CRÉEZ UN NAVIGATEUR WEB EN .NET

Bat'

28 mai 2016

Zeste de savoir

Table des matières

1 Introduction 5

2 Avant propos et préparation 7
- 2.1 Qu'est-ce qu'Awesomium exactement ? . . . 7
- 2.2 Téléchargement et installation : . . . 7

3 Présentation du WebBrowser et d'Awesomium 9

4 Un premier projet 11
- 4.1 Windows Form ou WPF ? . . . 11
- 4.2 Ça marche comment ? . . . 11
- 4.3 Design ! . . . 11

5 Et un peu de code ! 13

6 Du JavaScript ! 15
- 6.0.1 Comment ça marche ? . . . 15
- 6.0.2 Au travail ! . . . 15
- 6.1 TP ! . . . 15

7 TP Final : votre navigateur ! 17

8 Les autres contrôles 21
- 8.1 En Windows Form . . . 21
  - 8.1.1 Le *WebSessionProvider* . . . 21
  - 8.1.2 L'*AdressBox* . . . 21
  - 8.1.3 Le *WebControlContextMenu* . . . 21
- 8.2 En WPF . . . 21
  - 8.2.1 Le *WebDialogLayer* . . . 21

9 Annexes 23
- 9.1 Divers petits détails plus ou moins utiles : . . . 23
  - 9.1.1 Liste des fonctions et des propriétés les plus importantes . . . 23
  - 9.1.2 Bugs pouvant être rencontrés . . . 24

10 Conclusion 25

1 Introduction

Vous avez déjà essayé de créer un navigateur en C# ou en VB.NET, mais vous vous êtes rapidement rendu compte que le rendu de la plupart des sites web n'était pas correct ! Avec Awesomium tout cela va changer …

[[attention]] | Dans ce tutoriel, j'estime que vous connaissez les bases d'un des langages du .NET (C#, VB ou C++) et que vous avez un niveau correct en JavaScript car il sera utilisé. J'estime aussi que vous avez un minimum de connaissances en informatique (installations de programmes, etc.) et que vous disposez de Visual Studio pour Desktop ou de Visual Studio Community. Il faudra enfin que vous maîtrisiez le concepteur WindowsForm (ne vous inquiétez pas, il est assez intuitif) ou le langage XAML.

2 Avant propos et préparation

Awesomium est une bibliothèque .NET open source. Ses sources sont distribuées sur GitHub.

Dans ce tutoriel, j'utiliserai le C# mais vous pouvez utiliser le VB ou tout autre langage associé au .NET.

J'ai entendu certaines personnes dire que les tutoriels actuels n'étaient pas assez axés sur la partie TP. J'ai donc décidé qu'à chaque fois que je le pouvais je vous laisserai d'abord coder vous-même avant de vous donner la solution.

Maintenant, passons aux choses sérieuses :pirate : !

2.1 Qu'est-ce qu'Awesomium exactement ?

Awesomium est une bibliothèque pour le .NET qui fournit tous les éléments nécessaires à la réalisation d'un navigateur web. Il dispose en effet d'un contrôle pour afficher des pages web sans aucun bug comme avec le *WebBrowser* fourni par défaut. Il dispose également de nombreuses fonctions supplémentaires comme par exemple pouvoir exécuter du JavaScript.

Le *WebControl* n'est pas le seul contrôle qui y est fourni mais c'est le principal. Pour découvrir les autres, lisez la suite de ce tutoriel !

[[information]] | Si vous êtes sous Linux, vous serez heureux d'apprendre qu'Awesomium est compatible avec Mono !

2.2 Téléchargement et installation :

Pour télécharger Awesomium, allez sur le site officiel et cliquez sur "Get Awesomium". Installez-le. Une fois que c'est terminé, si vous lancez Visual Studio et que vous créez un nouveau projet Windows Form ou WPF vous découvrirez ... de nouveaux contrôles dans votre boîte à outils ! Génial !

3 Présentation du WebBrowser et d'Awesomium

Le *WebBrowser* est un contrôle fourni avec le framework .NET. Il va permettre d'afficher des sites internet et de créer un navigateur web. Malheureusement, il dispose du moteur de rendu d'IE 7 et la plupart des sites modernes provoquent donc des erreurs. Impossible de faire un vrai navigateur ainsi.

