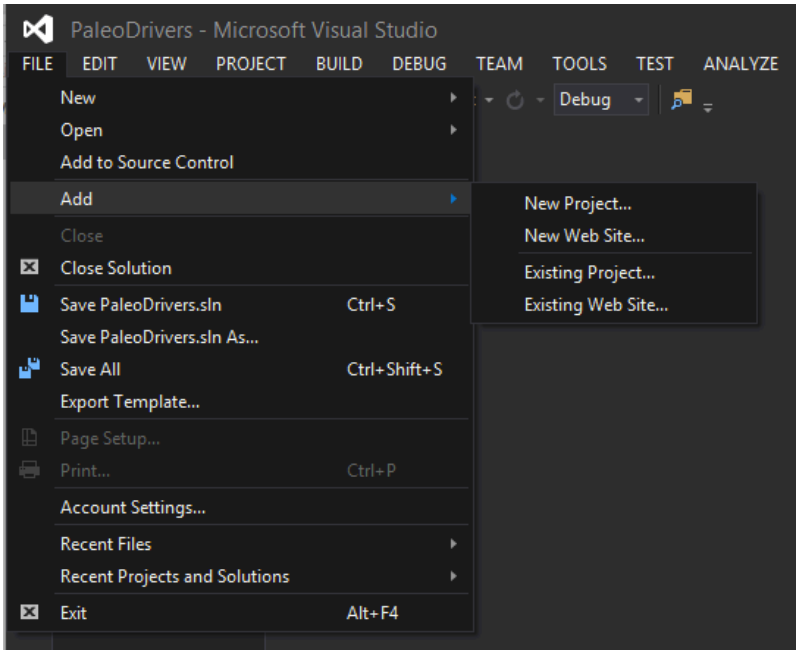
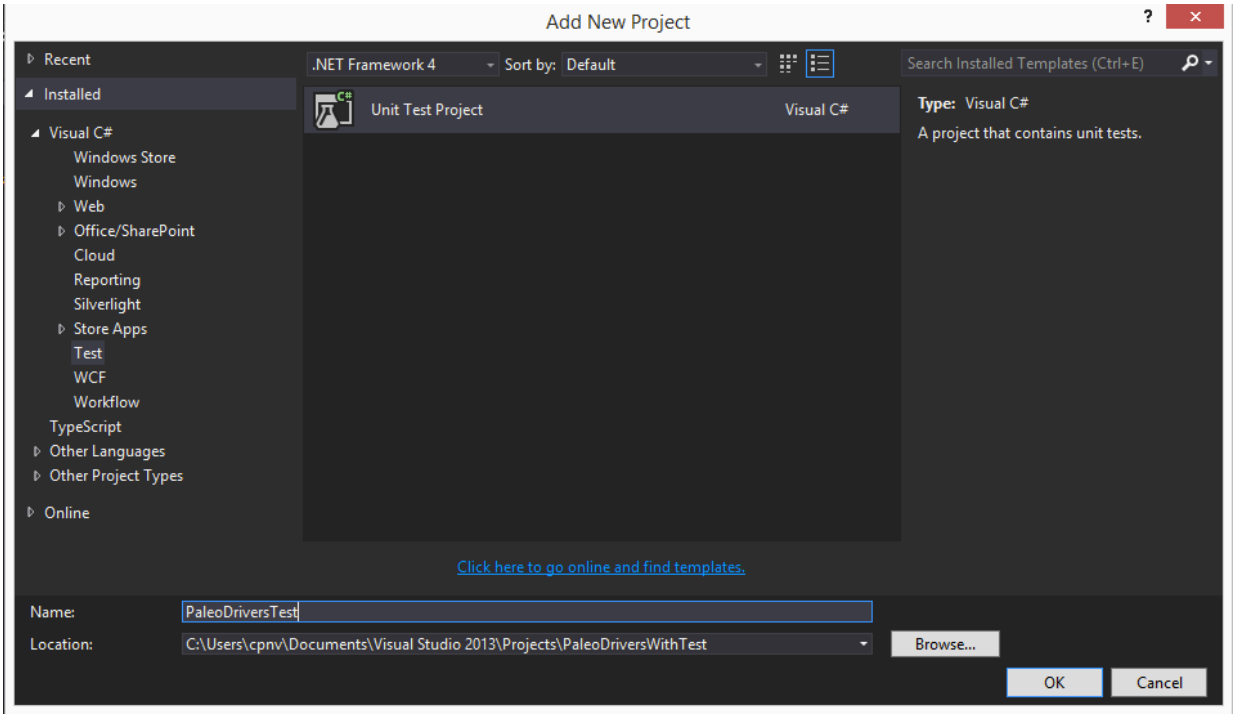
Tests unitaires

