

PicasaNet : Accédez à vos WebAlbums Google Picasa, restitution dans un ControlToolkit Ajax, Sérialisation et Désérialisation avec les technologies ASP.Net 2.0 et c#

20 Avril 2007

Par Laurent Geffroy

<http://www.laurentgeffroy.com>

[Version PDF à télécharger](#)
[Code Source](#)

Droit de diffusion et copyrights :

Ce tutoriel ne peut être diffusé sur d'autres sites Web ou tout autre support sans l'autorisation préalable de son auteur. L'utilisation des sources est totalement libre de droits.

Toutes les marques citées appartiennent à leurs auteurs, en particulier les marques Google, Picasa, ainsi que leurs logos respectifs sont des marques déposées par Google Inc.

Avant propos :

Cela fait bientôt un an que j'ai découvert le framework Atlas, renommé ASP.Net Ajax 1.0. La richesse de cette technologie offre de nombreuses voies pour des tutoriels et la sortie de Picasa2 me permet d'aborder sur un cas concret. Début Janvier 2007, me voila en route pour faire un SlideShow en Ajax. 3 jours plus tard, le voici disponible dans la AjaxControlToolkit, réduisant à néant mes premières approches !!! Restait à l'améliorer et à l'adapter quelque peu.

Picasa dispose donc d'une API assez simple, qui s'attaque par une requête http en GET toute simple. L'exposition se fait via un flux rss, assez verbeux et qu'il convient de nettoyer. Nous aborderons ici des points techniques que j'ai pu déjà exposer lors de précédents tutoriels. Mais mieux vaut se répéter et disposer d'un tutoriel complet.

Ainsi nous aborderons les points suivants :

- Récupération du flux XML via un HttpResponseMessage
- Transformation du flux via XSL
- Désérialisation des données dans une classe .Net
- L'utilisation du composant PicasaNet

Puis coté Ajax

- Le Webservice permettant de récupérer les requêtes
- Le Toolkit Ajax et son fonctionnement

Le tutoriel s'articule sur une solution disposant :

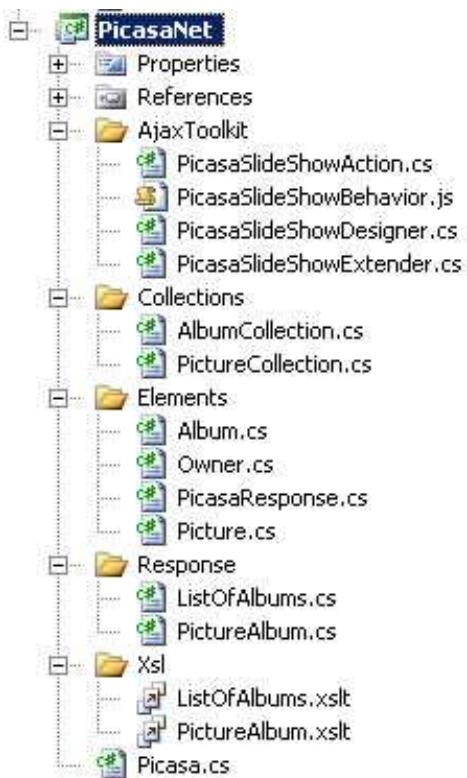
- Du projet PicasaNet qui permet d'interagir avec l'API de Picasa et qui dispose d'un AjaxControlToolkit
- Un projet Web qui exploite le composant Ajax et l'API de PicasaNet

1. PicasaNet

1.1 Qu'est-ce que c'est ?

PicasaNet est un composant gratuit et libre de droits qui vous permet d'interroger l'API de Google Picasa. Les données récupérées sont stockées dans des classes propres à PicasaNet, ce qui simplifie grandement l'accès, la manipulation et d'utilisation des informations fournies par Google Picasa. Le mode de récupération est plutôt évolutif. En ayant accès aux sources vous pouvez parfaitement enrichir les informations récupérées par PicasaNet.

La récupération est assez originale car elle se base sur une transformation XSL pour se caler aux structures des classes PicasaNet. Ce n'est peut être pas forcément la manière la plus efficace de faire, mais elle permet d'aborder une méthode différente et techniquement plus intéressante.



Le dossier AjaxToolkit dispose du composant Ajax qui permet de restituer le contenu d'un WebAlbum Picasa.

Notez que le fichier JS doit absolument être une ressource incorporée. Pour cela, sélectionnez votre fichier, puis dans l'écran de propriétés, Action de génération choisissez « Ressource incorporée ».

Les dossiers Collections et Elements disposent de classes qui permettent de stocker les données issues de Picasa.

Response dispose des classes de base à utiliser. ListOfAlbums pour disposer de l'ensemble des albums d'un compte et PictureAlbum pour avoir l'ensemble des images d'un album.

Dans Xsl vous trouverez les deux feuilles de style qui permettent de transformer le flux RSS Picasa vers la structure PicasaNet. **Ces feuilles de styles devront être déployées manuellement sur le site Web.**

Picasa.cs est la classe de base à instancier pour utiliser le composant.

