



<Lab Visual Studio .NET/>

Service Web XML

Sommaire

1	CONSOMMER UN SERVICE WEB	3
1.1	PREMIER CLIENT SERVICE WEB (APPLICATION WINDOWS)	3
1.2	DEUXIEME CLIENT SERVICE WEB (SITE WEB)	7
2	FOURNIR UN SERVICE WEB	11
2.1	CREER UN SERVICE WEB	11
3	TYPES DE DONNEES.....	21
3.1	UTILISATION DE TYPES COMPLEXES	21
3.2	MODIFICATION DE LA SERIALISATION XML	23
3.3	UTILISATION D'UN DOCUMENT XML	24
3.4	UTILISATION DES DATASETS	25
3.5	TRANSMETTRE DES DONNEES BINAIRES	34
4	DISPONIBILITE	36
4.1	CLIENT : APPEL ASYNCHRONE	36
4.2	SERVEUR : MISE EN CACHE	37
4.3	CLIENT : URL D'ACCES EN CONFIG	38
4.4	CLIENT : FAIL-OVER	39
4.5	GESTION D'UNE SESSION	39
5	SECURISATION D'UN SERVICE WEB.....	40
5.1	CRYPTER LA COMMUNICATION PAR SSL	40
5.2	AUTHENTIFICATION	40
5.2.1	<i>Authentication Basic</i>	41
5.2.2	<i>Authentication Windows</i>	43
5.3	DESACTIVATION DES PAGES HTML AUTO-GENERES	44
6	AUTRES SUJETS.....	45
6.1	RECUPERATION DE L'EXISTANT	45
6.2	UTILISATION DEPUIS OFFICEXP	45
6.3	CONFIGURATION DES VUES HTML.....	45

1 Consommer un Service Web

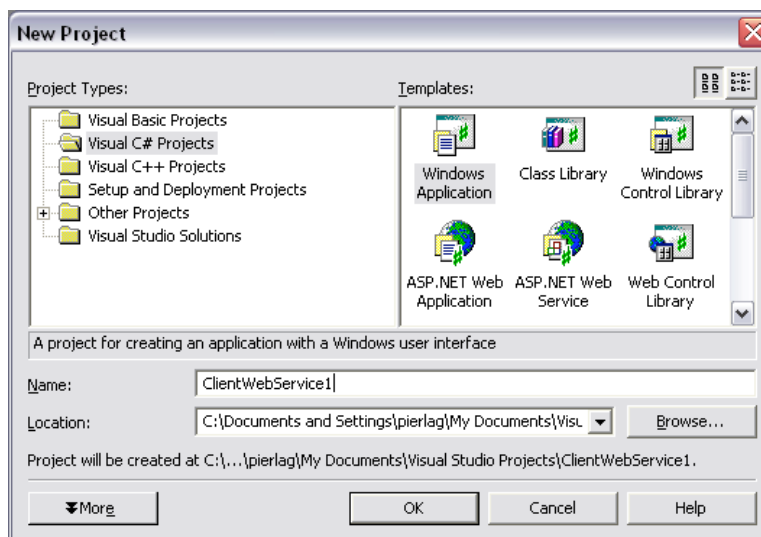
Dans ces exercices, vous allez apprendre à :

- Créer plusieurs clients service Web
- Interroger un Service Web via une application Windows
- Interroger un Service Web via une Site Web

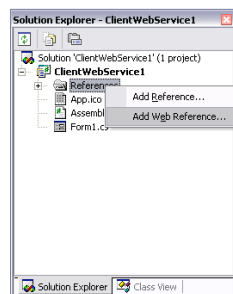
1.1 Premier Client Service Web (Application Windows)

Déroulement de l'exercice:

1. Lancer Visual Studio .NET
 - Démarrer > Programmes > Microsoft Visual Studio .NET > Microsoft Visual Studio
2. Créer une application WinForm
 - Fichier > Nouveau > Projet
 - Projet Type : Choisissez C#
 - Templates : Choisissez Windows Application
 - Nom du projet : ClientWebService1
 - Cliquer sur Ok

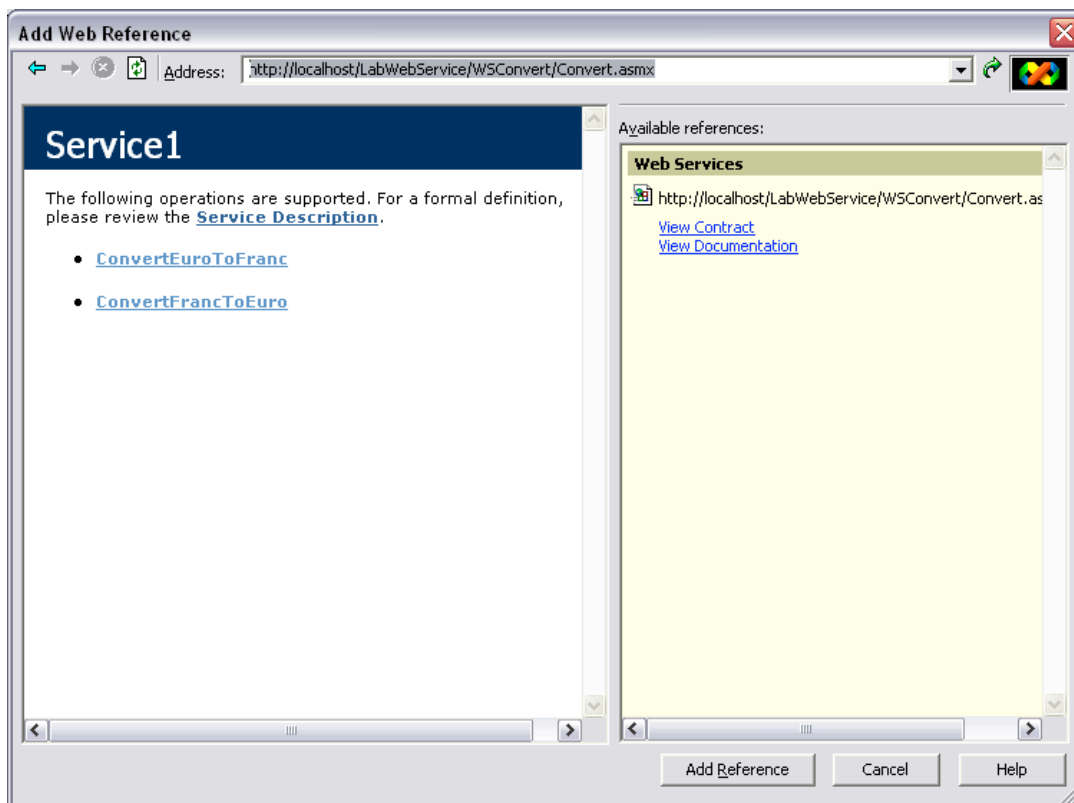


3. Regardez ce qui a été créé
 - Une form Form1.cs
4. Faire référence au WebService déjà disponible sur la machine (Convert Euro <-> Franc)
 - Ajouter la référence au WebService en cliquant sur le bouton droit sur référence et en sélectionnant « Add Web Référence »

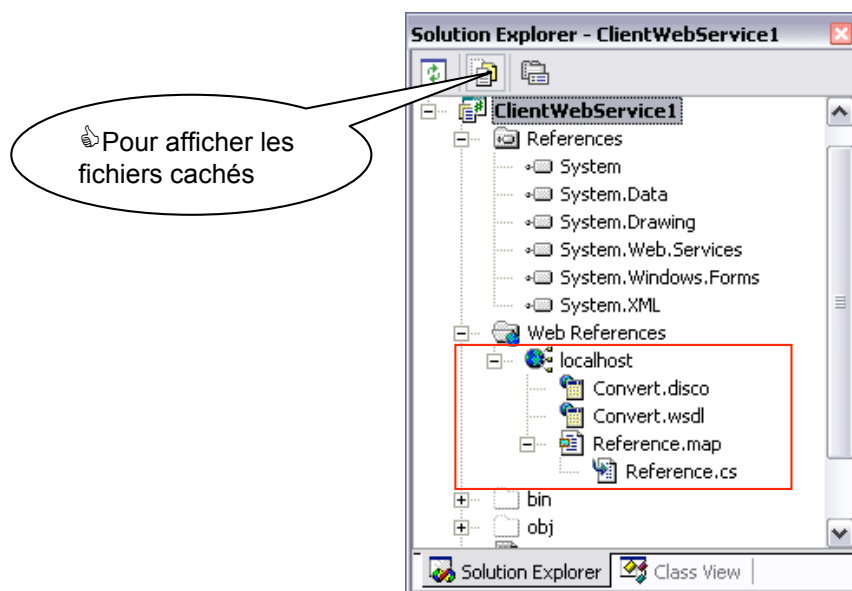


- Tapez l'URL dans l'adresse Web Reference :

http://localhost/LabWebService/WSConvert/Convert.asmx



- On a deux Services Web de disponible sur cette URL
- Cliquez sur Add Reference
- On obtient donc ceci dans les références projet :



Chapitre Créer un Webservice



Vous pouvez voir les sources de description du service à l'adresse suivante :

http://localhost/LabWebService/WSConvert/Convert.asmx?WSDL

L'ajout de la référence crée donc 4 fichiers : Reference.map et Convert.disco (fichiers d'inspections), Reference.cs (proxy local), Convert.wsdl (Description de l'interface du service)

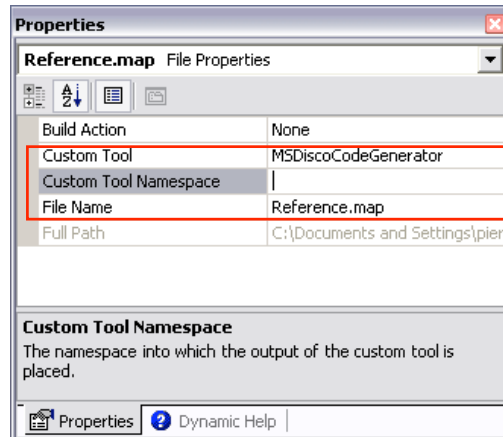


Si le Service Web change il suffit de mettre à jour la référence et les fichiers sont régénérés.

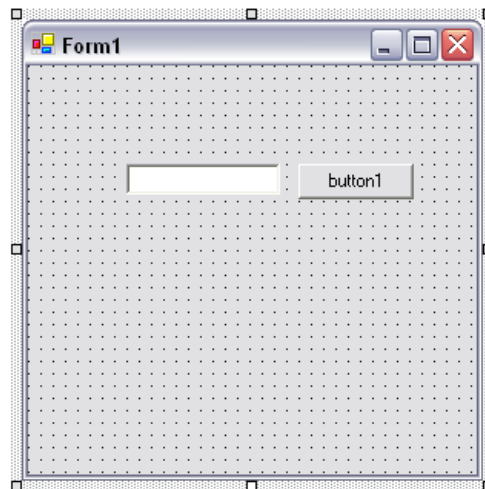
Erreur ! Nom de propriété de document inconnu.



Localhost fait office de namespace pour le proxy. Il est initialisé par le fichier WSDL mais peut être modifié par le panneau de Propriétés. On utilisera localhost.ServiceConvert pour appeler le Service Web.



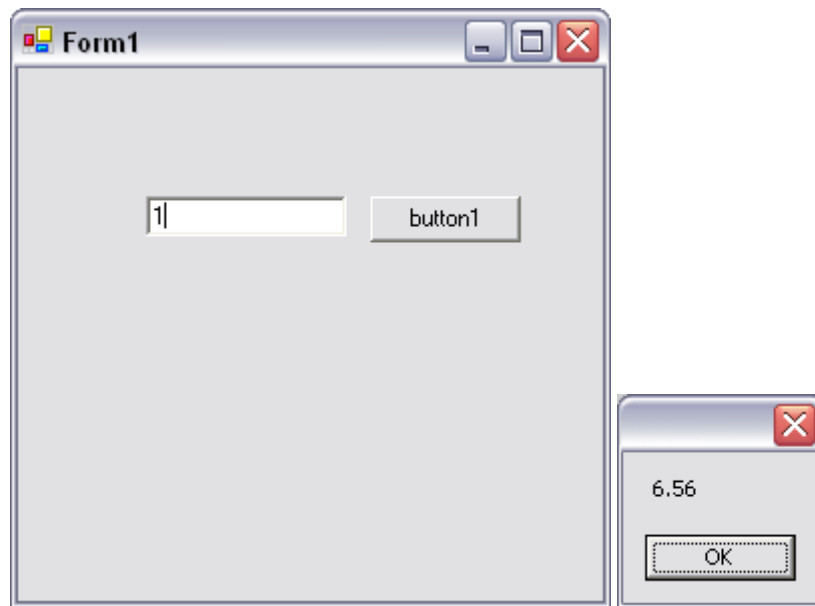
5. Placez un bouton et une zone d'édition sur la form pour appeler le service Web en utilisant la ToolBok par Glisser/Déplacer



6. Double cliquez sur le bouton pour déclenchez une action au moment du click.
 - Tapez ce code C#

```
private void button1_Click(object sender, System.EventArgs e)
{
    //Déclaration de la variable sur l'objet Proxy ServiceConvert
    localhost.ServiceConvert MonService;
    //Intanciation de la variable
    MonService = new localhost.ServiceConvert();
    //Conversion de la zone de saisie en double
    double Euro = double.Parse(textBox1.Text);
    //Appel du ServiceWeb
    double Franc = MonService.ConvertEuroToFranc(Euro);
    //Affichage du résultat
    MessageBox.Show(Franc.ToString());
}
```

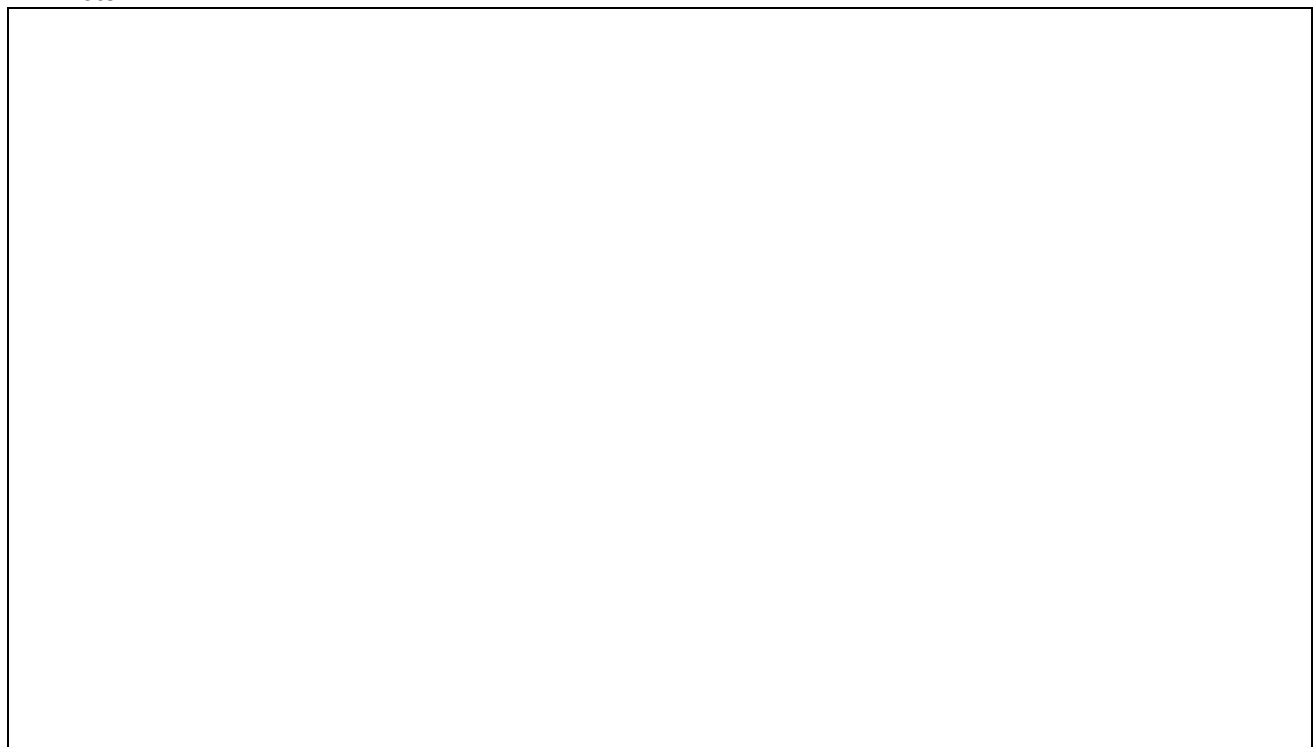
7. Lancez l'application : 



