



# Introduction au Web Service de Windows Live Search

---

Florian Casabianca

17/04/2007

---

## Remerciements

J'adresse ici tous mes remerciements à l'équipe de rédaction de "developpez.com" pour le temps qu'ils ont bien voulu passer à la correction et à l'amélioration de cet article.

---

## Table des matières

Introduction.....	3
L'application .....	3
Avant de commencer .....	4
Référencement du Web Service Windows Live Search.....	5
Principes d'utilisation .....	6
Construction de l'objet SearchRequest.....	7
Présentation .....	7
La propriété Requests .....	8
Traitement des résultats .....	10
Conclusion .....	16
Liens.....	16

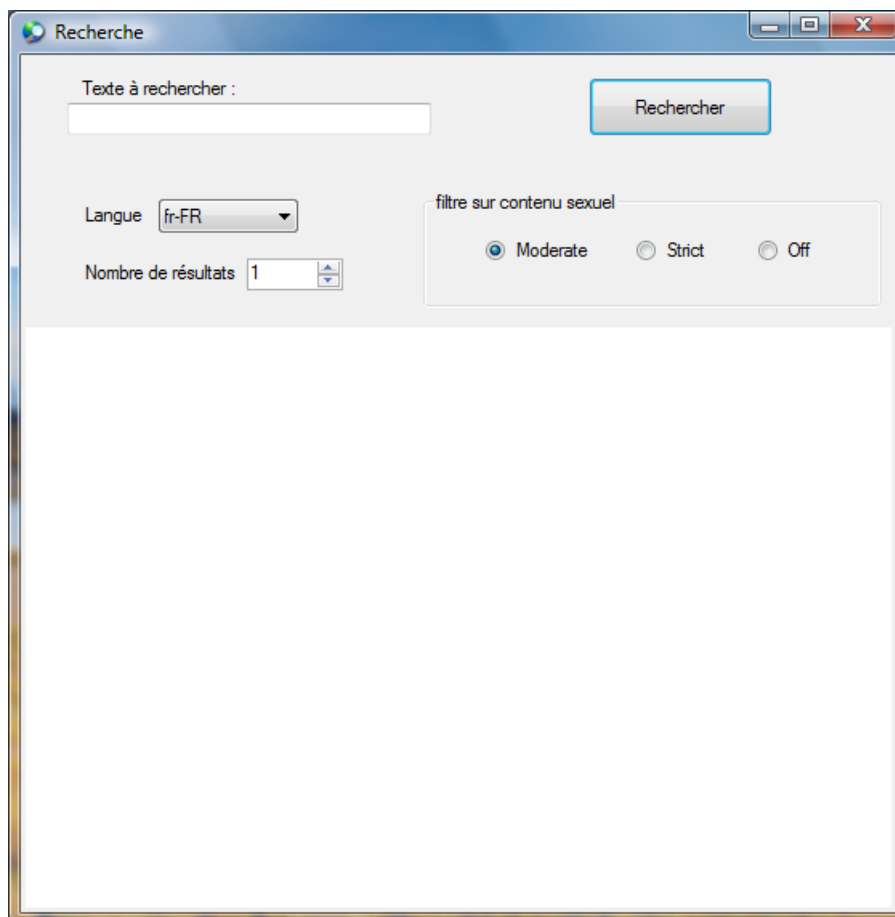
## Introduction

Vous connaissez tous <http://search.live.com> le moteur de recherche de Microsoft. Ce dernier possède une API (application programming interface) actuellement en version 1.1 qui va nous permettre de soumettre des requêtes par programmation au moteur de Live Search et d'en récupérer les résultats.

## L'application

Nous allons ici illustrer l'utilisation de ce web service en développant une petite application Winform qui enverra une requête définie par l'utilisateur et affichera les résultats transmis par Live Search.

Voici à quoi l'application va ressembler (notez le talent du designer):



Rien de bien compliqué : une *textbox* pour la requête, une *combobox* pour choisir la langue dans laquelle faire la requête, un *numericUpDown* pour sélectionner le nombre de résultats et enfin trois *radioButton* pour définir le niveau de filtrage du contenu du résultat de la requête. Les résultats s'afficheront dans un contrôle *webbrowser* (nous construirons donc un contenu HTML contenant les résultats que nous fournirons au *webbrowser*).

## Avant de commencer

Chacune des applications que vous allez développer et qui voudra utiliser le Web Service de **Live Search** devra disposer d'un identifiant unique (Application ID) fourni par le "*Developer Provisioning System*".