Il est à préciser que vous devez avoir installer l'environnement AJAX 1.0 pour modifier ou compiler ce composant. Rendez vous sur <http://ajax.asp.net/downloads/default.aspx?tabid=47> et installez :

- ASP.Net 2.0 AJAX Extensions
- ASP.Net AJAX Control Toolkit
- ASP.Net Ajax Futures CTP

1.2 La récupération des flux XML

1.2.1 Tester son application avec ces propres URL

Il vous est bien entendu possible de tester vos propres Url de flux RSS Picasa notamment lorsque vous être en mode déconnecté. L'objet Picasa dispose d'une propriété `AllowUrlChanging` que vous pouvez mettre à `True`. Cela permet ensuite d'être en mesure de fournir vos propres Url locales ou distantes en modifiant ensuite les propriétés `SetGoogleAlbumsListUrl` (pour le flux de liste des albums) et `SetGoogleAlbumUrl` (pour le flux d'un album). Si ces propriétés ne sont pas initialisées, l'API prendra l'Url par défaut de Google Picasa, à laquelle il ajoutera les données variables comme le compte Picasa et éventuellement le numéro d'Album.

1.2.2 La liste des albums

La méthode `GetAlbumsList()` vous permet de récupérer dans une classe `ListOfAlbums` l'ensemble des albums d'un compte Google Picasa déterminé.

La première étape consiste à vérifier si le compte Picasa est bien fourni. C'est la méthode privée `CheckRequiredArgs` qui s'en charge. Elle ne vérifie la présence des données obligatoires que si vous laissez à l'API le soin de définir les Url de destination (voir chapitre 1.2.1). En fonction, `BuildRequestForListOfAlbums` construira la requête dans la variable `_request` qui est une variable membre.

```
if (CheckRequiredArgs(_user))
{
    // Build httpRequest set _request variable
    BuildRequestForListOfAlbums();
    // Send Request the Google Picasa
    HttpWebResponse v_response = DoGetResponse(_request);
}
```

`DoGetResponse` récupère le contenu du flux RSS. Il s'agit d'invoquer un `HttpRequest` sur l'url construite. Il aurait été possible d'utiliser ici directement la variable membre `_request` plutôt qu'un paramètre dans la méthode.

```

private HttpResponseMessage DoGetResponse(string p_url)
{
    HttpRequest v_request = null;
    HttpResponseMessage v_response = null;

    // Initialise the web request
    v_request = (HttpRequest)HttpRequest.Create(p_url);

    v_request.UserAgent = USERAGENT;

    v_request.Timeout = _timeOut;
    v_request.KeepAlive = false;

    v_request.Method = "GET";

    if (Proxy != null)
        v_request.Proxy = Proxy;

    try
    {
        // Get response from the internet
        v_response = (HttpResponse)v_request.GetResponse();
    }
    catch (WebException ex)
    {
        throw new Exception("DoGetResponse:Error while requesting url " + p_url + ". Error Status " +
            ex.Status + " " + ex.Message);
    }
    catch (Exception exc)
    {
        throw new Exception("DoGetResponse:Error : " + exc.Message);
    }
    return v_response;
}

```

De retour dans notre méthode appelante, je place la réponse ainsi obtenue dans un `StreamReader`, car il me faut encore transformer le format RSS Picasa dans une structure `PicasaNet` que je pourrais désérialiser. La méthode `XslTransform` s'occupe à la fois de la transformation XSL mais également de la désérialisation. Ce point technique est détaillé dans le chapitre 1.3. Si la transformation et la désérialisation se sont bien déroulées, la variable de retour `v_result` est initialisée avec `_albumList` qui comporte la liste des albums. Cette variable membre est initialisée par la méthode `XslTransform`.

```

// I have a response
if (v_response != null)
{
    // Put the XML response into a StreamReader
    v_sr = new StreamReader(v_response.GetResponseStream());

    // Ready to be transformed while the appropriate XSL Style Sheet
    if (XslTransform(v_sr.BaseStream, BuildXsltPath(TransformationType.ALBUMS),
        TransformationType.ALBUMS))
        v_result = _albumList;
}

```

1.2.3 Un album en particulier

La récupération d'un album en particulier fonctionne de la même manière que la récupération d'une liste. Si l'Url est définie par l'application, le numéro de l'album est obligatoire. On utilisera la méthode `GetAlbum`. Cette méthode est surchargée et permet de passer le numéro de l'album via l'objet `PicasaNet.Picasa` ou bien en paramètre de la méthode.

1.3 La transformation des flux

La méthode privée `XslTransform` attend 3 arguments :

- Le `StreamReader` du flux RSS de Google Picasa
- Le chemin de la feuille de style
- Le type de transformation via une énumération

Le chemin de la feuille de style est initialisé à l'instanciation de l'objet `PicasaNet.Picasa`. Le nom de la feuille de style est fixe pour chaque type de transformation et ne doit pas être modifiée.

La transformation effectuée, vient ensuite la désérialisation. En fonction du type (`p_type`) , la méthode procède donc à la désérialisation et initialise la bonne variable membre.

```
// Deserialized the transformed XML
switch (p_type)
{
    case TransformationType.ALBUMS:
        // For the list of Albums
        v_serializer = new XmlSerializer(typeof(ListOfAlbums));
        _albumList = (ListOfAlbums)v_serializer.Deserialize(v_xreader);
        break;
    case TransformationType.ALBUM:
        v_serializer = new XmlSerializer(typeof(PictureAlbum));
        _album = (PictureAlbum)v_serializer.Deserialize(v_xreader);
        break;
}
v_result = true;
```

1.4 L'utilisation du composant

Le projet Web fournit 2 exemples, l'un pour la liste des albums, l'autre pour l'accès à un album en particulier. Pour chacun des cas, deux pages permettent d'afficher le flux XML. :

- Récupération d'un flux local, transformation XSL et affichage direct de la transformation
- Récupération d'un flux distant chez Google Picasa, transformation XSL, désérialisation et resérialisation, pour le fun...

