



ASP.NET 4.5 niveau 1

développement web



Benoît CHAUVENT

2

- 10 ans d'expérience sur la plateforme .NET
- Ingénierie web et gestion de projets
 - Eurosport.com : Mise en place de la communauté et du player Eurosport (VOD et Streaming)
 - Externis : Gestion de projets B2B en Application Service Provider pour les process logistique et animation de grands comptes (Kraft, Unilever, Nestlé...)
- Formation et conseil
 - Gestion de projet (Méthodes agiles, Scrum)
 - Conception (UML, concepts objet)
 - Programmation .NET (C#, ASP.NET)
- contact@benoitchauvet.com

Table des matières

1.	Rappels des concepts web et technologies .NET	4
2.	Fonctionnement des pages ASP.NET	16
3.	Contrôles serveur HTML	62
4.	Contrôles serveur web	74
5.	Conception de la structure d'un site web	86
6.	Contrôle des sources de données	107
7.	Gestion de la sécurité	143
8.	Configuration et déploiement	153
9.	Utilisation des services web	162

Rappels des concepts web et technologies .NET

Serveur Web

5

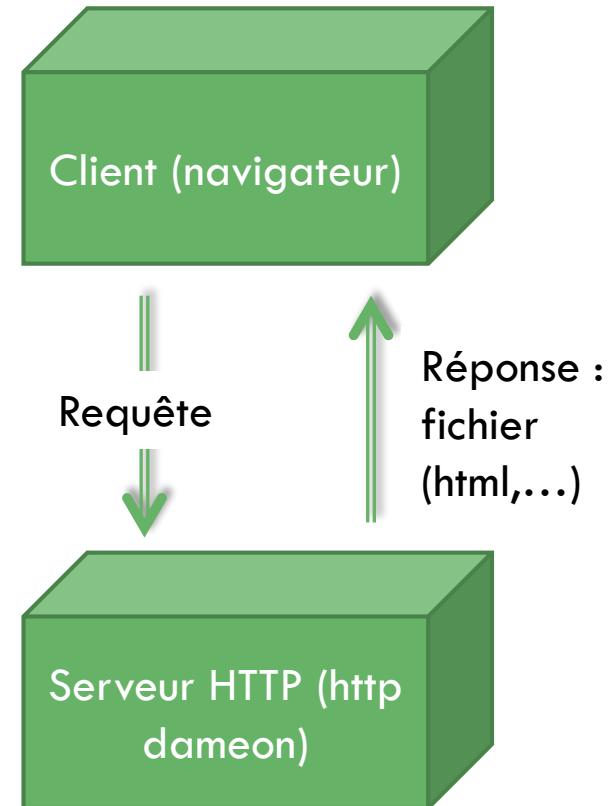
- Serveur web : protocole de communication client-serveur HTTP (HyperText Transfer Protocol)
- Requête / Réponse
- Quelques serveurs web :
 - Apache HTTP Server
 - Apache Tomcat (évolution d'apache pour J2EE)
 - IIS (Internet Information Services)
 - ...

Sites statiques

6

□ Sites statiques

- Serveur web = simple serveur de fichiers
- Simple
- Robuste
- Interactions limitées aux liens hypertexte
- Client essentiellement passif



Sites dynamiques

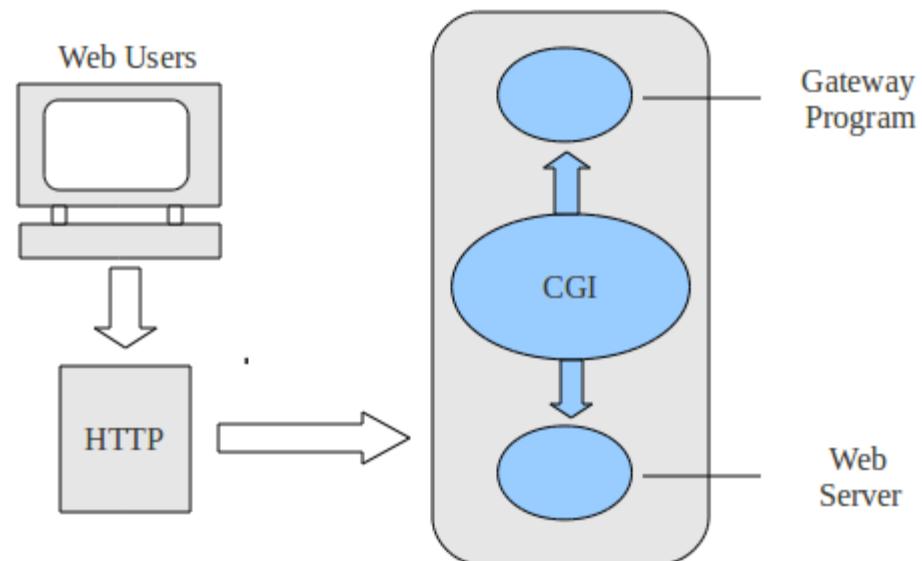
7

□ Sites dynamiques

- Pages web générées à la demande
- Traitements côté serveur
- Liaisons avec bases de données

□ Quelques exemples :

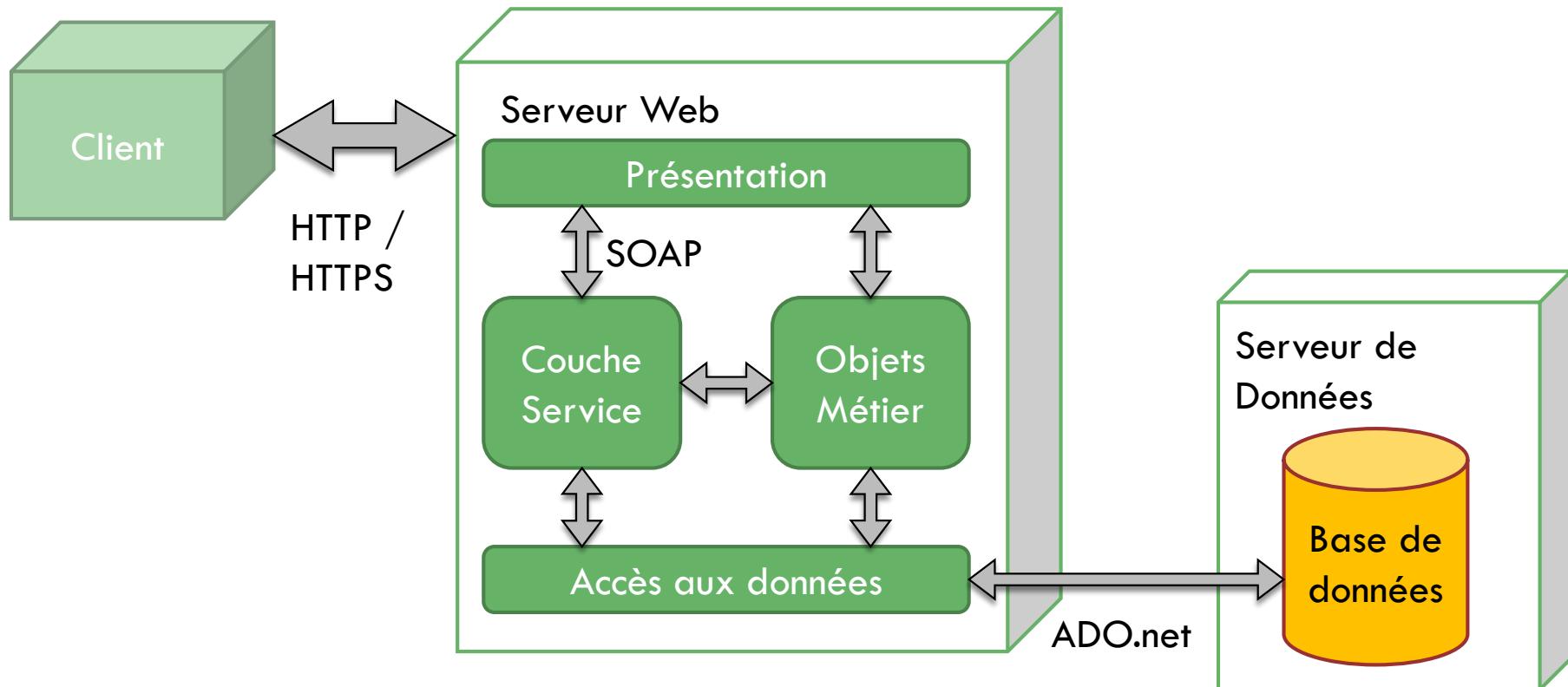
- CGI (Common Gateway Interface)
- Php
- JSP/Servlets
- Asp
- Asp.net...