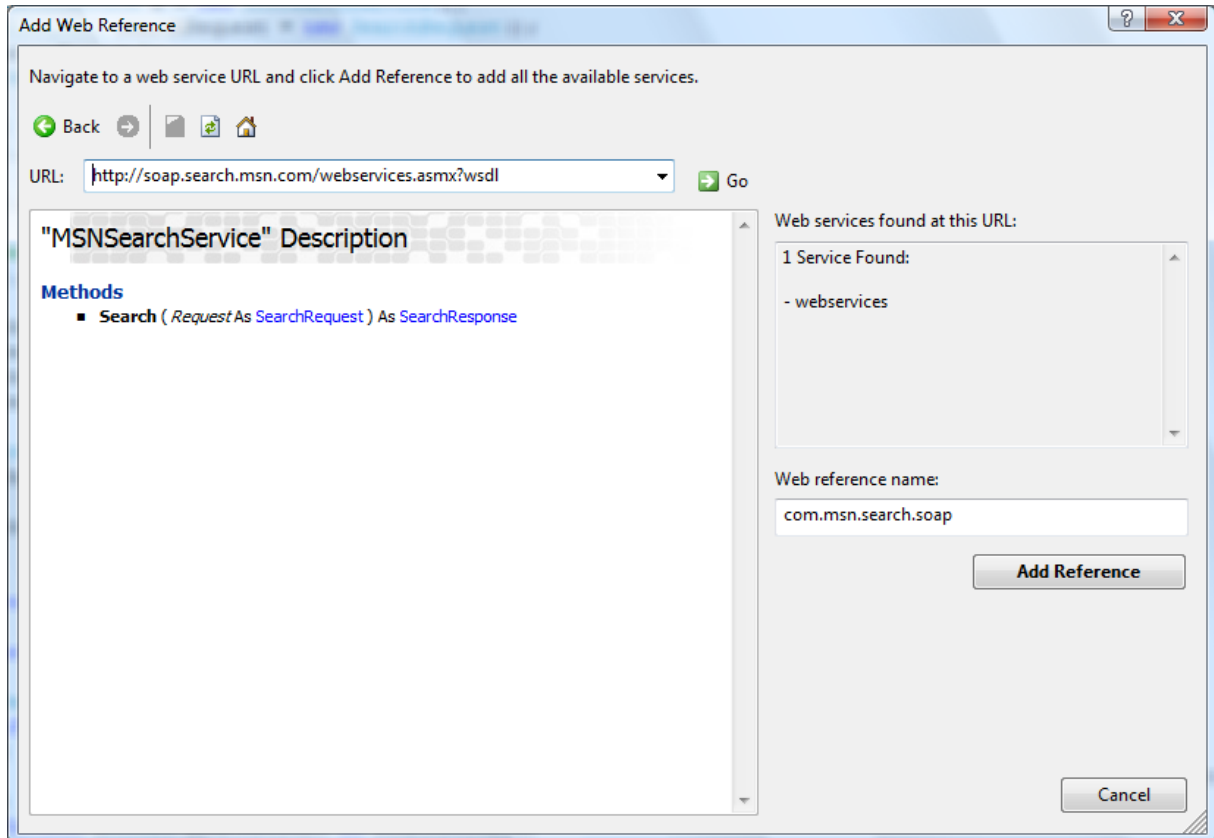
Suivez les instructions suivantes pour créer un identifiant pour votre application :

1. Rendez-vous à l'adresse : **<http://search.msn.com/developer>**.
2. Cliquez sur **Create and Manage Application IDs**.
3. Identifiez-vous avec votre compte Windows Live. Si vous n'en possédez pas, cliquez sur **Sign Up Now** pour en créer un.
4. Une fois connecté, cliquez sur **Get a new App ID** pour créer un nouvel identifiant pour votre application
5. Tapez un nom pour votre application dans le champ dédié.
6. Après avoir accepté les termes de la licence, vous pouvez sélectionner et copier l'identifiant qui vous est fourni et le coller dans votre code. Nous montrerons plus loin comment et où l'utiliser.

## Référencement du Web Service Windows Live Search

Votre projet doit référencer le Web Service de Live Search pour pouvoir l'utiliser. Voici la procédure à suivre dans Visual Studio 2005 :

Dans l'explorateur de solutions, faites un clic droit sur "référence" et choisissez "Ajouter une référence Web". Une nouvelle fenêtre s'ouvre. Saisissez l'adresse suivante dans le champ URL : "<http://soap.search.msn.com/webservices.asmx?wsdl>" :



Vous pouvez accepter le nom qui vous est proposé par défaut "**com.msn.search.soap**" ou le modifier par votre propre nom. Dans l'exemple que nous allons étudier, nous avons laissé la valeur par défaut.