L'utilisation de l'une ou l'autre est possible en ajoutant ou supprimant un argument fictif dans l'url. La requête avec argument interrogera directement Google Picasa.

Pour restituer le flux XML, on se place maintenant sur la solution Web et les pages ListOfAlbums.aspx et Album.aspx

Si vous utilisez des Thèmes dans votre application Web, veuillez bien à ajouter les attributs Theme, StylesheetTheme, EnableTheming à vos pages qui doivent retourner un flux XML ainsi que les attributs ResponseEncoding et ContentType :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ListOfAlbums.aspx.cs" Inherits="ListOfAlbums"
ContentType="text/xml" ResponseEncoding="utf-8" Theme="" StylesheetTheme="" EnableTheming="false" %>
```

1.4.1 Liste des albums

Dans le code behind de la page ListOfAlbums.aspx on dispose du code qui d'une part fait une simple transformation des fichiers locaux (disponibles dans le jeu de test) et d'autre part utilise l'API PicasaNet, toujours instanciée avec le compte Picasa et le chemin des feuilles de styles, ces données étant stockées dans le Web.Config. Dans le second cas, on procède à une sérialisation de la classe ListOfAlbums vers le `Response.OutputStream` .

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.ContentType = "text/xml";
    Response.ContentEncoding = System.Text.Encoding.UTF8;
    string _out = string.Empty;
    PicasaNet.Picasa picasa = new PicasaNet.Picasa(ConfigurationManager.AppSettings["GooglePicasaUser"],
        ConfigurationManager.AppSettings["XslPath"]);

    if (Request.QueryString.Count == 0)
    {
        // First solution
        // Loading 2 files, running transformation
        string _xsl = "http://localhost/WebPicasa/xslt/ListOfAlbums.xslt";
        string _xml = "http://localhost/WebPicasa/xmlesamples/Albums.xml";

        _out = picasa.RunXmlXslStandaloneTransformation(_xml, _xsl);

        Response.Write(_out);
    }
    else
    {
        // Second solution
        // Loading Picasa XML File, Deserialization and Serialization for the fun !
        PicasaNet.ListOfAlbums _liste = picasa.GetAlbumsList();
        XmlSerializer serializer = new XmlSerializer(typeof(PicasaNet.ListOfAlbums));
        serializer.Serialize(Response.OutputStream, _liste);
    }
    picasa.Dispose();
    Response.End();
}
```

```
}
```

1.4.2 Album spécifique

Pour un album spécifique, la technique est quasiment identique à celle du chapitre précédent. On s'attardera plus sur l'utilisation de l'API de PicasaNet, disponible dans le code behind de la page Album.aspx. L'exemple nettoyer montre bien d'une part l'instanciation de l'objet PicasaNet.Picasa avec ces valeurs de base (compte et chemin des feuilles de style) mais surtout dans un second temps l'appel de `GetAlbum` avec l'album à récupérer, dont l'ID est stockée dans le Web.Config.

```
Response.ContentType = "text/xml";
Response.ContentEncoding = System.Text.Encoding.UTF8;
string _out = string.Empty;
PicasaNet.Picasa picasa = new PicasaNet.Picasa(ConfigurationManager.AppSettings["GooglePicasaUser"],
ConfigurationManager.AppSettings["XslPath"]);

// Second solution
// Loading Picasa XML File, Deserialization and Serialization for the fun
PicasaNet.PictureAlbum _album = picasa.GetAlbum(ConfigurationManager.AppSettings["GooglePicasaDemoAlbumID"]);
XmlSerializer serializer = new XmlSerializer(typeof(PicasaNet.PictureAlbum));
serializer.Serialize(Response.OutputStream, _album);

picasa.Dispose();
Response.End();
```

Comme vous pouvez le voir, l'utilisation de l'API de PicasaNet est très simple, l'API de Google Picasa étant elle aussi très basique. Il est bien entendu possible d'enrichir les classes de stockage de PicasaNet, d'une part en ajoutant des accesseurs publics aux classes de structures, mais également en enrichissant les différentes feuilles de styles qui doivent refléter fidèlement la structure des classes.

Comme vous avez pu vous en rendre compte, certains attributs sont ajoutés aux accesseurs publics pour les transformer en attributs XML ou voir renommer des accesseurs côté XML. Ainsi, tous les accesseurs commencent par une majuscule, par contre, ceux transformés en attributs sont mis côté XML en minuscules.

Mais attention dans l'utilisation des données dans le composant AJAX : Dans la classe Album, j'ai un accesseur Id qui est mis en Attribut sous la forme id. Lors de l'utilisation de mon WebService qui va resérialiser cette classe, il mettra bien id dans le résultat de la requête SOAP. Cependant, dans le composant AJAX, lorsque vous voudrez parcourir votre flux XML, il conserve le nom original de l'accesseur, soit Id. Ce petit désagrément m'a valu un bon mal de crane.

```
[Serializable]
public class Album
{
    #region PRIVATE MEMBERS
    private string _id = string.Empty;
    private string _url = string.Empty;
    private string _name = string.Empty;
    private string _description = string.Empty;
    private bool _public = false;
    private int _qty = 0;
    #endregion

    #region PUBLIC MEMBERS
    /// <summary>
    /// Album ID
    /// </summary>
    [XmlAttribute(DataType = "string", AttributeName = "id")]
    public string Id
    {
        get { return _id; }
        set { _id = value; }
    }
}
```

2. Le contrôle Ajax

2.1 Les relations Extender / Javascript

L'un des principaux avantages de ce framework est la capacité à étendre des contrôles serveurs et de les manipuler via du code Javascript. La certaine homogénéité des navigateurs internet et les gros efforts notamment de Microsoft pour se rapprocher des

normes du W3C ont quelque peu réhabilité le Javascript qui a eu sa période de gloire dans les années 2000 et qui depuis tombait en désuétude.

