Introduction au langage C#

1 Introduction :

Langage développé par **Microsoft** afin de concurrencer le langage **Java** afin d'être exécuté sur un grand nombre de plates formes (même des linux). Le langage **C#** fait partir de la plate forme .NET. On trouve d'ailleurs beaucoup de points communs avec le langage java, dans la syntaxe & dans la mise en œuvre.

En effet, après avoir téléchargé le **framework .net** sur le site de Microsoft, on peut réaliser des applications consoles ou graphiques « manuellement », c'est à dire en écrivant les programmes sources avec un éditeur de texte (**notepad++** par exemple). Cela veut dire que pour qu'un programme C# fonctionne sur une autre machine, celle-ci devra être pré installée par le framework (si elle n'a pas Visual Studio).

1.1 Les différentes versions :

Où trouver le FrameWork : Il est disponible gratuitement sur le site http://www.microsoft.com/downloads/search.aspx?displaylang=fr .

Microsoft Visual Studio 2003 intègre la version 1.1 (sous répertoire SDK dans Visual), Visual 2008 intègre la version 3.5 et visual 2010 la version 4.

1.2 La Composition du framework :

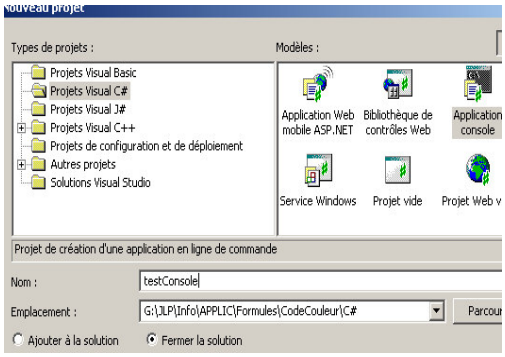
Après avoir téléchargé & installé le framework, on retrouve un dossier **Microsoft.NET/Framework** dans le dossier **Windows** et différentes versions. On trouve principalement le compilateur **csc.exe**

1.3 Notre outil de développement : Microsoft Visual Studio :

Les sources se développent avec n'importe quel éditeur de texte ou avec Microsoft Visual Studio en créant des projets C# (on bénéficie d'un assistant).

Pour la compilation des classes, on peut procéder de 2 manières :

- Utiliser l'outil Génerer de Visual Studio.
- Utiliser le compilateur **csc.exe** faisant partie du **Framework** avec un fichier de commande (extension **bat**).



2 Exemple : code couleur des résistances :

2.1 Première partie : utilisation de la classe en mode console :

Créer un projet Visual c# « Application console » ou **éditer** « à la main ces 2 fichiers »

2.1.1 Le code source :

testCodeCouleur.cs

```
using System;

namespace testCodeCouleur
{
  class testCodeCouleur
  {
     static void Main(string[] args)
     {
          Console.WriteLine("Test de la classe CodeCouleur");
```

CodeCouleur.cs

```
using System;

namespace testCodeCouleur
{
   public class CodeCouleur
   {
        private int r;
        private string [] table = new string [] {"noir", "brun",
              "rouge", "orange", "jaune","vert",
               "bleu", "violet","gris", "blanc"};
        public CodeCouleur()
        {
```

```
        CodeCouleur maCouleur = new CodeCouleur();

        Console.Write("Entrez une valeur de résistance ");

        string resist = Console.ReadLine();
        int r = System.Convert.ToInt32(resist);

        maCouleur.setValeur(r);
        string couleur1 = maCouleur.getCouleurs(1);
        string couleur2 = maCouleur.getCouleurs(2);
        string couleur3 = maCouleur.getCouleurs(3);

        Console.WriteLine("Couleurs " + couleur1 + "-" +
couleur2 + "-" + couleur3);
    }

  }

}
```

```
                this.r = 0;
            }
            public bool setValeur  ( int valeur )
            {
                if (valeur > 0)
                {
                    this.r = valeur;
                    return true;
                }
                else            return false;
            }
            public string  getCouleurs  ( int anneau)
            {
                if (r == 0)   return null;
                string couleurAnneau;

                double log10 =
Math.Log((double)this.r)/Math.Log(10.0);
                int C3 = (int)log10 -1 ;
                int C1 = (int)(this.r/Math.Pow (10.0, C3+1.0));
                int C2n = this.r-C1*(int)Math.Pow (10.0,
C3+1.0);

                int C2d = (int)Math.Pow (10.0, (double)C3);
                int C2 = C2n/C2d;
                switch (anneau)  {
                    case 1:  couleurAnneau = table[C1];    break;
                    case 2:  couleurAnneau = table[C2];    break;
                    case 3:  couleurAnneau = table[C3];    break;
                    default: couleurAnneau = null;         break;
                  }
                 return couleurAnneau;
            }
    }
}
```

