R4.02 – TP 2

TESTS AVEC ASSERTJ

Table des matières

Objectif 2
A lire avec attention 2
Travaux Pratiques 2
Création du dépôt et ajout des sources 2
Ajout de la chaine CI/CD 2
Ajout d'un jeu de données aléatoires individuelles 3
Test simple de conversion de texte en objet 3
Test du nombre d'individus chargés depuis le fichier CSV 4
Test des noms des 5 premiers individus chargés depuis le fichier CSV 4
Test des tailles des individus chargés depuis le fichier CSV 4
Test des titres des individus chargés depuis le fichier CSV 5
Test des dates de naissance des individus chargés depuis le fichier CSV 5
Test de répartition des titres des individus chargés depuis le fichier CSV 5

Objectif

L'objectif est de réaliser des tests avec AssertJ (https://assertj.github.io/doc/) et plus particulièrement la partie Core (https://assertj.github.io/doc/#assertj-core). AssertJ est intégrable grâce à une dépendances Maven : https://mvnrepository.com/artifact/org.assertj/assertj-core

A lire avec attention

Ne pas oublier de démarrer Docker.

Les énoncés sont à lire avec attention : le respect ou non-respect des indications sera pris en compte dans la notation

Les principales informations pour réaliser le TP sont dans le support de cours. Il n'est pas exclu de compléter/confirmer certaines de ces informations grâce à Internet.

Travaux Pratiques

Création du dépôt et ajout des sources

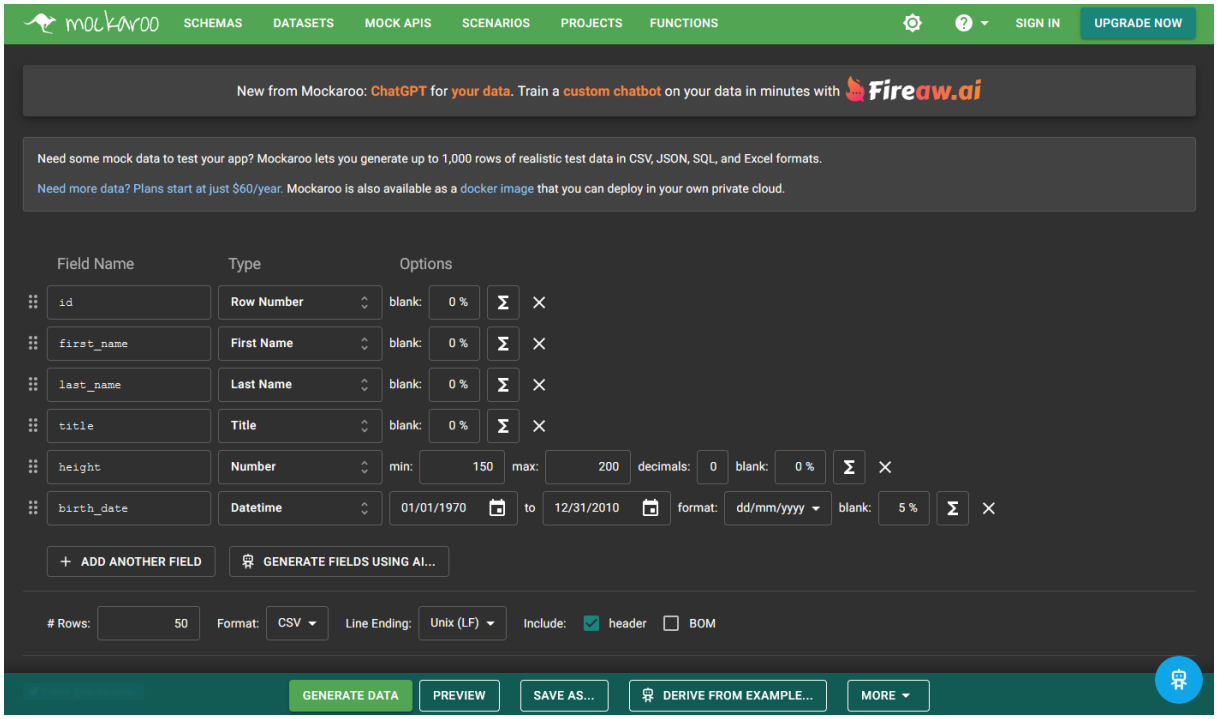
1. Créer un dépôt « `TP2` » dans le groupe précédemment créé et le cloner.
2. Ajouter les sources contenues dans « `TP2.zip` » et les pousser sur GitLab