Pour aller plus loin

1. Appeler le deuxième Service Web « ConvertFrancToEuro »
2. Renommer le namespace localhost
3. Mettez le fichier EXE de l'application Windows dans un répertoire virtuel et accédez-y par une URL dans Internet Explorer

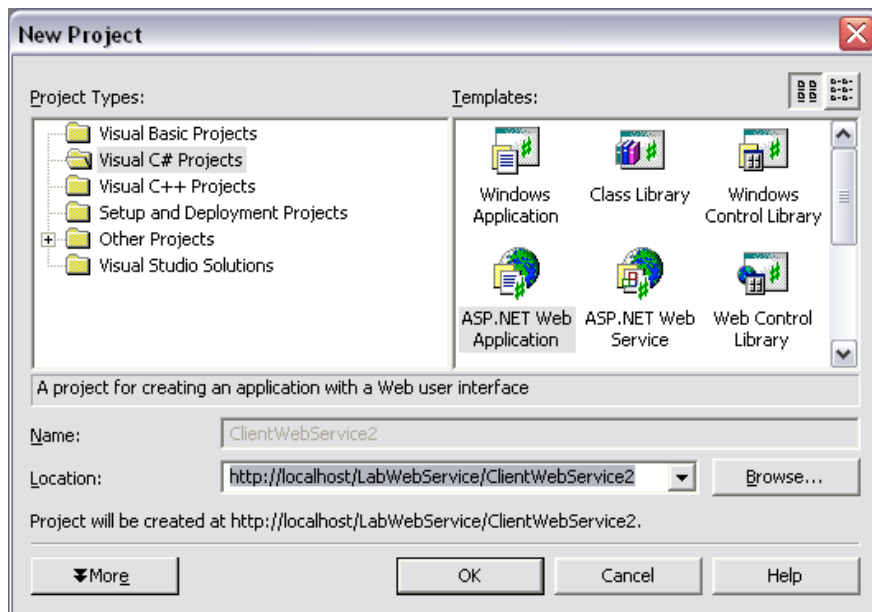
Note :



1.2 Deuxième Client Service Web (Site Web)

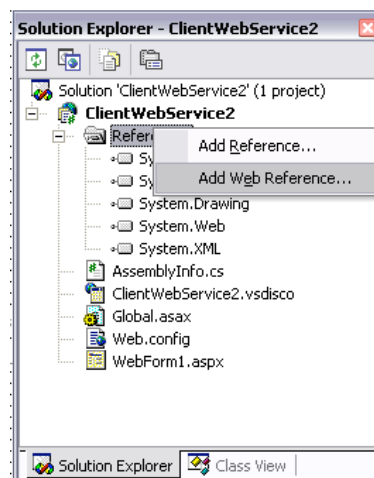
Déroulement de l'exercice:

1. Lancer Visual Studio .NET
 - Démarrer > Programmes > Microsoft Visual Studio .NET > Microsoft Visual Studio
2. Créer une application WinForm
 - Fichier > Nouveau > Projet
 - Projet Type : Choisissez C#
 - Templates : Choisissez ASP.NET WebApplication
 - Nom du projet : LabWebService/ClientWebService2
 - Cliquer sur Ok



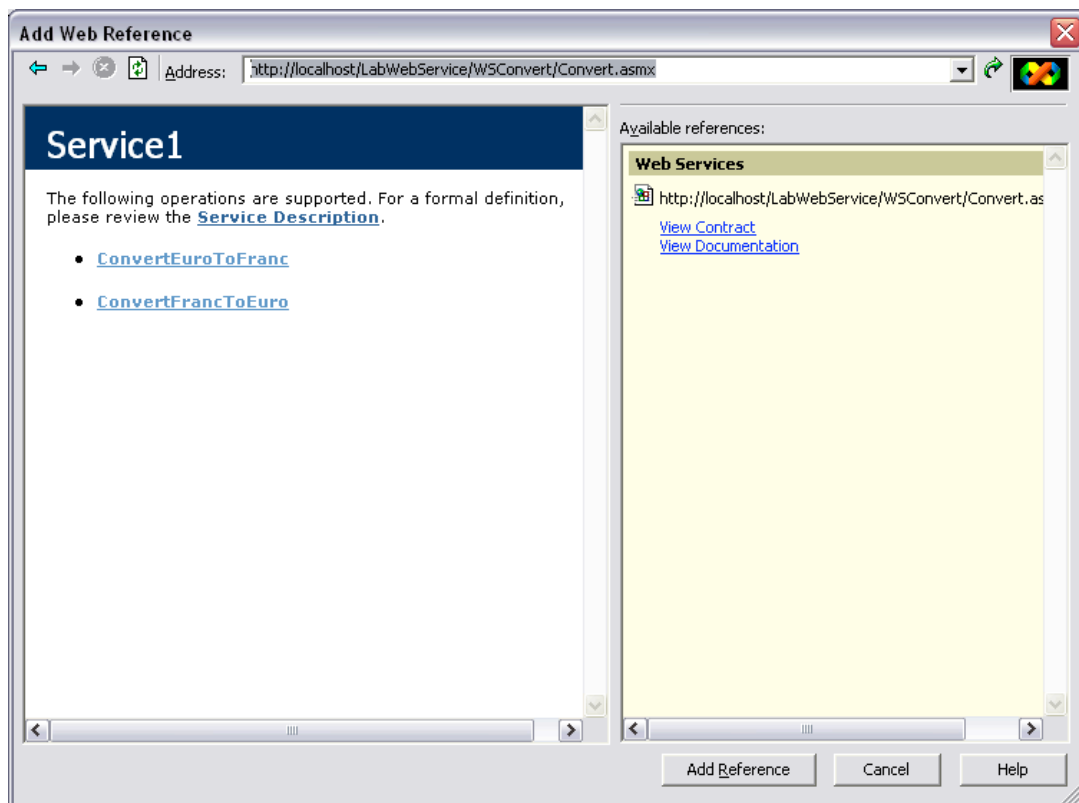
3. Regardez ce qui est créé
 - Une page WebForm1.aspx
4. Faire référence au Webservice déjà disponible sur la machine (Convert Euro <-> Franc)

Ajouter la référence au Webservice en cliquant sur le bouton droit sur référence et en sélectionnant « Add Web Référence »

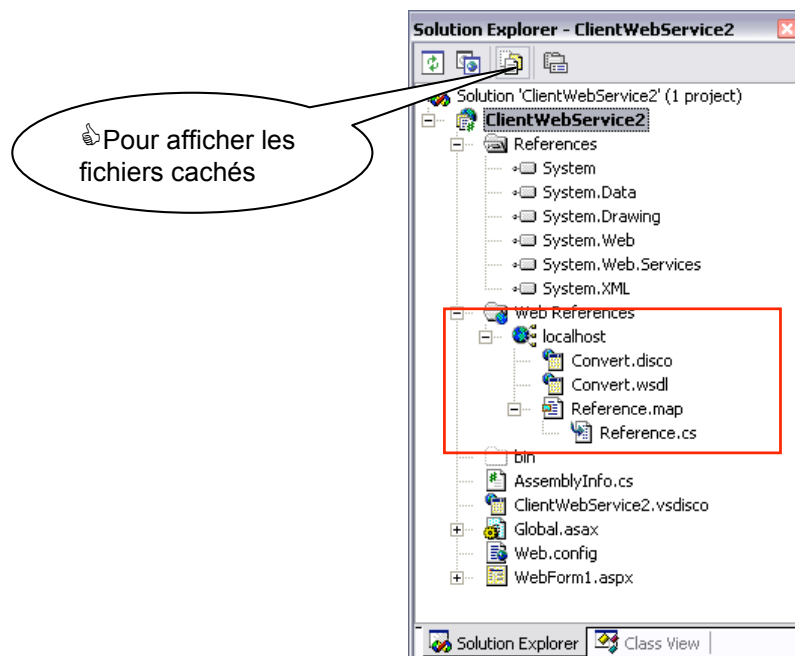


- Tapez l'URL dans l'adresse Web Reference :

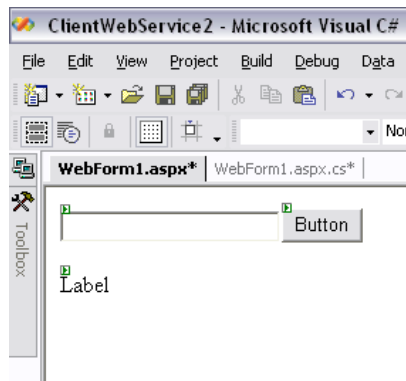
<http://localhost/LabWebService/WSConvert/Convert.asmx>



- On a deux Services Web de disponible sur cette URL
- Cliquez sur Add Reference
- On obtient donc ceci dans les références projet :



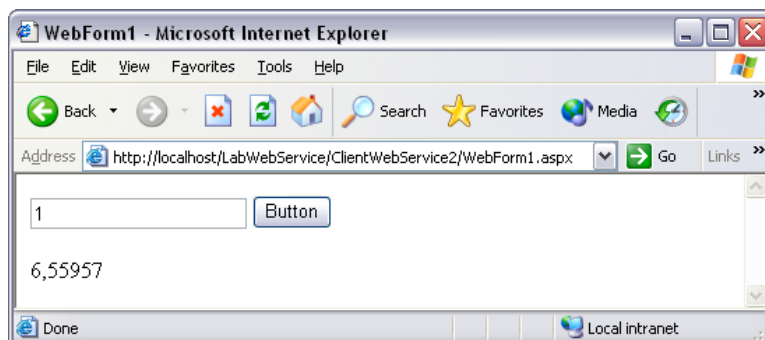
5. Placez un bouton, une zone d'édition et un label sur la page Web en utilisant la ToolBox et le glisser/déplacer.



6. Double cliquez sur le bouton pour déclenchez une action au moment du click.
 - Tapez ce code C#

```
private void Button1_Click(object sender, System.EventArgs e)
{
    //Déclaration de la variable sur l'objet Proxy ServiceConvert
    localhost.ServiceConvert MonService;
    //Intanciation de la variable
    MonService = new localhost.ServiceConvert();
    //Conversion de la zone de saisie en double
    double Euro = double.Parse(TextBox1.Text);
    //Appel du ServiceWeb
    double Franc = MonService.ConvertEuroToFranc(Euro);
    //Affichage du résultat
    Label1.Text = Franc.ToString();
}
```

7. Lancez l'application : 



Dans les applications Web l'appel au Service Web se fait de Serveur à Serveur. Un client interroge une URL (un serveur Web) et celui-ci pour répondre à la requête interroge par SOAP potentiellement un autre Serveur Web.



Attention le serveur Web est localisé. Attention à la saisie des nombres a virgules. Pour forcer une localisation allez dans le Web.Config et changer

En US (séparateur de décimal '.')

```
<globalization
    culture="en-US"
    requestEncoding="utf-8"
    responseEncoding="utf-8"
/>
```

En FR (séparateur de décimal ',')

```
<globalization
    culture="fr-FR"
    requestEncoding="utf-8"
    responseEncoding="utf-8"
/>
```



Pour aller plus loin

1. Appeler le deuxième Service Web « ConvertFrancToEuro »
2. Changer la globalisation
3. Gérer l'exception si l'appel génère une exception ou ne répond pas.

```
private void Button1_Click(object sender, System.EventArgs e)
{
    //Déclaration de la variable sur l'objet Proxy ServiceConvert
    localhost.ServiceConvert MonService;
    //Intanciation de la variable
    try
    {
        MonService = new localhost.ServiceConvert();
        //Conversion de la zone de saisie en double
        double Euro = double.Parse(TextBox1.Text);
        //Appel du ServiceWeb
        double Franc = MonService.ConvertEuroToFranc(Euro);
        //Affichage du résultat
        Label1.Text = Franc.ToString();
    }
    catch (Exception ex)
    {
        Label1.Text = ex.Message;
    }
}
```

 Note :

2 Fournir un Service Web

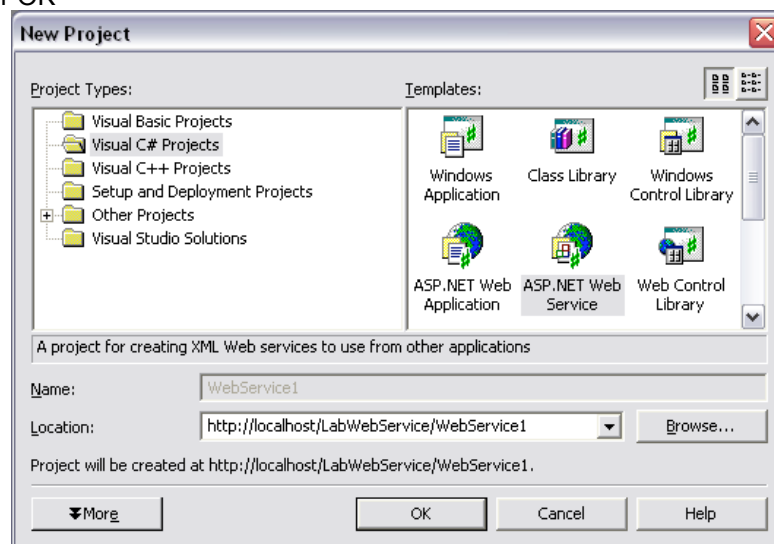
Dans ces exercices, vous allez apprendre à :

- Créer un Service Web avec une méthode de conversion Euro vers Franc
- Interroger le Service Web via son interface Web
- Ajouter des méthodes au service