Dès lors l'Extender ne fait que s'appuyer sur des contrôles serveurs, les exploite et permet de gérer le rendu coté client et non plus serveur.

Entre l'Extender et le Javascript, il existe forcément des liens forts et une rigueur de nommage est nécessaire d'une part pour que votre application fonctionne et d'autre part pour éviter de se perdre dans toutes ces variables. Le professionnalisme du développement des composants de l'Ajax ControlToolkit sont je pense une bonne base pour tout développeur qui souhaite construire son propre contrôle.

Dans le projet PicasaNet, examinons déjà le fichier PicasaSlideShowExtender.cs. L'attribut [ClientScriptResource](#) permet de faire le lien avec le JavaScript. Attention au nommage. Il s'agit ici de mettre d'abord le Namespace de l'Assembly, suivi par des points du chemin dans l'arborescence du projet. J'ai volontairement mis un nom différent du namespace à mon dossier comportant le Contrôle Ajax. Mon contrôle Ajax est dans le namespace PicasaNet.PicasaAjaxToolkit. Le chemin de la ressource n'est pas PicasaNet.PicasaAjaxToolkit.PicasaSlideShoxBehavior.js mais plutôt PicasaNet.AjaxToolkit.PicasaSlideShowBehavior.js car mon contrôle est dans le répertoire « AjaxToolkit ».

```
[Designer("PicasaNet.PicasaAjaxToolkit.PicasaSlideShowDesigner, AjaxControlToolkit")]
// AssemblyName [.] Js File path replacing '/' by '.'
[ClientScriptResource("PicasaNet.PicasaAjaxToolkit.PicasaSlideShowBehavior",
"PicasaNet.AjaxToolkit.PicasaSlideShowBehavior.js")]
[TargetControlType(typeof(System.Web.UI.WebControls.Image))]
[RequiredScript(typeof(BlockingScripts))]
[RequiredScript(typeof(CommonToolkitScripts))]
[RequiredScript(typeof(AnimationScripts))]
public class PicasaSlideShowExtender : ExtenderControlBase
{
```

Un ExtenderControlProperty est tout simplement une propriété de votre contrôle Ajax. Une valeur par défaut peut être assignée, ici 3. Le [ClientPropertyName](#) est le nom de la variable coté JavaScript. Mettez le même nom que votre propriété, mais avec la première lettre en minuscules. Les accesseurs Get et Set utiliseront obligatoirement les [GetProperty](#) et [SetProperty](#) fournis par l'Extender.

```
[ExtenderControlProperty]
[DefaultValue(3)]
[ClientPropertyName("thumbnailImages")]
public int ThumbnailImages
{
    get
    {
        return GetProperty("ThumbnailImages", 3);
    }
    set
    {
        SetProperty("ThumbnailImages", value);
    }
}
```

Coté Javascript, notre `thumbnailImages` ce retrouve de la façon suivante, toujours précédé d'un `get_thumbnailImages` et `set_thumbnailImages`. Notez que l'initialisation via le `set_thumbnailImages` se fait grâce au `this.raisePropertyChanged('thumbnailImages')` qui reprend le nom de nom variable coté client. La valeur est stockée coté client dans le `this._thumbnailImages` initialisé au début du script. Ici la variable est précédée du underscore pour une meilleure lisibilité.

```
get_thumbnailImages : function() {
    /// <value type="Number" integer="true" mayBeNull="true"/>
    return this._thumbnailImages;
},

set_thumbnailImages : function(value) {
    if (this._thumbnailImages != value) {
        this._thumbnailImages = value;
        this.raisePropertyChanged('thumbnailImages');
    }
},
```

Si l'on retourne maintenant à notre Extender, vous souhaitez peut être étendre pas seulement des variables mais également des contrôles comme des Panel, Image, LinkButton... Dans l'exemple ci-dessous, on étend un Panel, type comme un WebControl. Vous pouvez ici être plus précis et mettre le type exact de votre contrôle.

```
[ExtenderControlProperty]
[DefaultValue("")]
[IDReferenceProperty(typeof(WebControl))]
[ClientPropertyName("thumbnailPanelID")]
```

```

public string ThumbnailPanelID
{
    get
    {
        return GetPropertyValue("ThumbnailPanelID", "");
    }
    set
    {
        SetPropertyValue("ThumbnailPanelID", value);
    }
}

```

Coté Javascript, le fonctionnement est identique, avec les accesseurs get_ et set_ :

```

get_thumbnailPanelID : function() {
    /// <value type="String" mayBeNull="true">
    /// ID of the label that describes the current slide.
    /// </value>
    return this._thumbnailPanelID;
},

set_thumbnailPanelID : function(value) {
    if (this._thumbnailPanelID != value) {
        this._thumbnailPanelID = value;
        this.raisePropertyChanged('thumbnailPanelID');
    }
}

