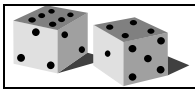
Cours n°1 : Informations de base.



Objectifs : Connaître les éléments de base du C#.
Difficultés : Aucune, hormis beaucoup d'éléments fondamentaux.

Présentation :

C# est une langage dit de "haut niveau". Il se positionne entre le C++, réputé complexe et Visual Basic.

Le C# est souple, c'est-à-dire qu'il peut être exécuté sur la machine sur laquelle il se trouve ou bien transmis par l'intermédiaire du web pour s'exécuter sur un ordinateur distant.
Le C# est aussi puissant que le C++, tant par la richesse de son langage que par sa vitesse d'exécution.
Le C# est facile à utiliser, les commandes générant des erreurs en C++ ont été modifiées pour les rendre plus sures.
Le C# est multi cibles, les programmes peuvent être définis pour s'exécuter en mode console, graphique, sur Pc, sur Pocket-Pc et même sur Linux grâce à mono et à Microsoft qui à fournit les sources de .net (Projet Rotor).
C# est prêt pour Internet, c'est le pivot de la nouvelle stratégie Internet de Microsoft, nommée .NET.
C# est sûr, comme tout langage destiné à une utilisation sur Internet il doit contenir les principes garantissant l'intégrité de la plateforme hôte.
C# est véritablement orienté objet.

Objectif des ces cours :

L'objectif des ces cours est de vous faire découvrir de façon simple les bases de C#, nous n'approfondirons pas les méandres de la programmation en C#, pour aller plus loin voici une liste d'ouvrage que je vous conseille:

Pratique de .NET et C# de Patrick Smacchia édité par O'Reilly
Formation à C# de Tom Archer édité par Microsoft Press.

Méthodologie de travail :

Pour reproduire l'ensemble des cours et des exercices vous pouvez utiliser Visual Studio ou Sharp Develop qui est une interface de développement gratuite téléchargeable sur le Net. Le cours va être illustré avec Sharp Develop que vous pouvez télécharger à l'adresse suivante :

http://www.sharpdevelop.net/OpenSource/SD/Download/

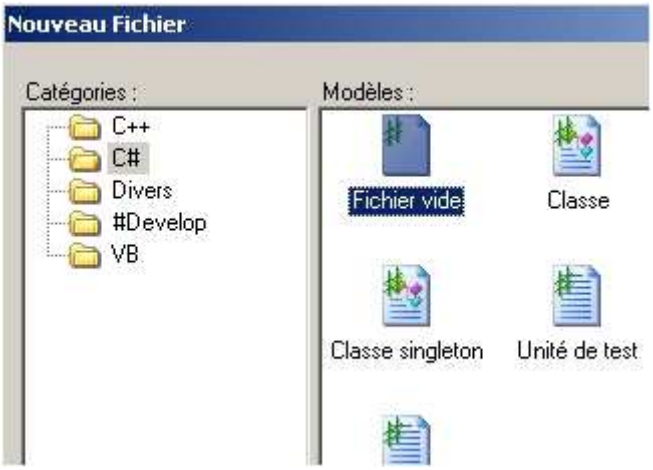
La progression doit être faite à votre rythme, ne passez pas à la leçon suivante tant que vous n'êtes pas à l'aise avec l'actuelle. Un conseil : passez à la leçon suivante lorsque vous êtes capable de réaliser les exercices sans le support. Le temps passé à faire et re-faire est du temps gagné pour la suite.

Dans le vif du sujet.