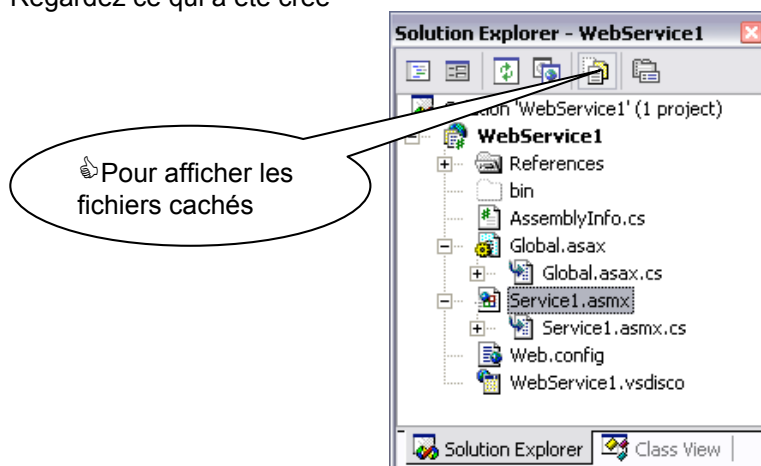
2.1 Créer un Service Web

Déroulement de l'exercice

1. Lancez Visual Studio .Net
 - Démarrer > Programmes > Microsoft Visual Studio .Net > Microsoft Visual Studio .Net
2. Créez un nouveau projet de Service Web
 - Fichier > Nouveau > Projet
 - Project Types : choisissez "C#"
 - Templates : choisissez "ASP.Net Web Service"
 - Nom du projet : LabWebService/WebService1
 - Cliquez sur OK



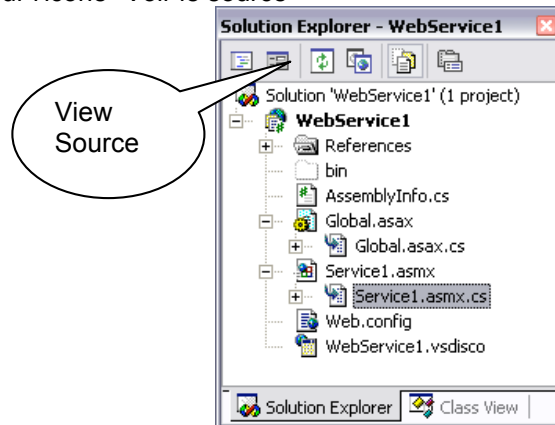
3. Regardez ce qui a été créé



- References : ne correspond pas à un fichier mais aux références sur les composants dont pourra avoir besoin votre service web
- Le répertoire "bin" qui va contenir la version compilée en DLL de votre service web
- AssemblyInfo.cs : fichier source contenant uniquement la définition des attributs d'Assembly
- Global.asax, Global.asax.cs, Global.asax.resx : le fichier d'application web avec son code source et ses ressources
- Service1.asmx, Service1.asmx.cs, Service1.asmx.resx : votre service web avec son code source et ses ressources
- Web.config : le fichier de configuration XML de votre application
- WebService1.vsdisco : le fichier DISCO d'inspection du service Web

4. Affichez le code source du service web

- Dans le panneau "Explorateur de solution"
- Sélectionnez le fichier "Service1.asmx"
- Cliquez sur l'icone "Voir le source"



- Dans la zone centrale, vous avez le code source du fichier Service1.asmx
- Prenez le temps de consulter le contenu du source

5. Créez une WebMethod

- Remplacez la méthode d'exemple par :

```
[WebMethod]
public double EuroVersFranc(double Euro)
{
    return Euro * 6.55957;
}
```



Seul l'attribut [WebMethod] est nécessaire pour faire d'une méthode normale une méthode accessible par votre service web

- Enregistrez les modifications (Ctrl+S)

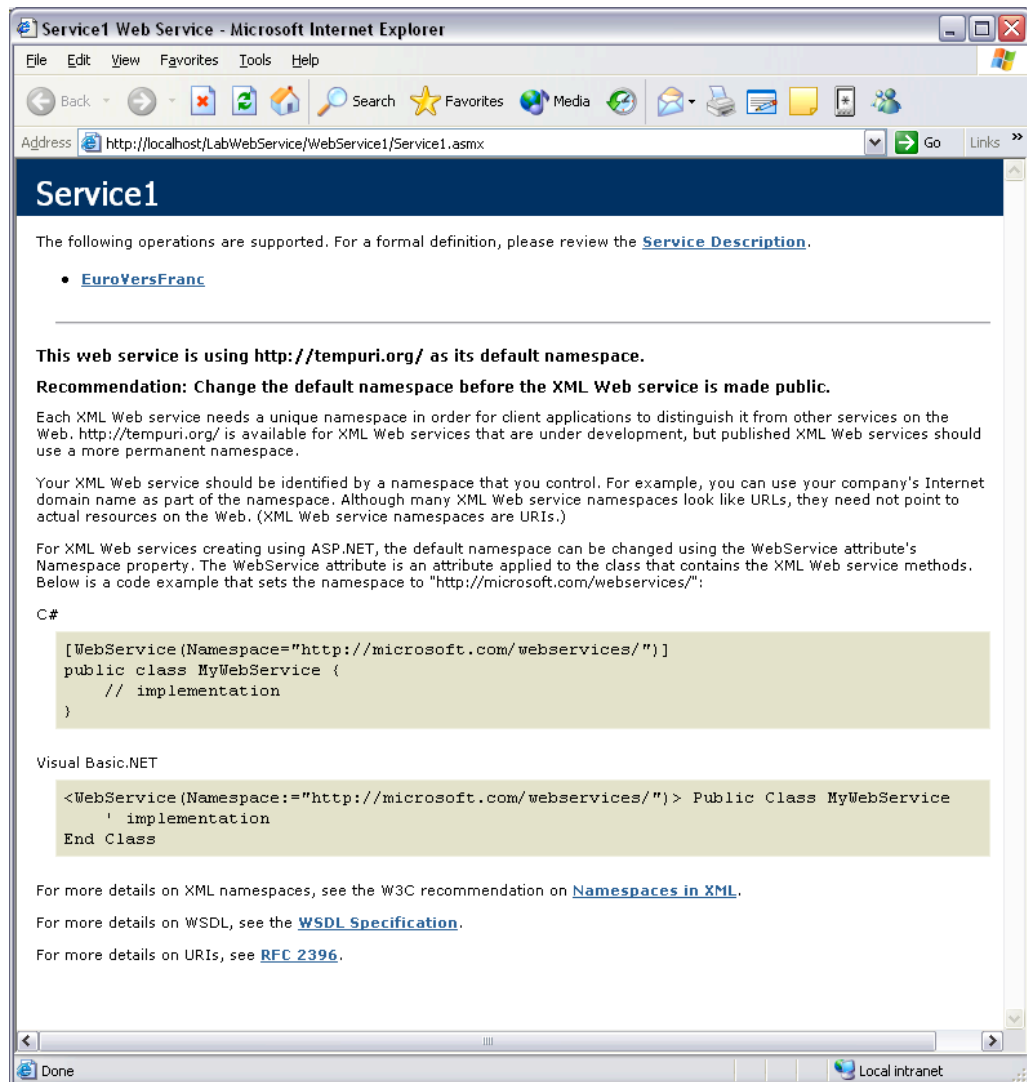
6. Testez directement votre service web

- Compilez la solution (Ctrl+Shift+B)
-



Le répertoire "caché" bin contient maintenant un fichier DLL, c'est lui qui va être réellement exécuté lors de l'appel du Webservice.

- Testez la solution (Ctrl+F5)
- VisualStudio a appelé Internet Explorer pour y afficher l'url de votre service web : <http://localhost/WebService1/Service1.aspx>



La page web que vous voyez est automatiquement générée par le Framework .Net et vous permet de tester votre service Web en utilisant votre navigateur

Elle contient la liste des méthodes de votre service web, donc ici uniquement la méthode "EuroVersFranc"



Remarquez le message vous indiquant que le namespace par défaut n'a pas été redéfini.

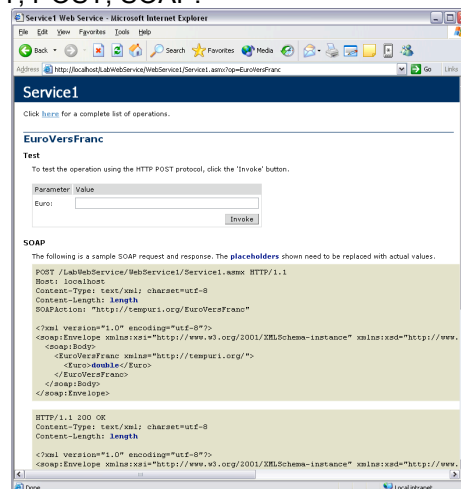


En cliquant sur le lien "Description du service" vous verrez s'afficher le document WSDL correspondant a votre service web. On s'en est servie pour écrire nos programmes client du début.

L'URL de cette page est l'URL du service web ajoutée de ?WSDL

```
<?xml version='1.0' encoding='utf-8' ?>
<definitions xmlns:http='http://schemas.xmlsoap.org/wsdl/http/'
  xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
  xmlns:s='http://www.w3.org/2001/XMLSchema' xmlns:s0='http://tempuri.org/'
  xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
  xmlns:mime='http://microsoft.com/wsdl/mime/textMatching/'
  xmlns:mime2='http://schemas.xmlsoap.org/wsdl/mime/'
  xmlns:tempuri='http://tempuri.org/' xmlns:tns='http://schemas.xmlsoap.org/wsdl/'>
  <portType name='EuroVersFranc' binding='http://schemas.xmlsoap.org/wsdl/soap/'>
    <operation name='EuroVersFranc'>
      <input message='s0:EuroVersFrancSoapIn' type='s0:EuroVersFrancSoapIn' />
      <output message='s0:EuroVersFrancSoapOut' type='s0:EuroVersFrancSoapOut' />
    </operation>
  </portType>
  <binding name='Service1Soap' type='s0:Service1Soap'>
    <operation name='EuroVersFranc'>
      <input message='s0:EuroVersFrancSoapIn' type='s0:EuroVersFrancSoapIn' />
      <output message='s0:EuroVersFrancSoapOut' type='s0:EuroVersFrancSoapOut' />
    </operation>
  </binding>
  <message name='EuroVersFrancSoapIn'>
    <part name='parameters' element='s0:EuroVersFranc' />
  </message>
  <message name='EuroVersFrancSoapOut'>
    <part name='parameters' element='s0:EuroVersFrancResponse' />
  </message>
  <portType name='Service1Soap'>
    <operation name='EuroVersFranc'>
      <input message='s0:EuroVersFrancSoapIn' />
      <output message='s0:EuroVersFrancSoapOut' />
    </operation>
  </portType>
  <binding name='Service1Soap' type='s0:Service1Soap'>
    <operation name='EuroVersFranc'>
      <input message='s0:EuroVersFrancSoapIn' type='s0:EuroVersFrancSoapIn' />
      <output message='s0:EuroVersFrancSoapOut' type='s0:EuroVersFrancSoapOut' />
    </operation>
  </binding>
  <schema targetNamespace='http://tempuri.org/'>
    <sequence base='xsd:string' minOccurs='1' maxOccurs='1' name='Euro' type='s:double' />
  </sequence>
  <complexType base='xsd:string' name='EuroVersFrancResponse' />
  <sequence base='xsd:string' minOccurs='1' maxOccurs='1' name='EuroVersFrancResult' type='s:double' />
  </sequence>
  <complexType base='xsd:string' name='EuroVersFrancResponse' />
  </complexType>
  </schema>
</definitions>
```

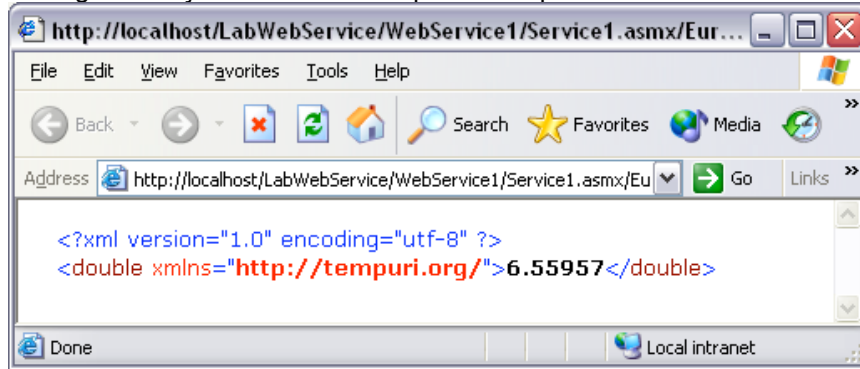
7. En cliquant sur le lien "EuroVersFranc", vous accédez à une page web (également générée automatiquement par le Framework) qui vous permet d'appeler le service web en donnant ses arguments. Vous voyez également quelles seraient les données envoyées au service web suivant trois méthodes possibles : GET, POST, SOAP.



Cette page permet de tester directement un service web en utilisant la méthode GET et que celle-ci ne peut fonctionner que pour les services dont les paramètres d'entrées sont une liste de types simples (ni tableaux, ni classes).