2.1.2 En dehors de Visual :

Compile.bat

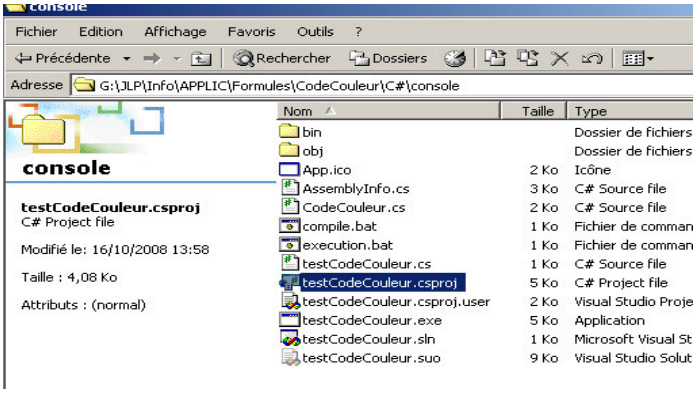
```
rem Variables d'environnement
@SET FrameWorkDir=D:\WINNT\Microsoft.NET\Framework
@SET
FrameWorkDir=C:\WINDOWS\Microsoft.NET\Framework
@SET FrameVersion=v1.1.4322

@SET
PATH=%DevEnvDir%;%FrameWorkDir%\bin;%FrameWorkDi
r%\%FrameVersion%;%PATH%;
csc *.cs
pause
```

Execution.bat

```
testCodeCouleur.exe
pause
```

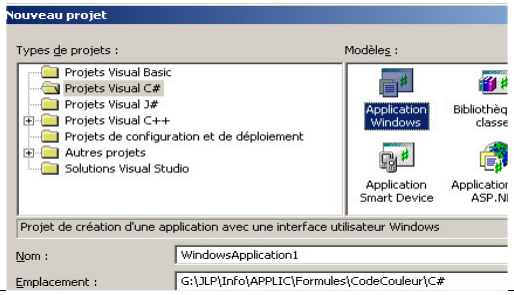
```
C:\WINDOWS\system32\cmd.exe

H:\JLP\Info\APPLIC\Formules\CodeCouleur\C#\console>testCodeCou
Test de la classe CodeCouleur
Entrez une valeur de résistance 470
Couleurs jaune-violet-brun

H:\JLP\Info\APPLIC\Formules\CodeCouleur\C#\console>pause
Appuyez sur une touche pour continuer...
```

Remarque :
On peut supprimer tous les fichiers à l'exception de **testCodeCouleur.cs** et **CodeCouleur.cs** !



2.2 Seconde partie : réutilisation de la classe dans une application graphique :

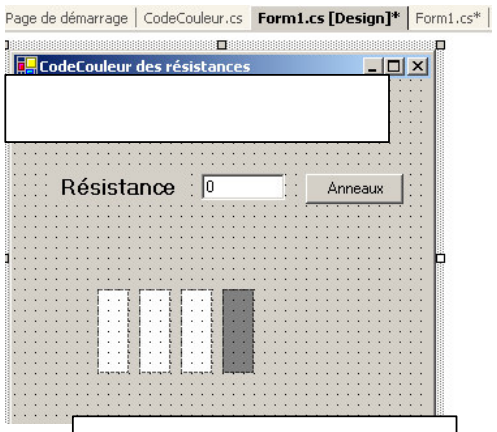
Créer un projet Visual c# « Application Windows »