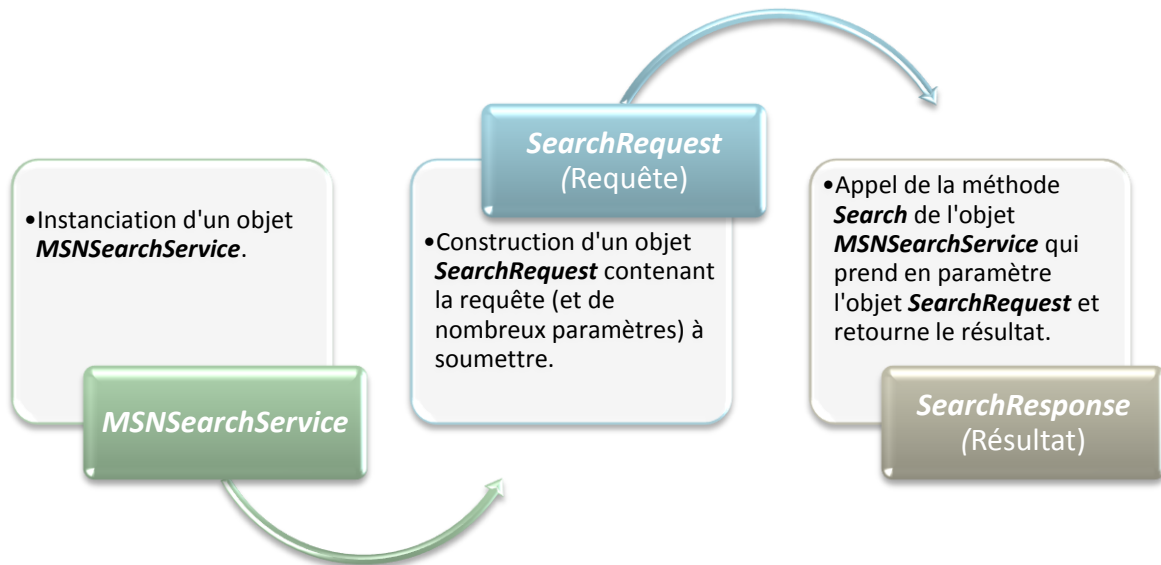
Cliquez ensuite sur le bouton "Ajouter la référence".

Maintenant dans votre code, vous devez utiliser la directive **using** afin d'utiliser le web service. Par exemple, si l'espace de nom de votre projet est "**WindowsLiveSearch**" vous devrez ajouter la ligne suivante dans votre code :

```
using WindowsLiveSearch.com.msn.search.soap;
```

## Principes d'utilisation

L'utilisation du Web Service de Live Search peut être résumée de la façon suivante :



La classe **MSNSearchService** va principalement nous servir pour exécuter la requête grâce à sa méthode **search**. La requête (au sens large) est représentée par l'objet **SearchRequest** qui contient un certain nombre de propriétés à remplir comme le texte de la requête, les sources où chercher (sites web, news, etc.), la langue, etc. Enfin, le résultat (au sens large) de la requête sera représenté par un objet **SearchResponse**.

Voici à quoi va ressembler le squelette de notre application :

```
MSNSearchService s = new MSNSearchService();  
  
SearchRequest searchRequest = new SearchRequest();  
  
//construction de l'objet SearchRequest  
...  
...  
  
SearchResponse searchResponse = s.Search(searchRequest);  
  
//traitement du résultat
```

Dans les deux prochaines parties nous allons voir comment construire un objet **SearchRequest** et comment traiter un objet **SearchResponse** pour récupérer les résultats de notre requête.

# Construction de l'objet SearchRequest

## Présentation

Voyons tout d'abord le diagramme de la classe **SearchRequest** :

▫

**AppID** (obligatoire) : identifiant de l'application pour l'utilisation du Web Service Live Search.

**CultureInfo** (obligatoire) : langue dans laquelle faire la recherche.

**Query** (obligatoire) : texte soumis au moteur de recherche.

**Requests** (obligatoire) : tableau de **SourceRequest** contenant des informations sur les sources où faire les recherches. Plus d'explications ci-après.

**SafeSearch** (optionnel) : option du filtre concernant le contenu à caractère sexuel.

Nous ne présenterons pas ici les propriétés **Flags** et **Location**.