1. Ajouter les Tests a un projet existant

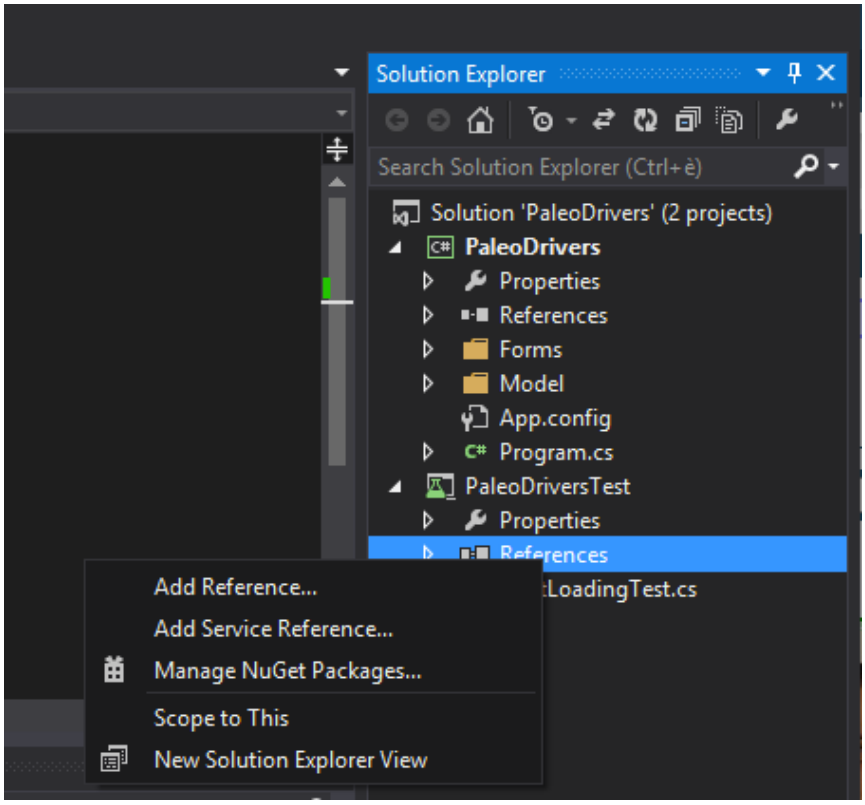
Ouvrez votre projet dans Visual Studio puis ajoutez un projet de test dans la solution existante, ceci en allant dans « FILE » « Add » « New Project » :



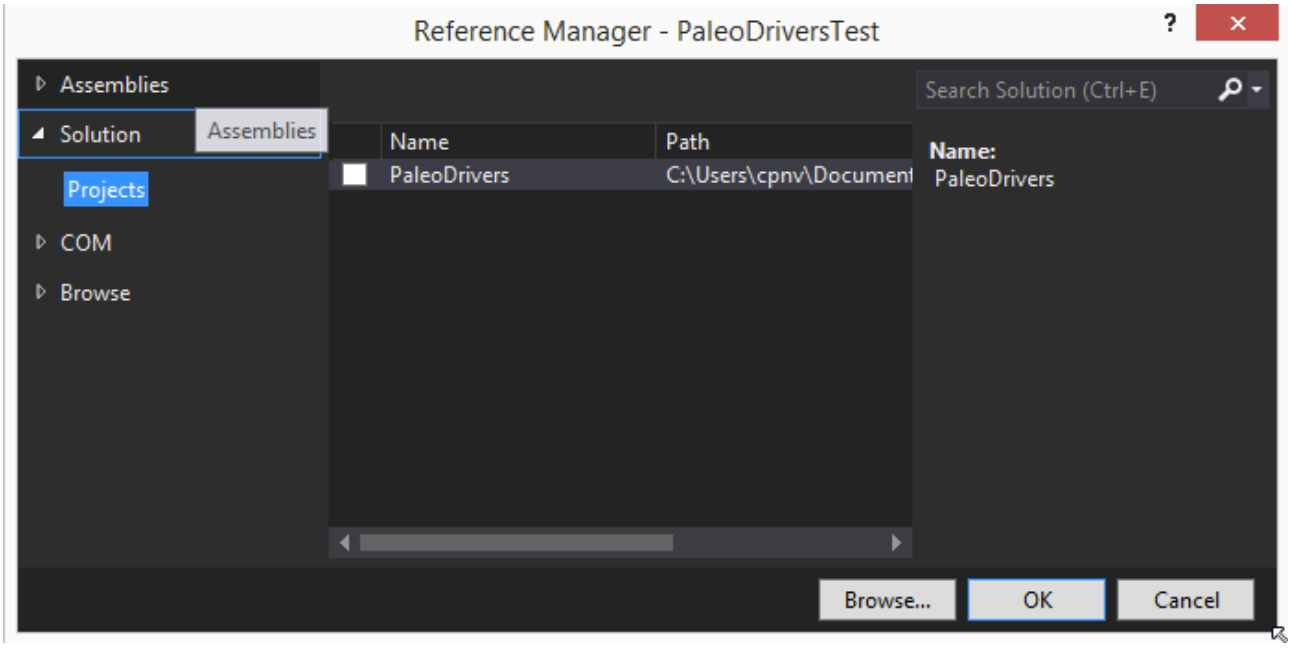
Choisissez à gauche « Visual C# » puis « Test », nommez le comme votre projet actuel avec comme suffixe « Test », exemple : « PaleoDriversTest »



