Cyril Seguenot

2016

Résumé

Support d’accompagnement d’une formation à la programmation avec C# faite en présentiel

Tests unitaires

Support de cours

Toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l’auteur ou de ses ayants droit ou ayants cause est illicite selon le Code de la propriété intellectuelle (Art L 122-4) et constitue une contrefaçon réprimée par le Code pénal.

Seules sont autorisées (Art L122-5) les copies ou reproductions strictement réservées à l’usage privé du copiste et non destinées à une utilisation collective, ainsi que les analyses et courtes citations justifiées par le caractère critique, pédagogique ou d’information de l’œuvre à laquelle elles sont incorporées, sous réserve, toutefois, du respect des dispositions des articles L 122-10 à L 122-12 du même Code, relatives à la reproduction par reprographie.

©Cyril Seguenot 2016

SOMMAIRE

[1 Introduction 4](#_Toc478336672)

[2 Création d’une suite de tests dans Visual Studio 4](#_Toc478336673)

[2.1 Ajout d’un projet de tests 4](#_Toc478336674)

[2.2 Création d’une classe de tests 5](#_Toc478336675)

[3 Bonnes pratiques 6](#_Toc478336676)

[4 Références bibliographiques 7](#_Toc478336677)

# Introduction

Le test d’une application est une étape essentielle pour s’assurer de sa fiabilité avant sa mise en production.

Les tests sont généralement réalisés au fur et à mesure des développements, pour ne pas avoir à tester l’intégralité d’une version de l’application juste avant sa sortie. Dans une approche itérative, les tests font partie de chaque itération.

Les tests peuvent être de plusieurs natures :

* Alphatests réalisés par le développeur au fur et à mesure ou à la fin d’un développement
* Tests réalisés par des testeurs (éventuellement développeurs) sur base d’un plan de test complet écrit au préalable, et portant principalement sur les aspects fonctionnels de l’application
* Tests automatisés mis au point par des développeurs/testeurs, et joués de façon automatique. Ces tests permettent de s’assurer que les résultats produits par l’application sont conformes à un jeu de résultats défini à l’avance (appelé master). Ils permettent :
  + Au départ, la mise au point du code
  + Ensuite, de façon récurrente, de s’assurer que le code n’a pas subi de régressions suite à d’éventuelles évolutions / corrections.

Les tests unitaires s’inscrivent dans la dernière catégorie.

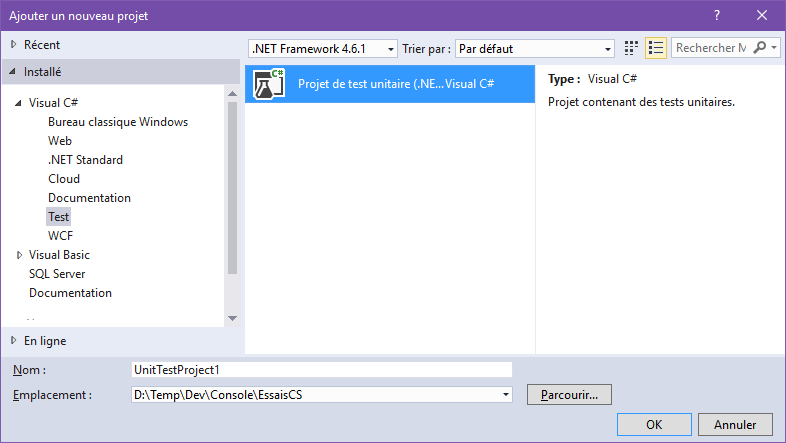
# Création d’une suite de tests dans Visual Studio

## Ajout d’un projet de tests

**Une suite de tests** est un enchaînement de tests unitaires appliqués aux classes présentes dans les projets d’une solution.

**Un test unitaire** est une méthode dont le but est de tester une petite unité de code homogène et isolable, et qui renvoie un résultat OK ou erreur. Typiquement, un test unitaire teste une seule méthode à la fois, mais il peut aussi en tester plusieurs.

Une suite de tests est créée dans Visual Studio au moyen d’un **projet de tests unitaires**, ajouté dans la solution contenant les projets à tester. L’image ci-dessous montre l’ajout d’un projet de tests :



Le projet de tests unitaire doit référencer tous les projets dont il souhaite tester les classes.

## Création d’une classe de tests

Voici la procédure pour créer une classe de test :

* Créer une classe de tests avec le nom de la classe à tester précédé de « Test » (ex : TestAnimal) et décorée avec l’attribut [TestClass]
* Ajouter un using sur l’espace de noms de la classe à tester pour appeler facilement ses méthodes
* (Optionnel) Créer une méthode statique décorée avec l’attribut [ClassInitialize] pour l'initialisation des tests avant chaque méthode de test
* (Optionnel) Créer une méthode statique décorée avec l’attribut [TestInitialize] pour l'initialisation des tests avant chaque méthode de test
* Créer des méthodes de test décorées avec l’attribut [TestMethode], avec le retour void et sans paramètres.
* (Optionnel) Créer une méthode décorée avec l’attribut [TestCleanup] pour le nettoyage à faire après chaque méthode de tests
* (Optionnel) Créer une méthode décorée avec l’attribut [ClassCleanup] pour le nettoyage à faire après chaque méthode de tests

Dans les méthodes de tests, utiliser les méthodes statiques de la classe Assert.

On peut utiliser un attribut placé au-dessus de la méthode pour tester qu'elle renvoie bien une exception du type souhaité. Ex :

[ExpectedException(typeof(FormatException))]

Pour lancer la suite de tests, cliquer sur le Menu Test \ Exécuter.

# Bonnes pratiques

Un test unitaire doit être véritablement unitaire. La classe testée doit être isolable, et ses méthodes doivent être testées une par une, ou en nombre réduit si on doit en tester plusieurs à la fois.

Ne testez qu’un comportement à la fois. Soyez raisonnable et gardez le test simple, lisible et concentré sur ce comportement.

Si une classe est difficile à tester, il est temps de faire du refactoring. Est-elle trop volumineuse ? Présente-t-elle trop de dépendances ? Profitez de l’occasion pour la découper et déplacer du code dans des classes annexes.

Il faut s’efforcer de faire des tests courts, car de nombreux tests exécutés les uns à la suite des autres peuvent finir par prendre beaucoup de temps.

Le code d’un test unitaire fait partie du code applicatif. Il doit donc, à l’image du reste de l’application, respecter des conventions de code.

Ne pas essayer de couvrir 100% du code par des tests unitaires. Choisir les parties de code les plus pertinentes à tester. Entre autres, les tests difficiles à réaliser par un humain tels que la vérification le résultats de calculs complexes, ou de requêtes complexes sur une base de données, sont de bons candidats.

Ne pas remettre le développement des tests unitaires à « plus tard ». Si on n'utilise pas l’approche TDD (Test Driven Development), développer tout de même les tests unitaires le plus tôt possible.

# Références bibliographiques

Procédure [MSDN](https://msdn.microsoft.com/fr-fr/library/ms182532.aspx) pas à pas sur la création et l’exécution de tests unitaires

Article « [Les 10 commandements des tests unitaires](http://blog.xebia.fr/2008/04/11/les-10-commandements-des-tests-unitaires/) » sur le blog Xebia