Heureusement, des développeurs ont eu l'idée de créer Awesomium ! Il fournit un nouveau *WebBrowser* qui respecte (presque) tous les standards du web. Ainsi, nous pouvons créer des navigateurs avec des fonctionnalités étendues. Fonctionnalités que nous découvrirons au fil de ce tutoriel.

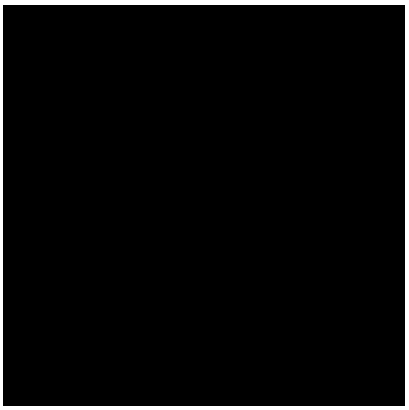

Figure 3.1 – À gauche Awesomium, à droite le *WebBrowser* du .NET avec l'erreur de script !

4 Un premier projet

4.1 Windows Form ou WPF ?

Pour utiliser Awesomium, notre projet devra être une application Windows Forms ou WPF. La console ou les applications ModernUI ne sont pas supportées par Awesomium. Dans ce tutoriel, je créerai un projet Windows Form, mais vous pouvez utiliser WPF car Awesomium ne diffère pas de l'un à l'autre (et puis je vous fournirai le XAML si vous n'avez pas trop le cœur à designer...).

Créez donc votre projet selon vos préférences et vos compétences !

4.2 Ça marche comment ?

Utiliser Awesomium est extrêmement simple. Une fois que vous aurez créé votre projet, ajoutez le contrôle appelé *WebControl* dans votre fenêtre. Changer sa propriété `Source` par l'adresse d'un site quelconque, compilez ! Et ça marche !

4.3 Design !

Vous êtes sur votre Form par défaut dans le designer de VS. Ajoutez-y :

- Un *WebControl d'Awesomium*, nommé *Navigateur* ;
- Une *TextBox*, nommée *AdressBox* ;
- Un *bouton*, nommé *Back* ;
- Un autre *bouton*, nommé *Forward* ;
- Un 3ème *bouton*, nommé *Home* ;
- Si on veut autre chose, on verra après.

Faites en sorte que tout cela soit un minimum esthétique. Chez moi, ça donne quelque chose comme ça :

Via le designer, pensez aussi à définir une page d'accueil sur votre *WebControl* avec la propriété Source. N'oubliez pas le *http ://* ou le *https ://* .

Et pour ceux qui préfèrent le WPF, voici le XAML :

```
<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:awe="http://schemas.awesomium.com/winfx" x:Class="Tuto_Awesomium_WPF.Mai
        Title="Mon premier navigateur avec Awesomium" Height="455.4" Width="659.8">
    <Grid>
```

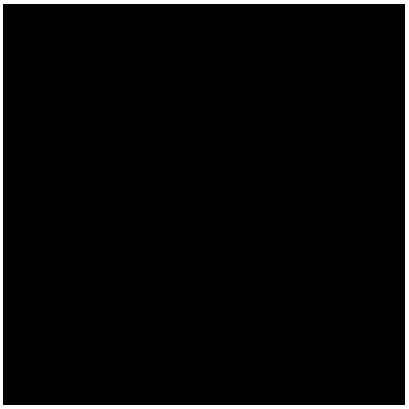

Figure 4.1 – Il faut aimer les couleurs flashs, mais bon…

```
        <Grid x:Name="Header" Height="81" VerticalAlignment="Top" Margin="0,0,-0.4,0">
            <Grid.Background>
                <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                    <GradientStop Color="#FFEEEEEE" Offset="1"/>
                    <GradientStop Color="White"/>
                    <GradientStop Color="White" Offset="0.519"/>
                </LinearGradientBrush>
            </Grid.Background>
            <TextBox x:Name="AdressBox" Height="23" Margin="10,46,10,0" TextWrapping="
            <Button x:Name="Back" Content="Précédent" HorizontalAlignment="Left" Margi
            <Button x:Name="Forward" Content="Suivant" HorizontalAlignment="Left" Marg
            <Button x:Name="Home" Content="Accueil" HorizontalAlignment="Right" Margin
        </Grid>
        <awe:WebControl x:Name="Navigateur" Margin="0,81,-0.4,-0.4" Source="https://ww

    </Grid>
</Window>
```

*[VS] : Visual Studio

5 Et un peu de code !

Maintenant, passons au code ! Prenons celui de notre Form principale. Il n'y a rien d'autre que le constructeur. On va ajouter un évènement *Back.Click* via le concepteur et dans notre méthode *Back_Click()* nous allons tester si l'on peut revenir en arrière et si oui, on le fait. Sinon on affiche un message comme "Impossible de revenir en arrière !". Je vous laisse faire !