```

Cependant, il vous faudra une autre variable car `this._thumbnailPanelID` dispose uniquement de l'ID du contrôle. Il est donc nécessaire à l'initialisation du contrôle coté Javascript de définir une variable qui sera vraiment votre objet, ici un `<div>` puisque c'est un Panel :

```

if (this._thumbnailPanelID)
{
    this._pthumbnail = document.getElementById(this._thumbnailPanelID);
}

```

2.2 Les principaux changements par rapport au SlideShow classique de l'AjaxControlToolkit

Les principales évolutions portent sur les points suivants :

- L'ajout d'arguments au Webservice permettent la récupération des Images du WebAlbum. Google Picasa ne permet aujourd'hui de récupérer que les images d'un album mais des évolutions laissent penser que l'on pourrait obtenir des images par rapport au tag. A ce jour, cela ne fonctionne pas dans leur API et de plus je ne vois pas ou dans l'interface graphique l'on peut ajouter des mots clés !
L'appel du Webservice est géré de cette manière. Le 3eme paramètre dispose des arguments du Webservice entre '{' et '}'. Les arguments doivent correspondre à ceux du Webservice.

```

Sys.Net.WebServiceProxy.invoke(
    this._slideShowServicePath,
    this._slideShowServiceMethod,
    false,
    { picasaAction : this._paramAction, picasaParam: this._paramValue },
    Function.createDelegate(this, this._initSlides),
    null,
    null);

```

- La disponibilité d'un visualisateur de vignettes, suivant pas à pas l'image en cours. Le visualisateur est initialisé à la récupération des données XML et mis à jour à chaque changement d'images.
- L'enrichissement du texte de l'image qui permet en fonction d'un paramètre de télécharger ou non l'image originale.
- La mise en œuvre d'un résumé, avec l'icône de l'album, sa description et son titre. Un lien permet un accès direct à l'album.
- L'étendue des images disponible qui va de la vignette jusqu'au fichier original en passant par un aperçu de taille moyenne.

2.3 L'accès aux données XML en Javascript

A chaque chargement de l'image, la fonction `this.updateImage` est exécutée. On transmet alors uniquement l'image à afficher via un `this._xml.Pictures[this._currentIndex]`. `this._xml` représente la racine du flux XML issu du Webservice.

```

_onImageLoaded : function(e) {
    /// <summary>
    /// Image loaded handler.
    /// </summary>
    /// <param name="e" type="Sys.UI.DomEvent" mayBeNull="false"/>
    /// <returns />
    this.updateImage(this._xml.Pictures[this._currentIndex]);
    this.resetButtons();
    this._cacheImages();
},

```

La ou cela devient étrange c'est dans la récupération des autres données. Dans la fonction updateImage, on récupère l'image réduite et son titre. Or dans le flux XML, les données sont des attributs nommés medium et title !! Bien que la sérialisation ait modifiée l'apparence du flux XML dans le Webservice, son utilisation coté client en Javascript n'en tient pas rigueur ! Ce qui est plus étrange c'est que le contrat WSDL ne mentionne nullement les noms des accesseurs publics de mes classes .Net.

```

this._elementImage.src = value.ResizedUrl;
this._elementImage.alt = value.Title;

```

Pire, si vous décidez de renommer la collection de Picture via XmlArray et XmlArrayItem, il devient alors impossible d'accéder aux nœuds coté Javascript.

2.4 Les prérequis

Vous devrez tout d'abord installer les composants ASP.Net AJAX disponibles à sur <http://ajax.asp.net/downloads/default.aspx?tabid=47> et précisés dans le point 1.1 de ce tutoriel.

Par la suite, votre Web.Config doit comporter un certain nombre d'informations pour faire fonctionner les composants AJAX

Dans le nœud <configuration> :

```

<configSections>
  <sectionGroup name="system.web.extensions"
type="System.Web.Configuration.SystemWebExtensionsSectionGroup, System.Web.Extensions, Version=1.0.61025.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35">
    <sectionGroup name="scripting" type="System.Web.Configuration.ScriptingSectionGroup, System.Web.Extensions, Version=1.0.61025.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35">
      <section name="scriptResourceHandler"
type="System.Web.Configuration.ScriptingScriptResourceHandlerSection, System.Web.Extensions, Version=1.0.61025.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" requirePermission="false" allowDefinition="MachineToApplication"/>
      <sectionGroup name="webServices"
type="System.Web.Configuration.ScriptingWebServicesSectionGroup, System.Web.Extensions, Version=1.0.61025.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35">
        <section name="jsonSerialization"
type="System.Web.Configuration.ScriptingJsonSerializationSection, System.Web.Extensions, Version=1.0.61025.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" requirePermission="false" allowDefinition="Everywhere"/>
        <section name="profileService"
type="System.Web.Configuration.ScriptingProfileServiceSection, System.Web.Extensions, Version=1.0.61025.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" requirePermission="false" allowDefinition="MachineToApplication"/>
        <section name="authenticationService"
type="System.Web.Configuration.ScriptingAuthenticationServiceSection, System.Web.Extensions, Version=1.0.61025.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" requirePermission="false" allowDefinition="MachineToApplication"/>
      </sectionGroup>
    </sectionGroup>
  </sectionGroup>
</configSections>