8. Rentrez un prix en Euro et cliquez sur le bouton "Invoquer"

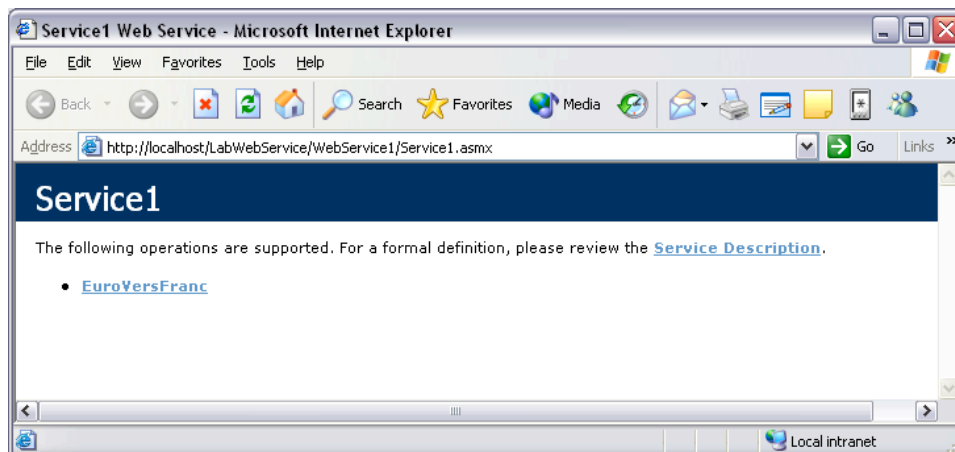
- Le navigateur reçoit et affiche telle-qu'elle la réponse XML du service web



9. Modifiez le "namespace" (espace de nom XML) de votre service web
 - Ajouter la ligne d'attribut suivante avant la ligne de déclaration de la classe "Service1"
 -

```
[WebService(Namespace="http://www.microsoft.com/france/msdn/lab/servicesweb/") ]
public class Service1 : System.Web.Services.WebService
{
    public Service1()
    ...
}
```

- Compilez et testez à nouveau, le message sur le namespace par défaut disparaît



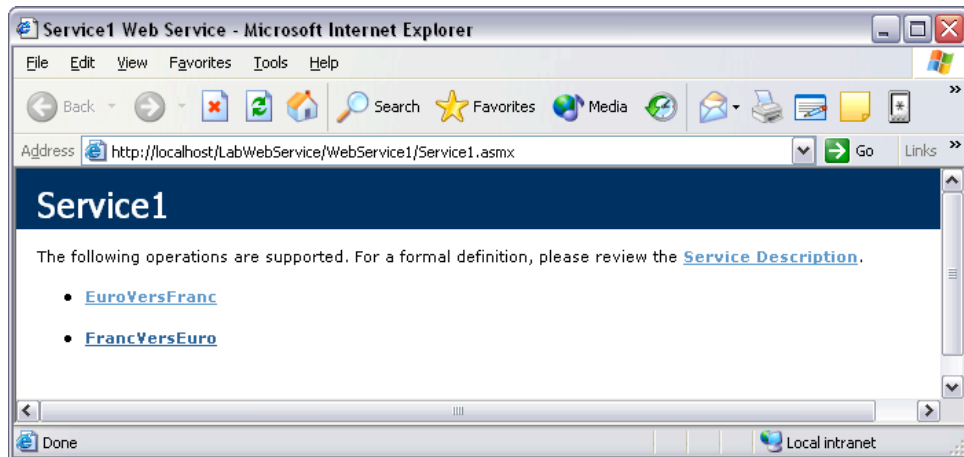
10. Ajoutez une méthode de conversion "FrancVersEuro"

```
[WebMethod]
public double FrancVersEuro(double Franc)
{
    return Franc / 6.55957;
}
```

- Compilez puis Testez



Vous avez cette fois-ci deux méthodes dans la page de présentation de votre service web



11. Testez l'importance de l'attribut WebMethod

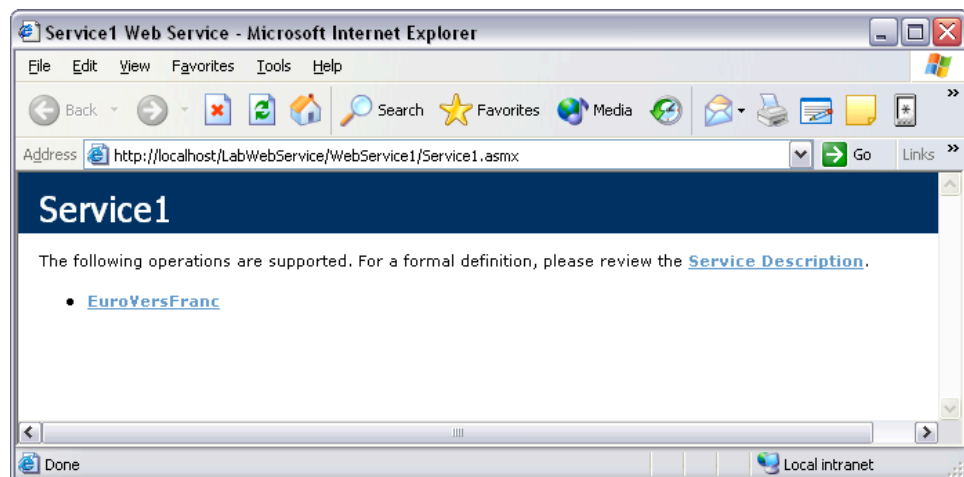
- Mettez en commentaire la ligne définissant l'attribut WebMethod sur la méthode FrancVersEuro
- Ajoutez une méthode de conversion "FrancVersEuro"

```
//[WebMethod]
public double FrancVersEuro(double Franc)
{
    return Franc / 6.55957;
}
```

- Recompilez puis testez



Seule la méthode EuroVersFranc apparait dans la liste des méthodes supportées par le service web



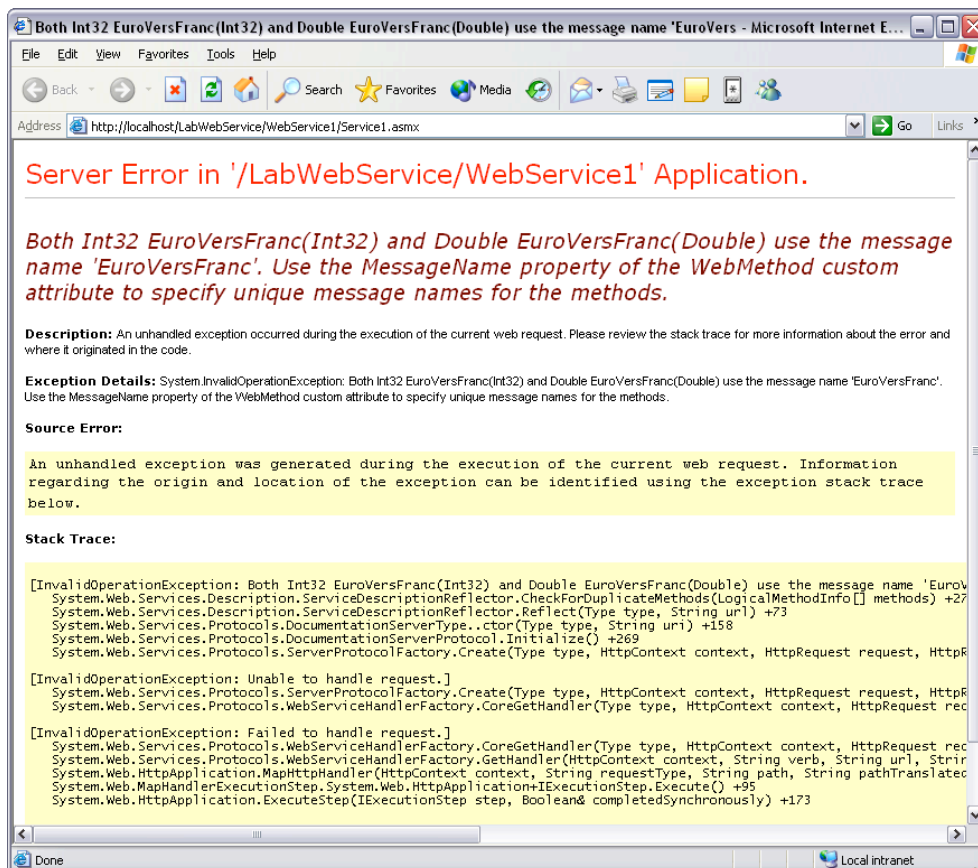
- Enlevez le commentaire pour que l'on puisse utiliser cette méthode dans la suite des exercices

12. Nommage pour les méthodes surchargées

- La propriété `MessageName` de l'attribut `WebMethod` permet de redéfinir le nom de la méthode au niveau du service web, par défaut c'est le nom de la méthode elle-même.
- Créez une nouvelle méthode `EuroVersFranc` qui travaille en chiffres entier (valeur * 100)

```
[WebMethod]
public int EuroVersFranc(int FrancCentimes)
{
    return (int)(FrancCentimes * 6.55957);
}
```

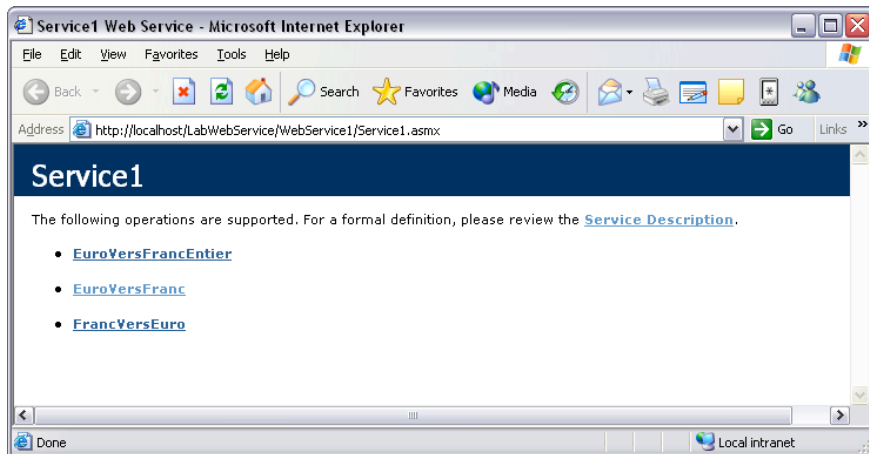
- Enregistrez, Compilez, Testez
 - Remarquez qu'il n'y a pas eu d'erreur à la compilation
 - Vous voyez que la page ASMX renvoie un message d'erreur vous indiquant que vous avez deux méthodes avec le même nom.



- Ajoutez la propriété `MessageName` à l'attribut `WebMethod`

```
[WebMethod(MessageName="EuroVersFrancEntier")]
public int EuroVersFranc(int FrancCentimes)
{
    return (int)(FrancCentimes * 6.55957);
}
```

- Cette fois-ci, la liste des méthodes du service s'affiche et inclue la nouvelle méthode.

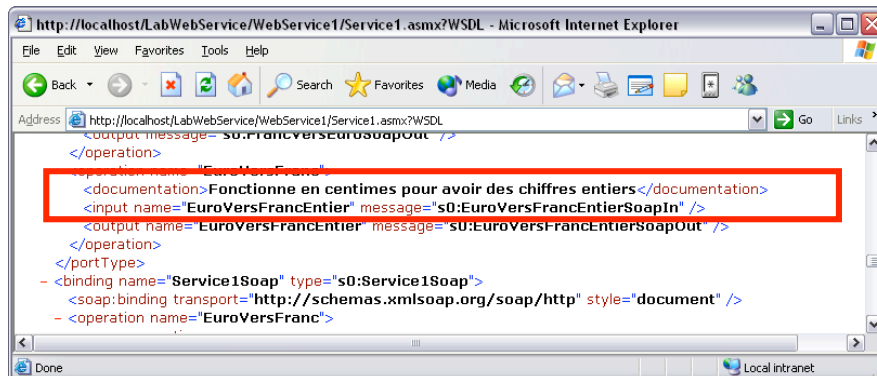


Lorsque vous créez un client pour cette méthode, vous allez utiliser le nom normal de la méthode et non pas le "MessageName".

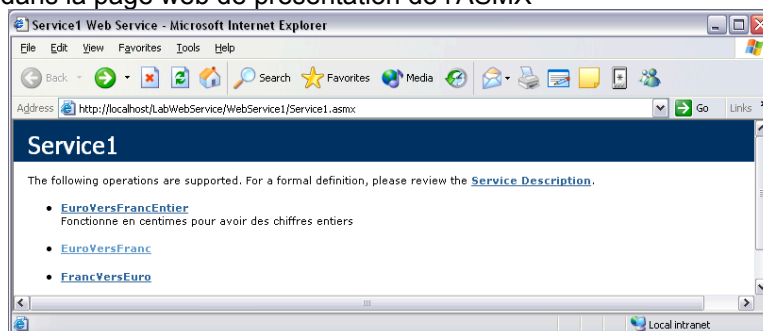
13. Ajoutez une description de la méthode

```
[WebMethod(MessageName="EuroVersFrancEntier", Description="Fonctionne en
centimes pour avoir des chiffres entiers")]
public int EuroVersFranc(int FrancCentimes)
{
    return (int)(FrancCentimes * 6.55957);
}
```

- Enregistrez, Compilez, Testez
- Cette description apparaîtra dans le document WSDL ...



- ... et dans la page web de présentation de l'ASMX





Pour aller plus loin

1. Ajouter dans ce WebService l'appel au premier WebService des 2 premiers exercices
2. Reprendre les 2 premiers exercices et les faire pointer sur votre nouveau Service Web.

2.2 Gestion des exceptions coté serveur

Soap autorise de lever une exception coté serveur pour être propagé ensuite coté client. L'exception va passer dans l'entête http.

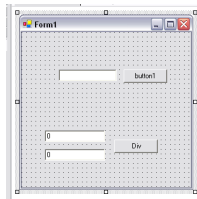
1. Faire un nouveau Service Web (**Dans le WebService fourni avec le Lab « WSConvert »**) qui fait une division. Vérifier si le deuxième paramètre n'est pas égal à zéro sinon lever une SOAP Exception.

Ajoutez :

```
using System.Web.Services.Protocols;
```

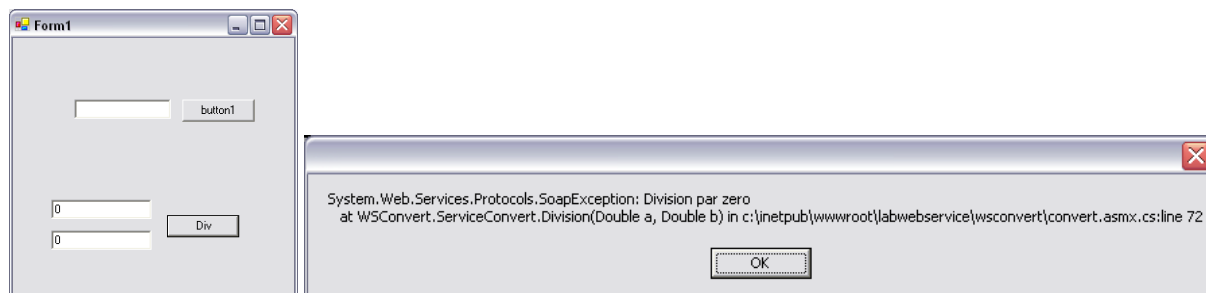
```
[WebMethod]
public double Division(double a, double b)
{
    if (b != 0)
    {
        return a / b;
    }
    else
    {
        //On leve l'exception SOAP
        throw new SoapException("Division par zero",
                                SoapException.ClientFaultCode);
    }
}
```


2. Ajouter à votre client WinForm du premier exercice l'appel à cette nouvelle méthode et encapsuler l'appel par un try...catch.



```
private void button2_Click(object sender, EventArgs e)
{
    try
    {
        localhost.ServiceConvert MonService =
            new localhost.ServiceConvert();
        MessageBox.Show(MonService.Division(double.Parse(textBox2.Text),
            double.Parse(textBox3.Text)).ToString());
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

3. Compiler et Tester



 Note :

3 Types de données

Dans ces exercices, vous allez apprendre à :

- Utiliser des types de données complexes
- Modifier la sérialisation XML
- Utiliser des documents XML
- Utiliser les DataSet de ADO.NET
- Utiliser des données binaires

3.1 Utilisation de types complexes

On va voir que l'on peut utiliser des classes, et même des tableaux de classes en entrée comme en sortie.

Dans le projet du service web, dans le fichier Service1.asmx.cs :

1. Ajoutez une classe publique qui représente le taux de change d'une devise par rapport à l'Euro

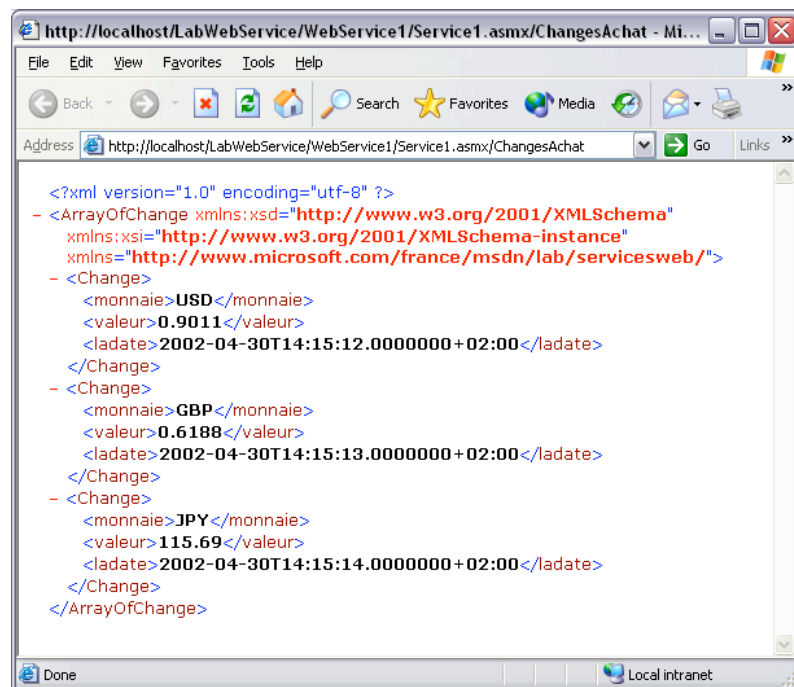
```
public class Change
{
    public string monnaie;
    public double valeur;
    public DateTime ladate;
}
```

2. Ajoutez une nouvelle méthode au service qui va renvoyer une liste de taux de change