[[information]] | Utilisez `Navigateur.CanGoBack()` et `Navigateur.GoBack()` !

Mon code :

```
using System;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;


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

        private void Back_Click(object sender, EventArgs e) // Événement déclenché au
        {
            if (Navigateur.CanGoBack() == true) // Si l'on peut revenir en arrière ...
            {
                Navigateur.GoBack(); // ... on revient en arrière.
            }
            else
            {
                MessageBox.Show("Impossible de revenir en arrière"); // Sinon on affic
            }
        }

        // La même chose avec GoForward().
        private void Precedent_Click(object sender, EventArgs e) // Événement déclench
        {
            if (Navigateur.CanGoForward() == true) // Si on peut revenir en avant ...
```

```
            {
                Navigateur.GoForward(); // ... on revient en avant.
            }
            else
            {
                MessageBox.Show("Impossible de revenir en avant"); // Sinon on affiche
            }
        }
    }
}
```

[[question]] | J'aimerais bien que…., ben qu'il y ait une adresse…

Vous vous souvenez de la propriété Source de votre *WebControl* ? Elle indique l'URL actuelle. Il suffirait de l'assigner à chaque fois que la page est prête autrement dit, avec l'événement *DocumentReady* ! Codez-la !

```
private void Awesomium_Windows_Forms_WebControl_DocumentReady(object sender, Awesomium
{
    adressBox.Text = Navigateur.Source.ToString(); //On  assigne l'adresse de notre We
}
```

Maintenant on va faire une fonction pour retourner à la page d'accueil. Vous avez tout ce qu'il vous faut pour la faire vous-même !

```
private void Home_Click(object sender, EventArgs e)
{
    Navigateur.Source = new Uri ("http://www.duckduckgo.com/"); //On assigne l'adresse
}
```

Facile, non ?

[[information]] | Je vous ai montré quelques exemples rapides. Je vous invite fortement à tester d'autres propriétés, les méthodes du *WebControl* de même que les autres éléments Awesomium.

Nous avons vu les fonctions de base d'Awesomium. Maintenant, passons à une partie plus amusante.

6 Du JavaScript !

[[information]] | Cette partie n'est pas essentielle, mais elle reste cependant intéressante.

Un des atouts d'Awesomium est qu'il permet d'exécuter du JavaScript. Ainsi nous pouvons manipuler nos pages web facilement. Je m'explique : j'aimerais bien avoir un bouton qui permet de traduire ma page rapidement : je peux le faire !

6.0.1 Comment ça marche ?

[[attention]] | Avant d'attaquer cette partie, il faut absolument avoir de bonnes bases en JavaScript. Si vous n'avez jamais créé un script vous-même, apprenez le JavaScript.

Le *WebControl* a deux méthodes pour utiliser le JavaScript :

- ExecuteJavascript() ;
- ExecuteJavascriptWithResult() ;

La première permet d'exécuter du JavaScript, tandis que la seconde l'exécute et nous renvoie le résultat.

6.0.2 Au travail !

J'ai trouvé un petit script qui permet de traduire une page avec Bing Translator.

[[secret]] | (function(){var s = document.createElement('script'); s.type = 'text/javascript'; s.src = 'http ://labs.microsofttranslator.com/bookmarklet/default.aspx?f=js&to=fr'; document.body.insertBefore(s, document.body.firstChild);})().

Ajoutons un bouton Traduire dans notre Form. Quand on clique (avec l'évènement Button.Click) dessus on lance le script. Allez-y !

Correction :

```
private void Traduction_Click(object sender, EventArgs e)
{
    Navigateur.ExecuteJavascript("(function(){var s = document.createElement('script')
    //On exécute du JavaScript contenant notre script. Pas besoin de récupérer le résu
}
```

6.1 TP !

6.1.0.0.1 Qu'est-ce qu'on fait ? Nous allons **bidouiller** YouTube (juste pour les gens qui utilisent votre navigateur, pas d'inquiétude ... :D).

