

[Visual C#] Utiliser les paramètres d'application

Par Saggah



www.openclassrooms.com

*Licence Creative Commons 4 2.0
Dernière mise à jour le 13/07/2011*

Sommaire


Sommaire	2
Lire aussi	1
[Visual C#] Utiliser les paramètres d'application	3
Les fichiers de configuration	3
Soyez prêt !	4
Création du projet !	4
Le fichier *.config	4
Personnalisons notre formulaire !	7
Utiliser les paramètres d'application !	7
Lire	8
Écrire	8
Dans notre programme	9
Q.C.M.	10
Partager	11



[Visual C#] Utiliser les paramètres d'application



Mise à jour : 13/07/2011

Difficulté : Facile  Durée d'étude : 1 heure, 30 minutes



Salut à tous et bienvenue dans ce tutoriel !

Vous avez besoin de stocker des données sans utiliser les bases de données (SQL et LINQ) ?
Vous êtes débutant et vous connaissez les bases du langage C# ?

Dans ce cas, ce tutoriel est fait pour vous !
Sommaire du tutoriel :



- Les fichiers de configuration
- Soyez prêt !
- Utiliser les paramètres d'application !
- Q.C.M.

Les fichiers de configuration

Les **paramètres d'application** sont des données comprises dans des **fichiers de configuration** qui sont écrits en **XML**.



Attention, ne pas confondre fichiers de configuration et paramètres d'application !

Si vous ne savez pas ce qu'est le XML, voici un exemple de code :