La valeur de **SafeSearch** est à choisir parmi l'énumération **SafeSearchOptions** :

Nom	Description
<b>Moderate</b>	Le moteur de Windows Live Search filtre seulement les images à caractère sexuel. C'est l'option par défaut.
<b>Strict</b>	Le moteur de Windows Live Search filtre à la fois les images et le contenu à caractère sexuel.
<b>Off</b>	Le moteur de Windows Live Search ne filtre pas le contenu.

Nous pouvons ainsi commencer à construire notre objet **SearchRequest** de cette façon :

```
MSNSearchService s = new MSNSearchService();

//instanciation de l'objet SearchRequest
SearchRequest searchRequest = new SearchRequest();

searchRequest.Query = textBoxQuery.Text; //texte de la requête
searchRequest.AppID = "mettez ici votre ID";
searchRequest.CultureInfo = comboBoxLangue.SelectedItem.ToString(); //par exemple fr-FR
if (radioButtonOff.Checked) //filtre sur le contenu
    searchRequest.SafeSearch = SafeSearchOptions.Off;
else if (radioButtonStrict.Checked)
    searchRequest.SafeSearch = SafeSearchOptions.Strict;
else
    searchRequest.SafeSearch = SafeSearchOptions.Moderate;
```

Il nous reste maintenant à voir comment remplir la propriété **Requests**.

## La propriété `Requests`

Cette propriété attend un tableau d'objets `SourceRequest` qui va déterminer où effectuer les recherches. Chaque objet `SourceRequest` contient des informations sur un lieu de recherche précis.

Voici le diagramme de la classe `SourceRequest` :

■

**Count** : spécifie le nombre de résultats par `SourceRequest`.

**FileType** : spécifie le type de fichiers (doc, pdf, etc.) à retourner. Uniquement si `Source` vaut `SourceType.Web`.

**ResultFields** : spécifie le ou les champs résultat à renvoyer (à choisir parmi l'énumération `ResultFieldMask`). On peut par exemple choisir de ne renvoyer que le titre, l'url, un résumé ou encore tous les champs.

**Source** : Source de recherche (à choisir parmi l'énumération `SourceType`). On peut par exemple effectuer la recherche dans les pages web, les news, les images, etc.

Les autres propriétés ne seront pas abordées.

Il ne peut y avoir qu'un seul objet `SourceRequest` par type de source (`SourceType`). Comme il n'existe que 8 types de sources, le tableau d'objets `SourceRequest` ne pourra contenir plus de 8 objets `SourceRequest`. S'il y a par exemple plusieurs objets `SourceRequest` dont la propriété `Source` vaut `SourceType.Web`, seul le premier sera pris en compte.

Voici le code de notre exemple nous permettant de remplir la propriété `Requests` :

```
int arraySize = 2;
SourceRequest[] sr = new SourceRequest[arraySize];

//recherche dans les pages web
sr[0] = new SourceRequest();
sr[0].Source = SourceType.Web;
sr[0].ResultFields = ResultFieldMask.All;
//nombre de résultats
sr[0].Count = Convert.ToInt32(numericUpDown1.Value);

//recherche dans les news
sr[1] = new SourceRequest();
sr[1].Source = SourceType.News;
//on ne veut que le titre et l'url
sr[1].ResultFields = ResultFieldMask.Title | ResultFieldMask.Url | ResultFieldMask.DateTime;
sr[1].Count = Convert.ToInt32(numericUpDown1.Value);
```

```
//recherche dans les images
sr[2] = new SourceRequest();
sr[2].Source = SourceType.Image;
sr[2].ResultFields = ResultFieldMask.All | ResultFieldMask.Image;
sr[2].Count = Convert.ToInt32(numericUpDown1.Value);

searchRequest.Requests = sr;
```

On crée ici un tableau d'objets **SourceRequest** de longueur 3 car on prévoit de ne faire des recherches que dans trois sources différentes (pages Web, news et images).

Pour la recherche dans les pages Web on souhaite récupérer tous les champs disponibles (**ResultFieldMask.All**).

Pour la recherche dans les news on ne veut récupérer que le titre, l'url et la date d'édition des quatre premiers résultats. Notez l'utilisation de l'opérateur "|" qui permet de combiner les champs à récupérer.

Pour la recherche dans les Images on souhaite aussi récupérer tous champs disponibles (**ResultFieldMask.All**) ainsi que ceux spécifiques aux images (**ResultFieldMask.Image**).

