La technologie .NET

ASP.NET et les Web Services

Samia Bouzefrane

Maître de Conférences

CEDRIC –CNAM

samia.bouzefrane@cnam.fr
http://cedric.cnam.fr/~bouzefra

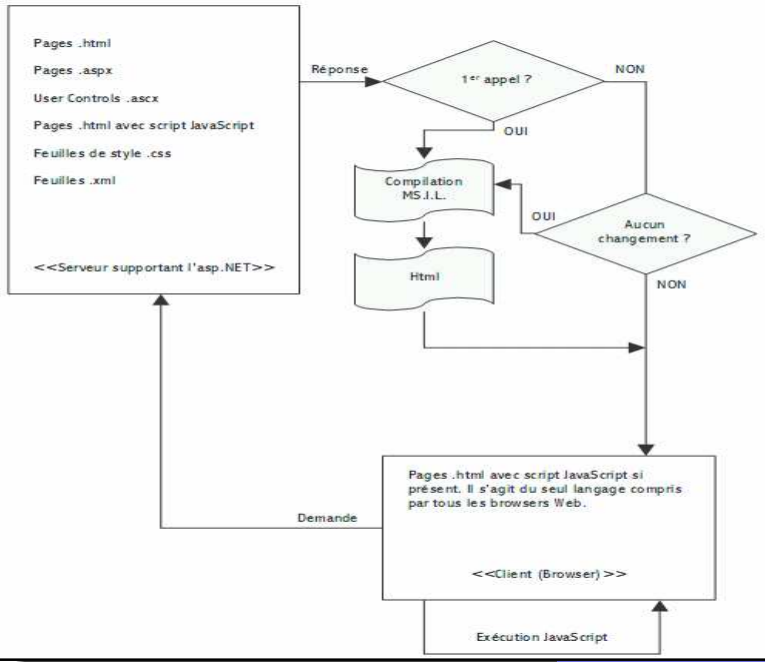
ASP.NET

ASP.NET

- ✓**Version 2.0**
- ✓**Successeur de ASP**
- ✓**Permet l'écriture de pages Web dynamiques (Web Form)**

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Architecture



RIC (CNAM) -

Structure d'une page ASP.NET

✓Une page ASP.NET possède une extension .aspx

✓Une partie fait apparaître les scripts (C#, Javascript).

✓Une partie : code html

✓Une partie <asp :…> et <% … %> : balise d'ouverture pour un WebForm

Exemple de Page ASP.NET

```
<html>
<head>
<script language= "c#" runat="server">
  type  fct (type i) { … }
</script>
<script language= "javascript">
  function f ;
</script>
</head>
<body>
  <table>
    …
  </table>
  <asp :Label [Propriétés]></asp :Label>
  <% =fct(5) %>
</body>
</html>
```

Script de traitement

- Tout script (écrit en C# par exemple) pour ASP.NET doit se trouver entre les balises:

```
<script language= "c#" runat= "server">
    …
</script>
```

- Le tout doit se trouver entre la balise html `<head>` et `</head>`.

`runat="server"` : script exécuté côté serveur.

- Dans une page, un seul langage .NET est accepté.

Exemples

```
<head>
<script language= "c#" runat= "server">
    string Demain()
    {
      DateTime Jour=DateTime.Now ;
      Jour=Jour.AddDays(1) ;
      return Jour.ToString() ;
    }
</script>
</head>
<body>
<form runat="server">
…
Date du jour :
<% =DateTime.Now.ToString() %>
<br>
Date de demain : <% =Demain() %>
…
</form>
</body>
```

```
<head>
</head>
<body>
<form runat="server">
  …
Date du jour : <% =
DateTime.Now.ToString() %>
<br>
Date de demain :
<%
DateTime Jour=DateTime.Now ;
Jour=Jour.AddDays(1) ;
Response.Write(Jour.ToString()) ;
 %>
…
</form>
</body>
```

Les composants/ contrôles

- Label
- Button
- TextBox
- CheckBox
- RadioButtonList
- ListBox
- DropDownList
- Image
- ImageButton
- Hyperlink
- LinkButton
- Panel
- Calendar