Ajout de la chaine CI/CD

1. Ajouter une chaîne d'intégration continue pour exécuter les tests dans le GitLab Runner sous Docker avec Maven.
2. Commiter et vérifier l'exécution de la chaine CI/CD.
3. Si création de la chaine CI/CD depuis GitLab, récupérer les sources depuis GitLab pour ne pas avoir de conflits.

Ajout d'un jeu de données aléatoires individuelles

1. Aller sur le site Mockaroo pour générer un fichier CSV (selon les champs ci-dessous) avec 50 valeurs



2. Placer le fichier généré dans le dossier « `src/test/resources` ».

```
id,first_name,last_name,title,height,birth_date
1,Deana,Syplus,Mr,187,26/07/1993
2,Gunilla,Kidder,Dr,178,07/04/1975
3,Leda,Kleehuhler,Honorable,176,
4,Farrand,Drayton,Honorable,174,09/07/1990
5,Bat,Baudou,Mrs,182,
6,Donny,Ravens,Rev,160,18/11/1988
```

3. Nommer ce fichier avec votre nom/prénom et l'extension CSV.
4. Commiter et vérifier l'exécution de la chaine CI/CD

Test simple de conversion d'objet en texte

Indice(s) : isEqualTo

1. Créer une classe « `IndividuUtilsTest` » dans le package « `iut.java.tests.unit` » du dossier « `src/test/java` »
2. Dans cette classe, créer une méthode de test nommée « `testGetLine` » sur la méthode « `getLine` » de « `IndividuUtils` »
3. Cette méthode devra découper les parties « ARRANGE », « ACT » et « ASSERT ».
4. Cette méthode devra être commentée (pas besoin de faire un roman)
5. Pour le jeu d'essai (nom et prénom, …), prendre par exemple un personnage ou un artiste (ou soi-même).
6. Commiter et vérifier l'exécution de la chaine CI/CD

Test simple de conversion de texte en objet

Indice(s) : isEqualTo et usingRecursiveComparison

1. Toujours dans cette classe de test, créer une méthode de test **avec AssertJ (et notamment l'assertion d'égalité basée sur la comparaison récursive)** nommée « `testGetIndividu` » sur la méthode « `getIndividu` » de la classe « `IndividuUtils` »
2. Cette méthode devra découper les parties « ARRANGE », « ACT » et « ASSERT ».
3. Cette méthode devra être commentée (pas besoin de faire un roman)
4. Il est possible (voire recommandé) d'utiliser une ligne du fichier CSV généré par Mockaroo pour faire le test.
5. Toutes les valeurs de l'individu doivent être vérifier.
6. Commiter et vérifier l'exécution de la chaine CI/CD

Test du nombre d'individus chargés depuis le fichier CSV

Indice(s) : hasSize

1. Créer une classe « `IndividuServiceTest` » dans le package « `iut.java.tests.integration` » du dossier « `src/test/java` »
2. Dans cette classe, créer une méthode de test **avec AssertJ** nommée « `testCountLoadedIndividus` » sur la méthode « `getIndividusList` » de la classe « `IndividuService` » après l'avoir instancié en utilisant le fichier généré. (Exemple : « `IndividuService service = new IndividuService("Mirko_DRAGIC.csv");` » si votre fichier s'appelle « `Mirko_DRAGIC.csv` »)
3. Le test doit vérifier le nombre d'individus de la liste
4. Cette méthode devra découper les parties « ARRANGE », « ACT » et « ASSERT ».
5. Cette méthode devra être commentée (pas besoin de faire un roman)
6. Commiter et vérifier l'exécution de la chaine CI/CD

