AssertJ & tests unitaires

Joël Costigliola
Développeur java freelance

@JoCosti

AssertJ : historique

- Crée en mars 2013
- Pourquoi avoir forké Fest Assert ?
  - Projet gelé
  - Peu ouvert à la communauté
  - Volonté de réduire le nombre d'assertions
- Philosophie d'AssertJ
  - Offrir un grand nombre d'assertions
  - Communtity driven !

AssertJ assertions

- Spécifique au type testé :
  - String, Date, Collection, File, Exception, ...
- Facile à utiliser :
  - IDE friendly
  - Auto discovery
  - Point d'entré unique : classe Assertions
  - little details (date
- Proche du langage naturel
- Documentées (javadoc et assertj.org)

AssertJ assertions

- Message d'erreurs multi-lignes à la mockito
- Extracting feature
- Soft assertions : lister toutes les erreurs d'un test
- Lenient equals assertions
- BDD assertions

AssertJ modules

- AssertJ modules:
  - assertj-core : assertions pour les types du JDK
  - assertj-guava : assertions pour Guava
  - assertj-joda-time : assertions pour Joda Time
  - assertj-neo4j : assertions pour Neo4J (Florent Biville)
  - assertj-examples : executable doc
  - en prévision : assertj-swing (Christian Rösch)

Exemple

```
// assertions simple
assertThat(gandalf.getName()).isEqualTo("Gandalf");

// il y a 3 anneaux elfiques dans le Seigneur des Anneaux
List<Ring> elvesRings = newArrayList(vilya, nenya, narya);

// assertions sur les collections
assertThat(elvesRings).isNotEmpty()
                      .hasSize(3)
                      .contains(nenya, vilya, narya)
                      .doesNotContain(oneRing);
```

AssertJ : extensions

- Utilisation de Condition
- Assertions pour ses propres classe (DSL)
- Générateur d'assertions :
  - Se base sur les properties de vos classes métiers
  - Se lance avec maven
  - Gain de temps important lorsque l'on veut écrire des assertions métiers pour un grand nombre de classes
  - Plugin eclipse/idea prévu

Démo !

Code source dans assertj-examples – branche octo

https://github.com/joel-costigliola/assertj-examples

AssertJ : limites

- Étendre les assertions des types existants :
  - Faisable mais difficulté sur le point d'entrée unique
- Fournir trop d'assertions peut perdre l'utilisateur
- Antipattern : assertThat(1 == 2)
- Messages d'erreur : à mieux intégrer dans les IDE
- Assertions sur les exceptions perfectibles
  - Utiliser : http://code.google.com/p/catch-exception/

Tests

- Utiliser des librairies :
  - assertj:)
  - mockito
  - ...
- Code coverage : pour voir ce que l'on n'a pas testé
- Si on manque de temps :
  - Tester le code commun
  - Tests de plus haut niveau (fonctionel / intégration)Code coverage : pour voir ce que l'on n'a pas testé