Architecture d'une application web

8

Exemple d'architecture 3 tiers (Présentation / Métier / Données)



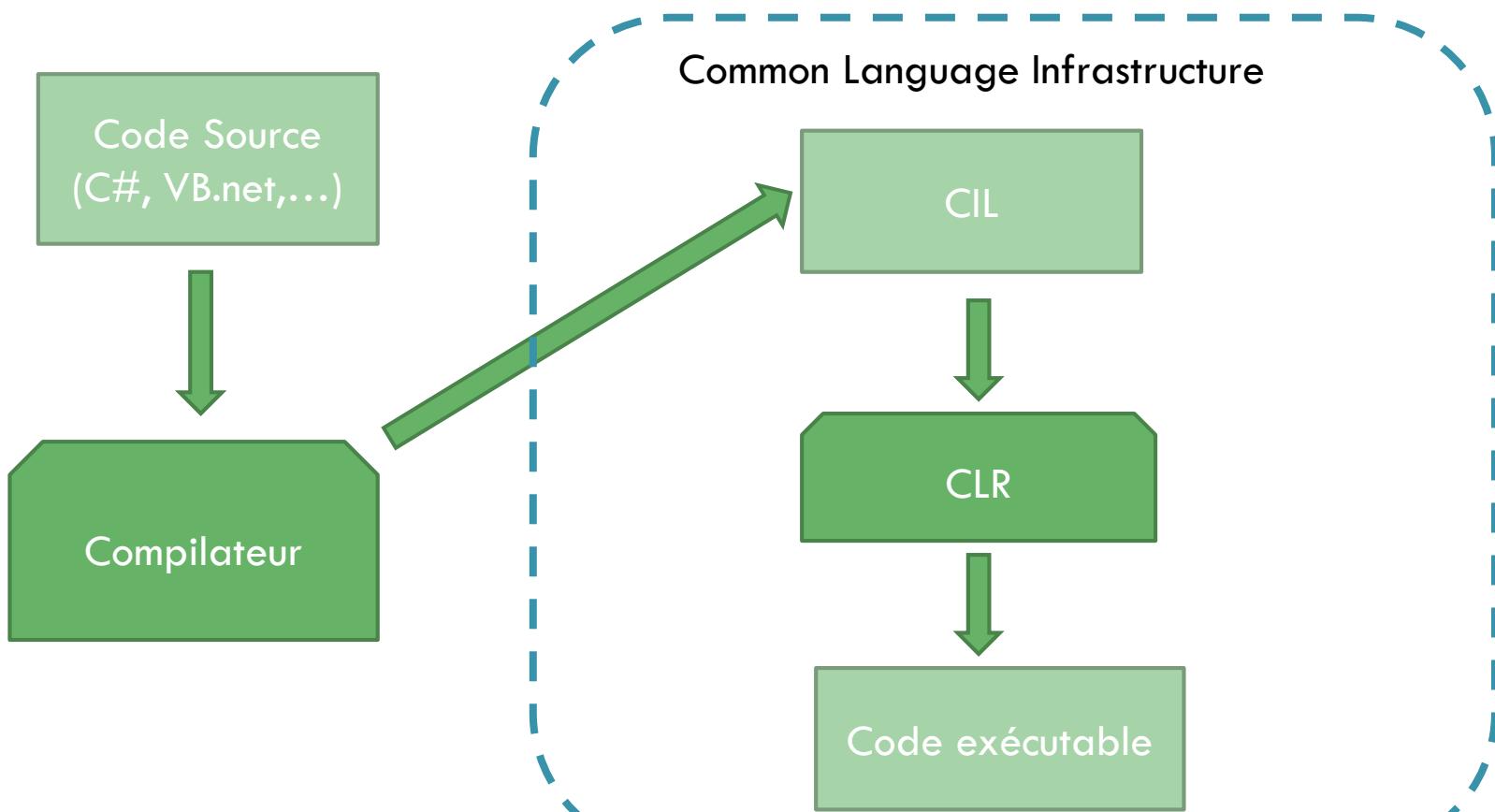
Plateforme .NET

9

- Indépendant du système (Windows, Unix, ...)
- Basé sur la norme Common Language Infrastructure (CLI) : indépendante du langage
 - Plusieurs langages disponibles : C#, VB.net, F#...
 - Common Language Specification (CLS)
 - Common Type System (CTS)
 - Génération de code intermédiaire : Common Intermediate Language (CIL, anciennement MSIL)
- Exécution du CIL par une machine virtuelle : Common Language Runtime (CLR)
- Compilation Just In Time (JIT) par la CLR, à l'exécution (Runtime)

Plateforme .NET

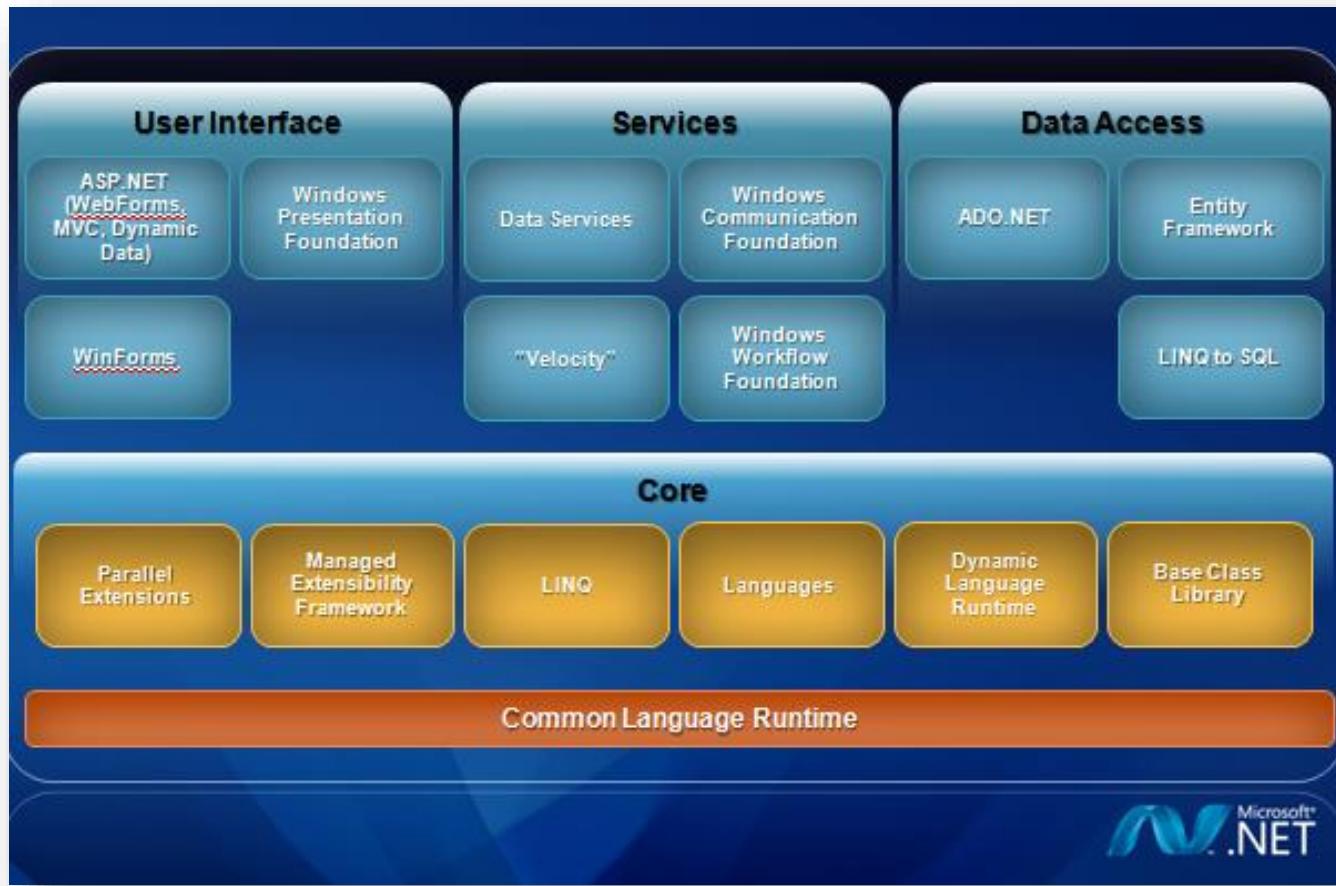
10



Plateforme .NET

11

Le Framework .net



Application Web en .NET.

12

□ ASP.net

- Couche de présentation web (génération dynamique de pages HTML)
- Traitement des requêtes, génération des pages basées sur le framework .net
- Framework de développement web, pour créer des applications web et des services web
- Bibliothèques d'outils et de composants réutilisables et personnalisables (Gestion de session, Webforms, Webcontrols...)