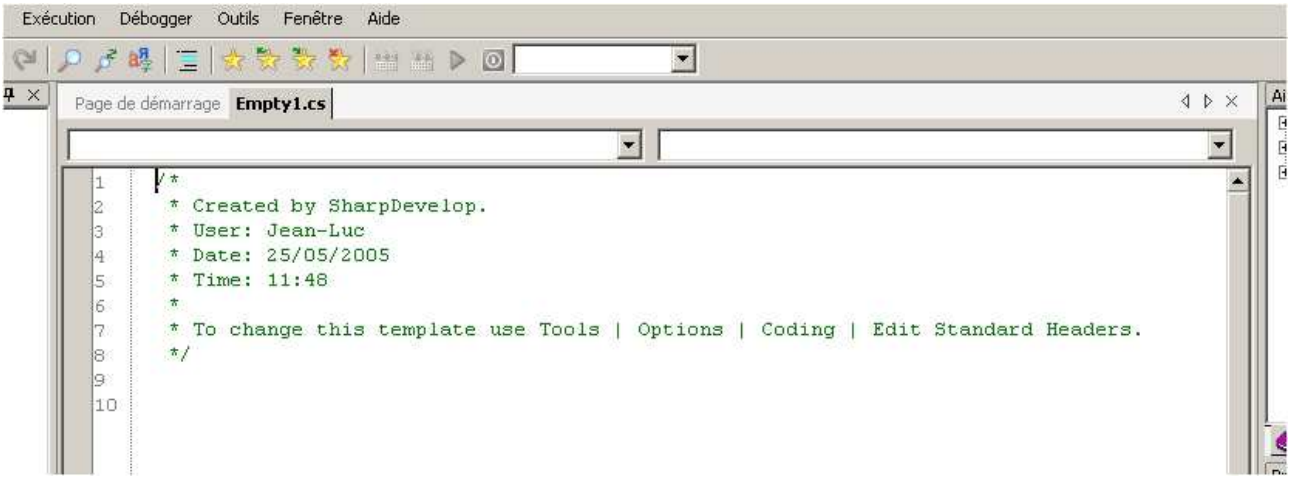
Notre premier programme ou première classe (nous verrons par la suite le concept de classe).

Lancez sharpdevelop, dans la barre de menu cliquez sur **Fichier** puis sur **Nouveau** et enfin **Fichier**.