2.2.1 Création de l'application graphique :

Form1.cs

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

namespace WindowsApplication1   {
  public class Form1 : System.Windows.Forms.Form    {
          private System.Windows.Forms.Button button1;
          private System.Windows.Forms.Label label1;
          private System.Windows.Forms.TextBox textBox1;
          private System.Windows.Forms.Panel panel1;
          private System.Windows.Forms.Panel panel2;
          private System.Windows.Forms.Panel panel3;
          private System.Windows.Forms.Panel panel4;
          private System.ComponentModel.Container components = null;
          private CodeCouleur maCouleur;

           public Form1()   {
                    InitializeComponent();
                    maCouleur = new CodeCouleur();
           }
             protected override void Dispose( bool disposing )  {
                    if( disposing )                    {
                      if (components != null)     {
                              components.Dispose();
                       }
                    }
                    base.Dispose( disposing );
           }
           private void InitializeComponent()              {
              this.button1 = new System.Windows.Forms.Button();
              this.label1 = new System.Windows.Forms.Label();
             ....
            }
           static void Main()
          {
                    Application.Run(new Form1());
           }
           private void button1_Click(object sender, System.EventArgs e)
          {
                    int r = Convert.ToInt32(this.textBox1.Text);
                    maCouleur.setValeur(r);
                    Color couleur1 = maCouleur.getCouleurs(1);
                    this.panel1.BackColor = couleur1;
                    Color couleur2 = maCouleur.getCouleurs(2);
                    this.panel2.BackColor = couleur2;
                     Color couleur3 = maCouleur.getCouleurs(3);
                     this.panel3.BackColor = couleur3;
           }
    }
}
```



2.2.2 Adaptation de la classe CodeCouleur :

- **Copier** dans le dossier cible et ajouter au projet le fichier **CodeCouleur.cs**
- **Modifiez** le namespace en **WindowsApplication1**
- **Ajoutez** une importation pour la classe Color :
  `using System.Drawing;`
- **Remplacez** systématiquement l'emploi de **string** par Color

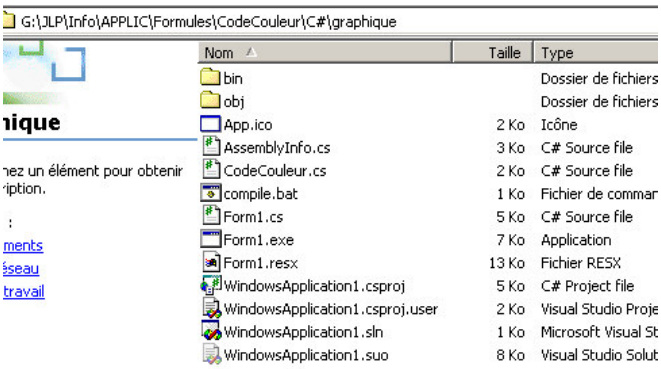
La table des couleurs devient :