Code : XML

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <sectionGroup name="userSettings"
type="System.Configuration.UserSettingsGroup, System,
Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" >
      <section name="Tutoriel.Properties.Settings"
type="System.Configuration.ClientSettingsSection, System,
Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
allowExeDefinition="MachineToLocalUser" requirePermission="false" />
    </sectionGroup>
  </configSections>
  <userSettings>
    <Tutoriel.Properties.Settings>
      <setting name="Setting" serializeAs="String">
        <value>Ceci est un exemple d'utilisation de notre
fichier XML pour une application C# !</value>
      </setting>
    </Tutoriel.Properties.Settings>
  </userSettings>
</configuration>
```

Pour de plus amples informations sur XML, vous pouvez lire [le tutoriel de Tangui](#) qui l'explique en détail.

Vous verrez plus tard, dans la troisième partie, comment utiliser ces fichiers par l'intermédiaire de C#. Je tiens à préciser que le **.NET Framework** permet d'accéder à ces fichiers très simplement à l'aide d'une seule méthode !

Le XML est utilisé de deux façons différentes :

- avec l'aide de XPath ;
- avec l'outil de Visual C#.

L'outil livré avec Visual C# est celui que nous utiliserons dans ce cours.

Soyez prêt !

Nous allons maintenant commencer.

J'utiliserai Microsoft Visual Studio 2010 avec le .NET Framework 4.0 dans ce tutoriel. N'ayez crainte, que vous ayez une version express de Visual Studio 2010 ou 2008, ça ne changera rien. 😊

Bon alors, commençons !

Création du projet !

Lancez Microsoft Visual C# (pas trop loin 😊).

Créez un nouveau projet : Fichier → Nouveau projet → Application Windows Forms / Application WPF. Après, à vous de choisir si vous voulez créer un projet WinForms ou WPF. Ça ne changera rien.

Nommez votre projet : « *Vive H@des* », par exemple.



Petite précision : je ne supporterai pas si vous mettez autre chose que mon nom ! 🙄

Maintenant, reste à créer le fichier *.config.

Le fichier *.config

Nous allons entamer la création et la construction du fichier *.config.

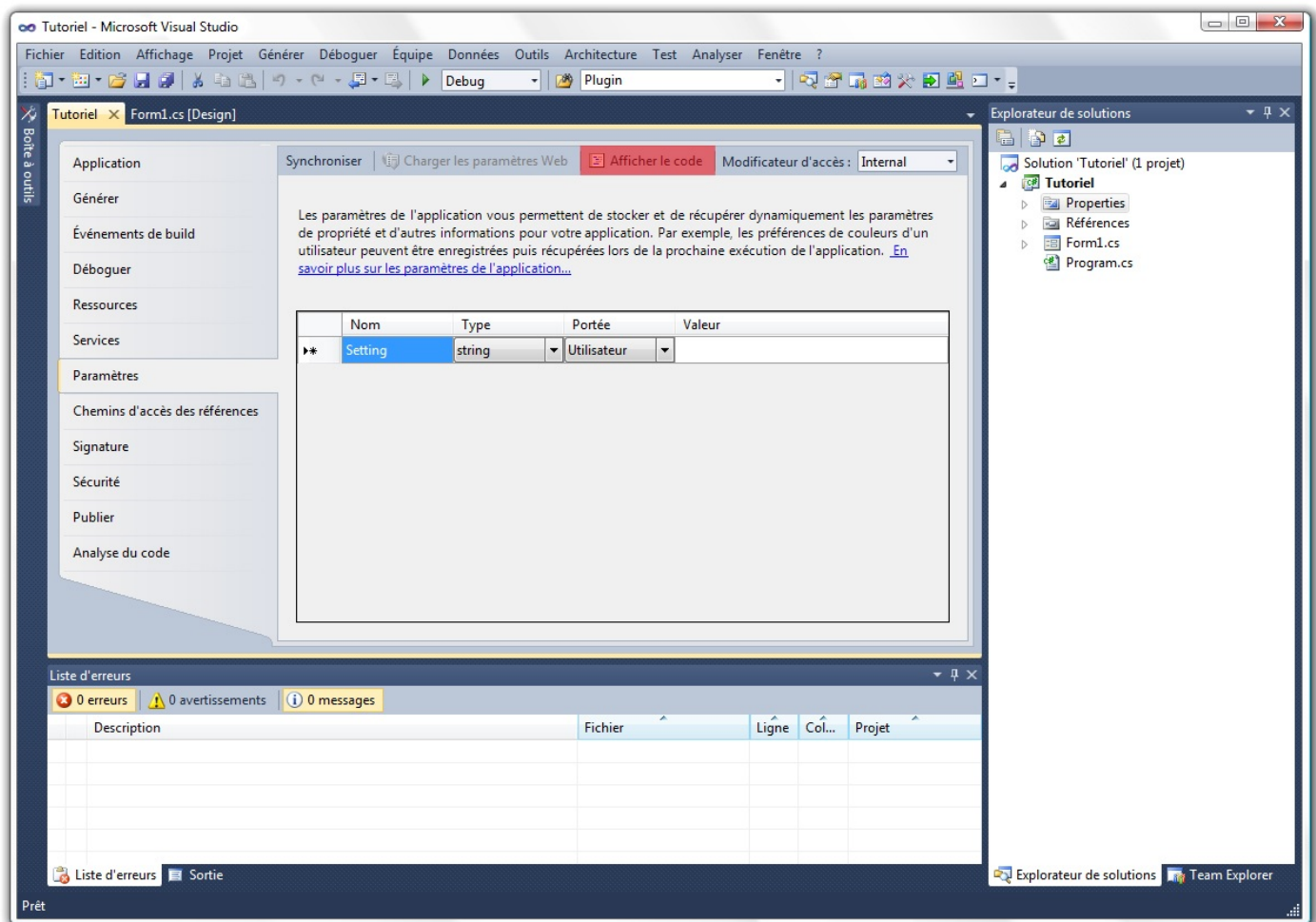
Pour cela, il vous faudra double-cliquer sur Properties ou Propriétés dans l'explorateur de solutions : vous devriez voir apparaître les **options de l'application**, mais pas encore les **paramètres de configuration**.



Attention à ne pas confondre !

Afin de voir ces paramètres, double-cliquez sur... Paramètres. 😊

Vous voyez donc apparaître ceci :



- Le nom correspond au nom du paramètre que vous utiliserez dans votre partie code source. Bien évidemment, n'utilisez pas d'espaces ni de caractères spéciaux.
- Le type sert à préciser de quel type sera la variable. Il existe aussi d'autres types, comme la taille.
- La portée. Deux choix sont possibles. Je vous en parlerai après.
- Et la valeur, qui sert, comme vous l'avez sûrement compris, à stocker la valeur que contiendra la variable. Vous pouvez très bien ne rien mettre, ce qui sera équivalent à un **null**.

Comme je l'ai dit, pour la portée du paramètre, il y a un choix à faire parmi :

- **Application.** Cette portée est utilisable seulement en lecture seule. Vous ne pourrez la modifier que par Visual Studio ou par exemple le célèbre Bloc-notes de Windows, mais surtout pas par l'application ; 😞
- **Utilisateur.** Contrairement à la portée Application, celle-ci est utilisable et modifiable à n'importe quel moment.

Vous l'avez sans doute remarqué, mais quand vous créez votre premier paramètre, un fichier *.config est créé dans votre solution. C'est lui qui contient toutes les données. Visual C# ne fait que le générer et l'interpréter !



Notez que quand vous utilisez la portée Utilisateur, les paramètres ne sont pas stockés dans le fichier *.config à la racine de votre application, mais dans un nouveau fichier [user].config où [user] désigne le nom de la session utilisateur. C'est comme ça que chaque utilisateur peut enregistrer ses valeurs préférées.

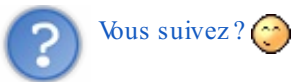


Mais il y a quelque chose de surligné en rouge sur l'image ! Qu'est-ce que c'est ?



Je reviendrai sur ce point tout à l'heure. 😊

Le T.P. de ce tutoriel est un navigateur web qui devra, au démarrage, lire ce paramètre qui est une URL et afficher la page web. Il y aura aussi un bouton afin de donner l'URL de la page web actuelle à notre paramètre.



Nous allons juste mettre une seule propriété :

Nom	Type	Portée	Valeur
PageDeDemarrage	String	Utilisateur 1	http://www.siteduzero.com/

¹ → Nous allons mettre Utilisateur car notre programme va consister, comme je l'ai dit, à ouvrir une page web et en définir une. Donc on a besoin de modifier cette valeur : → **public**. Une fois que tout ceci sera bon et que vous aurez validé, vous retournerez sur votre formulaire *Forms1.cs [Design]* ou *Window1.xaml*.


Maintenant, double-cliquez sur « *App.config* » ; vous devriez avoir ce code :

Code : XML

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <sectionGroup name="userSettings"
type="System.Configuration.UserSettingsGroup, System,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" >
      <section name="Projet.Properties.Settings"
type="System.Configuration.ClientSettingsSection, System,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
allowExeDefinition="MachineToLocalUser" requirePermission="false" />
    </sectionGroup>
  </configSections>
  <userSettings>
    <Projet.Properties.Settings>
      <setting name="PageDeDemarrage" serializeAs="String">
        <value>http://www.google.fr/</value>
      </setting>
    </Projet.Properties.Settings>
  </userSettings>
</configuration>
```