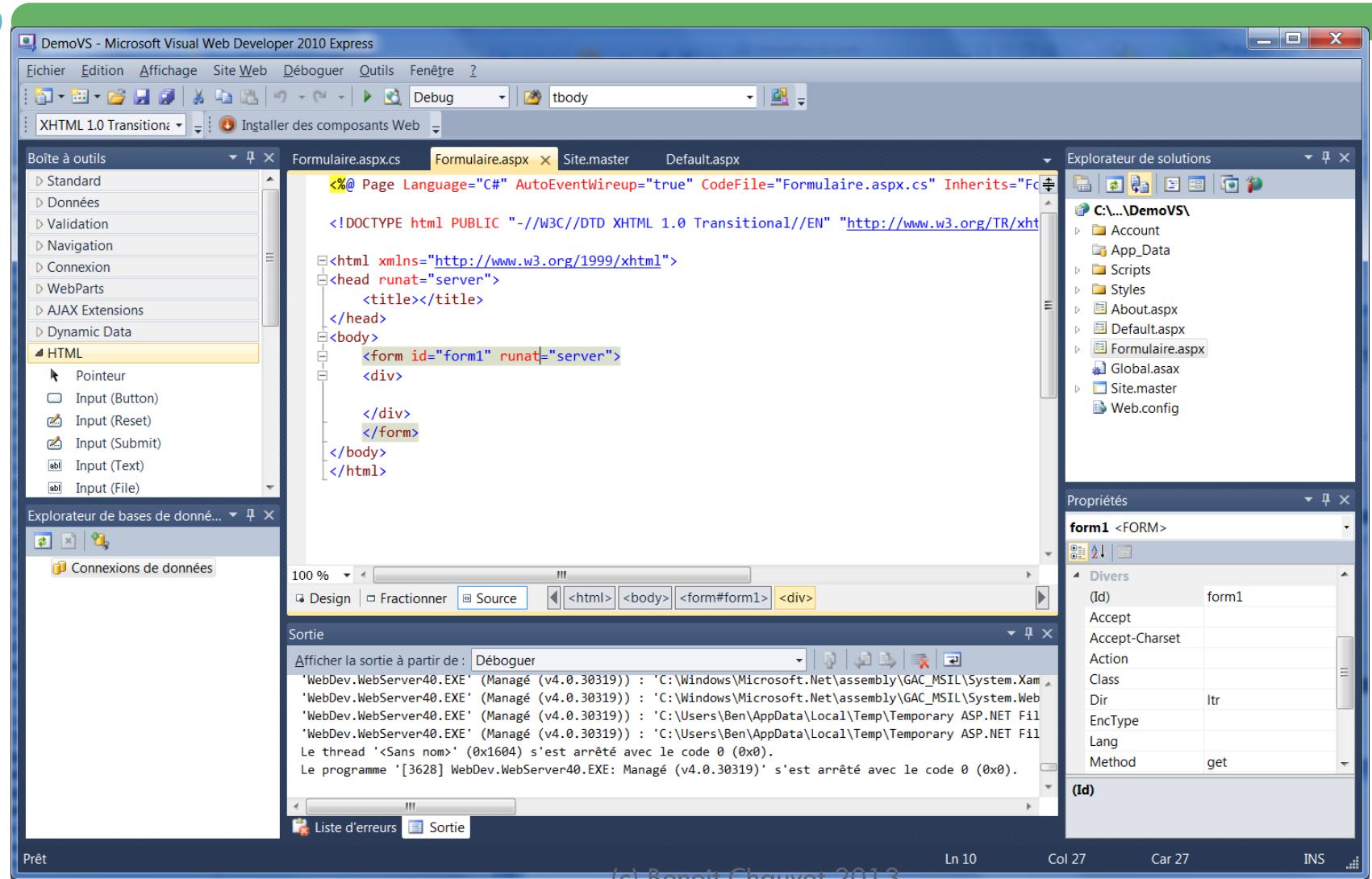
Visual Studio 2010

13

- IDE (Integrated Development Environment)
- Editeur de code avancé
 - Coloration syntaxique
 - Détection des erreurs à la volée
 - Intellisense
 - Outils de refactoring, génération de code
 - Editeur Wysiwyg pour les interfaces utilisateurs
- Outils de débogage (pile des appels, espion, points d'arrêt...)
- Explorateur de serveurs intégré (Bases de données)
- Intégration de tests unitaires, tests d'interface
- Gestionnaire d'extensions
- Outils spécifiques ASP.net
 - Serveur web intégré
 - Outils de débogage javascript
 - Gestionnaire de publication (client FTP intégré)

Visual Studio 2010

14



Travaux pratiques

15

- **Création d'un site web asp.net sous visual studio**
- **Création d'un formulaire simple (Hello World)**
- **Personnalisation de l'environnement (fenêtres, positionnement...)**
- **Lancement en mode debug**

Fonctionnement des pages ASP. NET

Principes des pages dynamiques

17

- Page dynamique : générée à la demande
- Contenu variable
 - Paramètres contenus dans la requête
 - Date / Heure
 - Contenu de la base de données de référence...
 - ...
- Mécanisme :
 - Envoi d'une requête au serveur http
 - Transmission au logiciel (interpréteur) correspondant à la requête (php, asp.net, jsp...)
 - Génération du contenu
 - Envoi de la réponse sous forme de page statique
- Problématique de l'indexation par les moteurs de recherche (« deep web » ou « web profond »)

Script côté client

18

- « Client-side scripting »
- Programmes s'exécutant sur le navigateur web du client.
- Concept de RIA (Rich Internet Application)
- Permet l'interactivité d'une page web
 - Langages de script : Javascript, DHTML, VBScript, Ajax...
 - Plugins compilés : applets java, Flash, ActiveX, Silverlight...

Script côté client

19

□ Exemple :

```
<html>
<head>
    <title></title>

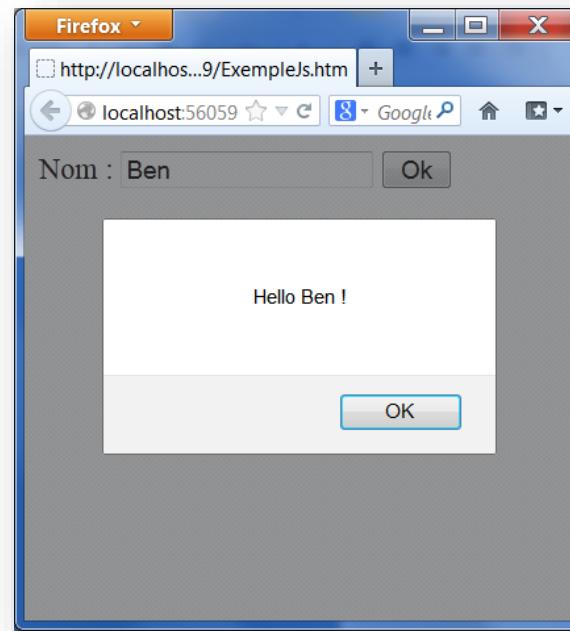
    <!-- Déclaration d'une fonction javascript -->
    <script language="javascript">
        function sayHello(nom) {
            alert("Hello " + nom + " !");
        }
    </script>

</head>
<body>

    <!-- zone de texte Nom -->
    Nom : <input id="Nom" type="text" />

    <!-- Bouton ok, avec déclenchement d'un appel javascript sur clic -->
    <input id="Button1" type="button" value="Ok" onclick="sayHello(Nom.value)"/>

</body>
</html>
```



Script côté serveur

20

- « Server-side scripting »
- Langage de script serveur (php, asp, jsp...) intégré au sein du code HTML
- Exécuté par le serveur à la réception de la requête (via interpréteur correspondant)
- Code non visible côté client : le contenu renvoyé est généralement du HTML généré
- Balises spécifiques
 - PHP : <?php ... ?>
 - ASP, JSP : <% ... %>

Script côté serveur

21

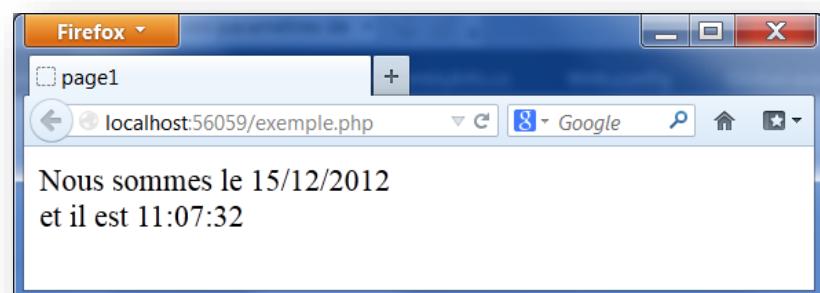
□ Exemple :

Exemple.php :

```
<html>
<head>
<title>page1</title>
</head>
<body>
<?php
$date = date("d/m/Y");
$heure = date("H:i:s");
echo("Nous sommes le $date <br>et il est $heure");
?>
</body>
</html>
```

HTML généré :

```
<html>
<head>
<title>page1</title>
</head>
<body>
Nous sommes le 15/12/2012 <br>et il est 11:07:32
</body>
</html>
```



Méthodes HTTP GET et POST

22

- 2 méthodes d'envoi de requête : GET ou POST
- Utilisation :
 - GET : récupérer des données
 - POST : envoyer des données en vue d'un traitement côté serveur

	GET	POST
Visibilité	Données visibles dans l'URL	Données non visibles dans l'URL
Boutons précédent / actualiser	Sans risque	Données re-soumises (avertissement du navigateur)
Cache	Mis en cache	Pas de mise en cache
Encodage	Url-encoding	Au choix
Historique	Enregistré	Non stocké
Restrictions de taille	Max 2048 caractères (url complète)	Pas de limite
Type de données	ASCII uniquement	Pas de restrictions (même binaire)
Sécurité	Aucune !	Un peu plus sécurisé
Marque pages	Possibles	Pas possibles

Méthode GET

23

□ Passage de paramètres dans l'url :

<url de la page>?<nomChamp1>=<valChamp1> [& <nomChamp2>=<valChamp2>]*

□ Exemple :

http://localhost/demoGet?nom=Admin&sujet=sport&saison=hiver

http://www.youtube.com/watch?feature=player_embedded&v=12PWq22E9ZQ

□ Récupération des paramètres côté serveur (ASP.net) :

```
string nom = Request.QueryString["nom"];
```



Méthode POST

24

- Données non visibles dans l'url
- Transmises dans le corps du message http :

Exemple :

POST /test/demo_form.asp HTTP/1.1

Host: w3schools.com

name1=value1&name2=value2

- Utilisé pour la transmission de formulaires
- Encodage variable (binaire : multipart form-data)

```
<html>
  <head><title></title></head>
  <body>
    <form id="form1" method="post" enctype="application/x-www-form-urlencoded">
      <input id="Nom" type="text" />
      <input id="Prenom" type="password" />
      <input id="Submit1" type="submit" value="submit" />
    </form>
  </body>
</html>
```

Principe des WebForms

25

- ASP.net introduit la notion de **code behind** : le code côté serveur est séparé du code html, dans un fichier distinct.
- Notion de classe partielle : Séparation du code behind en 2 fichiers
 - code généré
 - code utilisateur
- Une page ASP.net (Webform) est donc constituée de 3 fichiers :
 - Balises html (« concepteur de vues ») : *maPage.aspx*
 - Code behind généré : *maPage.aspx.designer.cs*
 - Code behind utilisateur : *maPage.aspx.cs*
- Les modifications du *.aspx* entraînent des modifs dans le *.aspx.designer.cs* (déclarations, événements...)
- Chaque page asp.net est une classe, dérivée de *System.Web.UI.Page*

Structure d'un Webform

26

maPage.aspx

```
<!-- Directive de page -->
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="maPage.aspx.cs" Inherits="WebApplication2.maPage" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <!-- TextBox ajoutée via la boîte à outils -->
            <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        </div>
    </form>
</body>
</html>
```

maPage.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace WebApplication2
{
    // Code Behind "utilisateur", classe PARTIELLE, dérivée de System.Web.UI.Page :
    public partial class maPage : System.Web.UI.Page
    {
        // Méthode de chargement (pour ajouter le code lié au chargement de la page)
        protected void Page_Load(object sender, EventArgs e)
        {
        }
    }
}
```

(c) Benoit Chauvet 2013

Structure d'un Webform

27

maPage.aspx.designer.cs

```
//-----  
// <généré automatiquement>  
// Ce code a été généré par un outil.  
//  
// Les modifications apportées à ce fichier peuvent provoquer un comportement incorrect et seront perdues si  
// le code est régénéré.  
// </généré automatiquement>  
//-----  
  
namespace WebApplication2 {  
  
    // Code behind "designer" : généré automatiquement, classe PARTIELLE  
    public partial class maPage {  
  
        /// <summary>  
        /// Contrôle form1.  
        /// </summary>  
        /// <remarks>  
        /// Champ généré automatiquement.  
        /// Pour modifier, déplacez la déclaration de champ du fichier de concepteur dans le fichier code-behind.  
        /// </remarks>  
        protected global::System.Web.UI.HtmlControls.HtmlForm form1;  
  
        /// <summary>  
        /// Contrôle TextBox1.  
        /// </summary>  
        /// <remarks>  
        /// Champ généré automatiquement.  
        /// Pour modifier, déplacez la déclaration de champ du fichier de concepteur dans le fichier code-behind.  
        /// </remarks>  
        protected global::System.Web.UI.WebControls.TextBox TextBox1;  
    }  
}
```

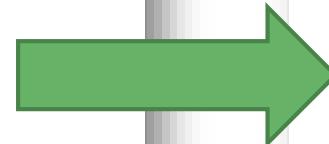
Le code imbriqué

28

- Possibilité de mettre du C# dans un fichier .aspx, sur le modèle des anciennes pages dynamiques.