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Propriétés communes

- `Id` : Tout composant a un nom Id.
- `OnClick` : est l'événement correspondant au clic sur un composant, par exemple un bouton.
- `Runat` : permet de spécifier où le traitement du composant doit être fait.
- `Text` : est la valeur affichée par le composant.
- `OnTextChanged` : est l'événement lorsque le texte change.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Propriétés des composants

	Id	OnClick	Runat	Text	OnTextChanged
Label	√		√	√	
Button	√	√	√	√	
TextBox	√		√	√	√
CheckBox	√		√		√
RadioButtonList	√		√		√
ListBox	√		√		
DropDownList	√		√		
Image	√		√		
ImageButton	√	√	√		
Hyperlink	√		√	√	
LinkButton	√	√	√	√	
Panel	√		√		
Calendar	√		√		

Label (zone d'affichage)

```
 <head>
</head>
<body>
  <form runat="server">
    <asp:Label  id="IdLabel" text="<b>Mon premier composant
                                   asp" runat="server"/>
  </form>
</body>
```

Button (bouton)

```
<head>
    <script language="c#" runat="server">
      void Envoi(Object Sender, EventArgs E)
      {
        IdLabel.Text="Notre deuxième composant asp fonctionne :)";
      }
    </script>
</head>
<body>
    <form runat="server">
      <asp:Label  id="IdLabel"  text="Mon premier composant asp"
runat="server"/>
      <br>
      <asp:Button  id="IdBouton"  text="Envoyer" OnClick="Envoi"
runat="server"/>
    </form>
</body>
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

TextBox (zone d'édition) et Propriétés

Une zone d'édition peut avoir les propriétés suivantes :

✓**Columns** : il s'agit de la largeur du contrôle exprimée en nombre de caractères.

✓ **MaxLength** : est le nombre maximal de caractères qui seront admis. Pour avoir un effet, il faut que **TextMode** corresponde à **SingleLine** ou **Password**.

✓ **Rows** : est le nombre de lignes visibles. Il faut dans ce cas que **TextMode** soit **MultiLine**. Défilement vertical si saisie de plus de lignes

✓ **TextMode** : Les trois modes possibles sont **SingleLine**, **MultiLine** ou **Password**.

✓ **Wrap** : stipule si oui ou non il y a passage automatique à la ligne dans une zone d'édition de plusieurs lignes.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

TextBox : Exemple

```
<head>
    <script language="c#" runat="server">
      void Envoi(Object Sender, EventArgs E)
      {
        IdLabel.Text=IdSaisie.Text;
      }
    </script>
</head>
<body>
    <form runat="server">
      <asp:Label  id="IdLabel"  text="Mon premier composant asp"
                    runat="server"/>
      <br>
      <asp:Button  id="IdBouton"  text="Envoyer"  OnClick="Envoi"
                  runat="server"/>
      <br>
      <asp:TextBox id="IdSaisie" runat="server"/>
    </form>
</body>
```


CheckBox (case à cocher)

- Une case à cocher permet de faire un choix de type oui/non ou vrai/faux.
- Propriétés :
  - ✓ **`Checked`** : est à true si la case est cochée et à false dans le cas contraire.
  - ✓ **`TextAlign`** : est la position du libellé par rapport à la case. Les valeurs possibles sont **`Right`** ou **`Left`**.

```
<asp:CheckBox id="IdCase" runat="server" text="S'incrire
à la mailing-list"/>
```

RadioButtonList (boutons radio)

- Composant utilisé lorsque l'on a plusieurs choix mais avec une seule valeur à choisir.
- Il y a deux syntaxes :

```
<asp:RadioButton  id="Madame"
runat="server"  text="Madame"
GroupName="rblMmeMlleMr"/>

<asp:RadioButton  id="Mademoiselle"
runat="server"  text="Mademoiselle"
GroupName="rblMmeMlleMr"/>

<asp:RadioButton  id="Monsieur"
runat="server"  text="Monsieur"
GroupName="rblMmeMlleMr"/>
```

```
<asp:RadioButtonList id="rblMmeMlleMr"
runat="server">

<asp:ListItem value="Madame" />
<asp:ListItem value="Mademoiselle" />
<asp:ListItem value="Monsieur" />
</asp:RadioButtonList>
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