Nous n'allons pas étudier ces lignes de code en XML. Si vous avez lu le lien que je vous avais donné au début du tutoriel, vous comprendrez. 🤔

Revenons à l'image de tout à l'heure ! Vous êtes peut-être impatient de savoir quelle est cette chose soulignée ! 🤔

 La chose soulignée est une fonction de Visual Studio qui permet de gérer les événements du fichier *.config. Ça y est, j'en ai trop dit. 🤔

Eh oui ! Visual Studio permet de gérer des événements du fichier *.config. C'est un outil puissant que nous avons là !

Quand vous cliquez dessus, Visual Studio vous ouvre un fichier source.

Code : C#

```
namespace ParametreApplication.Properties {

    // Cette classe vous permet de gérer des événements spécifiques
```

```

dans la classe de paramètres :
// l'événement SettingChanging est déclenché avant la
modification d'une valeur de paramètre ;
// l'événement PropertyChanged est déclenché après la
modification d'une valeur de paramètre ;
// l'événement SettingsLoaded est déclenché après le chargement
des valeurs de paramètre ;
// l'événement SettingsSaving est déclenché avant
l'enregistrement des valeurs de paramètre.
    internal sealed partial class Settings {

        public Settings() {
            // // Pour ajouter des gestionnaires d'événements afin
            // d'enregistrer et de modifier les paramètres, supprimez les marques
            // de commentaire des lignes ci-dessous :
            //
            // this.SettingChanging +=
this.SettingChangingEventHandler;
            //
            // this.SettingsSaving +=
this.SettingsSavingEventHandler;
            //
        }

        private void SettingChangingEventHandler(object sender,
System.Configuration.SettingChangingEventArgs e) {
            // Ajouter du code pour gérer l'événement
            SettingChangingEvent ici.
        }

        private void SettingsSavingEventHandler(object sender,
System.ComponentModel.CancelEventArgs e) {
            // Ajouter du code pour gérer l'événement
            SettingsSaving ici.
        }
    }
}

```

Tout est expliqué dans ce code source par l'intermédiaire des commentaires. Vous n'aurez pas besoin de moi. 😊

Personnalisons notre formulaire !

Maintenant, nous allons nous occuper de l'apparence du formulaire. Glissez un `WebBrowser` dans votre formulaire (je suppose que vous savez faire ça).

En WPF

Mettez le `VerticalAlignment` à `Stretch` ainsi que l'`HorizontalAlignment`.

En WinForm

Mettez-le tout simplement en `Dock.Fill`. Puis mettez un `Button` n'importe où dans votre formulaire qui contient le texte suivant : « Définir cette page comme page de démarrage. »

Et voilà, c'était tout pour cette sous-partie.

Utiliser les paramètres d'application !

Bon... Assez discuté !

Je vais enfin vous donner la syntaxe pour les paramètres d'application. Il n'y en a que deux ; ce sont des propriétés, générées automatiquement par Visual Studio, ce qui veut dire que ces paramètres sont utilisés de la même manière que des **Propriétés** `get{} set{} .`

Non, vous ne rêvez pas ! Je vous avais prévenu que c'était d'une simplicité ! C'est votre jour de chance !

Je vais commencer par mettre la syntaxe permettant de lire une de ces propriétés. Et puis comme vous savez aussi en définir une, je vous laisserai deviner la syntaxe pour le faire.

