Externalisation des données

Jean-Marc Farinone

Maître de Conférences
Conservatoire National des Arts et Métiers
CNAM Paris (France)

Etat de nos connaissances

- Jusqu'à alors, on a des classes de tests contenant des programmes (méthodes) de tests avec leurs propres données
- On ne sait qu'écrire de nouvelles méthodes de tests si on change de données
- On aimerait avoir, pour un programme de tests, un jeu (ensemble) de données qu'on va utiliser pour tester le programme
- Concrètement, on va indiquer qu'une méthode de test va prendre ces données à "l'extérieur" de la classe
- On veut faire de "l'externalisation de données"

Lancer plusieurs fois les tests : syntaxe

- Il y a seulement trois contraintes
- La classe de test doit avoir un et un seul constructeur. Il fabriquera l'objet de la classe de test sur lequel les tests seront lancés
- La classe de test doit être annotée `@RunWith(Parameterized.class)`
- La classe de test doit posséder la méthode :
  `@Parameters`
  `public static Collection<Object[]> data()`
- Par exemple :

```
@RunWith(Parameterized.class)
public class RationnelTest {
// Le seul et unique constructeur de cette classe de test
public RationnelTest() {...}
...
@Parameters
public static Collection<Rationnel[]> data() { ...}
}
```

La méthode `... data()`

- Sa syntaxe est :
```
@Parameters
public static Collection<Object[ ]> data()
```
- Elle retourne donc une Collection de tableaux ! C'est à dire une suite de tableaux !
- Chaque élément de cette Collection est donc un tableau d'`Object`
- Un tel tableau d'`Object` possède les arguments pour le seul et unique constructeur de cette classe de test
- Tous les tableaux retournés doivent donc avoir le même nombre d'élément : bons sens
- Finalement on obtiendra et on va ainsi lancer `nombreDeTableauRetournéParMaMethodeData * NombreDeMethodesDeTest` tests !

Externationalisation des données

- Si on met le jeu de données dans le code, on n'a guère avancé ! car si on veut changer (ajouter, enrichir) ce jeu de données, il faudra avoir le code du test, connaître Java, ne pas dégrader ce qui a déjà été fait, etc. ;-)
- On veut mettre ces données à l'extérieur du programme. Dans un fichier XML (bien pratique pour la compréhension)
- Pour cela on va créer :
  - un schéma XML qui décrit la structure des données de test,
  - un fichier XML contenant les valeurs des données de test
- La méthode `data()` ira lire le fichier XML, va transformer le contenu en une `Collection` de tableaux et retourner cette `Collection` (comme d'hab)

XML Schema

- Pour définir le format des données d'un fichier XML, on précise sa structure en indiquant qu'il vérifie les caractéristiques décrites dans un fichier XML Schema
- Dans les premières versions d'XML, on le précisait à l'aide de fichier dtd (Document Type Definition). Le principal défaut des dtd est que la syntaxe d'une dtd n'est pas en XML contrairement au fichier XML Schema
- Un fichier XML permet de définir, entre autre, des "types" de données
- On a des outils pour construire de tels fichiers (voir exercice sous eclipse)

Exemple de XML Schema

- Voici un exemple définissant certains "types" de données

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.example.org/RationnelSchema"
xmlns:tns="http://www.example.org/RationnelSchema" elementFormDefault="qualified">

<complexType name="Rationnel">
     <attribute name="numerateur" type="int"></attribute>
     <attribute name="denominateur" type="int"></attribute>
</complexType>

<complexType name="Data">
     <sequence>
          <element name="r1" type="tns:Rationnel"></element>
          <element name="r2" type="tns:Rationnel"></element>
          <element name="resultAddition" type="tns:Rationnel"></element>
          <element name="resultSoustraction" type="tns:Rationnel"></element>
     </sequence>
</complexType>

<element name="DataSet">
     <complexType>
          <sequence>
               <element name="Data" type="tns:Data" maxOccurs="unbounded"
                    minOccurs="0"></element>
          </sequence>
     </complexType>
</element>
</schema>
```

Données XML

- Ayant le schéma XML (la structure) des données, il faut construire des données comme jeu de tests. On aura par exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:Dataset xmlns:tns="http://www.example.org/testSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.example.org/testSchema testSchema.xsd ">

<tns:data>
     <tns:r1 denominateur="3" numerateur="2" />
     <tns:r2 denominateur="3" numerateur="4" />
     <tns:resultAddition denominateur="1" numerateur="2" />
     <tns:resultSoustraction denominateur="3" numerateur="-2" />
</tns:data>

<tns:data>
     <tns:r1 denominateur="9" numerateur="2" />
     <tns:r2 denominateur="3" numerateur="1" />
     <tns:resultAddition denominateur="9" numerateur="5" />
     <tns:resultSoustraction denominateur="9" numerateur="-1" />
</tns:data>

</tns:Dataset>
```

- Que représente ces données ?

Correspondance notions XML - code Java

- Il faut alors construire des classes qui manipulent les données XML
- Là aussi, des outils pour cette construction, sont donnés
- Par la suite à l'aide de ces classes, la méthode `data()`, pourra utiliser les données XML
- Et voilà !

Exercice

- Faire des données de tests externationalisées pour une classe à tester

Fin