```
<ul>
  <%
    string[] saisons = { "printemps", "été", "automne", "hiver" };

    foreach (string saison in saisons)
    {
      %>
      <li>
        <%
        Response.Write(saison);
        %>
      </li>
    <%} %>
</ul>
```



```
<ul>
  <li> printemps
  </li>
  <li> été
  </li>
  <li> automne
  </li>
  <li> hiver
  </li>
</ul>
```

Le code en ligne (inline)

29

- Possibilité de définir des traitements coté serveur dans le fichier aspx : <script runat=< server >>

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="codeInline.aspx.cs" Inherits="WebApplication2.codeInline" %>

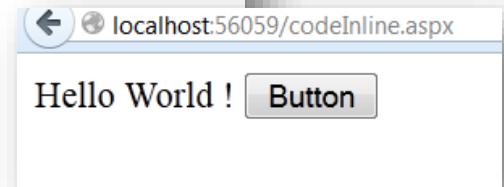
<script runat="server">

    void afficherMessage(object sender, EventArgs e)
    {
        Label1.Text = "Hello World !";
    }

</script>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
            <asp:Button ID="Button1" runat="server" Text="Button" OnClick="afficherMessage"/>
        </div>
    </form>
</body>
</html>
```



Les scriptlets

30

- ❑ Extraits de code dans une page aspx
- ❑ 4 types de scriptlets :

Scriptlet	Définition	Exemple
<code><% instructions %></code>	Instructions exécutées séquentiellement, dans l'ordre du code HTML	<code><% string[] saisons = { "printemps", "été", "automne", "hiver" }; foreach (string saison in saisons) { %> <% Response.Write(saison); %> <% } %></code>
<code><%= expression %></code>	Expression évaluée lors du rendu de la page	<code><%= DateTime.Now %> Équivaut à <% Response.Write(DateTime.Now);%></code>
<code><%# expression %></code>	Expression évaluée lors du DataBinding (cf ADO.net)	<code><ItemTemplate> <%# DataBinder.Eval(Container.DataItem, "Adresse") %></ItemTemplate></code>
<code><%\$ expression %></code>	Expression analysée à la compilation de la page, évaluée à chaque requête	<code><asp:Label ID="lbl1" runat="server" text=<%\$ ConnectionStrings:cx_annuaire %> /></code>

Utilisation des contrôles serveur

31

- Les contrôles serveur : éléments accessibles depuis le code behind.
- Déclaration générée dans le .aspx.designer.cs lors de l'ajout dans une page aspx.
- Définis par l'attribut **runat="server"**
- Référencés dans le code behind par leur ID
- Contrôles serveur HTML ou asp.net (web)

```
aspx    <input id="Text1" type="text" runat="server" />
        <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
```

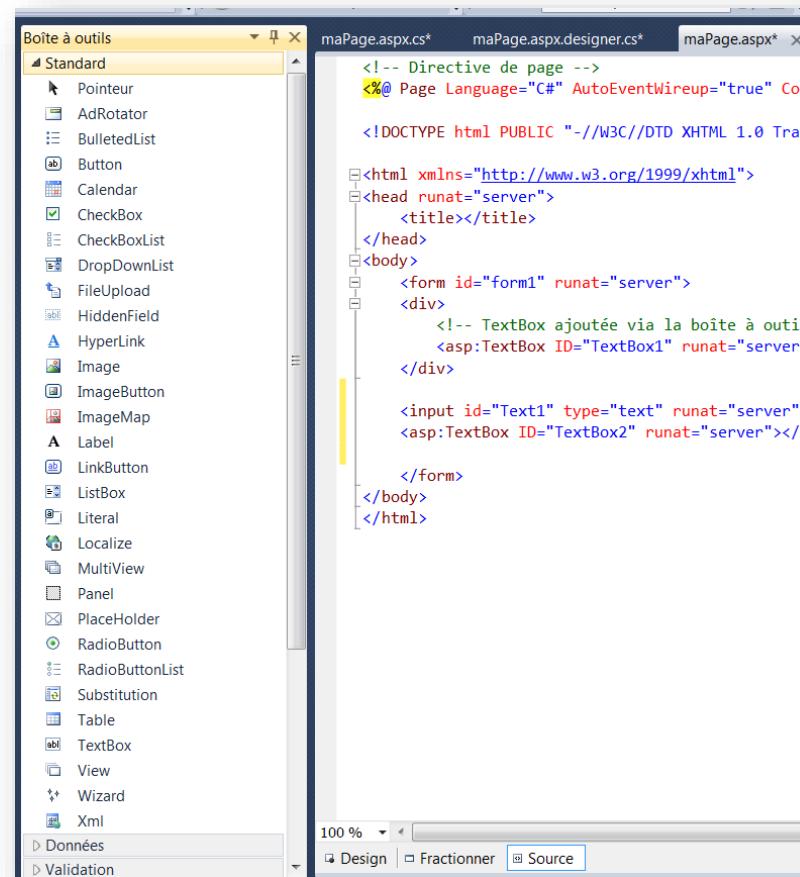
```
aspx.designer.cs    protected global::System.Web.UI.HtmlControls.HtmlInputText Text1;
                    protected global::System.Web.UI.WebControls.TextBox TextBox2;
```

```
aspx.cs    Text1.Value = "Hello";
            TextBox2.Text = "World";
```

Insertion de contrôles

32

- Via la boîte à outils Visual Studio
 - Glisser/déplacer vers la fenêtre source ou design
 - Génération automatique du code behind de la déclaration



The screenshot shows the Visual Studio IDE interface. On the left, the 'Boîte à outils' (Toolbox) is open, displaying a list of standard ASP.NET controls. On the right, the code editor window for 'maPage.aspx*' is open, showing the generated ASPX page code. The code includes directives, the DOCTYPE declaration, and the page structure with a form and two text box controls. The 'Source' tab is selected at the bottom of the code editor.

```
<!-- Directive de page -->
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="maPage.aspx.cs" Inherits="maPage" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/1999/xhtml1">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title></title>
  </head>
  <body>
    <form id="form1" runat="server">
      <div>
        <!-- TextBox ajoutée via la boîte à outils -->
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
      </div>
      <input id="Text1" type="text" runat="server" />
      <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
    </form>
  </body>
</html>
```

Gestion de la persistance

33

- HTTP : protocole déconnecté >> chaque requête est indépendante des autres, nativement il n'y a pas de moyen de faire le lien entre plusieurs requêtes.
- Plusieurs mécanismes pour répondre aux besoins de persistance :
 - Postback
 - Persistance d'état :
 - Champs cachés
 - Viewstate
 - QueryString et URI
 - Persistance des données :
 - Cookies
 - Objet Session
 - Objet Cache
 - Objet Application

Le Postback

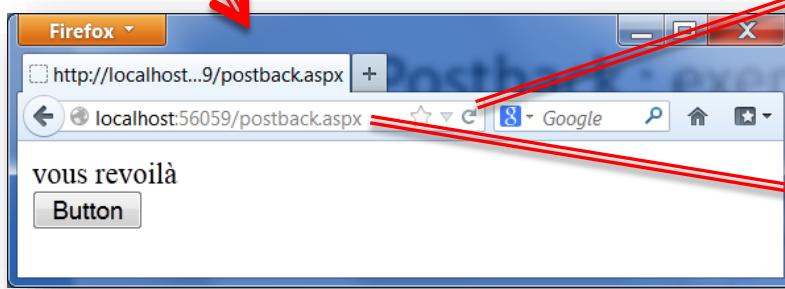
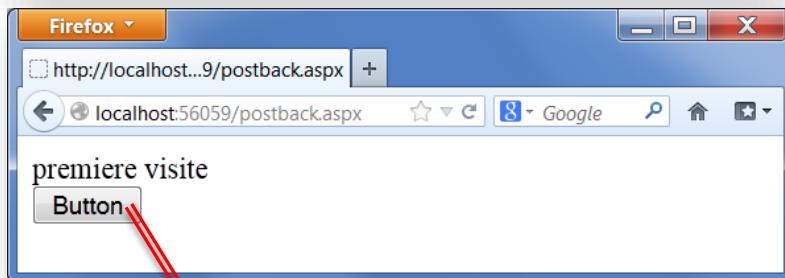
34

- Le modèle d'événements asp.net se base sur le post : un événement sur un webform déclenche un post du formulaire sur lui-même, donc une requête en POST, ou **postback**
- Le postback est un mécanisme permettant au serveur (code behind) de savoir si c'est la première requête sur la page (GET) ou s'il s'agit d'un post sur elle même (POST)
- Certains contrôles déclenchent automatiquement le postback (asp:button par exemple)
- L'Objet Page contient une propriété booléenne « `IsPostBack` » pour le savoir :
 - False = première requête
 - True = post du formulaire sur lui-même
- On utilise souvent cette propriété pour effectuer des actions d'initialisation dans le code behind