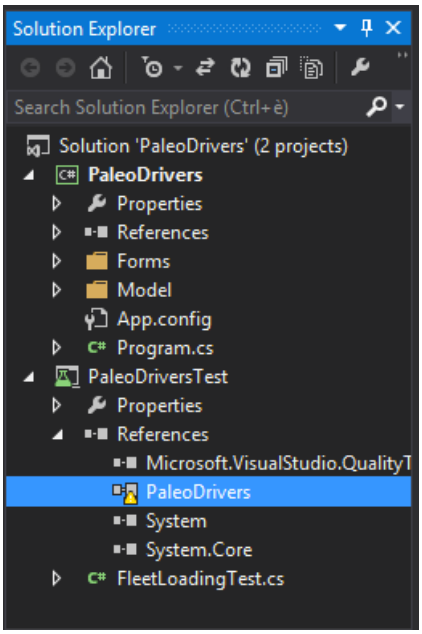
Il faut maintenant lier les deux projets. Dans l'explorateur de projet, ouvrez le projet Test et faites un click droit sur « References », choisissez « Add Reference… »



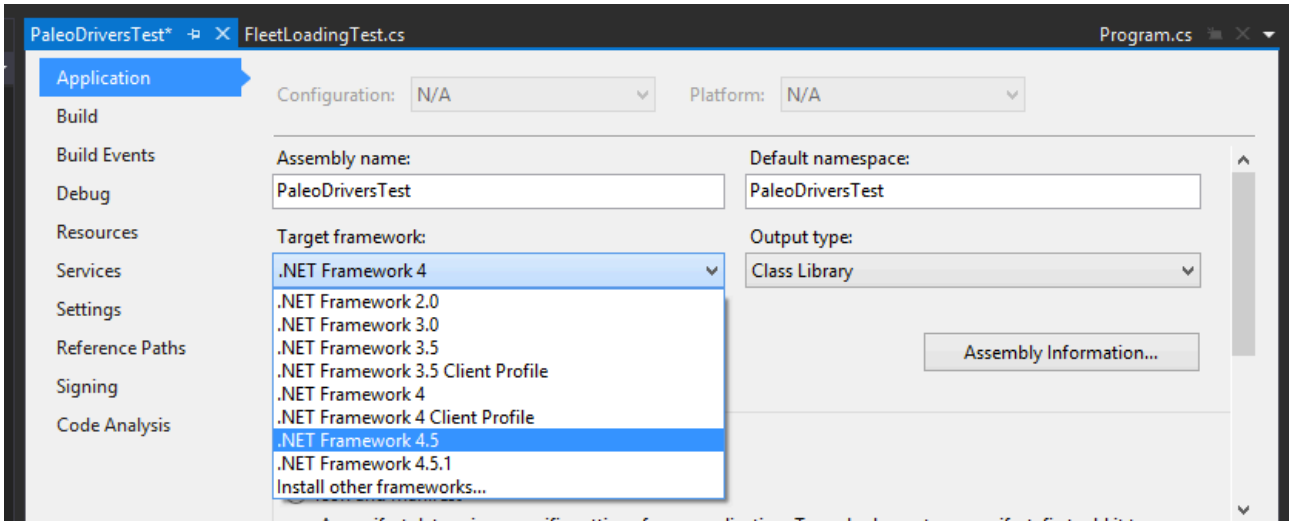
Dans la fenêtre suivante, sélectionnez dans « Projects » le projet que vous voulez tester (c'est le seul autre dans la solution actuelle)



Vous voyez maintenant le lien avec le projet à tester, s'il y a un triangle jaune (signe attention) comme dans la copie d'écran suivante, c'est que les deux projets ne visent pas la même version du Framework .NET.



Dans ce cas, ouvrez les propriétés du projet de test, afin d'ajuster la version du Framework



2. LES TESTS

Le terme test unitaire indique que chaque « unité » du logiciel sera testée de manière séparée.
Il ne reste plus qu'à déterminer les « unités ». Ce n'est pas par hasard que ces tests ne s'appellent pas « Tests de classes » ou « Tests de méthodes » ou « Tests de fichiers ». En effet, l'unité est laissée au libre choix du développeur.