Enfin à la dernière ligne on affecte le tableau d'objets **SourceRequest** à la propriété **Requests** de l'objet **SearchRequest**.

Nous spécifions ici le même nombre de résultats par **SourceRequest** (avec la propriété **Count**). Cependant nous aurions très bien pu définir des valeurs différentes. Par exemple, cinq résultats pour la recherche dans les pages web, trois pour les news et seulement deux pour la recherche dans les images.

Nous venons de terminer la construction de l'objet **SearchReques**. Il ne reste maintenant plus qu'à effectuer la recherche et à en récupérer les résultats.

## Traitement des résultats

L'exécution de la recherche s'effectue avec la méthode **Search** de l'objet **MSNSearchService** (voir première partie).

```
SearchResponse searchResponse = s.Search(searchRequest);
```

Cette méthode retourne un objet de type **SearchResponse** contenant l'ensemble des résultats (au sens large) de la requête.

L'objet **SearchResponse** ne possède qu'une propriété **Responses** renvoyant un tableau d'objets **SourceResponse**.

Ce tableau de **SourceResponse** correspond au tableau de **SourceRequest** de l'objet **SearchRequest**. Vous aurez donc autant d'objets **SourceResponse** qu'il y avait d'objets **SourceRequest**. Chaque **SourceResponse** contient les résultats de la recherche effectuée dans un type de source (web, news, etc.) défini dans l'objet **SourceRequest** correspondant.

Ainsi, dans notre exemple nous n'avons sélectionné que trois types de sources (Web, news et images) donc notre tableau ne contiendra que trois objets **SourceResponse** : un objet contenant les résultats pour la source Web, un objet contenant les résultats pour la source News et un objet contenant les résultats pour la source Images.

Voici le diagramme de la classe **SourceResponse** :

**Total** : nombre total estimé de résultats pour la source.

**Results** : tableau d'objets **Result**.

**Source** : type de source dont proviennent les résultats (web, news, etc.).

Les objets **Result** sont les résultats de la requête sur une source donnée. Ils possèdent un grand nombre de propriétés permettant de récupérer les informations concernant chaque résultat (titre, URL, résumé, date, etc.).

Voici le diagramme de la classe **Result** :

¶

Nous n'allons pas décrire ici chacune des propriétés, pour cela vous disposez de l'aide MSDN. Mais les noms des propriétés sont tout de même assez explicites.

Arrêtons-nous cependant un peu sur la propriété **Image**. Celle-ci renvoie un objet de type **Image**. Il ne s'agit pas du type `Image` du Framework .NET mais d'un autre type d'objet dont le diagramme se trouve ci-dessous. Il n'est bien sûr accessible que si vous avez effectué une recherche dans la source *Image* (voir plus haut).

¶

Cet objet rassemble les propriétés concernant une image qui a été retenue comme résultat de votre requête. Nous avons par exemple accès à sa taille, ses dimensions (hauteur et largeur) ou encore son URL.

L'équivalent est également disponible pour sa représentation sous forme de vignette (*Thumbnail*).

Comme vous le voyez nous n'avons pas accès à l'image elle-même (le rapatriement serait trop lourd). Pour la récupérer vous devrez utiliser l'URL fournie par cette classe. Utilisez de préférence l'URL de la vignette qui permet de récupérer la miniature de l'image pour sa présentation à l'utilisateur plutôt que directement l'URL de l'image complète (comme le font les moteurs de recherche).

Pour résumer, la récupération des résultats se fait en bouclant sur le tableau de **SourceResponse**, puis pour chaque **SourceResponse** en bouclant sur le tableau de **Result** pour en récupérer les propriétés.