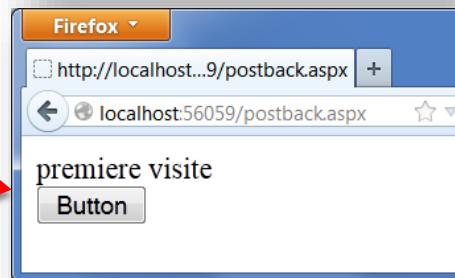
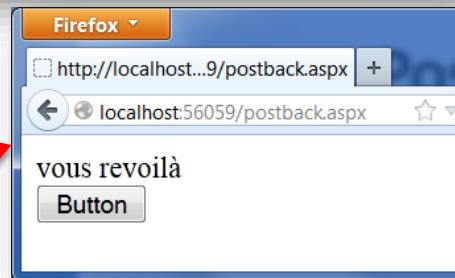
Postback : exemple

35

```
<form id="form1" runat="server">
<div>
    <asp:Button ID="Button1" runat="server" Text="Button"/>
</div>
</form>
```



```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        Response.Write("premiere visite");
    }
    else
    {
        Response.Write("vous revoilà");
    }
}
```



Cross Postback

36

- ASP.net donne la possibilité d'effectuer le postback sur une autre page >> « cross postback »
- Spécifié par l'attribut « postbackurl » sur un contrôle (Button ou Link par exemple)
- Pour s'assurer de l'origine du cross postback, la page d'arrivée doit vérifier la propriété PreviousPage.
- Le contenu des données transférées est accessible via Request.Form ou en transmettant PreviousPage

```
<form id="form1" runat="server">
<div>
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    <asp:Button ID="Button1" runat="server" Text="Button" PostBackUrl="~/CrossPostBackArrivee.aspx" />
</div>
</form>
```

CrossPostBackDepart.aspx

```
protected void Page_Load(object sender, EventArgs e)
{
    CrossPostBackDepart origine = this.PreviousPage as CrossPostBackDepart;
    TextBox txtNom = origine.FindControl("TextBox1") as TextBox;
    Response.Write("hello " + txtNom.Text);
}
```

CrossPostBackArrivee.aspx

Gestion de l'état : Champs cachés

37

- Une première solution pour garder des informations d'une requête à une autre est l'utilisation de champs cachés
- HTML propose les champs cachés qui permettent de stocker / modifier une valeur dans une page côté client sans l'afficher
- Syntaxe :
`<input name="champCache" type="hidden"
value="toto" />`
- Dans le code behind, il est possible de récupérer cette valeur lors de la requête suivante, via l'objet « Request »

Champs cachés : exemple

38

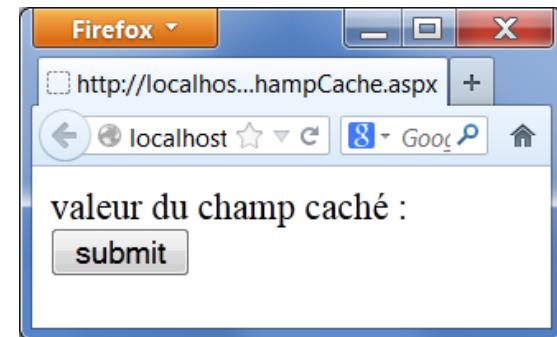
champCache.aspx

```
<form id="form1" runat="server">  
  
<input name="Hidden1" type="hidden" value="toto" />  
  
<input id="Submit1" type="submit" value="submit" />  
  
</form>
```

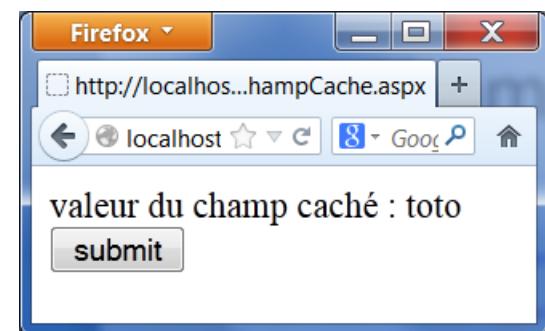
champCache.aspx.cs

```
public partial class ChampCache : System.Web.UI.Page  
{  
    protected void Page_Load(object sender, EventArgs e)  
    {  
        string champ = Request["Hidden1"];  
        Response.Write("valeur du champ caché : " + champ);  
    }  
}
```

1ere requête :



2eme requête (submit) :



Gestion de l'état : Viewstate

39

- ASP.net propose une alternative : le Viewstate
- Le Viewstate est un champ caché (name : __VIEWSTATE) qui contient sous forme sérialisée les informations d'état de tous les contrôles serveur de la page, afin de préserver leur état d'une requête à l'autre.

exemple : `<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="/wEPDwULLTE2MTY2ODcyMjlkZLRnaIQqtEKZymS+0sG46C2tthX9N+IGKgjAHUYatYZZ" />`

- Géré automatiquement par ASP.net
- Possibilité d'ajouter ses propres infos dans le viewstate (coté serveur)
- Attention ! Le Viewstate peut être lourd (plusieurs ko) >> il est possible de le désactiver sur les contrôles qui n'en ont pas l'utilité (EnableViewState="False")

Viewstate : Exemple

40

```
<form id="form1" runat="server">

    <asp:Label ID="monTexte" runat="server" Text="Label">Le viewstate c'est facile !</asp:Label>

    <br />

    <asp:DropDownList ID="ListCouleur" runat="server">
        <asp:ListItem Value="black" Text="noir"></asp:ListItem>
        <asp:ListItem Value="red" Text="rouge"></asp:ListItem>
        <asp:ListItem Value="green" Text="vert"></asp:ListItem>
    </asp:DropDownList>

    <asp:Button ID="Button1" runat="server" Text="ChangeCouleur" onclick="Button1_Click"/>

    <br />

    <asp:CheckBox ID="CheckItalique" runat="server" />

    <asp:Button ID="Button2" runat="server" Text="Maj Italique"
        onclick="Button2_Click" />

</form>
```

```
public partial class viewstate : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (ViewState["nbRequetes"] == null)
        {
            ViewState["nbRequetes"] = 1;
        }
        else
        {
            ViewState["nbRequetes"] = ((int)ViewState["nbRequetes"]) + 1;
        }

        Response.Write("Requetes : " + (int)ViewState["nbRequetes"] + "<br/>");
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        monTexte.ForeColor = System.Drawing.Color.FromName(ListCouleur.SelectedValue);
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        monTexte.Font.Italic = CheckItalique.Checked;
    }
}
```

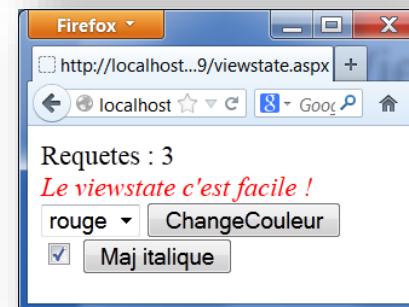
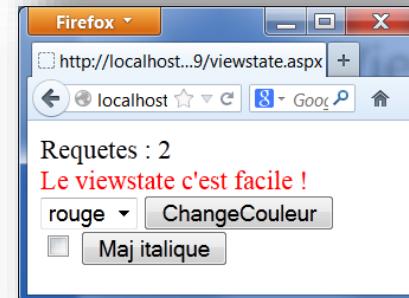
Viewstate : Exemple

41

1^{ère} requête : Eléments initialisés

2^{ème} requête : Sélection de « rouge »
et clic sur changeCouleur >> Couleur
mise à jour

3^{ème} requête : Sélection de italique et clic sur
Maj italique >> texte mis en italique
ET
Conservation de l'état précédent (couleur
rouge)



Gestion de l'état : **QueryString**

42

- Les données transmises en GET sont accessibles via la propriété **QueryString** de l'objet Request la Page (code behind).
- Objet Request : **System.Web.HttpRequest**
- **Request.QueryString** : **NameValueCollection** (lecture seule)
- Exemple :

URL :

maPage.aspx?nom=toto&age=23

Code behind :

```
string nom = Request.QueryString["nom"] ;  
int age = Int32.Parse(Request.QueryString["age"]);
```

Persistance des données : Cookies

43

- Cookie = segment de données émis par le serveur web.
- Quand le client reçoit un cookie, il l'intègre à chaque requête vers le serveur émetteur du cookie.
- Taille maximum 4 ko
- Peut contenir une ou plusieurs clés/valeurs
- Possibilité d'attribuer une date limite (cookie persistant), sinon le cookie est effacé à la fin de la session utilisateur (fermeture du navigateur)
- Asp.net : Récupération des cookies dans la propriété Request de la page (HttpRequest) : Request.Cookies (Collection)
- Asp.net : Envoi du cookie par l'objet Response : Response.Cookies.Add

Cookies : exemple

44

```
protected void Page_Load(object sender, EventArgs e)
{
    // cookie persistant (avec durée de vie)
    HttpCookie cp = Request.Cookies["cookiePersistant"];
    if (cp == null)
    {
        cp = new HttpCookie("cookiePersistant");
        cp.Value = "1";
        cp.Expires = DateTime.Now.AddMinutes(30);
    }
    else
    {
        cp.Value = (Int32.Parse(cp.Value) + 1).ToString();
        // remise à jour de la date d'expiration (non récupérée dans la requête)
        cp.Expires = DateTime.Now.AddMinutes(30);
    }
    Response.Cookies.Add(cp);
    Response.Write("Vous avez envoyé " + cp.Value + " requêtes en tout<br/>");

    // cookie de session (sans durée de vie)
    HttpCookie cs = Request.Cookies["cookieSession"];
    if (cs == null)
    {
        cs = new HttpCookie("cookieSession");
        cs.Values.Add("dateCreation", DateTime.Now.ToString("dd/MM/yyyy hh:mm:ss"));
        cs.Values.Add("compteur", "1");
    }
    else
    {
        cs.Values["compteur"] = (Int32.Parse(cs.Values["compteur"]) + 1).ToString();
    }
    Response.Cookies.Add(cs);
    Response.Write("vous avez envoyé " + cs.Values["compteur"] + " requêtes depuis le début de la session : " + cs.Values["dateCreation"]);
}
```

Persistance des données : Session

45

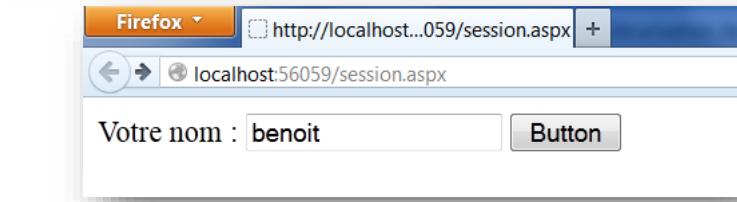
- Asp.net fournit un objet Session (System.Web.HttpSessionState), sous forme d'une table de hachage, pour stocker des informations relatives à la session courante de l'utilisateur.
- L'utilisateur transmet son id de session à chaque requête, par cookie ou par url (si cookies bloqués).
- Les données de session sont stockées sur le serveur (aspnet_wp.exe, SQL Server, asp.net state service)
- La durée de vie de la session est paramétrable
- On peut sécuriser une session
- L'objet Session peut contenir tout type d'objets
- Permet de maintenir des données liées aux préférences de l'utilisateur, critères de tri, sélection d'articles (panier), etc...