Une fois l'unité déterminée, le principe de découpage dans le projet de test est toujours le même :

- une classe de test par unité
- une méthode de test pour chaque cas test

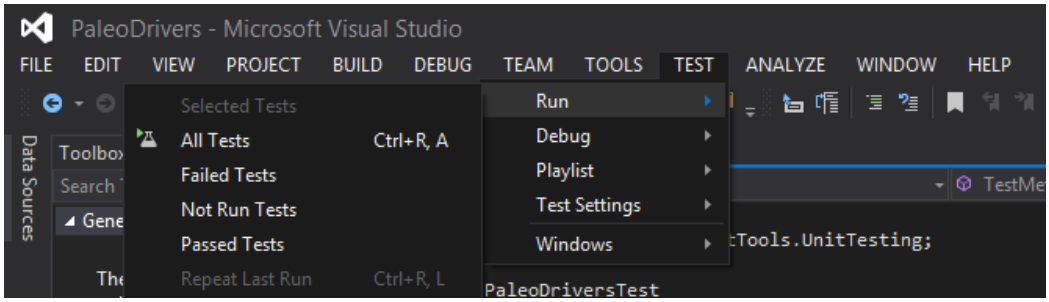
Dans notre exemple, commencez par renommer la classe de test créé automatiquement par le nom de l'unité à tester, nous voulons tester le chargement des véhicules depuis le fichier. Modifiez le nom de la classe de UnitTest1 à FleetLoadingTest (renommez également le nom du fichier de cette classe).

Ensuite pour chaque cas test, écrivez une méthode, nommez la pour identifier le cas test (ces noms sont affichés dans le rapport), ces méthodes doivent avoir l'annotation [TestMethod] :

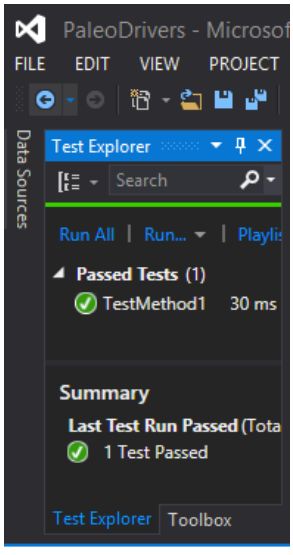
```
{
    [TestMethod]
    public void StringUpcaseConvertAlphabeticCharsToUpper()
    {
        string alpha = "abcd";
        Assert.AreEqual("ABCD", alpha.ToUpper());
    }

    [TestMethod]
    public void StringUpcaseDoesNotTouchDigits()
    {
        string digits = "1234";
        Assert.AreEqual("1234", digits.ToUpper());
    }
```

Pour lancer la batterie de tests, allez dans le menu « Test », « Run », « All Tests »



Les tests sont exécutés et un rapport est affiché à gauche, dans ce rapport vous avez des données statistiques ainsi que le nombre de tests passés (vert) ainsi que le nombre de tests échoués (rouge).



3. LES ASSERTIONS

Pour que le système de test puisse déterminer si les tests passent ou ne passent pas, vous devez affirmer des vérités. Si elles se vérifient, le test **passe**, si ce n'est pas le cas, le test **échoue**.

Ce principe est implémenté dans le code avec les **assertions**. Ce sont des instructions (méthodes de la classe Assert) qui permettent de comparer la valeur **attendue** dans le test par la valeur **réelle** donnée par le code à tester.

Méthode	Pour tester ?
`Assert.AreEqual(expected, tested)`	Les **valeurs** (contenu) sont les mêmes. Ex : int, double ou aussi string
`Assert.AreNotEqual(expected, tested)`	Les **valeurs** sont différentes
`Assert.AreSame(expected, tested)`	Les **objets** sont les mêmes (les références passées pointent vers le même objet)
`Assert.AreNotSame(expected, tested)`	Les **objets** sont différents (les références passées pointent vers des objets différents)
`Assert.IsFalse(tested)`	La valeur passée doit être false
`Assert.IsTrue(tested)`	La valeur passée doit être true
`Assert.IsNull(tested)`	La valeur passée doit être null
`Assert.IsNotNull(tested)`	La valeur passée ne doit pas être null

De plus pour explicitement échouer le test la méthode suivante existe :

```
Assert.Fail(message)
```

Lorsque le code testé doit jeter une exception (dans le comportement attendu), on n'utilise pas de Assert, il faut annoncer ceci par une annotation sur la méthode de test. Vous devez indiquer le type de l'exception attendue, exemple :