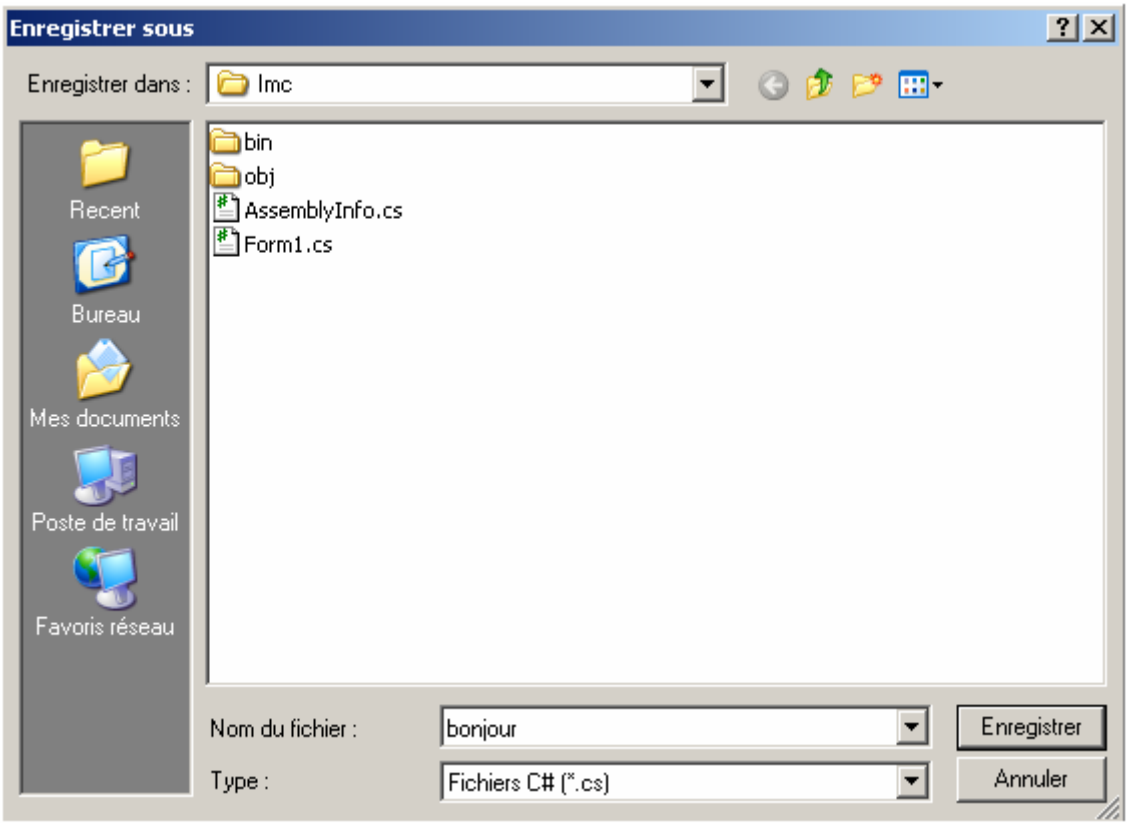
Dans la nouvelle fenêtre,



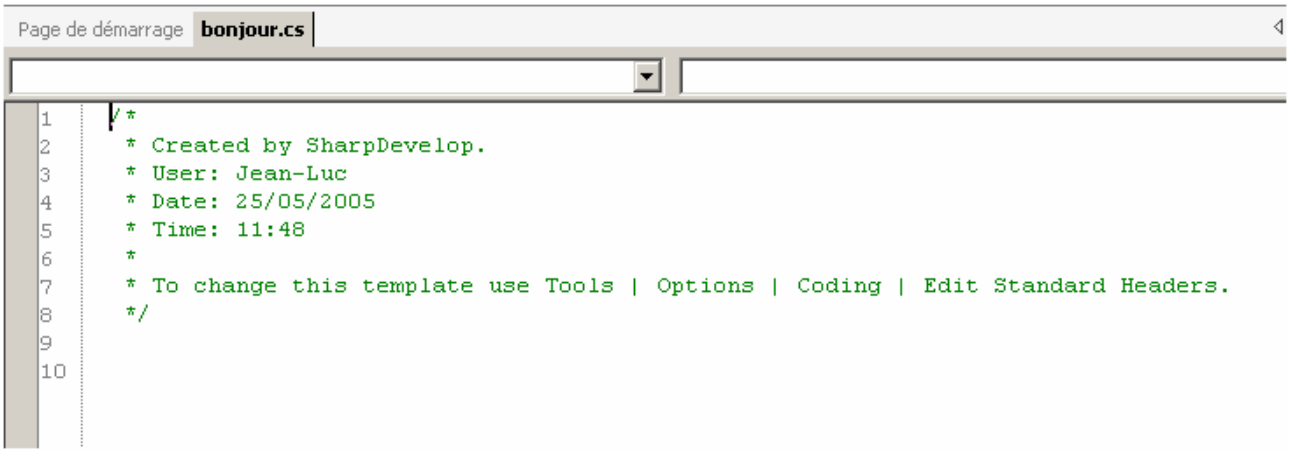
Vérifiez que la catégorie pointe sur C# et comme modèle prenez "Fichier Vide" ensuite validez en cliquant sur "Créer". Vous venez de créer une zone de saisie du code qui se nomme Empty.cs par défaut.



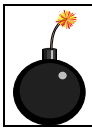
Nous allons renommer immédiatement cette page vide en "bonjour.cs". Pour cela cliquez sur F**ichier** puis **Enregistrer sous** et ensuite nommez le bonjour et ensuite validez en cliquant sur **Enregistrer**.



La feuille du classeur est passée de Empty.cs à bonjour.cs. Voici le nouvel affichage :



Comme vous pouvez le voir SharpDevelop rajoute en commentaire des informations en début du fichier. Comme elles nous sont de peu d'utilités supprimez les.



Remarque : /* et */ encadrent une zone de commentaires. N'hésitez pas à ajouter des commentaires dans votre code, vous aurez par la suite moins de difficultés de compréhension si votre code est commenté.