Session : Exemple

46

Session.aspx :

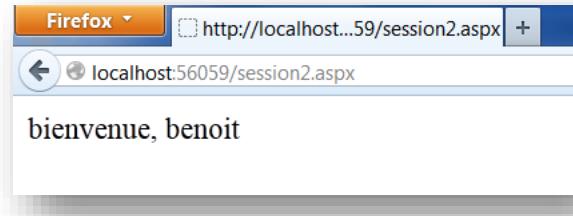
```
<form id="form1" runat="server">
<div>
    <asp:Label ID="Label1" runat="server" Text="Votre nom : "></asp:Label>
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    <asp:Button ID="Button1" runat="server" Text="Button" onclick="Button1_Click" />
</div>
</form>
```



```
protected void Button1_Click(object sender, EventArgs e)
{
    Session["Nom"] = TextBox1.Text;
    Session.Timeout = 10;
    Response.Redirect("session2.aspx");
}
```

Session2.aspx :

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["nom"] != null)
    {
        Response.Write("bienvenue, " + Session["nom"].ToString());
    }
}
```



Persistance des données : Application

47

- L'objet Application stocke les données communes à l'ensemble des utilisateurs du site.
- C'est l'équivalent de l'objet Session pour le site entier.
- Les données sont conservées tant que le processus de gestion de l'état le permet (aspnet_wp.exe, SQL Server, asp.net state service)
- Pour les problématiques d'accès concurrent : Système de verrouillage intégré (Lock)

Application : exemple

48

```
<form id="form1" runat="server">
<div>
    <asp:Label ID="lblNbCommandes" runat="server"></asp:Label>
    <asp:Button ID="Button1" runat="server" Text="Commander" OnClick="Commander_Click" />
</div>
</form>
```

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Application["nbCommandes"] == null)
    {
        Application.Lock();
        Application["nbCommandes"] = 0;
        Application.UnLock();
    }

    lblNbCommandes.Text = Application["nbCommandes"] + " commandes passées";
}

protected void Commander_Click(object sender, EventArgs e)
{
    Application.Lock();
    Application["nbCommandes"] = (int)Application["nbCommandes"] + 1;
    Application.UnLock();

    lblNbCommandes.Text = Application["nbCommandes"] + " commandes passées";
}
```

Persistance des données : Cache

49

- Contexte de données partagé entre tous les utilisateurs (comme l'objet Application) avec un système d'expiration (cache) :
 - Durée de vie Absolue
 - Durée de vie glissante (vs dernier accès)
 - Possibilité de liaison à une source de données (SQL Server ou fichier) : **dépendance de cache**
- Accès thread-safe géré par le Framework
- Nécessite le namespace System.Web.Caching

Ajout de données dans le Cache

50

```
public Object Add (string key, Object value, CacheDependency dependencies,
DateTime absoluteExpiration, TimeSpan slidingExpiration, CacheItemPriority
priority, CacheItemRemovedCallback onRemoveCallback)
```

- Méthode **Insert** (remplacement si clé déjà existante)
- Paramètres :
 - **key** : nom de la variable de cache
 - **value** : donnée mise en cache
 - **dependencies** : spécifie une dépendance de cache (fichier / bdd) >> **null** si pas de dépendance
 - **absoluteExpiration** : date/heure d'expiration, **Cache.NoAbsoluteExpiration** si pas d'expiration absolue
 - **slidingExpiration** : durée d'expiration glissante : intervalle entre le dernier accès et l'expiration du cache, **Cache.NoSlidingExpiration** (ou **TimeSpan.Zero**) si pas d'expiration glissante
 - **priority** : priorité des éléments en cache
 - **onRemoveCallback** : délégué pour spécifier une méthode qui sera appelée lorsque le cache est supprimé

Cache : Exemple

51

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        string valeur = "derniere maj : " + DateTime.Now.ToString("hh:mm:ss");

        Cache.Insert("cacheAbsolu", valeur, null, DateTime.Now.AddSeconds(30),
                    Cache.NoSlidingExpiration, CacheItemPriority.Default, CacheAbsoluRemoved);
        Cache.Insert("cacheGlissant", valeur, null, Cache.NoAbsoluteExpiration,
                    TimeSpan.FromSeconds(30), CacheItemPriority.Default, null);
    }

    Response.Write("cache absolu : " + (string)Cache["cacheAbsolu"] + "<br/>");
    Response.Write("cache glissant : " + (string)Cache["cacheGlissant"] + "<br/>");
}

private void CacheAbsoluRemoved(string key, object value, CacheItemRemovedReason reason)
{
    string valeur = "derniere maj : " + DateTime.Now.ToString("hh:mm:ss")
                  + " - cache recharge pour cause de " + reason.ToString();

    Cache.Insert("cacheAbsolu", valeur, null, DateTime.Now.AddSeconds(30),
                Cache.NoSlidingExpiration, CacheItemPriority.Default, null);
}
```



Cache Fichier : Exemple

52

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        LireFichier("news", string.Empty, CacheItemRemovedReason.DependencyChanged);
    }

    txtContenu.Text = (string)Cache["news"];
    lblMaj.Text = (string)Cache["maj"];
}

private void LireFichier(string key, object value, CacheItemRemovedReason reason)
{
    if (Cache["news"] == null)
    {
        string nomFichier;
        string contenu;

        nomFichier = Server.MapPath("/Fichiers/LastNews.txt");
        contenu = string.Empty;
        contenu = System.IO.File.ReadAllText(nomFichier);

        Cache.Insert("news", contenu, new CacheDependency(nomFichier), Cache.NoAbsoluteExpiration,
                    Cache.NoSlidingExpiration, CacheItemPriority.Default, LireFichier);
        Cache.Insert("maj", "Dernière mise à jour à " + DateTime.Now.ToString(), null, Cache.NoAbsoluteExpiration,
                    Cache.NoSlidingExpiration, CacheItemPriority.Default, null);
    }
}

<form id="form1" runat="server">
    <h1>
        Breaking news !</h1>
    <div>
        <asp:TextBox ID="txtContenu" runat="server" TextMode="MultiLine"></asp:TextBox>
        <asp:Label ID="lblMaj" runat="server" Text="Label"></asp:Label>
    </div>
    <input id="Submit1" type="submit" value="Mise à jour" />
</form>
```



La Classe HttpRequest

53

- Membre « Request » de la classe Page.
- Permet à ASP.NET de lire les valeurs HTTP envoyées par un client au cours d'une requête Web.
- Propriétés :
 - Headers : collection d'entêtes HTTP (Accessibles par des propriétés spécifiques : UserLanguages, UserAgent, Cookies...)
 - Url, RawUrl : URL de la requête
 - PhysicalPath : Chemin d'accès physique de la ressource demandée
 - QueryString : Collection de variables de querystring
 - Form : Collection des variables de formulaire
 - Files : Collection des fichiers téléchargés par le client (format Multipart MIME).

La classe `HttpResponse`

54

- Membre « `Response` » de la classe `Page`
- Encapsule les informations de réponse HTTP.
- Méthodes :
 - `Write(String)` : Ecrit une chaîne dans le flux de sortie HTTP
 - `Redirect(String)` : Redirige une requête vers une nouvelle URL
 - `End()` : Arrête l'exécution de la page et envoie au client le contenu actuel du flux de sortie.
 - `Flush()` : Envoie le contenu du flux de sortie actuellement en mémoire.

La classe ServerUtility

55

- Fournit des méthodes d'assistance pour traiter les requêtes Web.
- Membre « Server » de la classe Page
- Méthodes :
 - Execute(string) : Exécute l'appel à l'url spécifiée dans le contexte de la demande actuelle.
 - Transfer(string) : Termine l'exécution de la page actuelle et exécute l'appel à l'url spécifiée.
 - MapPath(string) : Retourne le chemin d'accès de fichier physique qui correspond au chemin d'accès virtuel spécifié sur le serveur Web.
 - HtmlEncode / HtmlDecode : encode / décode une chaîne en HTML
 - UrlEncode / UrlDecode : encode / décode une chaîne en URL