```
private Color [] table = new Color [] {Color.Black, Color.Brown, Color.Red,
                                       Color.Orange, Color.Yellow, Color.Green,
                  Color.Blue, Color.Purple, Color.Gray, Color.White};
```

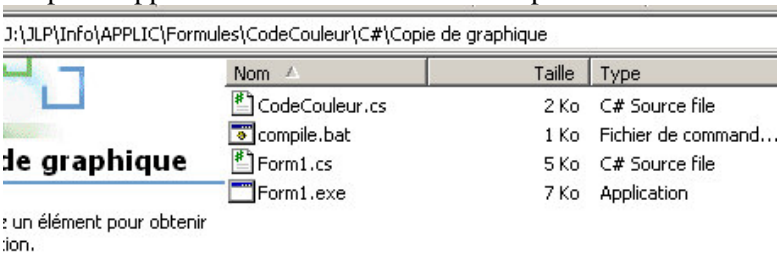
La valeur **null** devient Color.Pink.

2.2.3 En dehors de Visual :

Le fichier de compilation est identique à celui en mode console : un executable **Form1.exe** est alors crée :



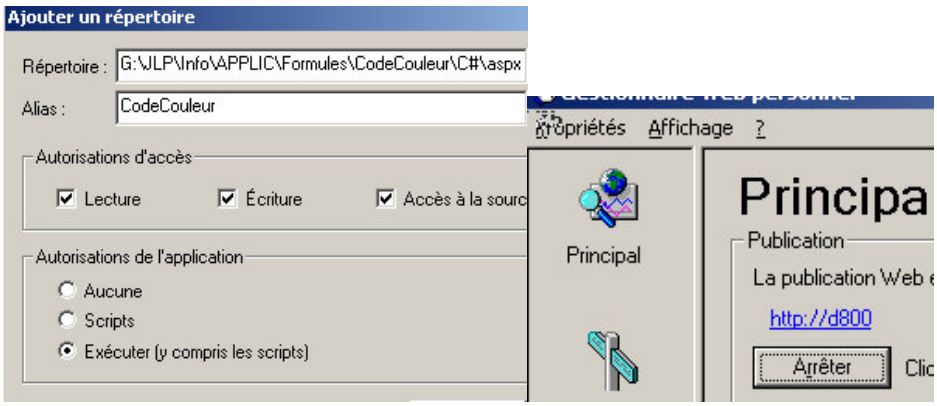
Remarque :
On peut supprimer tous les fichiers à l'exception de **Form1.cs** et **CodeCouleur.cs** !



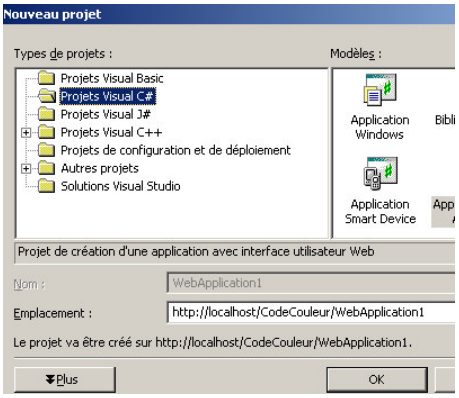
2.3 Troisième partie : réutilisation de la classe dans une page asp . net :

2.3.1 Configuration du serveur

On veille à ce que le serveur Apache soit arrêté et on démarre le serveur http Windows.
On crée un répertoire viruel.

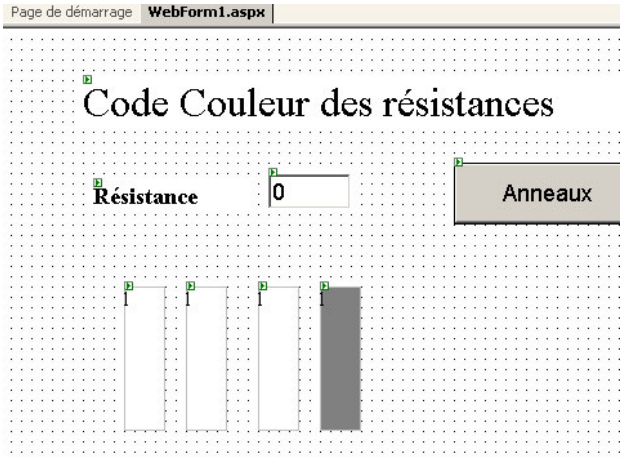


2.3.2 Création du projet



Créer un projet Visual c# « Application Web Asp .NET » en sélectionnant le répertoire virtuel précedemment crée.
Copier dans le dossier cible et ajouter au projet le fichier **CodeCouleur.cs**

Mettez en place l'interface graphique dans l'onglet **design**.

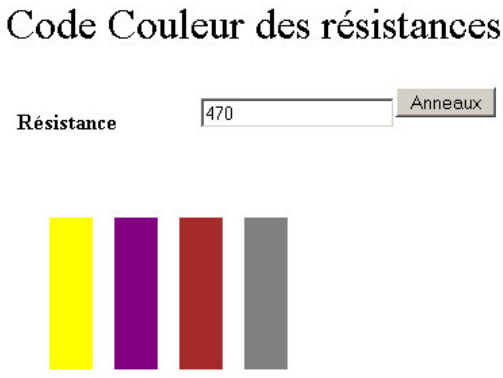


Ajoutez dans le source WebForm1.aspx.cs