Notre premier programme sera sans ambition, il affiche un message de bienvenue, la date et l'heure de l'ordinateur. Saisissez le code suivant en prenant garde a respecter la syntaxe et surtout la **différence entre les majuscule et minuscules.**

Fichier « bonjour.cs »

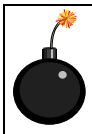
```
using System;
namespace bonjour
{
	public class bonjour
	{
		static void Main(string[] args)
		{
		DateTime d1 = DateTime.Now;
		Console.WriteLine("Bonjour nous somme le : ");
		Console.WriteLine(d1);
		}
	}
}
```

Saisissez-le tel quel et enregistrez le sous **bonjour**.

Voici les explications du code que vous venez de saisir.

Using System;

Au début de chaque fichier de programme il faut mettre la directive **using** pour spécifier au compilateur une liste de chemins de recherche implicites. Par exemple, quand le compilateur rencontre la méthode "Console.WriteLine("Bonjour nous somme le : ");", il s'aperçoit que son chemin n'est pas défini explicitement. Il balaie alors les espaces de noms spécifiés dans les using puis, une fois la méthode dénichée dans l'espace de nom System, il compile le code.



Remarque : Ainsi, vous avez évités ceci :
System.Console Console.WriteLine("Bonjour nous somme le : ");
Ou aussi:
DateTime d1 = System.DateTime.Now;

namespace bonjour

Cette ligne facultative défini un espace de nom qui va contenir les classes qui sont entre accolades dans notre exemple la classe bonjour. L'explication est la même dans le principe que celle donnée au dessus, je vous laisse vous y reporter.

public class bonjour

Une application C# est une collection de classes, une classe étant un ensemble de données et de méthodes. Une méthode est un ensemble d'instructions appliquant un traitement aux données de la classe, retournant ou non un résultat logique ou typé (valeur numérique, chaîne de caractères, référence d'objet, etc.…)

C'est le mot clé **class** qui introduit la déclaration de la classe, suivi du nom de la classe, d'une accolade ouvrante et d'une accolade fermante. Votre code sera placé entre les deux accolades.

```
static void Main(string[] args)
```

Cette méthode est la méthode principale (main in english) de votre programme bonjour, c'est ici que vous placerez les actions à faire exécuter à C#.

Le mot-clef **public** signifie que la méthode est accessible au monde extérieur. Le mot clé static indique au compilateur que la méthode **Main** est globale et que, par conséquent, il est inutile d'instancier la classe pour pouvoir appeler la méthode. Comme elle est statique, le compilateur prend son adresse comme point d'entrée. De cette façon, l'environnement .NET sait par où commencer l'exécution de l'application.

Le paramètre de **main( )** est un tableau d'objets de type **Strin**g. Le paramètre **args** n'est pas utilisé dans ce programme mais le compilateur insiste pour qu'il soit là car il contient les paramètres invoqués sur la ligne de commande.

```
DateTime d1 = DateTime.Now;
```

La ligne qui définie la date est assez intéressante :

DateTime d1 permet de créer un objet vide de type DateTime. Comme vous êtes doués pour les langues étrangères vous pouvez constater que d1 gérera des dates et heures. Vous avez compris ? Le **"= Datetime.now;"** va "graver" dans d1 la date et l'heure de l'ordinateur.
Donc si je récapitule cette ligne déclare une variable d'un type X et il la rempli avec un contenu Y. Simple, non ?

```
Console.WriteLine("Bonjour nous somme le : ");
```

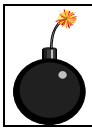
Pour l'instant intéressons-nous à WriteLine**( ),** qui veut dire « écris ce que je te donne sur une ligne de la console». C'est magnifique, vous savez faire écrire une phrase ! Ne vous inquiétez pas après l'écriture viendront les calculs !