DropDownList (boîte combo)

- La boîte combo possède les mêmes propriétés qu'une ListBox.
- Une boîte pour choisir le mois de naissance :

```
<asp:DropDownList  id="IdMoisNaissance"  runat="server" >
<asp:ListItem text="Janvier" />
<asp:ListItem text="Fevrier" />
     …
<asp:ListItem text="Octobre" />
<asp:ListItem text="Novembre" />
<asp:ListItem text="Decembre" />
</asp:DropDownList>
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Image

✓ **ImageUrl** qui est l'adresse de l'image (relative ou absolue).
✓ **AlternateText** est le texte à afficher si l'image ne peut l'être.
✓ **ImageAlign** est l'alignement de l'image par rapport au contour de son composant. On retrouve comme valeurs acceptées **AbsBottom**, **AbsMiddle**, **Bottom**, **Left**, **Middle**, **Right**, **Top**, etc.

```
<asp:Image ImageUrl="image.gif" runat="server" AlternateText="Made
with ASP.NET"/>
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

ImageButton

➢ A ce composant est associé un événement lors du clic.

Seule propriété : **ImageUrl**.

```
void ibClick(Object Sender, ImageClickEventArgs E)
    {
      IdImageButton.Text="Un bouton Image!";
    }
  ....
<asp:ImageButton  ImageUrl="image.gif"  runat="server"
AlternateText="Une  info" OnClick="ibClick"/>
<asp:Label id="IdImageButton" runat="server" text=""/>
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Hyperlink (hyper-lien)

- Les caractéristiques sont simples :
  - ✓ `ImageUrl` est l'image qui contient le lien (si l'on désire utiliser une image comme hyperlien). Si cette propriété est utilisée, il ne faut pas utiliser la propriété `Text`.
  - ✓ `NavigateUrl` est l'adresse destination.

```
<asp:Hyperlink  Text="Rubrique Enseignement"
NavigateUrl="http://enseignement.com" runat="server"/>
```


Panel (volet)

- Le panneau contient d'autres composants ASP.NET.
- Sert à afficher ou cacher un groupe de composants.

Exemple: désactiver des composants à l'aide de l'ImageButton. S'ils sont déjà désactivés, ils seront réactivés.

- Propriétés :
  - ✓ `BackImageUrl` qui est l'image de fond.
  - ✓ `HorizontalAlign` qui est l'alignement du contenu du volet (`Center`, `Justify`, `Left`, `NotSet` ou `Right`).

```
void ibClick(Object Sender, ImageClickEventArgs E)
    {
      pTout.Enabled=!(pTout.Enabled);
    }
```

avec :

```
 <asp:Panel id="pTout" runat="server">
…
    </asp:Panel>
```

Calendar (calendrier)

- Calendrier complet prêt à l'emploi.
- Propriétés principales :
  - ✓`VisibleDate` est la date visible (de type `DateTime`).

```
<head>
</head>
<body>
<form runat="server">
  <asp:Calendar VisibleDate="2008/05/11" runat="server"/>
</form>
</body>
```

<!Affiche le calendrier du mois de mai 2008>

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Le passage d'arguments

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Exemple de la saisie du login/password

Mapage.aspx

```
<%@ Page Language="C#" %>
<script runat="server">
  …
</script>
<html>
<head>
</head>
<body>
    <form runat="server">
        Login:
        <asp:TextBox id="TbLogin" Runat="server"/>
        <br />
        Passwd:
        <asp:TextBox id="TbPasswd" TextMode="Password" Runat="server"/>
        <br />
        <br />
        <asp:Button  id="BuLogin"  onclick="Check"  Runat="server"
Text="Login!"/>
    </form>
</body>
</html>
```


Passage d'arguments à une url

http://monsite.com/MesInfos.aspx?param1=val1&param2=val2.

Cette syntaxe est utilisée dans le traitement du clic sur le bouton de login. **MesInfos.aspx** est exécutée avec les paramètres fournis:

```
<%@ Page Language="C#" %>
<script runat="server">
public void Check(object sender, EventArgs e)
{ string[,] tab= {
        {"Toto", "aspnet"},
        {"Test", "Test"}
        };
int i=0;
while (i<tab.Length/2) // Nb total d'elements / nombre de dimensions...
{
        if (TbLogin.Text == tab[i,0].ToString() &&
                TbPasswd.Text == tab[i,1].ToString())
        Response.Redirect("MesInfos.aspx?L=" + TbLogin.Text
        + "&P=" + TbPasswd.Text);
        i++; }
Response.Redirect("MesInfos.aspx");
}
</script>
```

Passage d'arguments à une url

MesInfos.aspx

```
<html>
<head>
<%@ Page Language="C#" %>
<script runat="server">
void Page_Load(object sender, EventArgs e)
{
La.Text = "Nous sommes passés par le Page_Load() pour vérification";
if (Request.Params["L"]!=null)
    La.Text += "<br>Login: " + Request.Params["L"].ToString();
if (Request.Params["P"]!=null)
    La.Text += "<br>Pass: " + Request.Params["P"].ToString();
   }
</script>
</head>
<body>
<form runat="server"> Page MesInfos<br><br>
<asp:Label Runat=server ID=La></asp:Label>
</form>
</body>
</html>
```


Passage d'arguments à une url

`Request.Params()` : permet de vérifier que les variables existent bien et qu'elles sont initialisées.

Inconvénient : faire passer à chaque page le login et le mot de passe.

Une solution: les variables session.

Les variables session

- Les variables session sont stockées sur le serveur.
- Si trop de variables alors faibles performances du serveur.
- Variables accessibles du début jusqu'à la fin de la session du navigateur (ferméture de la session ou inactivité du navigateur pendant un timeout).
- Une session est un espace mémoire/disque sur le serveur associé à un et un seul utilisateur/navigateur.
- Une session possède un identifiant unique stocké au niveau du navigateur
- Les variables session sont stokées sur le serveur et sont accesssibles par toutes les pages Web associées à cette session.

Exemple

```
public void Check(object sender, EventArgs e)
{
string[,] tab= {
    {"Toto", "aspnet"},
    {"Test", "Test"}
};
int i=0;
while (i<tab.Length/2) // Nb total d'elements / nombre de dimensions...
{
    if (TbLogin.Text==tab[i,0].ToString() &&
    TbPasswd.Text==tab[i,1].ToString())
    {
        Session["Username"] = TbLogin.Text;
        break;
    }
    i++;
    }
    Response.Redirect("MesInfos2.aspx");
}
```

Exemple

Et pour la page d'information :

```
void Page_Load(object sender, EventArgs e)
{
La.Text = "Nous sommes passés par le Page_Load() pour
vérification";
if (Session["Username"]!=null)
La.Text += "<br>Login: " + Session["Username"].ToString();
else
La.Text + = "<br>Personne n'est connecté";
}
```


Les variables d'application

- Les variables d'application sont accessibles par toutes les sessions
- Exemple courant : comptabiliser le nombre d"utilisateurs connectés

Il faut ajouter dans la fonction Check() :

```
if (Application["NbUsers"]!=null)
                Application["NbUsers"]=

        Convert.ToInt32(Application["NbUsers"])+1;
else
                Application["NbUsers"] = 1;
```

Les variables d'application

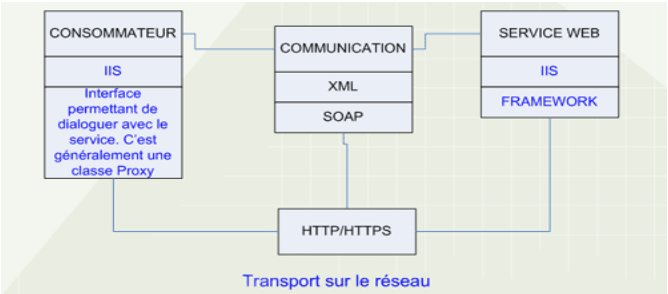
Dans la page d'informations, il faut récupérer cette valeur à l'aide de la syntaxe utilisée pour les variables de session.