```
[WebMethod]
public Change[] ChangesAchat()
{
    Change[] chgs=new Change[3];
    chgs[0]=new Change();
    chgs[0].monnaie="USD";
    chgs[0].valeur=0.9011;
    chgs[0].ladate=new DateTime(2002,4,30,14,15,12);
    chgs[1]=new Change();
    chgs[1].monnaie="GBP";
    chgs[1].valeur=0.6188;
    chgs[1].ladate=new DateTime(2002,4,30,14,15,13);
    chgs[2]=new Change();
    chgs[2].monnaie="JPY";
    chgs[2].valeur=115.69;
    chgs[2].ladate=new DateTime(2002,4,30,14,15,14);
    return chgs;
}
```

3. Enregistrez, Compilez et Testez

- Observez le résultat XML lorsque vous lancez cette méthode :



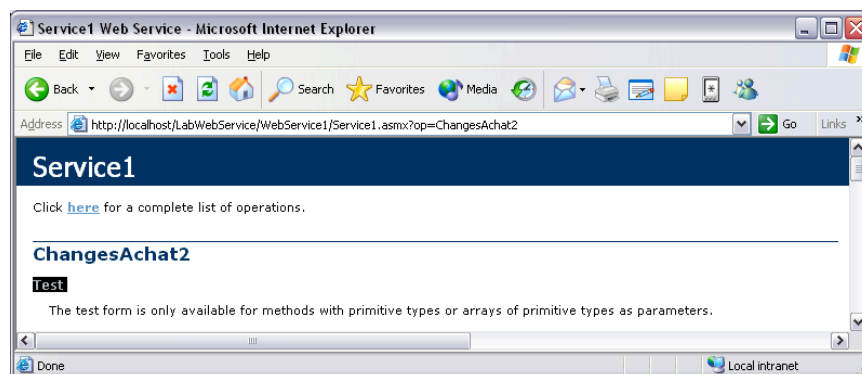
```
<?xml version="1.0" encoding="utf-8" ?>
- <ArrayOfChange xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.microsoft.com/france/msdn/lab/servicesweb/">
- <Change>
  <monnaie>USD</monnaie>
  <valeur>0.9011</valeur>
  <ladata>2002-04-30T14:15:12.0000000+02:00</ladata>
</Change>
- <Change>
  <monnaie>GBP</monnaie>
  <valeur>0.6188</valeur>
  <ladata>2002-04-30T14:15:13.0000000+02:00</ladata>
</Change>
- <Change>
  <monnaie>JPY</monnaie>
  <valeur>115.69</valeur>
  <ladata>2002-04-30T14:15:14.0000000+02:00</ladata>
</Change>
</ArrayOfChange>
```

- C'est le résultat de la sérialisation XML du tableau de classes



Pour aller plus loin

1. Créez un service web qui prend un tableau de classes « Change » en entrée.



Note :

3.2 Modification de la sérialisation XML

On va voir que l'on peut modifier la façon dont les données sont sérialisées dans le code XML.

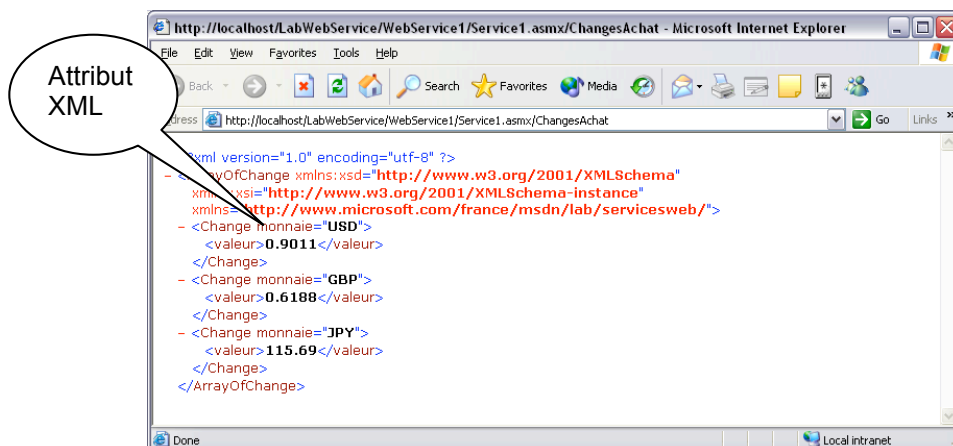
1. Ajouter le namespace "System.Xml.Serialization" au début du code source

```
...  
using System.Diagnostics;  
using System.Web;  
using System.Web.Services;  
//Ligne ajouté  
using System.Xml.Serialization;
```

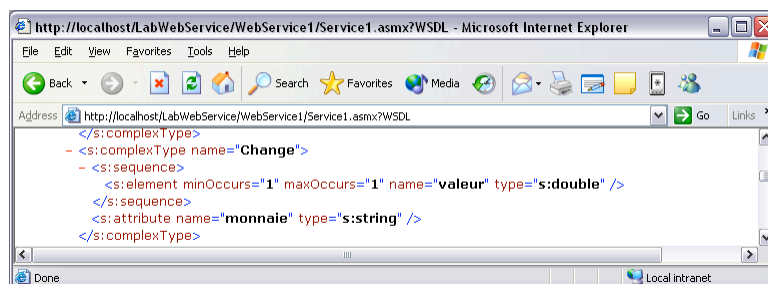
2. Ajouter des attributs de sérialisation XML aux variables de la classe Change

```
public class Change  
{  
    [XmlAttributeAttribute] public string monnaie;  
    public double valeur;  
    [XmlIgnore] public DateTime ladate;  
}
```

3. Enregistrez, Compilez, Testez
 - la variable "monnaie" est devenue un attribut de l'élément Change
 - la variable "ladate" n'apparaît plus dans le source XML



4. Regardez l'impact au niveau du client
 - Mettez à jour la référence web du client
 - Regardez le fichier WSDL de la référence, la variable monnaie est spécifiée en attribut



Un client Service Web ne verrait donc plus la variable « ladate ». Elle n'est plus exposée dans la description du Service Web.

3.3 Utilisation d'un document XML

Dans certains cas de figure, vous ne souhaitez pas gérer les données du service web par les structures de données du langage mais directement dans un document XML, par exemple issu d'un fichier ou d'une base de données XML. Il vous suffit de renvoyer directement un objet `System.Xml.XmlElement`.

1. Ajoutez une méthode qui renvoie directement un document XML


```
[WebMethod]
public System.Xml.XmlElement GetXMLFile()
{
    System.Xml.XmlDataDocument xd=new System.Xml.XmlDataDocument();
    xd.Load(Server.MapPath("Web.config"));
    return xd.DocumentElement;
}
```

2. Testez la nouvelle méthode : elle renvoie le contenu du fichier XML



Pour aller plus loin

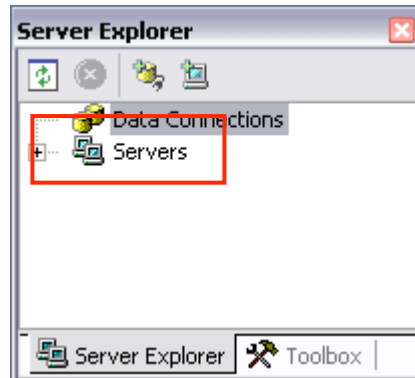
1. Créez un service web qui reçoit en argument un document XML
2. Créez le client associé.

 Note :

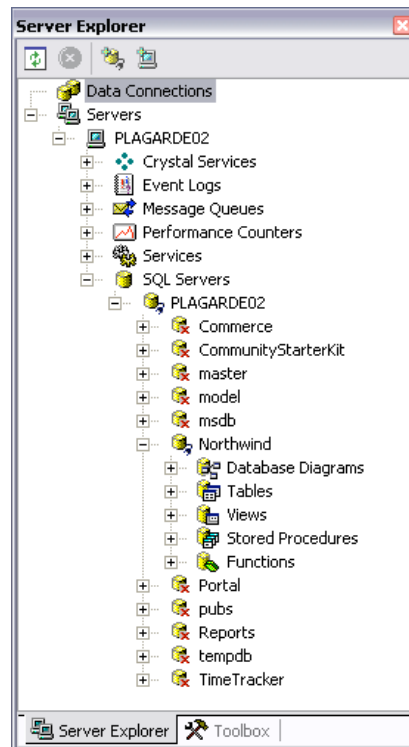
3.4 Utilisation des DataSets

Un objet DataSet est comme une petite base de données en mémoire : il peut contenir plusieurs tables, gère les relations entre elles, et peut se synchroniser avec la base de données réelle. On va voir que l'on peut utiliser un tel objet dans un service web pour exposer indirectement une base de données à une application cliente distante.

1. Sur le service Web, se mettre sur la vue Design du fichier Service1.asmx
 - Dans l'explorateur de solutions, double-cliquez sur le fichier "Service1.asmx"
 - Une page blanche apparaît au centre de Visual Studio
2. Se connecter à la base de données
 - Depuis "l'explorateur de serveurs", sur la première branche "bases de données"

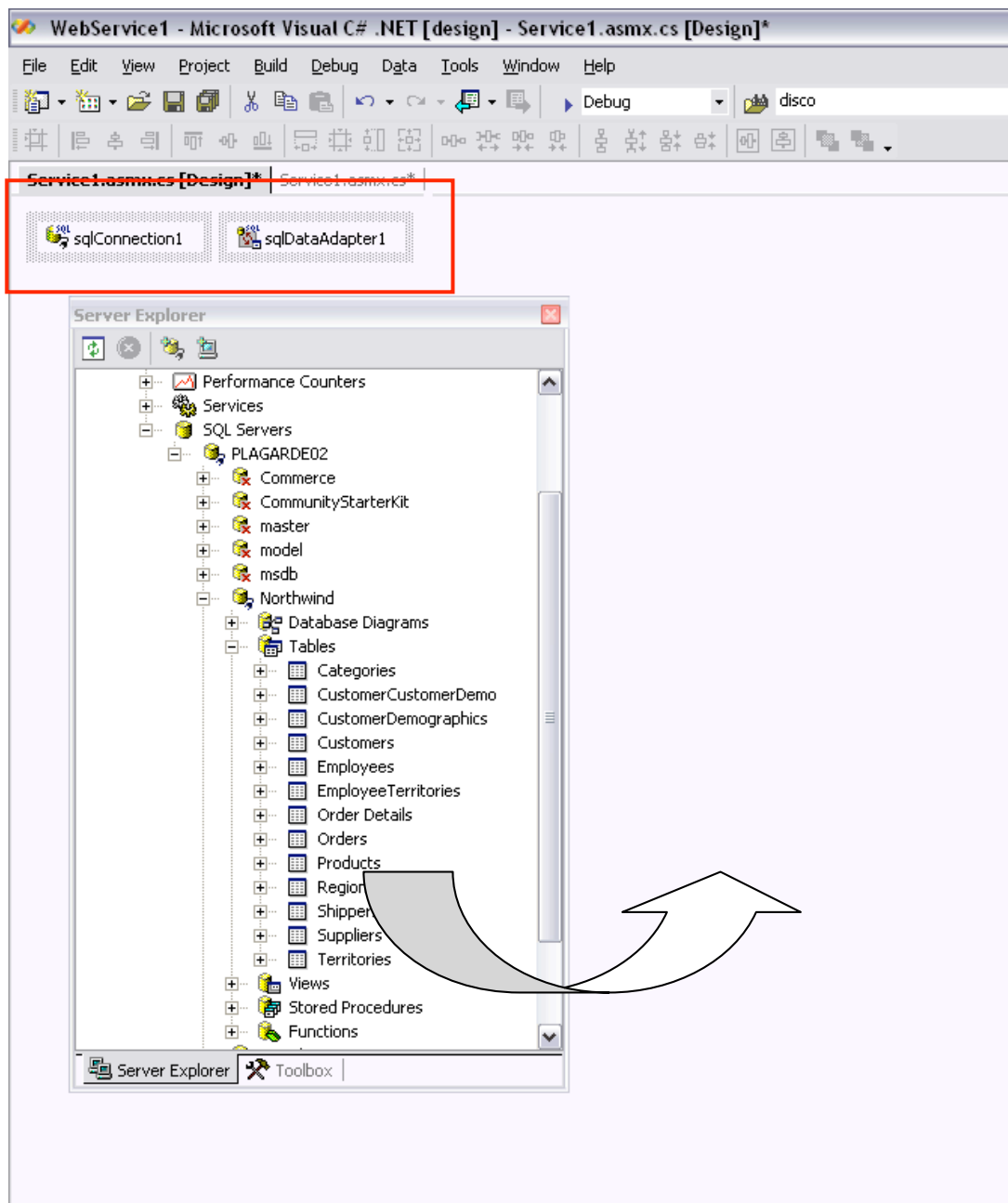


- Etablir une connection à cette base en utilisant
 - i. le nom de serveur "localhost",
 - ii. le mode d'authentification SQL (utilisateur "sa", mot de passe vide),
 - iii. sélectionnez la base "Northwind"



3. Définir une connection à la table "Products"

- sélectionnez la table "Products"
- Faites un Glisser-Déposer de la table dans la vue de conception
- Deux objets sont créés :



sqlConnection1 : contient les paramètres de connexion à la base
 sqlDataAdapter1 : contient les informations de traitement de la table

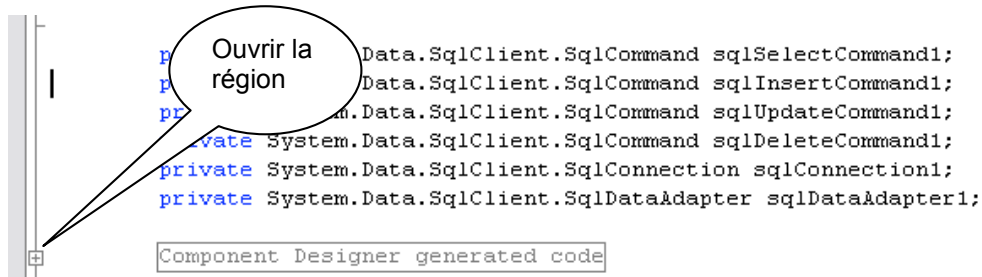


Ce qu'il s'est passé :

Affichez le code source du fichier ASMX (clique-droit > "voir le code")

Vous voyez que 6 objets privés ont été définis : une SqlConnection, un SqlDataAdapter, et 4 objets SqlCommand pour les opérations de sélection, insertion, mise à jour et suppression sur la table.

Ouvrez la région "code généré par le concepteur"



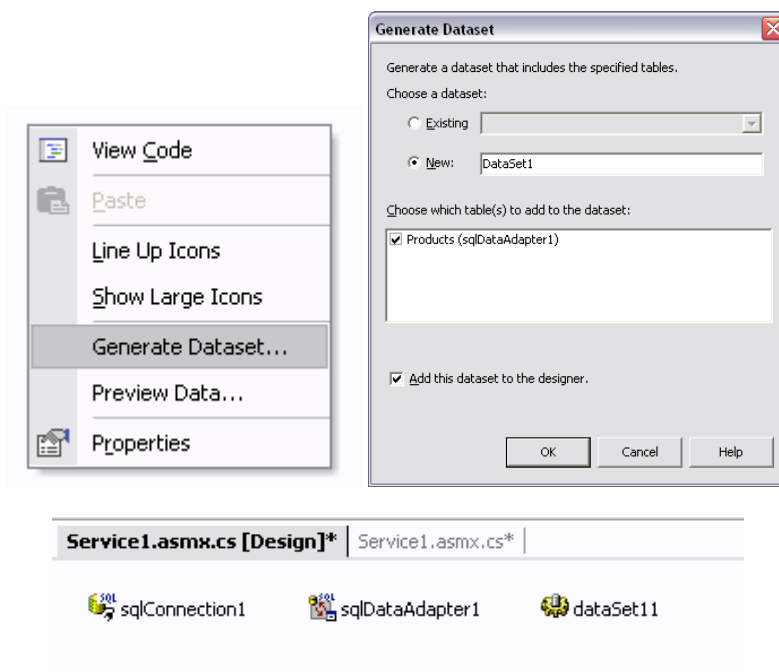
Dans la méthode InitializeComponent, vous voyez la déclaration des 6 objets et la définition de leurs propriétés.



Vous pouvez voir ces mêmes propriétés en revenant sur la vue de conception : en cliquant sur les objets SqlConnection et SqlDataAdapter, vous les voyez dans le panneau "Propriétés". Les objets SqlCommand n'apparaissent pas dans la vue de conception, mais vous pouvez accéder à leur propriétés depuis les celles du SqlDataAdapter, ou en sélectionnant leur nom dans le menu déroulant du panneau.

4. Générer un DataSet typé

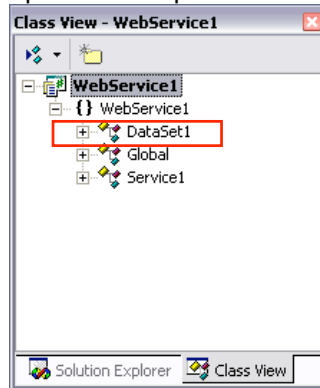
- Dans la vue de conception, Cliquez droit > Générer un DataSet : nouveau "DataSet1", ajouter au concepteur





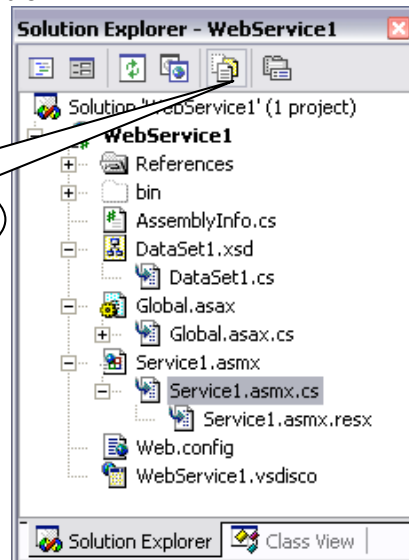
Ce qu'il s'est passé :

Visual Studio a ajouté un nouvel objet DataSet1 dont la classe DataSet1 a, elle aussi, été générée. Cette nouvelle classe apparaît dans le panneau "explorateur de classes".



Dans l'explorateur de solution, vous voyez un nouvel élément : DataSet1.xsd : il s'agit de la définition en XML de cette classe. Elle sera notamment utilisée par WSDL lors de la référence web par le client. En affichant les fichiers cachés, vous voyez également sa version source : DataSet1.cs. En regardant le code, vous voyez que cette classe hérite de DataSet, qu'elle est sérialisable en XML, et qu'elle contient notamment une classe interne "productsRow" qui hérite de "DataRow" et contient des propriétés correspondant aux colonnes de la table.

👉 Pour afficher les fichiers cachés



5. Renvoyer le dataset

- Dans le service web, ajoutez la méthode "GetProducts".