Astuce : Nous aurions put concaténer d1 à la suite de "Bonjour nous sommes le : " la syntaxe aurait alors été :

```
Console.WriteLine("Bonjour nous somme le : "+d1);
```

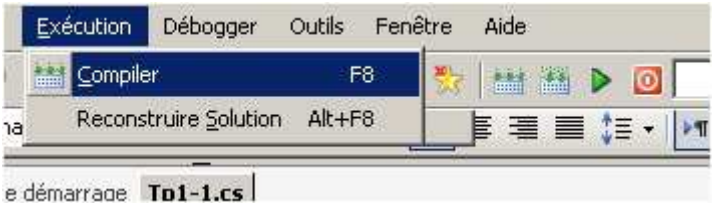
```
Console.WriteLine(d1);
```

Est-ce nécessaire que j'explique ? Vous êtes suffisamment corticalisé pour comprendre.

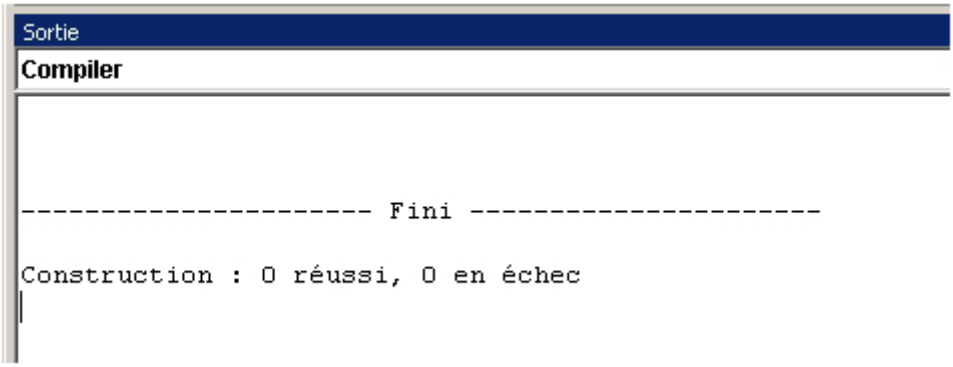


Remarque : Observez les accolades qui forment les blocs d'instructions et les points virgules qui terminent une instruction.

Voilà le premier programme est fait, passons à son exécution (En joue, feu…Oupps, pardon ça m'a échappé!).



Donc, cliquez sur **Exécution** puis **Compiler**, si il n'y a pas d'erreur dans votre code vous ne devriez pas avoir de messages de la part du compilateur.



Maintenant il faut activer ce programme :

Cliquez sur **Débogger** et **Exécuter** ou alors repérez la flèche verte dans la barre d'outil. Selon l'interface de développement vous aurez une fenêtre console qui apparaîtra avec le résultat de votre programme ou alors il vous faudra l'activer pour voir le résultat à l'écran. Si tel est le cas, cliquez sur **Affichage** puis **Outils** puis **Console** ainsi vous aurez un rendu de l'aperçu tel qu'il serait sous le mode console (ex mode Dos).

Il est évident que ce premier programme est vraiment simpliste, mais son but est de vous montrer le fonctionnement de base d'un programme C# et sa structuration spécifique. N'oubliez pas d'indenter (faire des retraits) vos sources, vous y gagnerez en clarté, lisibilité, vous maintiendrez plus facilement vos programmes.

On passe aux exos ?

Exercices applicatifs

Le but de ces exercices est de vous familiariser avec le langage, vous travaillez en autonomie, à votre rythme. Il n'y a aucune surprise vous avez vu dans la leçon tous les éléments nécessaires pour résoudre les exercices demandés. Donc courage… !

Exercice 1 :

Créez une classe nommée etatcivil qui affiche sur la première ligne votre nom votre prénom, sur la deuxième votre date de naissance, sur la troisième votre ville de naissance.

Exercice 2 :

En vous aidant du code de la classe etatcivil (copier / coller), créez une classe nommée etatciv2. Modifiez le code pour que la ville de naissance soit sur la même ligne que la date.