```

Dans le nœud <system.web>. Le premier nœud sert uniquement à ajouter une référence à l'assembly PicasaNet pour utiliser le composant Ajax. Le second sert à utiliser les composants AJAX tels que les UpdatePanel.

```

<pages>
  <controls>
    <add assembly="PicasaNet" namespace="PicasaNet.PicasaAjaxToolkit" tagPrefix="picasa"/>
    <add tagPrefix="asp" namespace="System.Web.UI" assembly="System.Web.Extensions, Version=1.0.61025.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"/>
  </controls>
</pages>

```

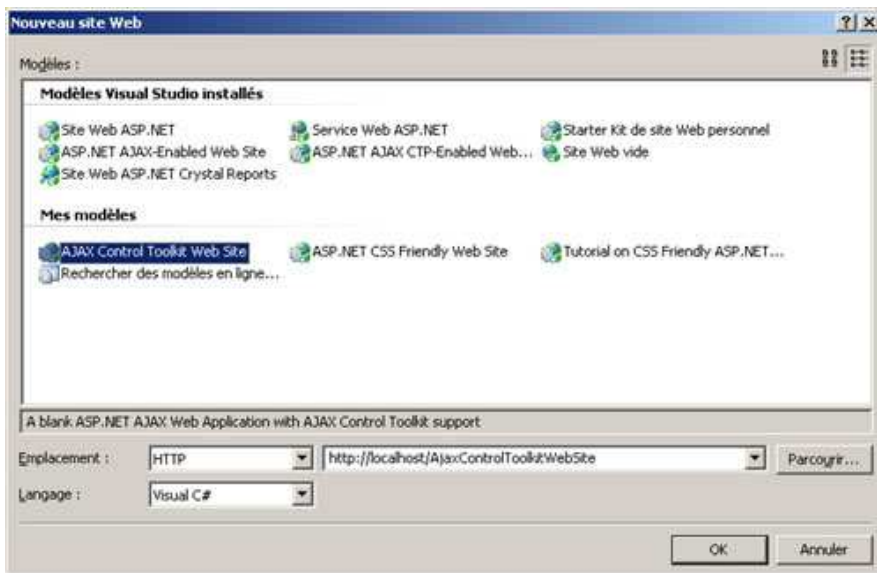
Toujours dans le nœud `<system.web>`, une série de nœud doit être ajoutée dans `<compilation>`, `<httpHandlers>` et `<httpModules>` :

```
<compilation debug="true">
  <assemblies>
    <add assembly="System.Web.Extensions, Version=1.0.61025.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35"/>
    <add assembly="System.Design, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=B03F5F7F11D50A3A"/>
    <add assembly="System.Drawing.Design, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=B03F5F7F11D50A3A"/>
    <add assembly="System.Web.Extensions.Design, Version=1.0.61025.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
    <add assembly="System.Windows.Forms, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=B77A5C561934E089"/> </assemblies>
  </compilation>
  <httpHandlers>
    <remove verb="*" path="*.asmx"/>
    <add verb="*" path="*.asmx" validate="false"
type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions, Version=1.0.61025.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35"/>
    <add verb="*" path="*_AppService.axd" validate="false"
type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions, Version=1.0.61025.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35"/>
    <add verb="GET,HEAD" path="ScriptResource.axd"
type="System.Web.Handlers.ScriptResourceHandler, System.Web.Extensions, Version=1.0.61025.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" validate="false"/>
  </httpHandlers>
  <httpModules>
    <add name="ScriptModule" type="System.Web.Handlers.ScriptModule, System.Web.Extensions,
Version=1.0.61025.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"/>
  </httpModules>
```

Enfin, deux nouveaux nœuds sont ajoutés, `<system.web.extensions>` et `<system.webServer>` :

```
<system.web.extensions>
  <scripting>
    <webServices>
      </webServices>
    </scripting>
  </system.web.extensions>
  <system.webServer>
    <validation validateIntegratedModeConfiguration="false"/>
    <modules>
      <add name="ScriptModule" preCondition="integratedMode"
type="System.Web.Handlers.ScriptModule, System.Web.Extensions, Version=1.0.61025.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35"/>
    </modules>
    <handlers>
      <remove name="WebServiceHandlerFactory-Integrated"/>
      <add name="ScriptHandlerFactory" verb="*" path="*.asmx" preCondition="integratedMode"
type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions, Version=1.0.61025.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35"/>
      <add name="ScriptHandlerFactoryAppServices" verb="*" path="*_AppService.axd"
preCondition="integratedMode" type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions,
Version=1.0.61025.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"/>
      <add name="ScriptResource" preCondition="integratedMode" verb="GET,HEAD"
path="ScriptResource.axd" type="System.Web.Handlers.ScriptResourceHandler, System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35"/>
    </handlers>
  </system.webServer>
```

Autre méthode pour éviter les grosses manipulations de votre Web.Config, lorsque vous créez votre projet Web, l'installation des composants AJAX a rajoutée un modèle de Site Web. Choisissez alors le Modèle `Ajax Control Toolkit Web Site`.