```
[WebMethod]
public DataSet1 GetProducts()
{
    DataSet1 ds=new DataSet1();
    sqlDataAdapter1.Fill(ds);
    return ds;
}
```

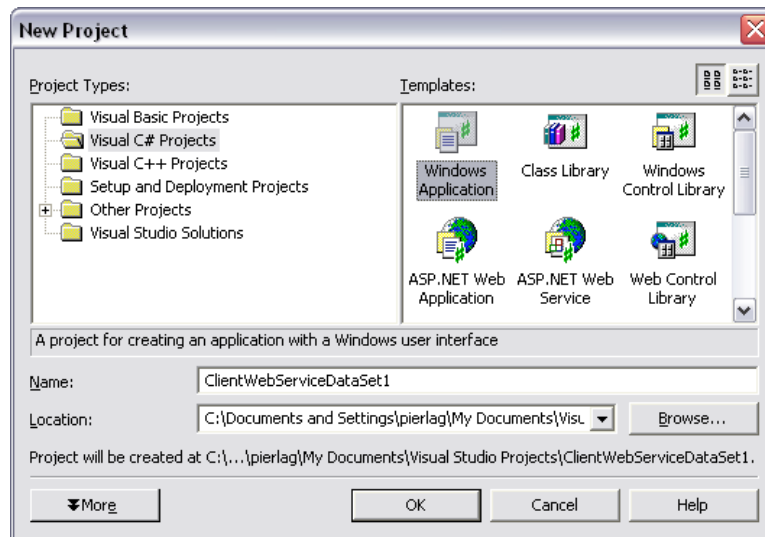
6. Mettre à jour le dataset

- Dans le service web, ajoutez la méthode "UpdateProducts".

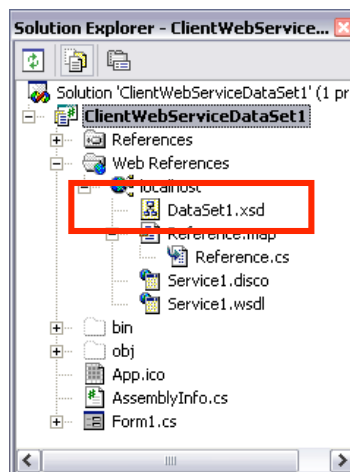
```
[WebMethod]
public DataSet1 UpdateProducts(DataSet1 ds)
{
    if (ds != null)
    {
        sqlDataAdapter1.Update(ds);
    }
    return ds;
}
```

- Compiler le projet

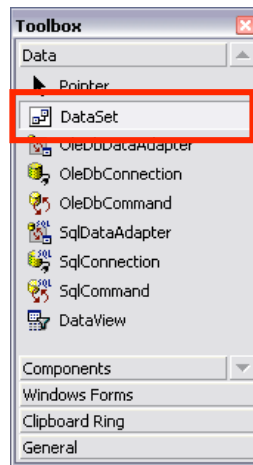
7. Créer une nouvelle application Client (Voir Exercice 1) mais cette fois si sur ce WebService :
<http://localhost/LabWebService/WebService1/Service1.asmx>



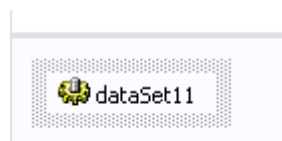
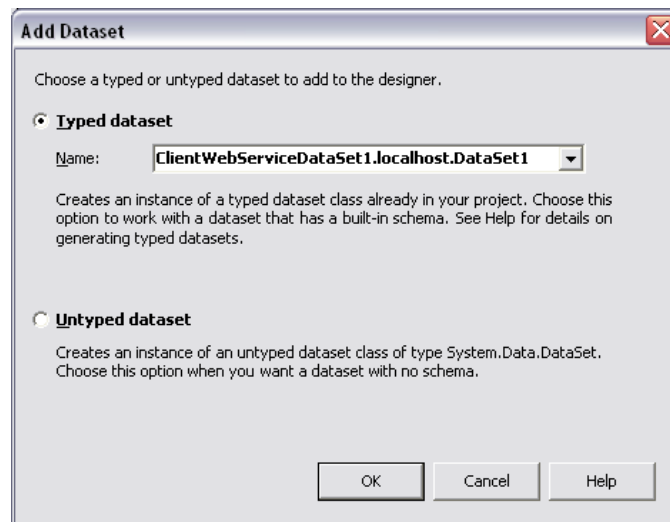
Le fichier DataSet1.xsd qui s'est ajouté et le contenu du fichier source Reference, augmenté de la classe DataSet1.



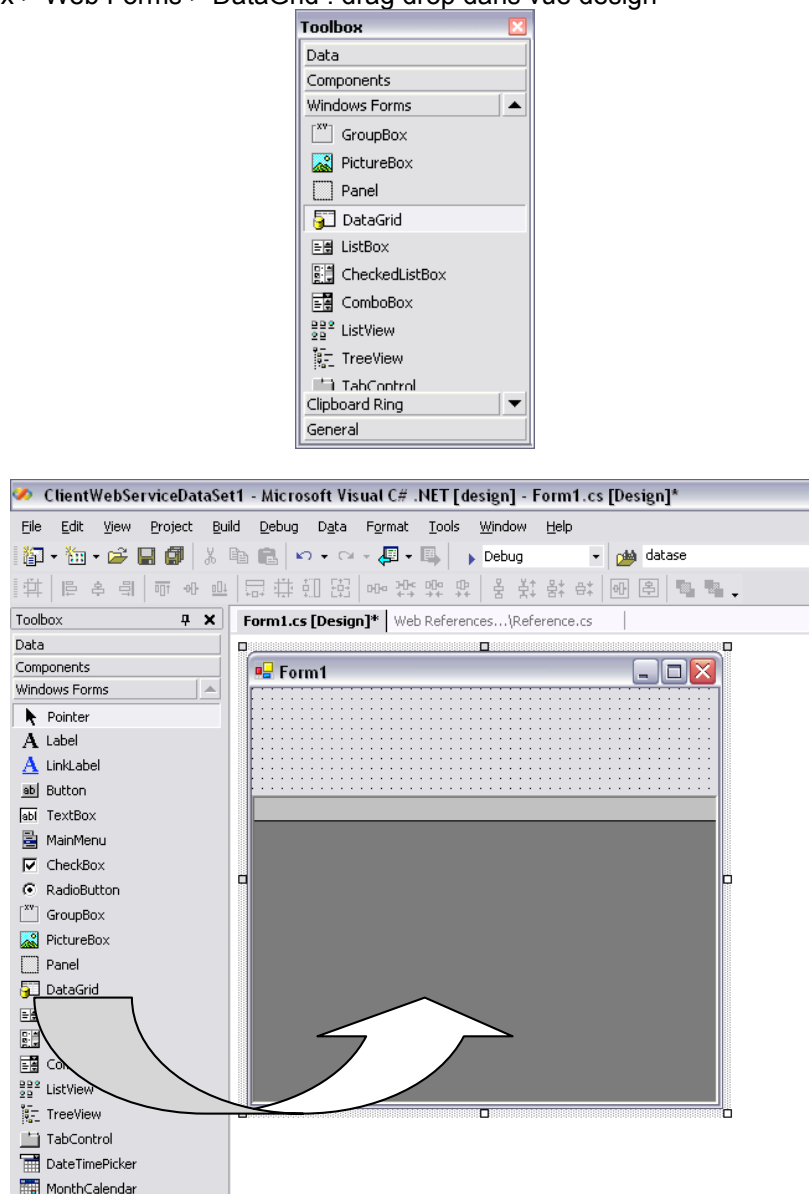
8. Sur l'application cliente : récupérer le dataset, le modifier et le mettre à jour
- Ajoutez un DataSet dans la vue de conception
ToolBox > Data Tab > DataSet : drag drop dans vue design



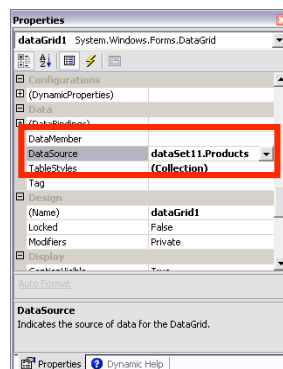
Choisir DataSet typé : localhost.DataSet1, il va se nommer "dataSet11"



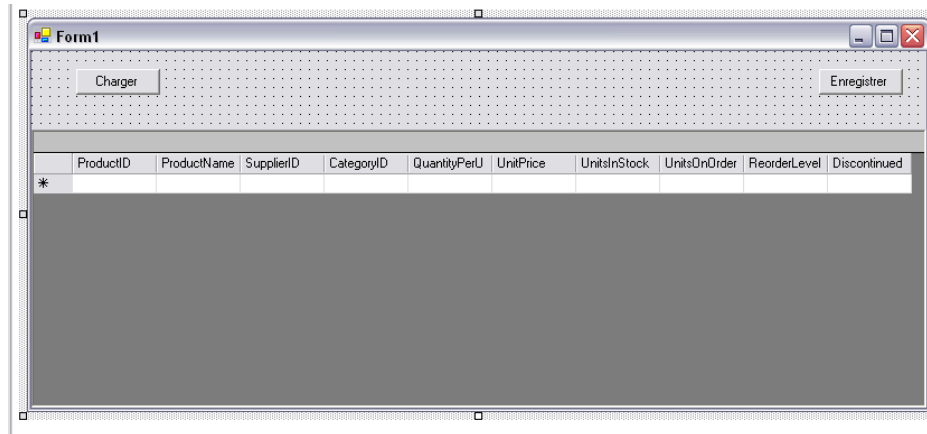
- Ajoutez un composant visuel : DataGrid
ToolBox > Web Forms > DataGrid : drag drop dans vue design



- Connectez le DataGrid au DataSet
Dans les propriétés du DataGrid, spécifiez DataSource="DataSet11.Products"



- Ajoutez deux boutons "Charger" et "Enregistrer" sous le DataGrid



- Entrez les codes source suivant pour traiter l'activation des boutons "Charger" :