Voici ce que nous allons faire : quand on arrive sur YouTube, je souhaite que la barre supérieure et le menu soient jaunes et qu'un pop-up disant : "Pour savoir ce qu'il se passe allez ici : zestedesavoir.com/tutoriels/399/creez-un-navigateur-web-en-net/" apparaisse.

[[information]] | Pour obtenir les `id`, utilisez les outils de développement de votre navigateur.

Il se peut que le chargement soit long et que le JavaScript ne s'exécute pas tout de suite, car Awesomium est un peu lent pour ça ...

On code !

6.1.0.0.2 Correction !

[[secret]] | csharp | | private void Page_Chargee(object sender, Awesomium.Core.UrlEventArgs e)// Méthode déclenchée lorsque la page est entièrement chargée (impossible d'exécuter du JS avant). | { | if (Navigateur.Source.ToString().StartsWith("https://www.youtube.com") == true) //Si l'adresse commence par "https://www.youtube.com" ... | { | | //Pour des raisons de lisibilité j'ai fait mon script en 3 parties différentes : | string script1 = "var barreSup = document.getElementById('yt-masthead-container'); barreSup.style.backgroundColor = 'yellow';"; | // Cette partie définit le jaune comme couleur de fond de la barre supérieure. | string script2 = "var menu = document.getElementById('guide-container'); menu.style.backgroundColor = 'yellow';"; | // Celui-ci fait la même chose avec le menu. | string script3 = "alert(\"Pour savoir ce qu'il se passe allez ici : http://zestedesavo\");"; | // Et enfin cette partie affiche un popup avec marqué "Pour savoir ce qu'il se passe ..." | | Navigateur.ExecuteJavascript(script1 + script2 + script3);// On exécute nos 3 scripts. | } | } |

Et le résultat en image :

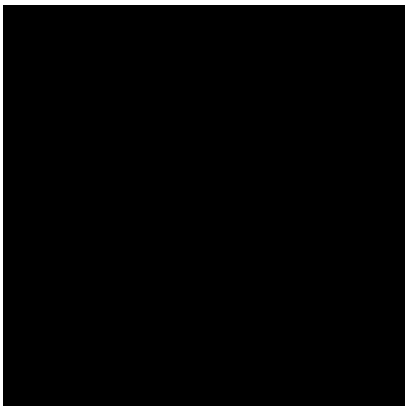

Figure 6.1 – On a tout cassé !

Ça marche :D ! Vous pouvez vous amuser avec ces méthodes autant que vous voulez, car avec elles les possibilités sont infinies.

7 TP Final : votre navigateur !

7.0.0.0.1 On fait quoi ? Vous l'avez vu dans le titre, **vous** allez créer un navigateur.

Un navigateur, oui. Mais avec quoi dedans ? Il nous faudra :

- Les boutons de base : Précédent, suivant, accueil ;
- Une barre d'adresse ;
- Une barre de recherche que vous pouvez fusionner avec la barre d'adresse ;
- Des outils de développement permettant de voir le HTML, et une console JavaScript ;
- Et un historique.

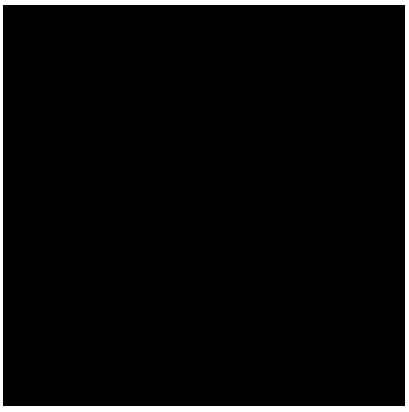

Figure 7.1 – Voici mon super navigateur !