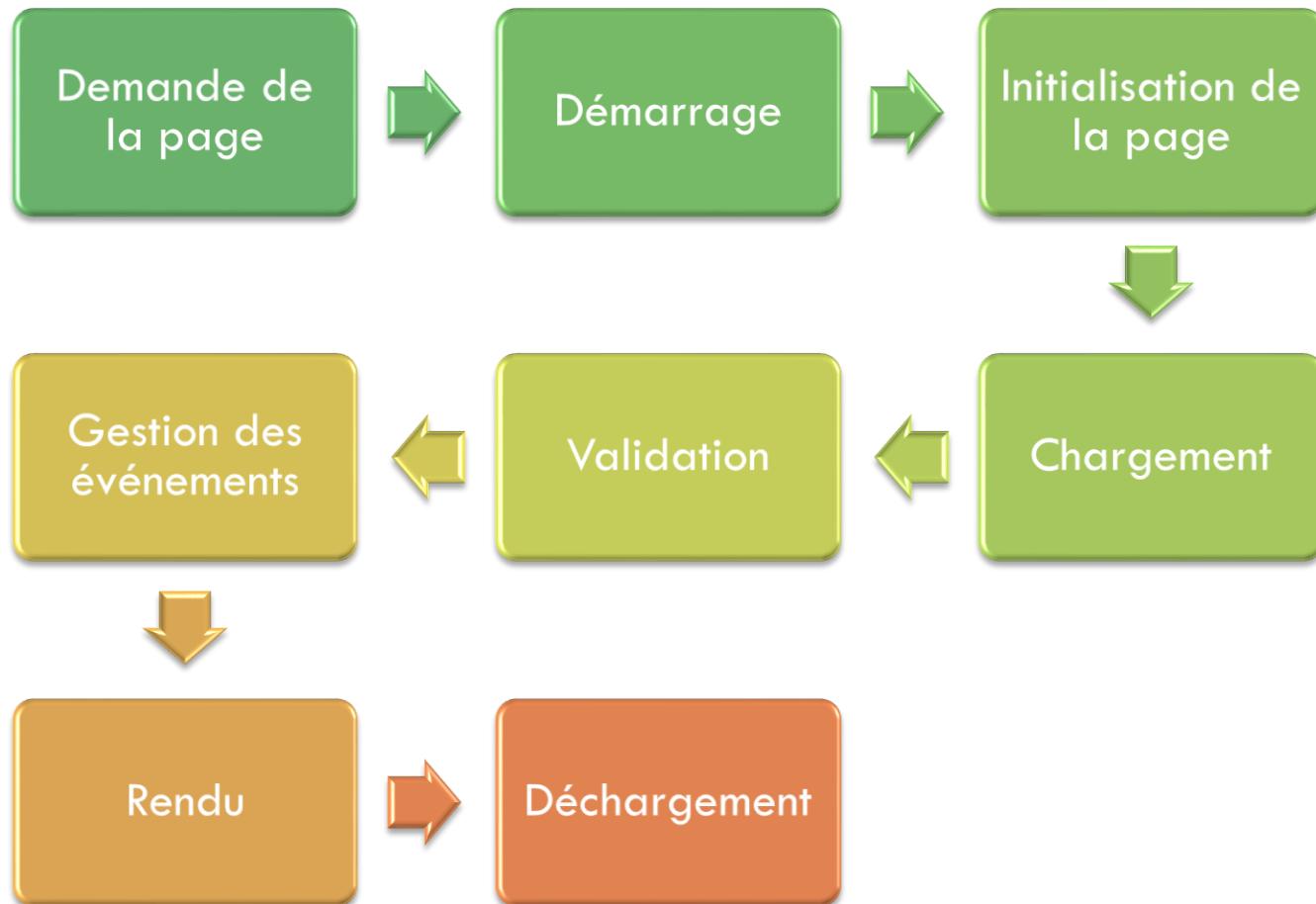
Exercice

56

- Faire une page aspx qui affiche les informations de l'utilisateur
- Tester le comportement des méthodes suivantes :
 - Response.Redirect
 - Response.Flush
 - Server.Transfer
 - Server.Execute

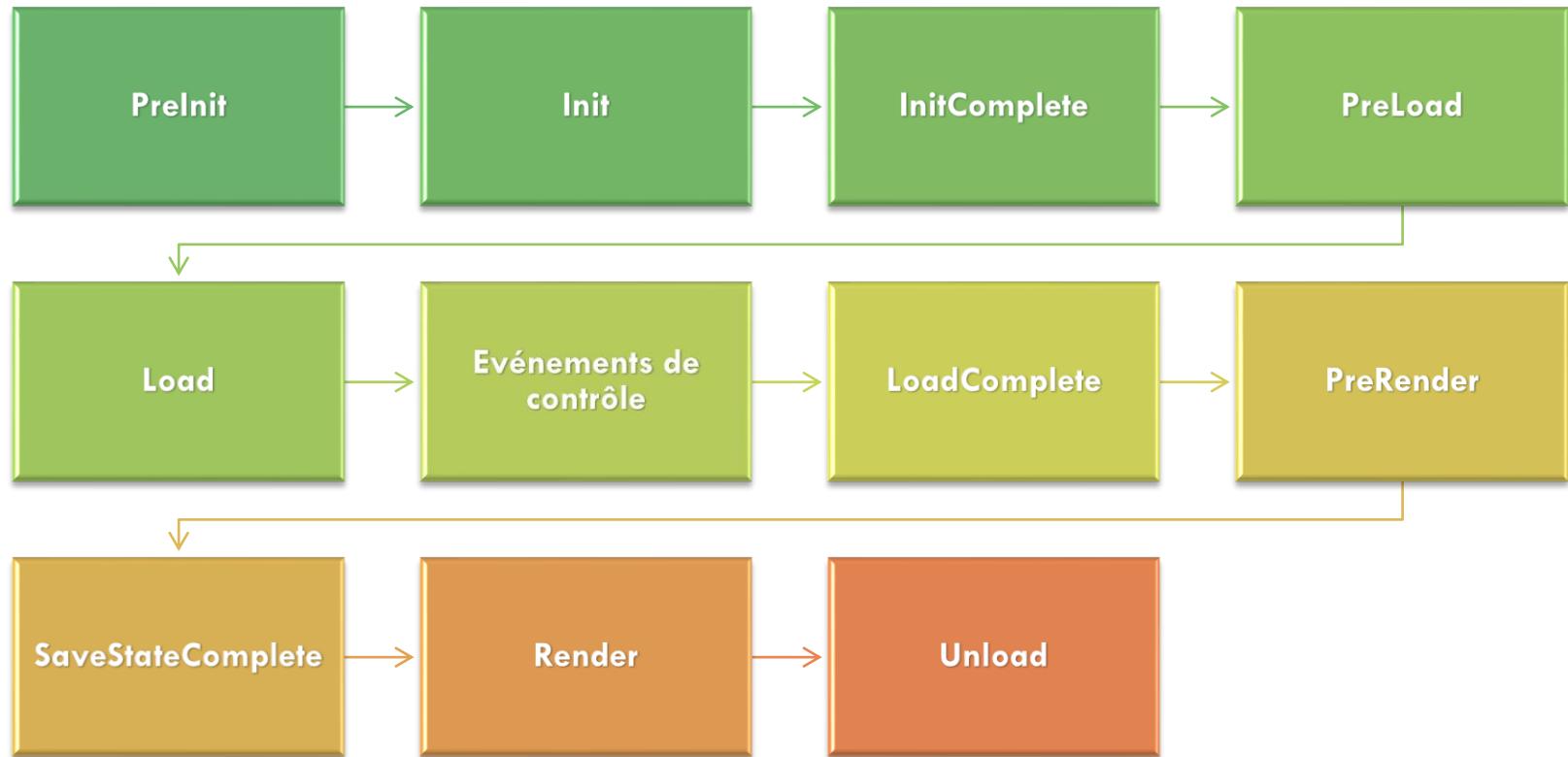
Cycle de vie d'une page ASP.net

57



Événements du cycle de vie

58



Gestion des événements de l'application : fichier global.asax

59

- Permet de définir des traitements sur des événements-clés de l'application :
 - **Application_Start** : Démarrage de l'application
 - **Application_End** : Arrêt de l'application
 - **Application_BeginRequest** : Réception d'une requête
 - **Application_Error** : Erreur non gérée
 - **Session_Start** : Démarrage d'une Session
 - **Session_End** : Fin d'une Session (mode sessionState=InProc uniquement, pas déclenché en mode StateServer ou SQLServer)

Global.asax : exemple

60

```
void Application_Start(object sender, EventArgs e)
{
    Application.Add("nbVisiteurs", 0);
    Application.Add("nbVisiteursEnCours", 0);
}

void Application_End(object sender, EventArgs e)
{
    System.Diagnostics.Trace.WriteLine("Arrêt de l'application : " + DateTime.Now.ToString());
}

void Application_Error(object sender, EventArgs e)
{
    System.Diagnostics.Trace.WriteLine(Server.GetLastError().ToString());
}

void Session_Start(object sender, EventArgs e)
{
    int nbVis = (int)(Application["nbVisiteurs"]);
    nbVis++;
    Application["nbVisiteurs"] = nbVis;
}

void Session_End(object sender, EventArgs e)
{
    int nbVis = (int)(Application["nbVisiteursEnCours"]);
    nbVis--;
    Application["nbVisiteursEnCours"] = nbVis;
}
```

Travaux pratiques

61

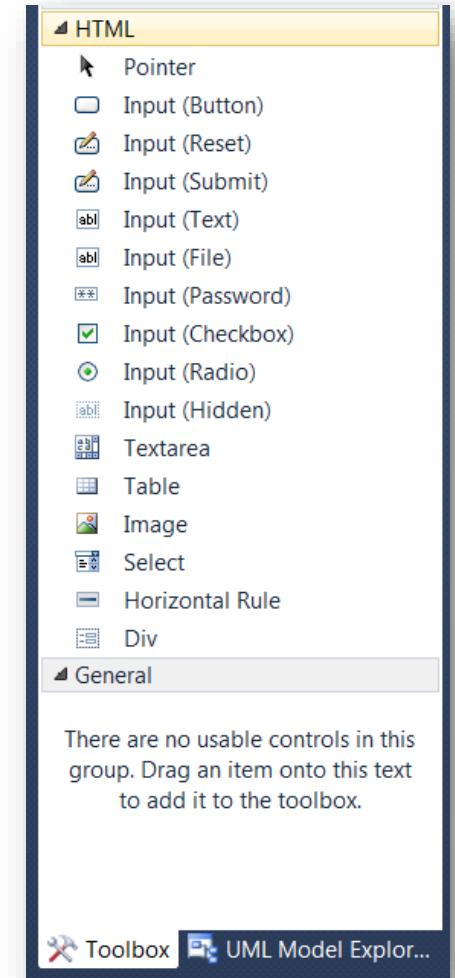
- Ecriture des premières pages ASP.NET basées sur des contrôleurs serveur standard (listes, boutons, etc.) et gérant des événements. Mise en oeuvre des mécanismes nécessaires aux applications professionnelles (session, cookie, etc.).

