Travaux Pratiques 6 – QTL

Dans ce TP, vous allez implémenter et tester un système simple de gestion des utilisateurs en utilisant un **fichier CSV comme base de données**. L'accent sera mis sur l'écriture de **tests d'intégration** pour vérifier le bon fonctionnement du système.

Structure du Projet

```
my-maven-project
├── src
│   ├── main/java/com/example
│   │   ├── User.java
│   │   ├── UserService.java
│   ├── test/java/com/example
│   │   ├── UserServiceIT.java  <-- Test d'intégration
├── users.csv
├── pom.xml
```

Création du Fichier CSV

Créez un fichier nommé users.csv, ce fichier représente une base de données simple avec trois champs : ID, Nom et Email.

```
1,John Doe,john@example.com
2,Jane Smith,jane@example.com
3,Houssam Kanso,houssam@example.com
```

Création de la Classe User

Créez un fichier User.java dans src/main/java/com/example/ :

```
package com.example;

public class User {
    private int id;
    private String name;
    private String email;

    public User(int id, String name, String email) {
        this.id = id;
        this.name = name;
        this.email = email;
    }

    public int getId() { return id; }
    public String getName() { return name; }
    public String getEmail() { return email; }
}
```

Implémentation du UserService pour Lire le CSV

Créez un fichier UserService.java dans src/main/java/com/example/ :

```
package com.example;

import com.opencsv.CSVReader;
import com.opencsv.exceptions.CsvException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class UserService {
    private static final String CSV_FILE = "users.csv";

    public List<User> getUsers() {
        List<User> users = new ArrayList<>();
        try (CSVReader reader = new CSVReader(new FileReader(CSV_FILE))) {
            List<String[]> records = reader.readAll();
            for (String[] record : records) {
                users.add(new User(Integer.parseInt(record[0]), record[1], record[2]));
            }
        } catch (IOException | CsvException e) {
            e.printStackTrace();
        }
        return users;
    }

    public User getUserById(int id) {
        return getUsers().stream().filter(user -> user.getId() == id).findFirst().orElse(null);
    }
}
```

Configuration du Plugin Failsafe

Modifiez votre fichier pom.xml pour inclure le **plugin Failsafe** :

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-failsafe-plugin</artifactId>
      <version>3.0.0-M8</version>
      <executions>
        <execution>
          <id>integration-tests</id>
          <goals>
            <goal>integration-test</goal>
            <goal>verify</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

Écriture des Tests d'Intégration

Dans src/test/java/com/example/, créez un fichier UserServiceIT.java. pour faire les tests d'intégration.

Pour les tests d'intégration, il est normal d'avoir plusieurs asserts dans le même test.

Vous devez implémenter les tâches suivantes :

1. **Test de récupération de tous les utilisateurs** : Vérifier que la liste des utilisateurs n'est pas vide et contient exactement 3 éléments.
2. **Test de récupération d'un utilisateur par ID** : Vérifier que l'utilisateur avec l'ID 2 est bien "Jane Smith".
3. **Test avec un ID invalide** : Vérifier que la méthode retourne null ou une valeur appropriée lorsqu'un ID inexistant est fourni.
4. **Modification du fichier CSV** : Ajouter un champ "solde" pour chaque utilisateur dans le fichier CSV.
5. **Ajout d'un fichier CSV pour les produits** : Créer un fichier CSV avec la liste des produits, leurs quantités et prix respectifs.
6. **Test d'achat de produit** : Écrire un test qui prend la quantité de produit et l'ID de l'utilisateur et vérifie si cet utilisateur peut acheter les produits en fonction de la somme d'argent disponible et la quantité disponible.
7. **Test d'intégration de votre choix**