Voici mon XAML :

```
<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:awe="http://schemas.awesomium.com/winfx" x:Class="Tuto_Awesomium_WPF.Mai
        Title="MainWindow" Height="505.8" Width="799">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition/>
        </Grid.RowDefinitions>
        <Grid x:Name="Header" Height="81" VerticalAlignment="Top" Margin="0,0,-0.2,0">
            <Grid.Background>
                <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                    <GradientStop Color="#FFEEEEEE" Offset="1"/>
                    <GradientStop Color="White"/>
                    <GradientStop Color="White" Offset="0.519"/>
                </LinearGradientBrush>
            </Grid.Background>
```

```
            <TextBox x:Name="AdressBox" Height="23" Margin="10,46,10,0" TextWrapping="
            <Button x:Name="Back" Content="Précédent" HorizontalAlignment="Left" Margi
            <Button x:Name="Forward" Content="Suivant" HorizontalAlignment="Left" Marg
            <Button x:Name="Home" Content="Accueil" HorizontalAlignment="Right" Margin
            <Button x:Name="Translate" Content="Traduire" HorizontalAlignment="Right"
            <Button x:Name="Go" Content="Go !" HorizontalAlignment="Right" Margin="0,46
            <Button x:Name="Historique" Content="Historique" HorizontalAlignment="Righ
            <Button x:Name="Devs" Content="Outils de dévellopement" HorizontalAlignmen
        </Grid>
        <awe:WebControl x:Name="Navigateur" Margin="0,81,-0.2,0" Source="http://www.du
        <ListBox x:Name="HystoryList" HorizontalAlignment="Right" Height="193" Margin=
        <Grid x:Name="DevTools" Height="175" Margin="0,0,-0.2,0" VerticalAlignment="Bo
            <awe:WebControl x:Name="HTML" Margin="10,10,10,27" IsSourceView="True"/>
            <TextBox x:Name="JSCommand" HorizontalAlignment="Left" Height="23" Margin=
            <Button x:Name="ExecuteJS" Content="Executer" HorizontalAlignment="Left" M
        </Grid>

    </Grid>
</Window>
```

À vos claviers … Prêts … Codez !

7.0.0.0.2 Corrigeons : [[secret]]|csharp | using System; | using System.ComponentModel; | using System.Drawing; | using System.Text; | using System.Windows.Forms; | | //Les using peuvent changer selon si vous faites du WindowsForm ou du WPF. | | | namespace Tuto_ZdS___Awesomium | { | public partial class Form1 : Form | { | public Form1() | { | InitializeComp | } | | private void Back_Click(object sender, EventArgs e) | { | // même code qu'au début | | if (Navigateur.CanGoBack() == true) | { | Navigateur.GoBac | } | else | { | MessageBox.Show(" de revenir en arrière"); | } | } | | private void Forward_Click(object sender, EventArgs e) | { | //même code qu'au début | | if (Navigateur.CanGoForward() == true) | { | Navigateur.GoForward(); | } | else | { | MessageBox.Show("Impossible de revenir en avant"); | | } | } | | private void Awesomium_Windows_Forms_WebControl_TargetURLChanged(object sender, Awesomium.Core.UrlEventArgs e) | { | // même code qu'au début | | adressBox.Text = Navigateur.Source.ToString(); | | } | | private void Home_Click(object sender, EventArgs e) | { | // même code qu'au début | Navigateur.Source = new Uri("http://www.duckduckgo.com"); | } | | private void Page_Chargee(object sender, Awesomium.Core.UrlEventArgs e) | { | // On crée et ajoute nos éléments à l'historique | ListViewI adresseActuelle = new ListViewItem(adressBox.Text); | HistoryList.Items.Ad | } | | private void Go_Click(object sender, EventArgs e) | { | if (adressBox.Text.StartsWith("http://") ||

```
adressBox.Text.StartsWith("https://") == true)  |             { |                 Navi
= new Uri(adressBox.Text); |                 // si le texte tapé comence
par http:// ou https:// on va a l'adresse tapée |  |             } |             else
|             { |                 Navigateur.Source = new Uri("https://www.duckduckgo.
+ adressBox.Text); |  |                 //Sinon on le recherche |             }
|         } |  |         private void Dev_Click(object sender, EventArgs
e) |         { |             HTML.Source = Navigateur.Source; |             DevTools.V
= true; |             //on obtient la source et on affiche les outils de
développement |         } |  |         private void OkJs_Click(object sender,
EventArgs e) |         { |             try |             { |                 //
On tente d'exécuter le JavaScript entré par l'utilisateur  |                 Navigateu
|             } |             catch (Exception ex) |             { |                 /
gère les exceptions (notamment s'il tape n'importe quoi) |                 MessageBox.
d'exécuter le JavaScript : \n \n " + ex.Message); |             } |         }
|     } | } ||
```

Remarque :

Oui, avec ce code notre historique est un peu **bizarre**, mais on pourrait facilement l'arranger, soit en utilisant un `string[]` et un `foreach`, soit en coupant le texte du *ListViewItem*.

8 Les autres contrôles

Jusqu'ici nous n'avions vu que le *WebControl*, mais si vous êtes observateurs vous avez déjà dû remarquer la présence d'autres contrôles, comme l'*AdressBox*. À la fin cette partie, vous ne vous demanderez plus jamais à quoi ils servent.

8.1 En Windows Form

8.1.1 Le *WebSessionProvider*

Ce contrôle va surtout nous permettre de configurer plus en détail notre navigateur. Ajoutez un *WebSessionProvider* et déployez sa propriété *Preferences*. Vous avez ici toutes sortes de propriétés pour modifier les paramètres de votre navigateur. Par exemple, avec *WebGL*, vous autorisez ou non l'affichage de WebGL dans votre navigateur. Vous pouvez faire une personnalisation avancée de votre navigateur ainsi, mais si vous lancez, vous verrez que rien n'a changé. Il faut en fait une autre étape, indispensable : associer votre *WebControl* avec votre *WebSessionProvider*. Pour cela, modifiez simplement la propriété *WebSessionProvider* de votre *WebControl*.

8.1.2 L'*AdressBox*

Ce contrôle va nous faire gagner du temps : il fait apparaître automatiquement l'adresse de la page qui lui est associée. Plus besoin de changer l'adresse manuellement à chaque nouvelle page. Il n'est pas très esthétique mais on peut facilement changer cela. De toute façon l'esthétique ne nous intéresse pas vraiment. :p

8.1.3 Le *WebControlContextMenu*

Derrière ce nom à rallonge se cache un contrôle qui ne fonctionne pas. Du moins pas avec moi. Il devrait normalement afficher un menu contextuel au clic droit, mais on a beau appuyer, il ne se passe rien. Je ne détaillerai pas ses fonctionnalités ici (en plus ça ressemble à un *ContextMenu* normal).

8.2 En WPF

8.2.1 Le *WebDialogLayer*

Il sert à afficher les popups. Ainsi on peut les positionner là où on veut ou choisir leur style.

9 Annexes

9.1 Divers petits détails plus ou moins utiles :

9.1.1 Liste des fonctions et des propriétés les plus importantes

Fonctions :

Fonction	Description
CanGoBack()	Retourne un *bool* indiquant si l'on peut revenir en arrière
GoBack()	Renvoie à la page précédente
CanGoBack()	Retourne un *bool* indiquant si l'on peut revenir en avant
GoForward()	Renvoie à la page suivante
ExecuteJavascript()	Exécute du JavaScript dans la page en cours (attention : attendre l'évènement DocumentReady pour utiliser cette méthode)

Fonction	Description
ExecuteJavascriptWithResult()	Exécute du JavaScript dans la page en cours et renvoie le résultat sous la forme d'un *JSValue* (attention : attendre l'évènement DocumentReady pour utiliser cette méthode)
Refresh()	Actualise la page
Stop()	Arrête le chargement de la page

Propriétés :

Propriété	Description	Type
Source	Définit ou obtient l'adresse de la page	*System.Uri*
HTML	Obtient le code HTML de la page	*string*
Selection	Obtient la sélection actuelle	*Awesomium.Core.Selection*
TargetURL	Obtient l'adresse du lien survolé par l'utilisateur	*System.Uri*
Title	Obtient le titre de la page	*string*
Zoom	Obtient ou définit le niveau de zoom de la page	*int*

9.1.2 Bugs pouvant être rencontrés

— Certains sites (comme SoundCloud) se servent du UserAgent pour détecter si votre navigateur est récent. Seulement, Awesomium a pour UserAgent celui de Chromium 18, ce qui fait que ces sites nous détectent comme trop ancien et nous empêchent parfois même l'accès !
— Certaines propriétés CSS ne sont pas encore gérées (je n'en connais pas la liste exacte), par exemple, sur Zeste de Savoir, le menu latéral est affiché en bas de la page !
— Le WebGL n'est pas encore supporté pour Windows 8.
— Certaines touches de clavier font perdre le focus au *WebControl*, il est donc impossible de les écrire.

10 Conclusion

C'est déjà la fin de ce tutoriel. J'espère qu'il vous aura plu, et que vous arriverez à détrôner Google Chrome :D ! N'hésitez pas à aller voir la doc sur le site d'Awesomium (vous avez aussi un wiki local dans les documents publics de votre PC). Si vous avez un problème vous pouvez poster un commentaire (ou envoyer un MP si je ne réponds pas), aller sur le site Question/Réponse officiel (anglais uniquement) ou poster une description de votre problème sur le forum de programmation de ZdS.

Vous savez maintenant créer de superbes navigateurs web ! Amusez-vous bien !