```
private CodeCouleur maCouleur;  // comme attribut de la classe WebForm1
maCouleur = new CodeCouleur();  // dans la méthode Page_Load
```

Cliquez le bouton et faites un copier-coller du code du fichier Form1.cs de l'application graphique :

- **Modifiez** le namespace de **CodeCouleur.cs** en **WebApplication1**
- **Génerez et tester** (soit à l'intérieur de visual soit en utilisant n'importe quel navigateur et an appelant la page serveur).



3 Des points de programmation

3.1 Les tableaux :

On a vu plus haut que l'on pouvait déclarer un tableau de la manière suivante :
string [] table = **new string** [] {"noir", "brun", "rouge", "orange", "jaune","vert", "bleu", "violet","gris", "blanc"};

Si on veut juste déclarer un nombre d'éléments, la syntaxe est : **string** [] table = **new string** [10];

Remarque : la syntaxe est la même qu'en java alors qu'en c++, on aurait écrit string *table = new string [10] ;

3.2 Le mot clef partial :

Si une classe s'avère être trop grosse, on peut la dispatcher dans deux classes.
Exemple :

CodeCouleur.cs

```
using System;

namespace testCodeCouleur
{
  public partial  class CodeCouleur
  {
      private int r;
      private string [] table = new string []
     {"noir", "brun", "rouge", "orange",
      "jaune","vert","bleu","violet","gris",
       "blanc"};
      public CodeCouleur()
      {
               this.r = 0;
      }
      public bool setValeur  ( int valeur )
      {
         if (valeur > 0)
         {
              this.r = valeur;
              return true;
         }
              else   return false;
       }
  }
}
```

CodeCouleur2.cs

```
using System;

namespace testCodeCouleur
{
  public partial  class CodeCouleur
  {
     public string  getCouleurs  ( int anneau)      {
      if (r == 0)      return null;
      string couleurAnneau;

       double log10 = ath.Log((double)this.r)/Math.Log(10.0);
        int C3 = (int)log10 -1 ;
        int C1 = (int)(this.r/Math.Pow (10.0, C3+1.0));
       int C2n = this.r-C1*(int)Math.Pow (10.0, C3+1.0);

        int C2d = (int)Math.Pow (10.0, (double)C3);
        int C2 = C2n/C2d;
        switch (anneau)  {
          case 1:  couleurAnneau = table[C1];    break;
          case 2:  couleurAnneau = table[C2];    break;
          case 3:  couleurAnneau = table[C3];    break;
          default: couleurAnneau = null;         break;
       }
        return couleurAnneau;
      }
    }
}
```

3.3 Le mot clef var :

Permet à une variable locale de voir son type déduit automatiquement.
Exemple :
CodeCouleur maCouleur = new **CodeCouleur**(); peut être réécrit **var** maCouleur = new **CodeCouleur**();

3.4 Passage de paramètres :

Tout comme en c++, les types de base (ou primitifs) sont passés par valeur et les tableaux par référence (ou adresse).
Les classes sont passées par référence.
Si on veut forcer le type de passage , le C# introduit 3 mots clefs, **in**, **out** et **ref** à mettre devant les paramètres.
Si on utilise **ref**, à l'appel les paramètres doivent avoir été initialisés à la différence de **out**.

Exemple :

TestCodeCouleur.cs

```
string couleur1, couleur2, couleur3;
maCouleur.getCouleurs(out couleur1, out couleur2, out couleur3);
```

CodeCouleur.cs

```
public bool  getCouleurs  (out string couleur1,out string couleur2,out string couleur3)
{
        if (r == 0)
        {
                couleur1 = couleur2 = couleur3 = null;
                return false;
        }

        double log10 = Math.Log((double)this.r)/Math.Log(10.0);
        int C3 = (int)log10 -1 ;
        int C1 = (int)(this.r/Math.Pow (10.0, C3+1.0));
        int C2n = this.r-C1*(int)Math.Pow (10.0, C3+1.0);

        int C2d = (int)Math.Pow (10.0, (double)C3);
        int C2 = C2n/C2d;
        couleur1 = table[C1];
        couleur2 = table[C2];
        couleur3 = table[C3];

        return true;
}
```