Test des noms des 5 premiers individus chargés depuis le fichier CSV

Indice(s) : extracting et startsWith

1. Toujours dans cette classe de test, créer une méthode de test **avec AssertJ** nommée « `testLoadedIndividusName` » sur la méthode « `getIndividusList` » de « `IndividuService` »
2. Le test doit vérifier le nom des 5 premiers individus de la liste
3. Le code de l'instanciation de la classe « `IndividuService` » ne doit pas être présent dans chaque méthode de test. Il faut qu'il soit une seule fois dans la classe de test pour être exécuté avant chaque test.
4. Cette méthode devra découper les parties « ARRANGE », « ACT » et « ASSERT ».
5. Cette méthode devra être commentée (pas besoin de faire un roman)
6. Commiter et vérifier l'exécution de la chaine CI/CD

Test des tailles des individus chargés depuis le fichier CSV

Indice(s) : allMatch

1. Toujours dans cette classe de test, créer une méthode de test **avec AssertJ** nommée « `testLoadedIndividusHeight` » sur la méthode « `getIndividusList` » de « `IndividuService` »

2. Le test doit vérifier que tous les individus de la liste ont une taille (« `height` ») comprise entre 150 et 200.
3. Cette méthode devra découper les parties « ARRANGE », « ACT » et « ASSERT ».
4. Cette méthode devra être commentée (pas besoin de faire un roman)
5. Commiter et vérifier l'exécution de la chaine CI/CD

Test des titres des individus chargés depuis le fichier CSV

Indice(s) : containsOnly

1. Toujours dans cette classe de test, créer une méthode de test **avec AssertJ** nommée « `testLoadedIndividusTitle` » sur la méthode « `getIndividusList` » de « `IndividuService` »
2. Le test doit vérifier les différents titres de l'intégralité des individus de la liste. Tous les titres présents doivent être vérifiés et cela qu'ils soient présents une fois ou plusieurs fois.
3. Cette méthode devra découper les parties « ARRANGE », « ACT » et « ASSERT ».
4. Cette méthode devra être commentée (pas besoin de faire un roman)
5. Commiter et vérifier l'exécution de la chaine CI/CD

Test des dates de naissance des individus chargés depuis le fichier CSV

Indice(s) : filteredOn, LocalDate.of(), LocalDate.isAfter(), LocalDate.isBefore()

1. Toujours dans cette classe de test, créer une méthode de test **avec AssertJ** nommée « `testLoadedIndividusBirthDate` » sur la méthode « `getIndividusList` » de « `IndividuService` »
2. Le test doit vérifier :
    - Le nombre d'individus avec la date de naissance qui est « `null` »
    - Le nombre d'individus avec la date de naissance qui antérieure au « `01/01/2000` »
    - Le nombre d'individus avec la date de naissance qui postérieure au « `01/01/2000` »
3. Cette méthode devra découper les parties « ARRANGE », « ACT » et « ASSERT ».
4. Cette méthode devra être commentée (pas besoin de faire un roman)
5. Commiter et vérifier l'exécution de la chaine CI/CD

Test de répartition des titres des individus chargés depuis le fichier CSV

Indice(s) : containsOnly

1. Toujours dans cette classe de test, créer une méthode de test **avec AssertJ** nommée « `testLoadedIndividusTitleRepartition` » sur la méthode « `getTitleRepartition` » de « `IndividuService` »
2. Le test doit vérifier le nombre d'individus par titre (le contenu de la Map retournée par la méthode à tester)
3. Cette méthode devra découper les parties « ARRANGE », « ACT » et « ASSERT ».
4. Cette méthode devra être commentée (pas besoin de faire un roman)
5. Commiter et vérifier l'exécution de la chaine CI/CD