Cours n°13

Programme du cours

Activité	Durée
Introduction au tests unitaires	15m
Création d'un test	30h

Auteur : Eric-Nicolas Preisig

Les tests unitaires

Objectif

Comprendre l'utilité et comment utiliser les tests unitaires

Introduction

Les tests unitaires sont un type de tests logiciels où des unités individuelles de code source, telles que des fonctions ou des méthodes, sont testées pour vérifier qu'elles se comportent comme prévu. Ils font partie des méthodologies de développement de logiciels et sont généralement écrits et maintenus par les développeurs eux-mêmes.

Voici quelques points clés sur les tests unitaires :

1. Isolation : Un test unitaire doit tester une unité de code de manière isolée, indépendamment des autres parties du système. Cela signifie généralement que les dépendances externes, comme une base de données ou une API, sont simulées ou "mokées" afin de ne pas affecter le test.

2. Cohérence : Les tests unitaires doivent être cohérents et produire le même résultat chaque fois qu'ils sont exécutés, quel que soit l'environnement.

3. Vitesse : Les tests unitaires doivent être rapides à exécuter. Comme ils sont généralement exécutés très fréquemment lors du développement, leur lenteur peut entraîner une baisse de productivité.

4. Automatisation : Les tests unitaires sont généralement automatisés et intégrés dans le processus de construction et de déploiement du logiciel.

5. Couverture de code : L'objectif est d'obtenir une couverture de code aussi complète que possible avec les tests unitaires, afin de détecter les erreurs potentielles dans tous les chemins d'exécution du code.

Les tests unitaires sont un outil précieux pour maintenir la qualité du code, faciliter les refactoring, et aider à prévenir les régressions (lorsqu'une fonctionnalité précédemment fonctionnelle cesse de fonctionner comme prévu). Ils sont généralement complétés par d'autres types de tests, comme les tests d'intégration et les tests d'acceptation, pour tester le système à différents niveaux.

Par ChatGPT

Auteur : Eric-Nicolas Preisig

Mise en place

- Nous allons créer un nouveau projet console nommé « Calculatrice »
- Ajoutons ensuite une class « Calculator »

```
public class Calculator
{
   public int Add(int a, int b)
   {
      return a + b;
   }

   public int Multiply(int a, int b)
   {
      return a * b;
   }
}
```

- Nous allons créer un nouveau projet de type « Projet de tests MSTest » dans la solution. La plupart du temps le nom du projet est formaté comme ceci : [LeNomDuProjet].Tests
- Ajouter une référence vers le projet « Calculatrice »
- Dans le projet de tests nous allons ajouter une class de test :

```
[TestClass]
public class CalculatorTests
{
   private Calculator _calculator;

   [TestInitialize]
   public void Setup()
   {
      _calculator = new Calculator();
   }

   // Ici, vous ajouterez vos tests
}
```

Mise en place

Pour tester la méthode « Add » de la calculatrice, nous pouvons ajouter les méthodes tests suivantes dans la class « CalculatorTests » :

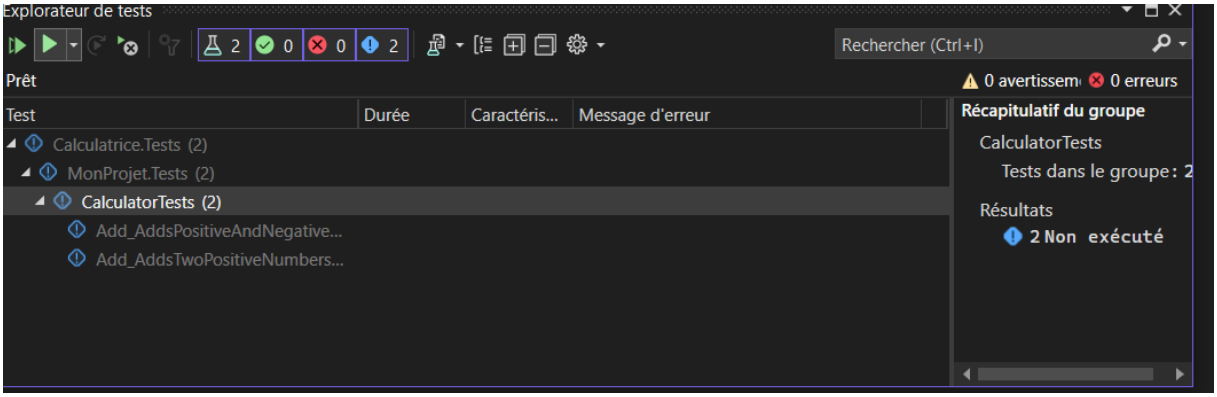
```
[TestMethod]
public void Add_AddsTwoPositiveNumbersCorrectly()
{
   var result = _calculator.Add(5, 10);
   Assert.AreEqual(15, result);
}

[TestMethod]
public void Add_AddsPositiveAndNegativeNumberCorrectly()
{
   var result = _calculator.Add(10, -5);
   Assert.AreEqual(5, result);
}
```

Sur la base de cet exemple, implémenter deux tests pour tester la méthode « Multiply »

Pour lancer les tests :

Aller dans le menu affichage de Visual Studio, puis cliquer sur le menu « Explorateur de tests »



Depuis cette vue vous pouvez soit lancer les tests un à un soit lancer tous les tests de l'application