Voici le code permettant de récupérer les résultats de notre requête en utilisant deux boucles imbriquées. Nous affichons les résultats dans un contrôle *WebBrowser*, c'est pourquoi nous construisons une chaîne au format HTML.

```
private void AfficherResultas(SearchResponse searchResponse)
{
    StringBuilder sb = new StringBuilder();
    sb.Append("<HTML><BODY>");
    foreach (SourceResponse sourceResponse in searchResponse.Responses)
    {
        Result[] sourceResults = sourceResponse.Results;
        if (sourceResponse.Total > 0)
        {
            // Type de la source
            sb.Append("<h3><b>Source : " + sourceResponse.Source.ToString() + " -
                Nombre de résultats: ");
            // Nombre de résultats estimé
            sb.Append(sourceResponse.Total.ToString() + "</b></h3><hr>");
        }
        foreach (Result sourceResult in sourceResults)
        {
            if ((sourceResult.Title != null) && (sourceResult.Title != String.Empty))
                sb.Append("<b>Titre: </b>" + sourceResult.Title + "<br>");
            if ((sourceResult.Description != null) && (sourceResult.Description !=
                String.Empty))
                sb.Append("<b>Description: </b>" + sourceResult.Description + "<br>");
            if ((sourceResult.Url != null) && (sourceResult.Url != String.Empty))
                sb.Append("<b>URL: </b>" + sourceResult.Url + "<br>");

            if ((sourceResult.CacheUrl != null) && (sourceResult.CacheUrl != String.Empty))
                sb.Append("<b>CacheUrl: </b><a href=\"\" + sourceResult.CacheUrl + \"\">\" +
                    sourceResult.CacheUrl + "</a><br>");
            if ((sourceResult.Source != null) && (sourceResult.Source != String.Empty))
                sb.Append("<b>Source: </b>" + sourceResult.Source + "<br>");

            if (sourceResult.DateTime != null)
            {
                int year = sourceResult.DateTime.Year;
                int month = sourceResult.DateTime.Month;
                int day = sourceResult.DateTime.Day;
                int hour = sourceResult.DateTime.Hour;
                int minute = sourceResult.DateTime.Minute;
                int second = sourceResult.DateTime.Second;

                System.DateTime newsDateTime = new System.DateTime(year, month, day, hour,
                    minute, second);
                // Affichage de la date pour les news.
                sb.Append("<B>Date: </B>" + newsDateTime.ToString() + "<BR>");
            }
        }
    }
}
```

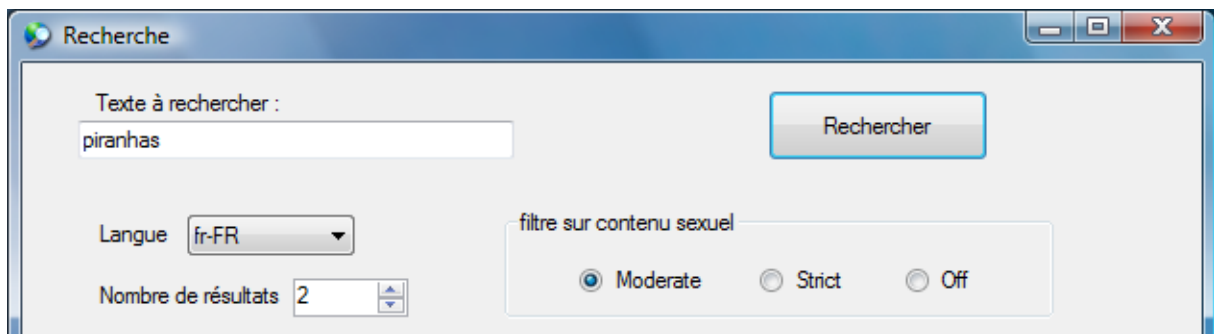
```

// Affichage des infos pour les images.
if (sourceResult.Image != null)
{
    if (sourceResult.Image.ThumbnailFileSizeSpecified)
        sb.Append("<B>Taille de la vignette: </B>" +
            sourceResult.Image.ThumbnailFileSize.ToString() + "<BR>");
    if (sourceResult.Image.ThumbnailHeightSpecified &&
        sourceResult.Image.ThumbnailWidthSpecified)
        sb.Append("<B>Hauteur vignette: </B>" +
            sourceResult.Image.ThumbnailHeight.ToString() +
            ", <B>Largeur vignette: </B>" +
            sourceResult.Image.ThumbnailWidth.ToString() + "<BR>");
    sb.Append("<B>URL vignette: </B>" + sourceResult.Image.ThumbnailURL +
        "<BR>");
    if (sourceResult.Image.ImageFileSizeSpecified)
        sb.Append("<B>Taille: </B>" + sourceResult.Image.ImageFileSize.ToString()
            + "<BR>");
    if (sourceResult.Image.ImageHeightSpecified &&
        sourceResult.Image.ImageWidthSpecified)
        sb.Append("<B>Hauteur: </B>" + sourceResult.Image.ImageHeight.ToString() +
            ", <B>Largeur: </B>" + sourceResult.Image.ImageWidth.ToString() +
            "<BR>");
    sb.Append("<B>URL: </B>" + sourceResult.Image.ImageURL + "<BR>");
    sb.Append("" + "<BR>");
}
sb.Append("<br>");
}
}
sb.Append("</BODY></HTML>");
webbrowser1.DocumentText = sb.ToString();
}