```
if (Application["NbUsers"]!=null)
                La.Text += "<br>NbUsers: " +


        Application["NbUsers"].ToString();
else
                La.Text += "<br>NbUsers: 0!";
```


Les Web Services

Architecture des Services Web



Les protocoles HTTP et HTTPS servent à transporter les données sur le réseau entre le consommateur et le service web. Les données sont sous le format XML et/ou SOAP. Le serveur où se trouve le service web devra être capable d'exécuter le code de celui-ci et nécessitera IIS & le framework. La machine du consommateur devra quant à elle disposer de IIS.

Comment ajouter une référence Web dans son projet ?

Sous Visual Studio Express 2008

- Il faut connaître l'adresse Web du WebService
(par ex. le Service Web *http://ws.soatrader.com/delimiterbob.com/0.1/YellowPages?WSDL* du site http://www.xmethods.com/ qui contient des WebServices qu'on peut tester en ligne).

- Une fois l'adresse du WebService trouvée, choisir le menu **"References"** et choisir "**Ajouter une référence Web**".

- On arrive sur l'écran qui permet la saisie de l'adresse du WebService,
Il faut cliquer sur le bouton "**Aller à**".

- Une barre de progression qui s'affiche permet de récupérer la liste des méthodes disponibles.

- Il faut juste cliquer sur le bouton "**Ajouter la référence**" pour pouvoir ajouter ce WebService en tant que référence à ce projet.

- Si la référence est correctement ajoutée, elle devient visible dans l'explorateur de solutions.

Les Services Web

- Les Services Web :
  - ✓ sont des fonctions stockées sur un serveur Web et mises à la disposition de tout le monde.
  - ✓ ressemblent à des classes classiques.
  - ✓ sont basés sur des protocoles tels que XML et SOAP transportés par les protocoles http ou https.
- Le framework .NET génère lui-même les en-têtes SOAP et XML.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Le contrat WSDL

- Chaque service web est identifié par un contrat WSDL.
- Ce contrat représente la structure SOAP/XML du Service Web.
- Il est nécessaire de faire référence au contrat WSDL du Service Web si on désire le consommer.
- Le contrat WSDL est visible
  - * soit en cliquant sur le lien "Service description"
  - * soit en ajoutant "?WSDL" à la fin de l'url où se trouve le Service Web.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Consommation d'un Service Web

- Pour consommer un Service Web :
  - il faut connaître son emplacement
  - il faut obtenir son contrat WSDL nécessaire à la création d'une classe proxy à utiliser par le consommateur.

- Pour générer la classe proxy, il suffit d'utiliser la commande wsdl :

```
wsdl url_du_service_web?WSDL /out:lenomdelaclasseproxy.cs
```

- On peut ajouter ce fichier source à notre projet où en faire une dll à rajouter en tant que référence dans le projet.

- Pour créer cette DLL, on peut utiliser l'utilitaire :

```
csc /target:library lenomdelaclasseproxy.cs
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Exemple de ServiceWeb

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;

public class Exemple : System.Web.Services.WebService {

[WebMethod] /*Cet attribut spécifie que la méthode est
utilisable par une application cliente d'un service Web*/
    public int Additionne(int a, int b) {
        return a + b;
        }
 }
```

L'attribut [WebMethod] permet de rendre une méthode accessible aux consommateurs des services web. Le résultat du ServiceWeb est retourné au format XML. Ajouter le fichier source `petitExemple.cs` ou la dll `petitExemple.dll` au projet.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Génération du proxy

➢Pour générer la classe proxy, il suffit d'utiliser la commande wsdl :

```
wsdl url_du_service_web?WSDL /out:lenomdelaclasseproxy.cs
```

Dans MicroSoft Visual Studio 8, lancer l'utilitaire wsdl qui se trouve dans :

```
$cd <Rep>\SDK\v2.0\Bin si <Rep> est le répertoire d'installation
$wsdl http://localhost:1178/WebSite1/Service.asmx?wsdl /out:MonService.c
écriture du fichier <Rep>\SDK\v2.0\Bin\MonService.cs
```

On peut rajouter ce fichier source à notre projet où en faire une dll à rajouter en tant que référence dans le projet.

➢Pour créer cette DLL, on peut utiliser l'utilitaire s'il est disponible:

```
csc /target:library lenomdelaclasseproxy.cs
```


Exemple de page ASP.NET

```
<@ Page="c#" Codebehind="Program.aspx.cs"
AutoEventWireup="true"
Inherits="Program.Consommateur" >
<HTML>
 <body MS="GridLayout">
<form id="Form1" method="post" runat="server">
<asp:Label id="rep_webservice" runat="server">Label</asp:Label>
</form>
</body>
</HTML>
```

Cette page ASP.NET contient un contrôle label qui recevra la réponse du Service Web.

Le programme Program.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

public class Consommateur : System.Web.UI.Page {
protected System.Web.UI.WebControls.Label rep_webservice;

private void Page_Load(object sender, System.EventArgs e) {
        /*On instancie le service web*/
       Exemple petitExemple = new Exemple();
       /*On appelle la méthode "additionne" qui retournera "15"*/
       rep_webservice.Text=petitExemple.Additionne(5,10).ToString();
   }
}
```


Les attributs de WebMethod

- **[WebMethod (EnableSession=true)] : permet l'usage de sessions**

  Par défaut, l'attribut *EnableSession* est à false,
  Le consommateur doit accepter les cookies.

- **[WebMethod (BufferResponse=false)] : égal à true par défaut**

  Lorsque l'on définit cet attribut à false, le service web envoie sa réponse au fur et à mesure qu'il la sérialise, pas de stockage préalable dans un tampon. Si cet attribut est à false, les en-têtes SOAP sont désactivées et de ce fait, certains objets complexes ne peuvent plus être véhiculés vers le consommateur/service.

Les attributs de WebMethod

➢**[WebMethod (CacheDuration=nombre de secondes)]**
Par défaut, aucune rétention de données en cache n'est effectuée.
On peut changer ce comportement en disant au service web de garder les réponses en cache pendant un certain temps.

➢**[WebMethod (Description="une description")]**

➢**[WebMethod (MessageName="alias nom de méthode")]**
Si cet attribut n'existe pas c'est le nom de la méthode elle-même qui sera utilisé.

Les attributs de WebMethod

➢**[WebMethod (TransactionOption=TransactionOption.[Disabled ou Required ou Supported ou NotSupported ou RequiresNew])]**

- ✓*Disabled*: les transactions sont désactivées
- ✓*Required*: les transactions sont obligatoires, l'appel à la méthode du service doit être au sein d'une transaction
- ✓*Supported*: les transactions sont supportées
- ✓*NotSupported*: les transactions ne sont pas supportées
- ✓*RequiresNew*: chaque méthode doit démarrer une transaction

Les appels asynchrones

- Par défaut les appels sont synchrones
- Une classe proxy est créée pour un service web à l'aide de l'outil wsdl pour travailler avec les méthodes synchrones et asynchrones.
- Les méthodes synchrones portent le même nom que les méthodes du service Web.
- Les méthodes asynchrones sont par contre préfixées :

Begin<nom de la méthode du service> démarre la communication asynchrone

End<nom de la méthode du service> reçoit la réponse de la méthode en fin de commumnication

Exemple

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
public class Consommateur : System.Web.UI.Page {
        protected System.Web.UI.WebControls.Label reponse_webservice;
        private Exemple petitExemple = new Exemple();
        private void Page_Load(object sender, System.EventArgs e) {
            //Déclaration d'un objet résultat d'appel asynchrone et
            démarrage asynchrone de l'appel*/
        IAsyncResult Reponse = petitExemple.BeginAdditionne(10,15,null,
            null); Response.Write("Le traitement continue");
/* Ici on bloque l'exécution du code jusqu'à ce que la réponse de l'appel
asynchrone ait été obtenue */
            Reponse.AsyncWaitHandle.WaitOne();
            /*Ici on récupère la réponse --> 25*/
            rep_webservice.Text=petitExemple.EndAdditionne(
                    Reponse).ToString(); } } }
```

Bibliographie

- Pratique de .NET2 et C#2, Patrick Smacchia, Ed. Oreilly, 2005.
- //ditch.developpez.com/ par Didier Danse
- msdn.microsoft.com
- Apprentissage du Langage C#, par Serge Tahé, ISTIA, Université d'Angers
- « Créer et Consommer un service web avec .NET », Par Stéphane Eyskens.
- Visual Studio 2008 (versions Express gratuites) http://www.microsoft.com/express/

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -