

Exercice : Parser

# Enoncé

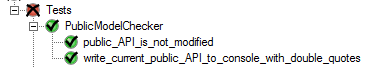
L'objectif de cet exercice est de comprendre le fonctionnement d’un parser JSON et d'en implémenter un qui le renvoie JSON à sa structure en C# spécifié sous la forme de tests unitaires. Autrement dit : l'objectif de l'exercice est de faire passer en vert un jeu de tests.

Vous disposez pour cela d'une solution comprenant 3 projets :

* ITI.Parser.Tests, contient les tests unitaires
* ITI.Parser.Model, un modèle à implémenter, constitué exclusivement d’abstractions
* ITI.Parser.Impl, une implémentation minimale du modèle (le code écrit ne permet rien de plus que de pouvoir compiler sans erreur)

Pour faire tourner les tests unitaires il vous suffit de configurer le projet ITI.Parser.Tests en tant que projet de démarrage puis d'exécuter la solution.

Comme vous pouvez le constater, pour le moment, tous les tests sont rouges, à l’exception de 2 d’entre eux :



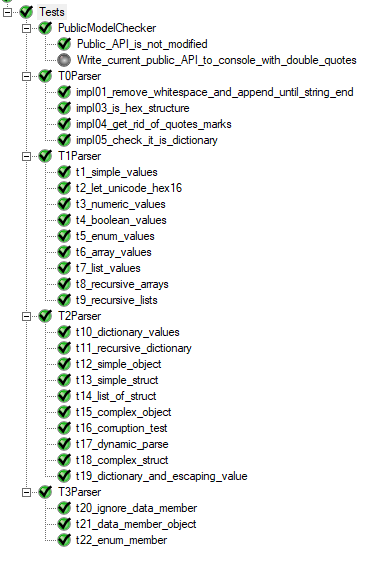
Ces deux tests analysent les objets implémentés dans ITI.Parser.dll et en extraient l’API public sous forme XML. Le deuxième se contente d’afficher l’XML dans la console, le premier, plus intéressant, compare l’API public avec celle d’origine qui est mémorisée : si d’aventure, vous altérez l’API public des objets (en ajoutant une méthode, une propriété ou un champ public), il passe au rouge : vous devez vous assurer qu’il reste au vert, votre implémentation ne doit pas modifier l’API publique !

Pour les autres tests, à vous de faire en sorte qu'ils passent en vert. Pour cela, vous avez le droit de faire ce que bon vous semble dans le projet ITI.Parser (modifier l'implémentation des méthodes, des propriétés ou des constructeurs, ajouter des membres *internal* ou *private*, créer de nouveaux types etc.) En revanche, vous ne devez pas modifier le projet ITI.Parser.Tests.

Les tests unitaires sont là pour spécifier de façon détaillée les fonctionnalités attendues. Cependant voici une courte description de ce qui est attendu :

* Il faut retirer les espaces de la chaîne de caractères que reçoit le parser
* On peut parser des string
* On peut parser des numériques :
  + entiers
  + flottants
* On peut échapper les caractères spéciaux
* On peut parser des objets/classes plus complexes :
  + Enum
  + Array
  + List
  + Dictionary
  + Structure
  + Objets dans des objets, etc…

Il y a 26 tests à passer au vert :



Au boulot !