Lire

Syntaxe

Code : Autre

```
[Type] [NomDuType] = Properties.Settings.Default.[NomDuParamètre];
```



Vous ne devez pas mettre les crochets ! C'est pour vous montrer ce qu'il y a à modifier.

Dans notre cas

Code : C#

```
string Url = Properties.Settings.Default.PageDeDemarrage;  
webBrowser1.Navigate(Url);
```



Notez que vous pouvez utiliser directement le paramètre ! Rien de tel pour se simplifier la vie !

Code : C#

```
webBrowser1.Navigate(Properties.Settings.Default.PageDeDemarrage);
```

Écrire

Syntaxe

Code : Autre

```
Properties.Settings.Default.[NomDuParamètre] = [Valeur];
```



[Type] et [NomDuType] doivent être de même type. Les casts fonctionnent quand même, mais je ne vois pas l'intérêt puisque [NomDuType] peut s'agir de n'importe quel type de variable.



Si vous voulez réutiliser la même valeur pour le type lors d'une prochaine session, vous devez sauvegarder les changements (dans l'événement `_Closing` de votre application). Ça ne sert à rien de sauvegarder le fichier à chaque fois qu'il y a un changement, mettez-le uniquement à la sortie de votre application !

Code : C#

```
Properties.Settings.Default.Save();
```


Dans notre cas

Code : C#

```
Properties.Settings.Default.PageDeDemarrage =  
"http://www.siteduzero.com/";  
Properties.Settings.Default.Save();
```

C'est bon ! Vous maîtrisez les fichiers de configuration ! Maintenant, place à la pratique !

Dans notre programme

Je vais vous demander de gérer l'événement : `_Load` pour votre fenêtre et l'événement : `_Click` pour votre bouton. Donc dans : `_Load`, je vais vous demander de charger une URL de type `string` dans votre composant : `webBrowser1`, sachant que pour charger une URL à partir de ce composant, la syntaxe est la suivante :

Code : C#

```
webBrowser1.Navigate([IciVousMettezL'UrlAChargerSansLesCrochets]);
```

Et dans l'événement `_Click` de notre bouton, je voudrais que la page actuelle sur le composant `webBrowser1` soit définie dans le paramètre `PageDeDemarrage`. Pour obtenir l'URL de la page actuelle, vous devez utiliser la propriété `Url` du composant `WebBrowser` :

Code : C#

```
webBrowser1.Url();
```

Je veux aussi que vous sauvegardiez le paramètre pour une prochaine session !

Bon ! J'en ai trop dit ! Bonne chance !

Je vous laisse 5 minutes !

3...

2...

1...



GO !!!

Voici la correction :

Secret (cliquez pour afficher)

Code : C#

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Windows.Forms;
```

```
namespace ParametreApplication
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load()
        {
            webBrowser1.Navigate(Properties.Settings.Default.PageDeDemarrage);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Properties.Settings.Default.PageDeDemarrage =
            webBrowser1.Url.ToString();
        }

        private void Form1_FormClosing(object sender,
        FormClosingEventArgs e)
        {
            Properties.Settings.Default.Save();
        }
    }
}
```

Maintenant, nous pouvons exécuter notre programme et vous voyez le navigateur charger la page définie dans le fichier *.config, et quand on clique sur le bouton, il donne l'URL de la page actuelle à notre Type dans le fichier *.config pour qu'il puisse charger la page lors d'une prochaine session.

Q.C.M.

Le premier QCM de ce cours vous est offert en libre accès.
Pour accéder aux suivants

[Connectez-vous](#) [Inscrivez-vous](#)

Que sont les fichiers de configuration ?

- ☐ Ce sont les options de l'application.
- ☐ Ce sont des fichiers qui contiennent les paramètres de l'application.
- ☐ Ce sont des fichiers qui contiennent les options de l'application.

Imaginons qu'on veuille mettre de l'argent sur votre compte en banque par l'intermédiaire des paramètres de l'application, qu'écririez-vous ?

- ☐ Properties.Settings.argentEnBanque = System.Console.ReadLine();
- ☐ argentEnBanque = System.Console.ReadLine();
- ☐ Properties.Settings.Default.argentEnBanque = System.Console.ReadLine();

Correction !

[Statistiques de réponses au QCM](#)

Ceci marque la fin du tutoriel !

Comme vous le voyez, ce système d'utilisation pour stocker vos données est très simple et est aujourd'hui utilisé dans la plupart des programmes.

Je tenais à remercier [Thunderseb](#), [Orwell](#) et [Coyote](#) pour m'avoir aidé à rendre ce tutoriel publiable.
Et aussi aux [zCorrecteurs](#) zoubab et [Fihld](#) pour avoir corrigé mon tutoriel très rapidement. 🙏

Partager



Ce tutoriel a été corrigé par les [zCorrecteurs](#).