getName())";
// no need for semi-colon; it is added by the SnippetExpression
final CtCodeSnippetExpression loggerExpression = getFactory().Code
    ().createCodeSnippetExpression(expression);
loggerField.setDefaultExpression(loggerExpression);
// Add the field to the concerned class
elementClass.addFieldAtTop(loggerField);
```

Listing 2 – Déclaration de l'attribut Logger pour l'auto-import via Spoon

Le code cette application est accessible sur le dépôt GitLab public suivant : https://gitlab.com/yanisallouch/hai913i-tp5-spoon-logging

4 Question 4

Pour répondre à cette question, nous définissons les trois utilisateurs suivants :

1. Utilisateur n°1 décrit de la façon suivante :

```
ID=1, name=Ahmed, age=24, email=ahmed.kaci@etu.umontpellier.fr,
    password=123

```

2. Utilisateur n°2 décrit de la façon suivante :

```
ID=2, name=Yanis, age=22, email=yanis.allouch@etu.umontpellier.
   fr, password=1234

```

3. Utilisateur n°3 décrit de la façon suivante :

```
ID=3, name=Zaara, age=23, email=zaara@domain.fr, password=1235

```

Voici les profils d'utilisateurs que nous définissons :

1. L'utilisateur n°1 et n°3, feront un grand nombre d'ajouts et de mise à jour dans la base de données,
2. L'utilisateur n°1, ajoutera quelques produits et fera beaucoup de lecture dans la base de données,

3. L'utilisateur n°3, cherchera les trois produits les plus chers (de façon répétée afin de le distinguer des autres).

Les logs résultant de notre scénario sont disponibles dans le dossier "spooned" à l'adresse GitLab suivante :
https://gitlab.com/yanisallouch/hai913i-tp5-projet-de-test/-/blob/main/output/spooned/src/Repository.log

5 Question 5

Dans cet exercice, nous développons un nouveau projet Java nous permettant de parser les logs générés et d'en extraire les informations requises à la construction des profils utilisateurs. Et ce, après avoir récupéré l'ensemble des logs du projet sous tests réalisé dans l'exercice 4.

Pour répondre à la question de cet exercice qui consiste à implémenter le code Java d'un parseur de log XML et de produire les profils utilisateurs, nous avons suivi les étapes suivantes :

- Pour commencer, nous réifions le concept de LPS dans la classe *LPSPersonalised* selon deux critères, le nom de l'opération tracé et le message de l'opération tracé
  - La logique métier de ce processus, est encapsulée dans la classe *LogParserProcessor*.
- Ensuite, pour chaque LPS, on extrait l'information des 4W [1] qu'on a réifié dans les classes *User*, *Product* et *UserProfile*
  - La logique métier de ce processus, est encapsulée dans les classes *ProductProcessor* et *UserProfileProcessor*.
- Enfin, nous affichons dans la console pour chaque profil, l'utilisateur qui correspond le mieux aux critères (définis dans l'exercice 3 et détaillé dans l'exercice 4)
  - Cette logique métier, a été réparti en trois classes. La classe *BestReadingUserProfiles*, *BestWritingUserProfiles* et *BestExpensiveUserProfile*. Chaque classe se préoccupe respectivement de calculer le profil utilisateur ayant fait le plus de lecture, le plus d'écriture et ayant cherchés les produits les plus chers dans la base de données.

Le code cette application est accessible sur le dépôt GitLab suivant :
https://gitlab.com/kaciahmed3/hai913i-tp5-question5 par Ahmed Kaci, Yanis Allouch et M. Abdelhak-Djamel Seriai (sous le pseudo @rechercheseriai).

1. Les 4W est un concept abordé en cours, faisant référence aux quatre mots anglais suivants : Who, What, Where et When.

Nous avons aussi joint en annexe de ce TP la trace d'exécution permettant d'afficher les résultats obtenus à partir de l'application du parsing des logs XML fourni dans le dépôt https://gitlab.com/yanisallouch/hai913i-tp5-projet-de-test.

6 Annexe

6.1 Traces d'exécution

Question 5

Trace d'exécution du parsing des logs sur l'application disposé sur ce lien GitLab public.

```
********** The userProfile with the best reading operations **********

User [ID=1, name=Ahmed, age=24, email=ahmed.kaci@etu.umontpellier.fr, password=123]

The number of reading operations is : 29

List of implicite and explict reading opeartions done

  addProduct
  fetchProduct
  addProduct
  fetchProduct
  addProduct
  fetchProduct
  addProduct
  fetchProduct
  addProduct
  fetchProduct
  updateProduct
  deleteProduct
  fetchProduct
  addProduct
  fetchProduct
  updateProduct
  deleteProduct
  fetchProduct
  addProduct
  fetchProduct
  updateProduct
  deleteProduct
  fetchProduct
  addProduct
  fetchProduct
  fetchProduct
  fetchProduct
  fetchProduct
  fetchProduct

********** The userProfile with the best writing operations **********

```

```
User [ID=1, name=Ahmed, age=24, email=ahmed.kaci@etu.umontpellier.fr, password=123]

The number of writing operations is : 14

List of writing opeartions done

  addProduct
  addProduct
  addProduct
  addProduct
  addProduct
  updateProduct
  deleteProduct
  addProduct
  updateProduct
  deleteProduct
  addProduct
  updateProduct
  deleteProduct
  addProduct

********** The userProfile with the best number of searching most expensives products operations **********

User [ID=3, name=Zaara, age=23, email=zaara@domain.fr, password=1235]

The number of searching most expensives products operations is : 11

List of most expensives products searched

  Product [ID=5, name=Ordinateur de Luxe, price=100000, expirationString=11/12/30]]
  Product [ID=1, name=cheese omelette, price=10.0, expirationString=11/11/27]]
  Product [ID=0, name=fresh ground steak, price=20.0, expirationString=11/11/26]]
```

Listing 3 – Résultat du parsing du fichier Repository.log

Références bibliographiques

[1] Jean Michel DOUDOUX. *19. Le logging*. URL : https://www.jmdoudoux.fr/java/dej/chap-logging.htm.

[2] Renaud PAWLAK et al. « Spoon : A Library for Implementing Analyses and Transformations of Java Source Code ». In : *Software : Practice and Experience* 46 (2015), p. 1155-1179. DOI : 10.1002/spe.2346. URL : https://hal.archives-ouvertes.fr/hal-01078532/document.

[3] Abdelhak-Djamel SERIAI. *Évolution et maintenance des systèmes logiciels*. Paris : Hermès Science publications Lavoisier, 2014. ISBN : 9782746245549.