```
private void buttonCharger_Click(object sender, System.EventArgs e)
{
    //Creation de l'objet Proxy Service Web
    localhost.Service1 sw = new localhost.Service1();
    //Efface le DataSet local
    dataSet11.Clear();
    //Replis le DataSet local par les données renvoyé par le Webservice
    dataSet11.Merge(sw.GetProducts());
}
```

"Enregistrer"

```
private void buttonEnregistrer_Click(object sender, System.EventArgs e)
{
    //Creation de l'objet Proxy Service Web
    localhost.Service1 sw=new localhost.Service1();
    //Creation de l'objet DataSet Distant
    localhost.DataSet1 ds=new localhost.DataSet1();
    //Si le DataSet local a changé
    if(dataSet11.HasChanges())
    {
        //Recupere les modifications effectuées et les met dans "ds"
        ds.Merge(dataSet11.GetChanges());
        //On envoi en mise à jour que les modifications effectuées
        sw.UpdateProducts(ds);
    }
    else
    {
        //Si le DataSet local n'a pas été modifié on ne fait rien
        MessageBox.Show("Rien à mettre à jour");
    }
}
```

- Enregistrez, compilez, testez

Form1

Charger Enregistrer

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerU	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
▶	1	Chai	1	1	10 boxes x 20	18	39	0	10	<input type="checkbox"/>
	2	Changia	1	1	24 - 12 oz bot	19	17	40	25	<input type="checkbox"/>
	3	Aniseed Syru	1	2	12 - 550 ml b	10	13	70	25	<input type="checkbox"/>
	4	Chef Anton's	2	2	48 - 6 oz jars	22	53	0	0	<input type="checkbox"/>
	5	Chef Anton's	2	2	36 boxes	21.35	0	0	0	<input checked="" type="checkbox"/>
	6	Grandma's B	3	2	12 - 8 oz jars	25	120	0	25	<input type="checkbox"/>
	7	Uncle Bob's	3	7	12 - 1 lb pkgs	30	15	0	10	<input type="checkbox"/>
	8	Northwoods	3	2	12 - 12 oz jar	40	6	0	0	<input type="checkbox"/>
	9	Mishi Kobe Ni	4	6	18 - 500 g pk	97	29	0	0	<input checked="" type="checkbox"/>
	10	Ikura	4	8	12 - 200 ml ja	31	31	0	0	<input type="checkbox"/>
	11	Queso Cabral	5	4	1 kg pkg.	21	22	30	30	<input type="checkbox"/>
	12	Queso Manc	5	4	10 - 500 g pk	38	86	0	0	<input type="checkbox"/>



Pour allez plus loin

1. Utilisez les fonctions `AcceptChanges` et `RejectChanges` interne au `DataSet` et Observez

Form1

Charger AcceptChange RejectChanges Enregistrer

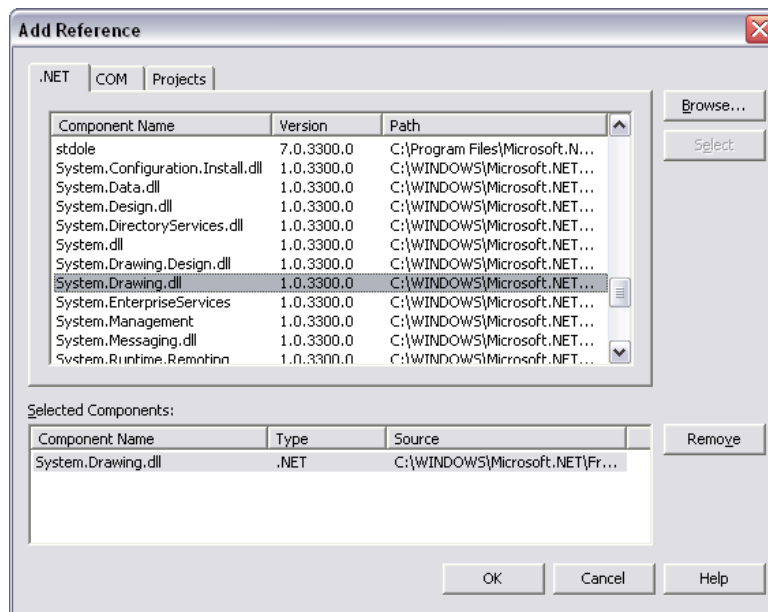
	ProductID	ProductName	SupplierID	CategoryID	QuantityPerU	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
▶	1	Chai	1	1	10 boxes x 20	18	39	0	10	<input type="checkbox"/>
	2	Changia	1	1	24 - 12 oz bot	19	17	40	25	<input type="checkbox"/>
	3	Aniseed Syru	1	2	12 - 550 ml b	10	13	70	25	<input type="checkbox"/>
	4	Chef Anton's	2	2	48 - 6 oz jars	22	53	0	0	<input type="checkbox"/>
	5	Chef Anton's	2	2	36 boxes	21.35	0	0	0	<input checked="" type="checkbox"/>
	6	Grandma's B	3	2	12 - 8 oz jars	25	120	0	25	<input type="checkbox"/>
	7	Uncle Bob's	3	7	12 - 1 lb pkgs	30	15	0	10	<input type="checkbox"/>
	8	Northwoods	3	2	12 - 12 oz jar	40	6	0	0	<input type="checkbox"/>
	9	Mishi Kobe Ni	4	6	18 - 500 g pk	97	29	0	0	<input checked="" type="checkbox"/>
	10	Ikura	4	8	12 - 200 ml ja	31	31	0	0	<input type="checkbox"/>
	11	Queso Cabral	5	4	1 kg pkg.	21	22	30	30	<input type="checkbox"/>
	12	Queso Manc	5	4	10 - 500 g pk	38	86	0	0	<input type="checkbox"/>

Note :

3.5 Transmettre des données binaires

Pour transmettre simplement des données binaires (ex: images) dans les services web, on peut utiliser l'encodage Base64 (comme pour les emails).

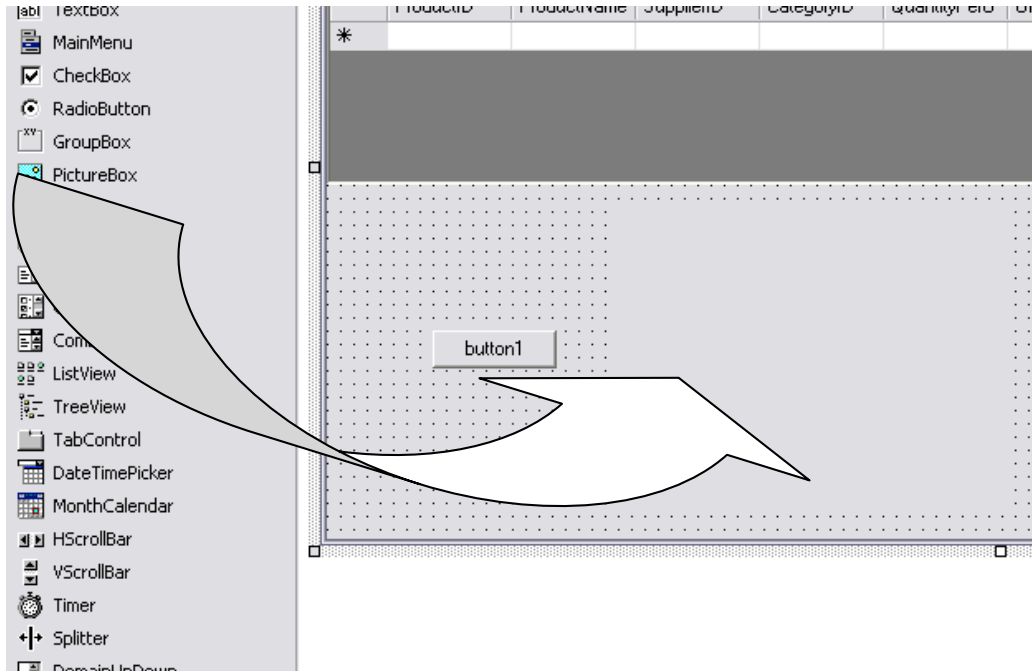
1. Le service web renvoie une image.
 - Ajouter une Reference sur System.Drawing
 - Dans l'explorateur de solution, pour le service web
 - Cliquez droit sur "Référence" > "Ajouter une référence"
 - Dans l'onglet ".Net", sélectionnez "System.Drawing.dll"
 - Cliquez sur le bouton "Sélectionner", puis OK



- Ajoutez la méthode suivante qui renvoie une image JPEG sous forme de string :

```
[WebMethod]
public string GetImage ()
{
    try
    {
        //Récupère le chemin physique de l'image
        string filename= Server.MapPath("/LabWebService/trafic.bmp");
        //Charge l'image en mémoire
        System.IO.MemoryStream ms=new System.IO.MemoryStream();
        System.Drawing.Bitmap bm=new System.Drawing.Bitmap(filename);
        //Transformation de l'image en Jpeg
        bm.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
        //Convert le flux binaire en flux text en base 64
        return Convert.ToString(ms.GetBuffer(),0,(int) ms.Length);
    }
    catch (Exception e)
    {
        return e.Message;
    }
}
```

2. Sur le client : récupérer l'image du service web et l'afficher
 - Ajouter un "Button" et un "PictureBox".



- Entrez le code suivant pour le déclenchement du bouton :

```
private void buttonImage_Click(object sender, System.EventArgs e)
{
    //Creation de l'objet Proxy Service Web
    localhost.Service1 sw=new localhost.Service1();
    //Création d'un espace mémoire réceptacle de l'image
    System.IO.MemoryStream ms=new System.IO.MemoryStream();
    //Réception de l'image et conversion de base 64 en binaire
    Byte[] ba=Convert.FromBase64String(sw.GetImage());
    //Ecriture dans le réceptacle mémoire
    ms.Write(ba,0,ba.Length);
    //Affichage de l'image
    pictureBox1.Image=new Bitmap(ms);
}
```

- Compilez et testez



N'oubliez pas de mettre à jour les références du Service Web pour que GetImage soit reconnu.

- Sélectionnez, sous l'onglet "Références Web", votre service web "localhost"
- Cliquez droit > "Mise à jour de la référence"

 Note :

4 Disponibilité

Dans ces exercices, vous allez apprendre à :

- Appeler un service web de façon asynchrone
- Mettre en cache les résultats des méthodes
- Avoir l'URL du service web dans un fichier de configuration
- Changer d'URL si la première ne répond pas
- Voir comment gérer des sessions

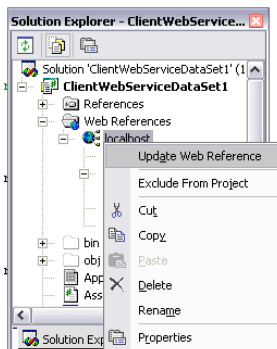
4.1 Client : Appel Asynchrone

Si le client est une application graphique, il ne faut pas bloquer pendant l'appel du Webservice.

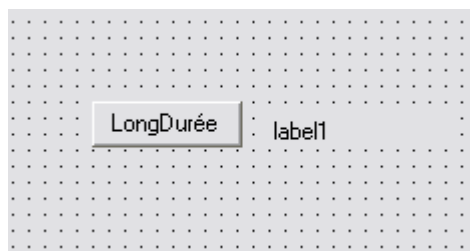
1. Réalisez une méthode de service web qui met du temps à répondre
 - Ajoutez la méthode suivante à votre service web

```
[WebMethod]
public string LongueDuree ()
{
    System.Threading.Thread.Sleep(10000);
    return System.DateTime.Now.ToString("HH:mm:ss");
}
```

- Enregistrez, Compilez.
2. Ajoutez à votre application cliente l'appel à cette méthode
 - On va d'abord mettre à jour la référence au service web
 - Sélectionnez, sous l'onglet "Références Web", votre service web "localhost"
 - Cliquez droit > "Mise à jour de la référence"



- Ajoutez un bouton dont le texte est "Appel Longue Durée" à votre formulaire
- Ajoutez un label qui contiendra le texte de retour



- Cliquez sur le bouton pour éditer la fonction qu'il déclenche

- Appelez la méthode "LongueDuree" de votre service web :

```
private void LongDuree_Click(object sender, System.EventArgs e)
{
    //Creation de l'objet Proxy Service Web
    localhost.Service1 sw=new localhost.Service1();
    //Mise en attente de l'utilisateur
    label1.Text="en cours ...";
    //Appel du Web Service Long
    label1.Text=sw.LongueDuree();
}
```

3. Enregistrez, Compilez, Testez



Remarquez que quand vous appelez la méthode LongueDuree, votre application ne peut plus bouger et son affichage ne se réactualise pas.

4. Ajoutez un bouton "Appel asynchrone", son label associé, et le code source correspondant au clique sur le bouton.
Explication : Plutôt que d'appeler directement la méthode, on passe par une autre méthode (qui a été définie automatiquement lors de la création de la référence web), qui est préfixée par "Begin" et à laquelle on donne l'adresse d'une méthode de callback et l'objet de service web en cours. Cette méthode "Begin" va lancer un nouveau Thread puis rend la main. C'est dans ce nouveau Thread qu'est réalisé l'appel au webservice puis l'appel à la méthode de callback.
en C#

```
private void Asynchrone_Click(object sender, System.EventArgs e)
{
    //Creation de l'objet Proxy Service Web
    localhost.Service1 sw=new localhost.Service1();
    //Mise en attente de l'utilisateur
    label1.Text="en cours ...";
    //Lancement de l'appel au Service Web dans un autre thread
    sw.BeginLongueDuree(new AsyncCallback(LongueDureeAFini), sw);
}
```

5. Spécifiez la fonction de callback. Celle-ci sera appelée lorsque le service web aura répondu.

```
public void LongueDureeAFini(IAsyncResult state)
{
    //Récupération de l'objet Proxy passé en paramètre
    localhost.Service1 sw=(localhost.Service1) state.AsyncState;
    //Affichage du résultat (ici la fonction coté serveur est fini
    label1.Text=sw.EndLongueDuree(state);
}
```

6. Enregistrez, Compilez, Testez



Remarquez que quand vous utilisez l'appel asynchrone, votre application est toujours active.

4.2 Serveur : Mise en cache

Lorsqu'une méthode nécessite beaucoup de ressources pour fonctionner mais que son résultat n'a pas besoin d'être recalculé sur une période donnée, on peut très facilement le mettre en cache en utilisant la propriété "CacheDuration" de l'attribut "WebMethod". Cette propriété définit la durée de vie du cache.

Mettez un cache de 30 secondes sur la méthode "LongueDuree()".