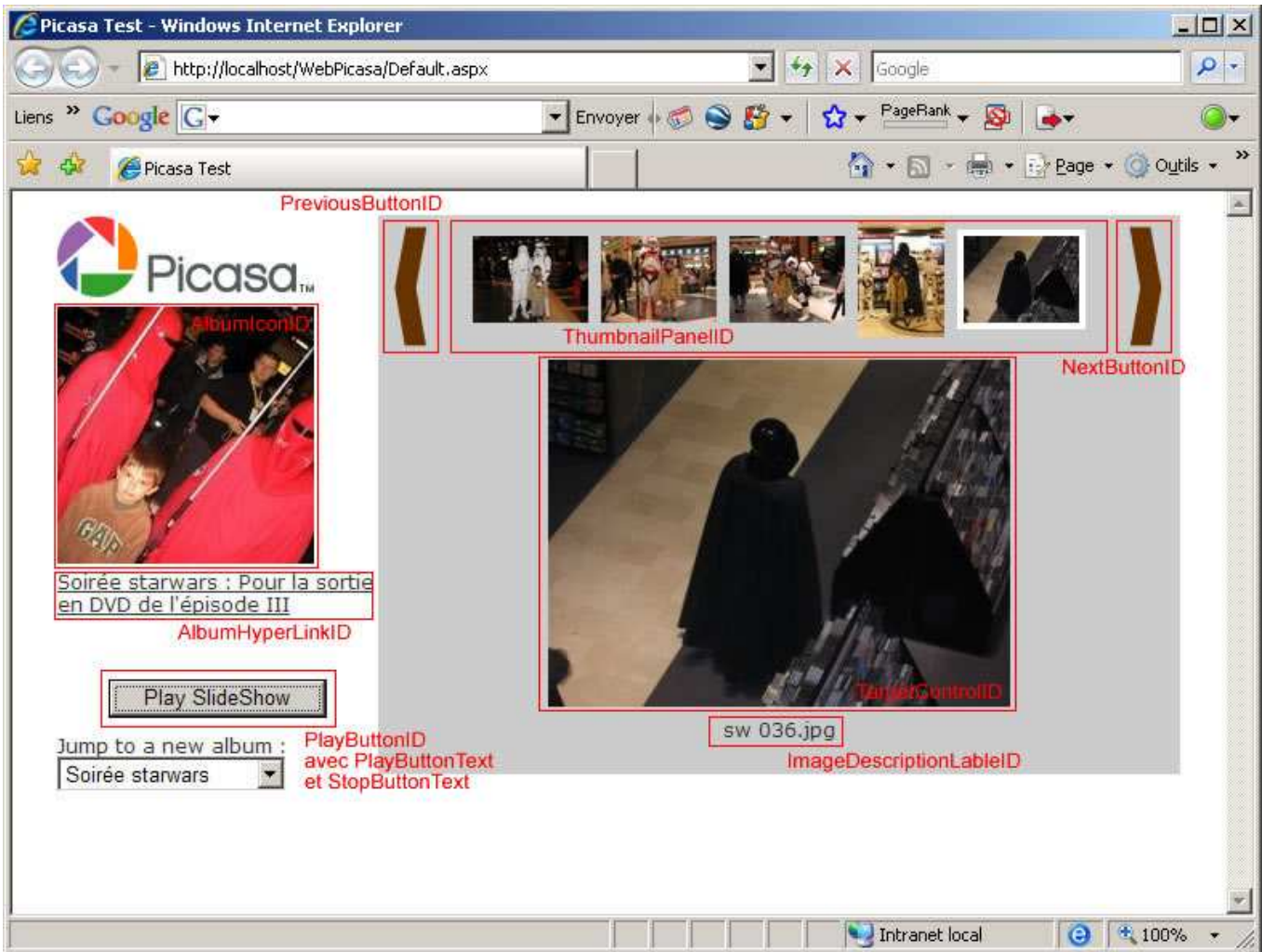


2.5 Les paramètres du composant Ajax PicasaNet

Dans votre page aspx en mode source, commencez à taper <picasa. L'IntelliSense vous proposera les contrôles disponibles et notamment vous trouverez votre PicasaSlideShowExtender.

```
<picasa:PicasaSlideShowExtender runat="server" ID="picasaSlideShow"
  PlayInterval="5000"
  TargetControlID="PictureDestination"
    AlbumIconID="IconImage"
    AlbumHyperLinkID="AlbumHyperLink"
  ThumbnailImages="5"
  CurrentThumbnailPictureCssClass="selectedThumbnail"
  ThumbnailPanelID="thumb"
  SlideShowServicePath="pictures.asmx"
  SlideShowServiceMethod="PicasaRequest"
  ParamAction="ImagesFromAlbum"
  ParamValue="5046867294354988737"
  AutoPlay="true"
  ImageDescriptionLabelID="imageLabel1"
  NextButtonID="nextButton"
  PlayButtonText="Play SlideShow"
  StopButtonText="Stop SlideShow"
  PreviousButtonID="prevButton"
  PlayButtonID="playButton"
  Loop="true"
  AllowImageAccess="false"/>
```