```

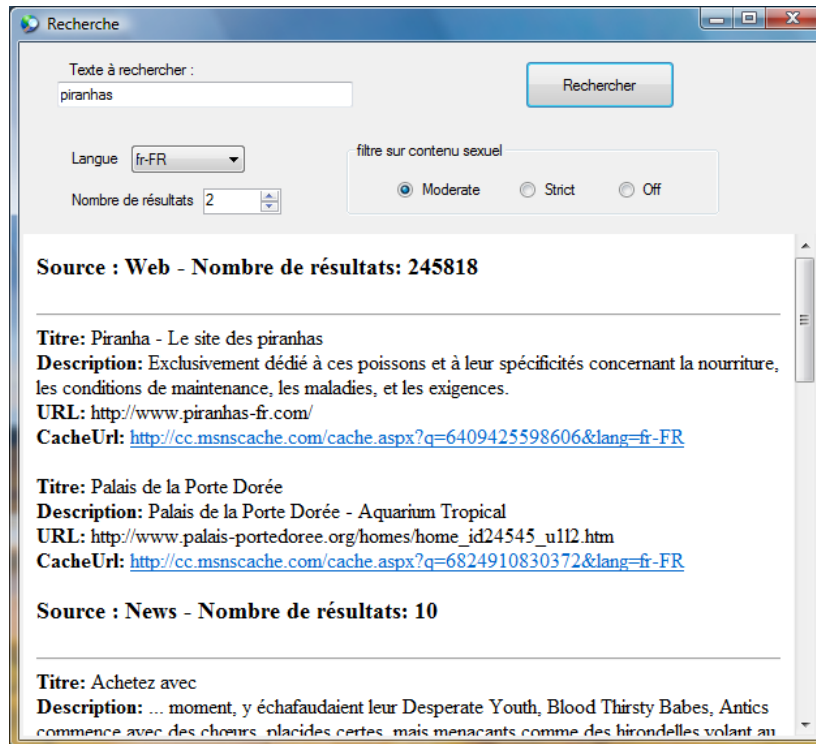
Voyons maintenant ce que cela donne avec notre petite application.

Nous allons faire une recherche sur les piranhas. La recherche s'effectuera sur des sources en français et nous ne souhaitons récupérer que deux résultats par type de sources (web, news et images).

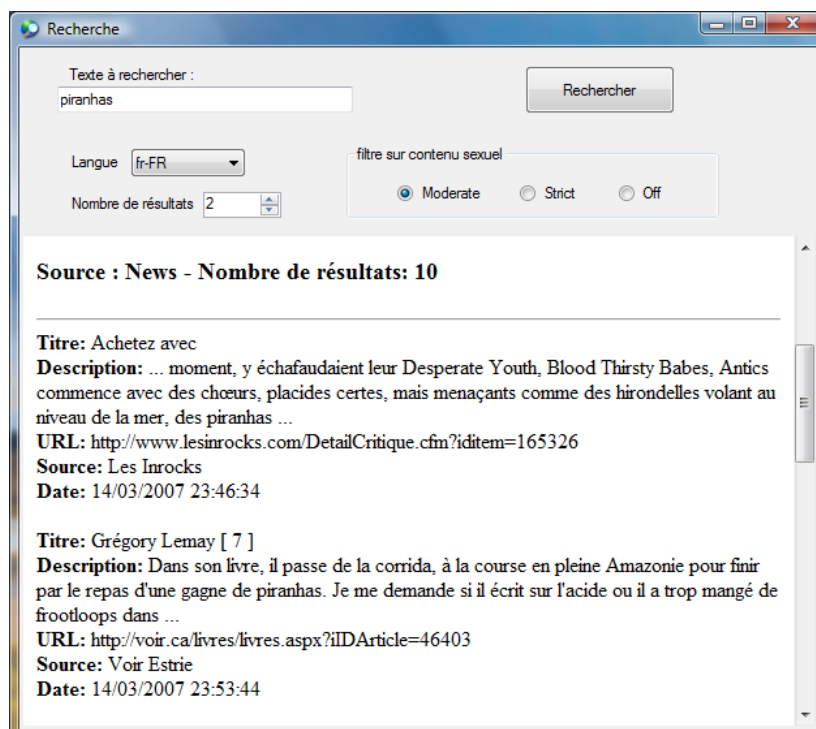


Voici donc le résultat obtenu :

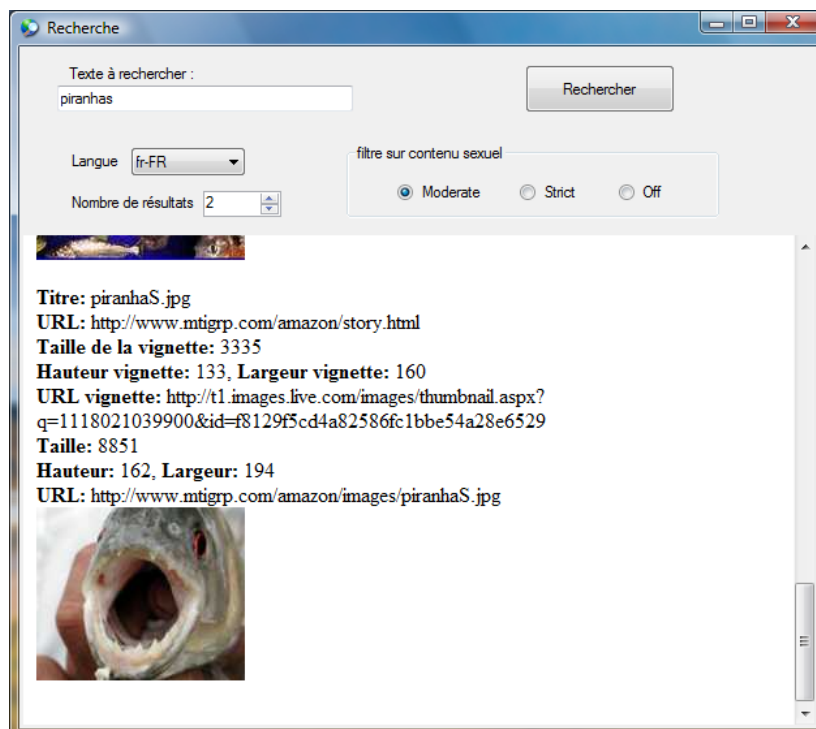
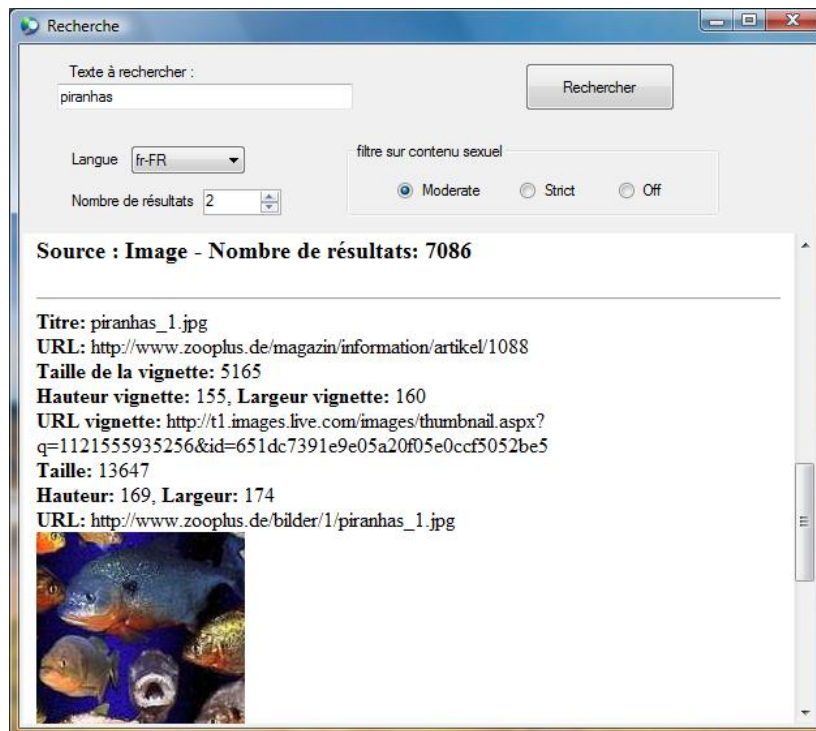
Tout d'abord les deux résultats dans la source Web.



Puis les résultats de type News :



Et enfin les images dont nous n'affichons que les miniatures :



Attention, ça mord !

## Conclusion

Nous venons de voir dans cet article un bref aperçu des possibilités de l'API de Windows Live Search en version 1.1. Les fonctionnalités évoquées ne sont bien sûr pas exhaustives et c'est pourquoi je vous invite à découvrir les nombreuses autres possibilités de l'API, qui va s'en doute s'enrichir plus tard, sur le site MSDN.

## Liens

[Windows Live Search API sur MSDN](#) 