```
[WebMethod(CacheDuration=30)]
public string LongueDuree()
{
    System.Threading.Thread.Sleep(10000);
    return System.DateTime.Now.ToString("HH:mm:ss");
}
```

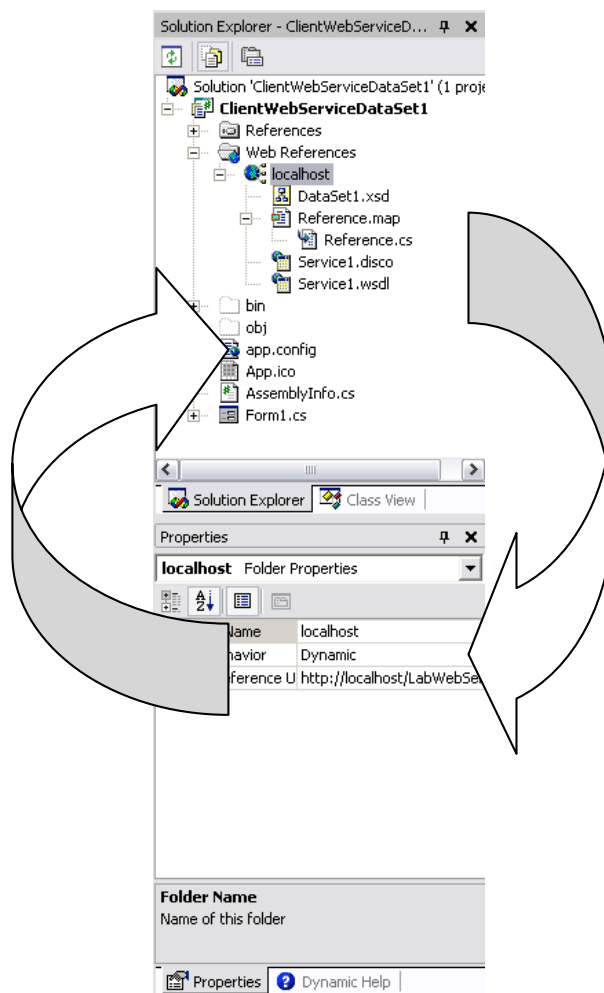
Enregistrez, Compilez puis Testez l'appel asynchrone depuis le client

4.3 Client : URL d'accès en config

Le proxy contient en dur l'URL d'accès au webservice telle que définie dans le fichier WSDL.

On peut néanmoins utiliser une autre URL définie dans un fichier de config.

- Sélectionnez "Explorateur de solutions" > "clientsw" > "Références Web" > "localhost"
- Dans "Propriétés" > "URL Behavior" : sélectionnez "Dynamic"



- Un fichier XML "app.config" est créé, il contient une paire clé/valeur qui définit l'URL d'accès au service web :

```
<configuration>
```

```
<appSettings>
  <add key="ClientWebServiceDataSet1.localhost.Service1"
    value="http://localhost/LabWebService/WebService1/Service1.asmx"/>
</appSettings>
</configuration>
```



Dans le fichier source "Reference.cs", dans le constructeur, la définition de l'URL d'accès au webservice se fait maintenant en allant la rechercher dans le fichier de configuration de l'application.

```
public Service1()
{
    string urlSetting =
System.Configuration.ConfigurationSettings.AppSettings["ClientWebServiceDataSet1.localhost.Service1"];
    if ((urlSetting != null)){
        this.Url = string.Concat(urlSetting, "");
    }
    else {
        this.Url = "http://localhost/LabWebService/WebService1/Service1.asmx";
    }
}
```



Lors du déploiement de l'application cliente, vous pourrez donc modifier manuellement ce fichier "App.config" pour spécifier une autre URL d'accès au service web.

4.4 Client : Fail-Over

On définit une liste d'URLs d'accès au service web dans le fichier App.config. Lorsqu'une requête ne fonctionne pas (on prend garde de récupérer l'exception avec un try/catch), on utilise l'URL suivante dans le fichier de configuration jusqu'à trouver celle qui fonctionne.

Diminuez la durée d'attente avant erreur (Timeout) : propriété "Timeout" de la classe proxy

4.5 Gestion d'une session

Il est possible de gérer les sessions sur les services web, mais cela n'est pas conseillé car il faut plutôt concevoir les services web en terme de couplage faible.

On active la session au niveau de chaque méthode en utilisant la propriété "EnableSession"

```
WebMethod(EnableSession=True|False)
```

On peut alors utiliser l'objet "Session" comme dans une page ASPX.

Ce mécanisme de session utilise obligatoirement un cookie.

L'application cliente doit donc pouvoir être capable de le reconnaître, le conserver, et le renvoyer à chaque appel de méthode.



L'utilisation ou pas des sessions n'est pas dans le WSDL !

5 Sécurisation d'un service Web

Dans ces exercices, vous allez apprendre à :

-
- Crypter la communication en SSL
 - Authentifier le client en mode Basic et Intégré Windows
 - Définir des autorisations différentes suivant le client authentifié
 - Désactiver l'accès au service web depuis un navigateur web
-

5.1 Crypter la communication par SSL

1. Installer un certificat SSL sur le serveur
2. Configurer le répertoire virtuel en SSL uniquement
3. Installer un certificat sur le client (optionnel)
4. Configurer l'application cliente

Le document WSDL, qui est renvoyé directement par la page "asmx?WSDL" du service web, définit par défaut l'URL d'accès au service en mode HTTP.

Lorsque l'on définit une référence web en utilisant ce document, VisualStudio.Net utilise cette URL dans la création du fichier source du proxy client.

Pour mettre cette URL en HTTPS, vous pouvez

- soit utiliser une URL en HTTPS lors de la création de la référence web
- soit modifier directement l'URL dans le fichier source du proxy (Reference.xx)

5.2 Authentification

Il existe plusieurs modes d'authentification :

- Basic sur SSL : en Basic sur HTTP, le mot de passe n'est pas crypté, c'est pourquoi il faut obligatoirement être en mode HTTPS
- Certificat SSL : il faut relier, au niveau d'IIS, chaque certificat à un compte NT
- Windows (uniquement IE) : NTLM ou Kerberos ; authentification cryptée
- Digest (uniquement IE) ; authentification cryptée



Pour connaître les modes d'authentification possibles sur votre système, vous pouvez utiliser le code suivant dans une application Windows contenant un ListBox:

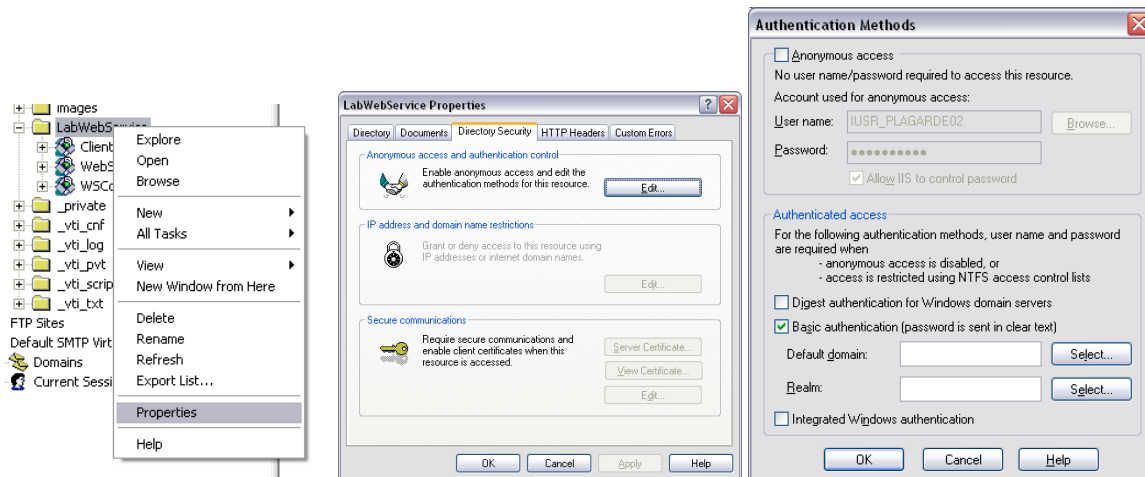
```
private void button1_Click(object sender, System.EventArgs e)
{
    System.Net.IAuthenticationModule am;
    IEnumerator ar;
    ar=System.Net.AuthenticationManager.RegisteredModules;
    ar.Reset();
    while (ar.MoveNext())
    {
        am = (System.Net.IAuthenticationModule)ar.Current;
        listBox1.Items.Add(am.AuthenticationType);
    }
}
```


5.2.1 Authentication Basic

L'authentification "Basic" ne peut fonctionner avec un mot de passe vide, il faut donc tout d'abord mettre un mot de passe "ms" pour l'utilisateur "ms" : il faut passer par le panneau de configuration.

1. Création du service web

- Configurez IIS : il faut que l'accès au répertoire virtuel ne puisse se faire qu'avec une authentification "Basic"



- Créez une classe qui contiendra les informations d'authentification

```
public class MyIdentity
{
    public string AuthenticationType;
    public bool IsAuthenticated;
    public string Name;
}
```

- Créez un service web qui renvoie les informations d'authentification

```
[WebMethod]
public MyIdentity GetAuth()
{
    MyIdentity id=new MyIdentity();
    id.AuthenticationType=User.Identity.AuthenticationType;
    id.IsAuthenticated=User.Identity.IsAuthenticated;
    id.Name=User.Identity.Name;
    return id;
}
```

- Enregistrez, Compilez, Testez

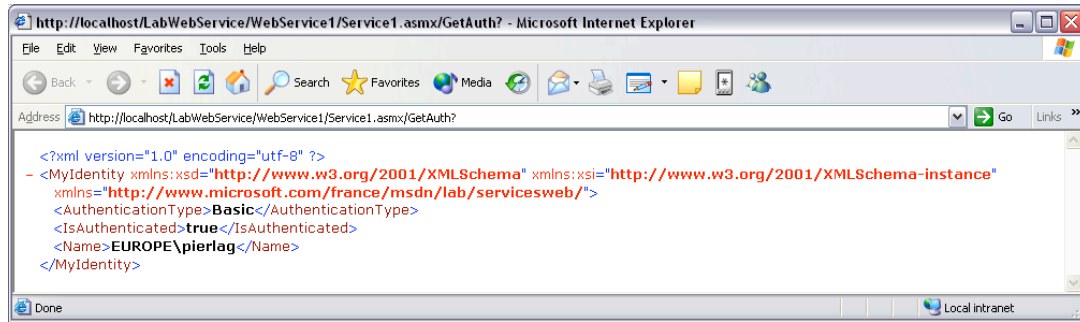


Lors de l'accès à la page du service web, votre navigateur vous demande de rentrer un login et mot de passe. Entrez celui de votre compte actuel : login = "ms", mot de passe = "ms"



- Appelez la méthode "GetAuth"

Elle vous renvoie les informations d'authentification suivantes :



2. Créer une application cliente du service web.

- Créez une nouvelle application Windows contenant un bouton pour l'appel au service web et trois labels qui contiendront les trois données d'authentification.
- Créez la référence web sur le service
- Utilisez le namespace System.Net

```
using System.Net;
```

- Mettez le code suivant pour le déclenchement du bouton
On ajoute une authentification en mode "Basic" avec les bons login/passwd à une liste de "Credentials" que l'on affecte au proxy du service web.

```
private void button1_Click(object sender, System.EventArgs e)
{
    localhost.Service1 sw=new localhost.Service1();
    CredentialCache cc=new CredentialCache();
    cc.Add(new Uri(sw.Url), "Basic", new NetworkCredential("ms","ms"));
    sw.Credentials=cc;
    localhost.MyIdentity mi=sw.GetAuth();
    label1.Text=mi.Name;
    label2.Text=mi.AuthenticationType;
    label3.Text=mi.IsAuthenticated.ToString();
}
```

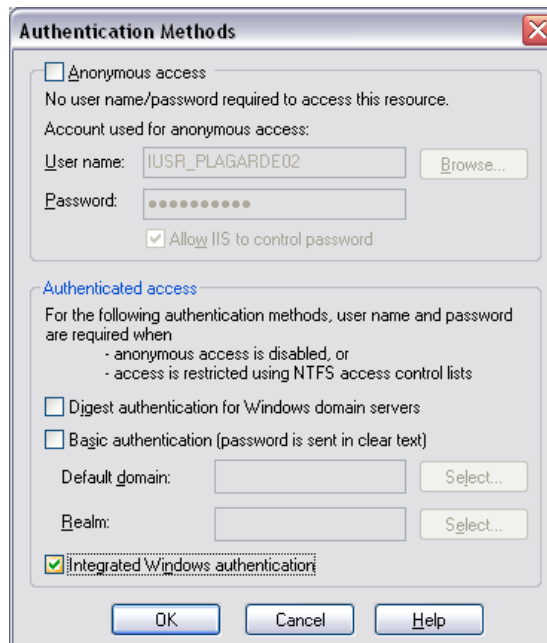
 Note :

5.2.2 Authentication Windows

On utilise l'authentification intégrée windows et on prend le compte NT de l'utilisateur de l'application cliente.

1. Le service web

- Configurez IIS : il faut que l'accès au répertoire virtuel ne puisse se faire qu'avec une authentification intégrée Windows



- Configurer le Service Web

Dans le fichier Web.config , vérifiez que vous avez :

```
<authentication mode="Windows"/>
```

2. Le client

```
private void button2_Click(object sender, System.EventArgs e)
{
    localhost.Service1 sw=new localhost.Service1();
    sw.Credentials=CredentialCache.DefaultCredentials;
    localhost.MyIdentity mi=sw.GetAuth();
    label1.Text=mi.Name;
    label2.Text=mi.AuthenticationType;
    label3.Text=mi.IsAuthenticated.ToString();
}
```

 Note :

5.3 Désactivation des pages HTML auto-générés

Lors du déploiement sur un serveur de production, il est conseillé de désactiver les aides à la programmation.

Il faut modifier la section « webservice » des fichiers de configuration, soit celui global du serveur (Machine.config), soit celui local de l'application (Web.config).

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  ...
  <system.web>
    ...
    <webServices>
      <protocols>
        <add name="HttpSoap"/>
        <!--
        <add name="HttpPost"/>
        <add name="HttpGet"/>
        <add name="Documentation"/>
        -->
      </protocols>
    
```

6 Autres sujets

Voici d'autres sujets que vous pouvez explorer.

6.1 *Récupération de l'existant*

- Transformer un composant .Net en Webservice
- Transformer un composant COM (ActiveX) en Webservice
- Transformer une DLL en Webservice



Il n'y a là rien de particulier par rapport aux services web, on utilise juste les capacités du Framework .Net

- Transformation des données d'une page HTML en Webservice : Comment récupérer le contenu d'une page web à l'aide d'expressions régulières. C'est un moyen d'accéder à des données non structurées pour les servir en service web.

6.2 *Utilisation depuis OfficeXP*

A l'aide de l'Office XP WebServices Toolkit, créez une macro dans Office XP qui utilise un service web pour effectuer un calcul.

6.3 *Configuration des vues HTML*

On peut modifier l'apparence des pages HTML générées automatiquement par le moteur des pages ASMX. Cette modification des pages HTML vaut pour tout le serveur.

A bientôt et merci d'être venus...

Vous pouvez vous inscrire aux prochains séminaires MSDN et aux labs .NET:

En vous rendant à l'adresse suivante:

<http://www.microsoft.com/france/events/espace.Asp?space=MSDN>

Par courrier électronique :

Inscriptions-microsoft@ctm.fr

Par téléphone

0825 827 829 -0-1188#

Par télécopie

01 41 16 47 01

The Microsoft logo is displayed in white, bold, italicized sans-serif font against a black rectangular background. The word "Microsoft" is followed by a registered trademark symbol (®).