Paramètre	Description
PlayInterval	Durée d'affichage d'une image en millisecondes
TargetControlID	Nom du contrôle Server asp:Image dans lequel l'image s'affichera
AlbumIconID	Nom du contrôle Server asp:Image dans lequel l'icône du WebAlbum s'affichera
AlbumHyperLinkID	Nom du contrôle Server asp:HyperLink dans lequel sera mis le nom et la description du WebAlbum
ThumbnailImages	Nombre maximum de vignettes à afficher
CurrentThumbnailPictureCssClass	Nom de la classe à utiliser pour faire ressortir la vignette actuellement affichée
ThumbnailPanelID	Nom du contrôle Server asp:Panel dans lequel seront mises les vignettes
SlideShowServicePath	Chemin du Webservice
SlideShowServiceMethod	Nom de la WebMethod à utiliser dans le Webservice
ParamAction	Action à réaliser (issue d'une énumération)
ParamValue	Valeur à passer au Webservice, dépendante du ParamAction
AutoPlay	Le déroulé des images doit-il démarrer ?
ImageDescriptionLabelID	Nom du contrôle Server asp:Label pour afficher le titre et la description de l'image
NextButtonID	Nom du contrôle Server de type LinkButton ou Button pour accéder à l'image suivante
PreviousButtonID	Nom du contrôle Server de type LinkButton ou Button pour accéder à l'image précédente
PlayButtonText	Texte à afficher dans le bouton d'arrêt ou de marche du déroulé pour le relancer
StopButtonText	Texte à afficher dans le bouton d'arrêt ou de marche pour arrêter le déroulé
PlayButtonID	Nom du contrôle Server de Type Button destiné à démarrer ou arrêter le déroulé
Loop	Reprend au début lorsqu'il est arrivé à la fin ?
AllowImageAccess	Propose ou non le lien pour télécharger l'image originale ?



2.6 Le Webservice d'interrogation

C'est finalement un Webservice plutôt standard qui retournera des données sous la classe PictureAlbum qui sera sérialisée. Notez les attributs [System.Web.Script.Services.ScriptService] et [System.Web.Script.Services.ScriptMethod] respectivement sur la classe et la WebMethod, nécessaires pour que le Webservice soit interopérable coté client.

```
[System.Web.Script.Services.ScriptService]
[WebService(Namespace = "http://www.laurentgeffroy.com/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class pictures : System.Web.Services.WebService
{
    public pictures()
    {
    }

    [WebMethod(Description="Get Google Picasa Web Album pictures")]
    [System.Web.Script.Services.ScriptMethod]
    public PicasaNet.PictureAlbum PicasaRequest(string picasaAction, string picasaParam)
    {
        // set the return value
        PicasaNet.PictureAlbum v_result = null;

        // New PicasaNet objet with Username and XSLT path for transformation
        PicasaNet.Picasa _img = new PicasaNet.Picasa(ConfigurationManager.AppSettings["GooglePicasaUser"],
        ConfigurationManager.AppSettings["XslPath"]);

        // For testing, you can change the Url of Picasa Feed
        _img.AllowUriChanging = true;
        _img.SetGoogleAlbumUrl = "http://localhost/WebPicasa/xmlsamples/Album.xml";

        // Set the Album name to retrieve
        _img.AlbumName = picasaParam;

        // Depending on action
        switch (picasaAction)
```

```

{
    case "0":
        // Get the list of Pictures from an album
        v_result = _img.GetAlbum();
        break;
    case "1":
        // Not implemented by Google
        // Get the list of Pictures from a tag
        break;
}
// Return the value
return v_result;
}
}

```

2.7 L'utilisation du contrôle

La page Default.aspx vous propose une utilisation du contrôle PicasaNet. Elle s'articule autour d'une liste déroulante qui récupère l'ensemble des WebAlbums d'un compte et qui affecte le premier WebAlbum au contrôle Ajax.

Au changement de valeur dans la liste, le contrôle Ajax est mis à jour avec le nouvel ID d'album. Le tout est contrôlé par un UpdatePanel qui évite le rechargement entier de la page.

```

protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        LoadListAlbums();
    }
}

private void LoadListAlbums()
{
    PicasaNet.Picasa _albums = new PicasaNet.Picasa(ConfigurationManager.AppSettings["GooglePicasaUser"],
    ConfigurationManager.AppSettings["XslPath"]);
    PicasaNet.ListOfAlbums _list = _albums.GetAlbumsList();
    if (_list.Total > 0)
    {
        ListItem _it;
        for (int i = 0; i < _list.Total; i++)
        {
            _it = new ListItem();
            _it.Text = _list.Albums[i].AlbumName;
            _it.Value = _list.Albums[i].Id;
            ddlb_albums.Items.Add(_it);
        }
        // Select the first album
        ddlb_albums.SelectedIndex = 0;
        // Send the value of the first Album to the Picasa Extender
        picasaSlideShow.ParamValue = ddlb_albums.Items[0].Value;
    }
    else
    {
        // No Album !!!!
        ddlb_albums.Visible = false;
        picasaSlideShow.Enabled = false;
    }
}

protected void ddlb_albums_SelectedIndexChanged(object sender, EventArgs e)
{
    picasaSlideShow.ParamValue = ddlb_albums.SelectedValue;
}

```

Et voila le résultat :



3. En savoir plus

Rendez vous sur <http://www.laurentgeffroy.com> pour suivre l'évolution de ce composant. Soyez libre de l'utiliser, de le modifier. Si vous apportez une modification majeure, faites le moi savoir, je pourrai peut être l'intégrer dans ma version si vous êtes d'accord.

Vous pouvez me contacter par email à [lgeffroy\[a\]gmail.com](mailto:lgeffroy[a]gmail.com)

A voir également la possibilité de sécuriser l'appel de votre Webservice. Je vous recommande un très bon article de Cyril, un fou d'Ajax :

http://www.aspfr.com/codes/MODULE-AUTHENTIFICATION-POUR-WEBSERVICES-AJAX_41816.aspx

A lire également la documentation du Framework Ajax.

<http://ajax.asp.net/docs/>