Contrôles serveur HTML

Contrôles serveur HTML

63

- Contrôle Serveur : balise `runat=<server>`
>> reconnu dans le code behind
- Contrôles HTML : éléments HTML, rendus tels quels dans le navigateur.
- Manipulables côté client par des scripts client (javascript, DHTML...)
- Déclarés et manipulables dans le code behind (accès aux attributs)
- Namespace `System.Web.UI.HtmlControls`



Contrôles serveur HTML : Exemple

64

.aspx

```
<input id="Text1" type="text" runat="server" />
```

.aspx.designer.cs

```
///<summary>
/// Text1 control.
///</summary>
///<remarks>
/// Auto-generated field.
/// To modify move field declaration from designer file to code-behind file.
///</remarks>
protected global::System.Web.UI.HtmlControls.HtmlInputText Text1;
```

.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    Text1.Value = "Hello Control !";
}
```

Source html côté client

```
<input name="Text1" type="text" id="Text1" value="Hello Control !" />
```

Principaux contrôles - Utilisation

65

□ Les Boutons

➤ Input (Button)

- Bouton standard associé à une action côté client
- Ex : `<input id="Button1" type="button" value="Aide" onclick="javascript:alert('Good Luck');"/>`

➤ Input (Submit)

- Envoi du formulaire courant vers l'url actuelle par défaut (postback), ou vers l'url indiquée par les attributs « action » ou « method » du formulaire.

➤ Input (Reset)

- Annule les données saisies dans le formulaire et restaure les valeurs par défaut (sans postback).

Principaux contrôles - Utilisation

66

□ Les zones de texte

- **Input (Text)**
 - Zone de texte standard
- **Input (Password)**
 - Zone de saisie de mot de passe (caractères remplacés par des *)
- **Input (File)**
 - Zone de texte verrouillée + bouton « parcourir... ». Lors de l'envoi du formulaire, ce champ contient le flux de données correspondant au fichier uploadé.
 - L'attribut « accept » permet de contrôler le type de fichier
- **TextArea**
 - Zone de texte multiligne, redimensionnable et scrollable (pour textes longs)

Principaux contrôles - Utilisation

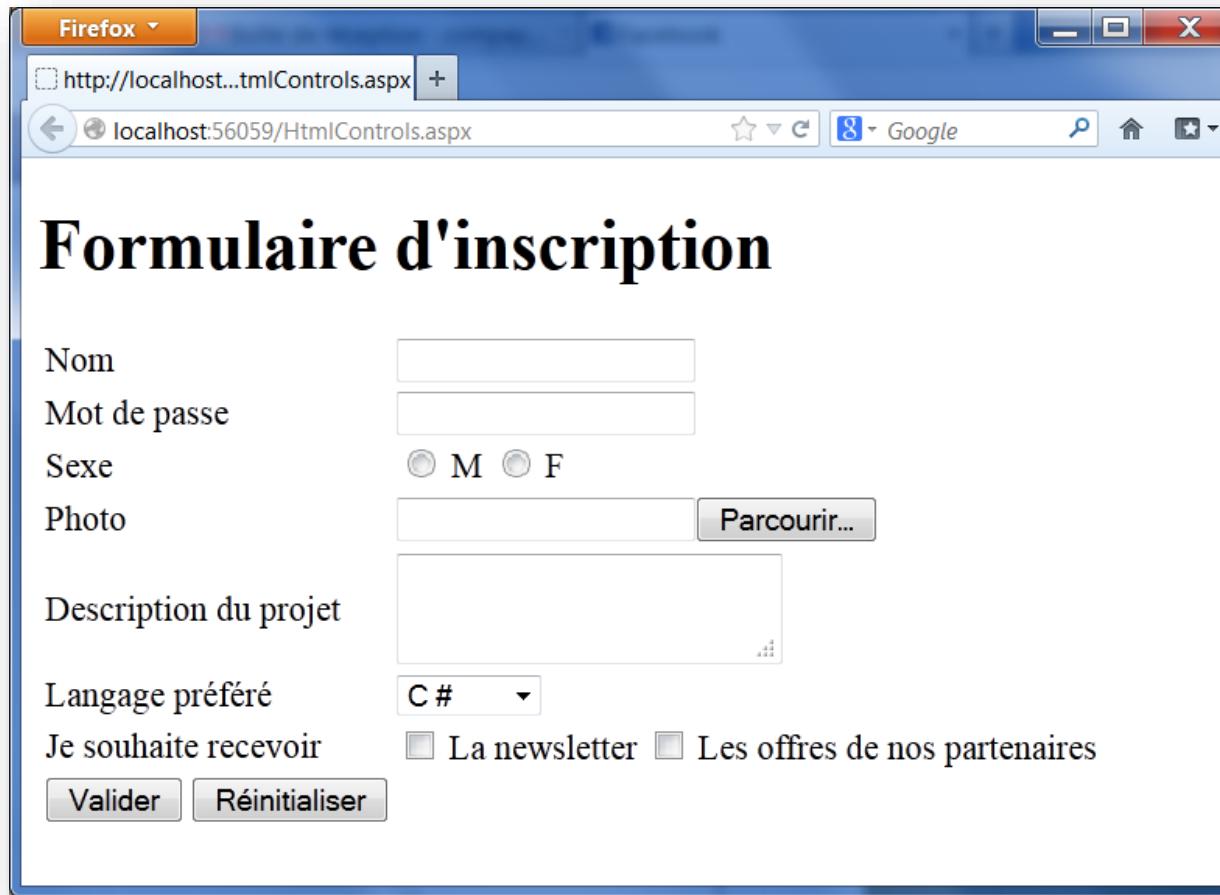
67

□ Autres contrôles :

- **Input (Checkbox)**
 - Case à cocher, on peut lui spécifier un attribut `checked` (sans valeur) pour la pré-cocher par défaut.
- **Input (Radio)**
 - Bouton radio. Plusieurs boutons radios sont regroupés par leur attribut `name`.
- **Select**
 - Liste déroulante. Chaque élément de la liste est une balise `<Option>` imbriquée.
- **Input (hidden)**
 - Zone de texte non visible, utile pour stocker des infos côté client.

Contrôles HTML : Exemple

68



The screenshot shows a Firefox browser window with a registration form. The browser title bar says "Firefox" and the address bar shows "localhost:56059/HtmlControls.aspx". The form is titled "Formulaire d'inscription" and contains the following fields:

- Nom: An input text field.
- Mot de passe: An input text field.
- Sexe: Radio buttons for "M" and "F", with "M" selected.
- Photo: An input file field with a "Parcourir..." button.
- Description du projet: A text area.
- Langage préféré: A dropdown menu showing "C #".
- Je souhaite recevoir:
 - La newsletter
 - Les offres de nos partenaires
- Buttons: "Valider" and "Réinitialiser".

Contrôles HTML : Exemple (2)

69

```
<h1> Formulaire d'inscription</h1>

<table style="width: 100%;">
  <tr>
    <td>Nom</td>
    <td>
      <input id="txtNom" type="text" runat="server" />
    </td>
  </tr>
  <tr>
    <td>Mot de passe</td>
    <td>
      <input id="txtPassword" type="password" runat="server"/>
    </td>
  </tr>
  <tr>
    <td>Sexe</td>
    <td>
      <input id="radSexeM" type="radio" name="radSexe" runat="server"/> M
      <input id="radSexeF" name="radSexe" type="radio" runat="server"/> F
    </td>
  </tr>
  <tr>
    <td>Photo</td>
    <td>
      <input id="photo" type="file" accept="image/*" runat="server"/>
    </td>
  </tr>
```

Contrôles HTML : Exemple (3)

70

```
<tr>
    <td>Description du projet</td>
    <td>
        <textarea id="txtDescription" cols="20" rows="2" runat="server"></textarea>
    </td>
</tr>
<tr>
    <td>Langage préféré</td>
    <td>
        <select id="selLangage" runat="server">
            <option value="java">java</option>
            <option value="cs" selected>C #</option>
            <option value="vbnet">VB.Net</option>
            <option value="cpp">C ++</option>
            <option value="Ada">Ada</option>
        </select>
    </td>
</tr>
<tr>
    <td>Je souhaite recevoir</td>
    <td>
        <input id="chkNewsletter" type="checkbox" selected runat="server"/> La newsletter
        <input id="chkPartenaires" type="checkbox" runat="server"/> Les offres de nos partenaires
    </td>
</tr>
<tr>
    <td colspan="2">
        <input id="Submit1" type="submit" value="Valider" />
        <input id="Reset1" type="reset" value="Réinitialiser" />
    </td>
</tr>
</table>
```

Scripts clients JavaScript.

71

- Interactivité côté client : javascript
- Action sur les contrôles html de la page
- Avantage : évite des allers-retours inutiles
- Obtenir un contrôle : `document.getElementById('<ID contrôle');`
- VS2010 propose un débuggeur javascript (seulement avec IE)
- Déboggage côté client pour autres navigateurs : plugins (WebDevelopper, Firebug...)

Javascript : exemple

72

```
<script type="text/javascript" language="javascript">
    function affichPartenaires(visible) {
        if (visible == true) {
            document.getElementById("listePartenaires").style.display = 'block';
        }
        else {
            document.getElementById("listePartenaires").style.display = 'none';
        }
    }
</script>
```

```
<input id="chkPartenaires" type="checkbox" runat="server" onclick="javascript:affichPartenaires(this.checked)"/> Les offres de nos partenaires
<ul id="listePartenaires" style="display:none;">
    <li><input id="chkPart01net" type="checkbox" selected runat="server"/> 01net.com</li>
    <li><input id="chkPartDeveloppez" type="checkbox" selected runat="server"/> Developpez.com</li>
    <li><input id="chkPartCcm" type="checkbox" selected runat="server"/> CommentCaMarche.net</li>
</ul>
```

Je souhaite recevoir

La newsletter Les offres de nos partenaires

La newsletter Les offres de nos partenaires

Je souhaite recevoir

- 01net.com
- Developpez.com
- CommentCaMarche.net

Travaux pratiques

73

- Utilisation des contrôles serveur HTML afin de rendre dynamique une page du côté client.
- Récupération des infos côté serveur

Contrôles serveur Web

Principe des contrôles Web.

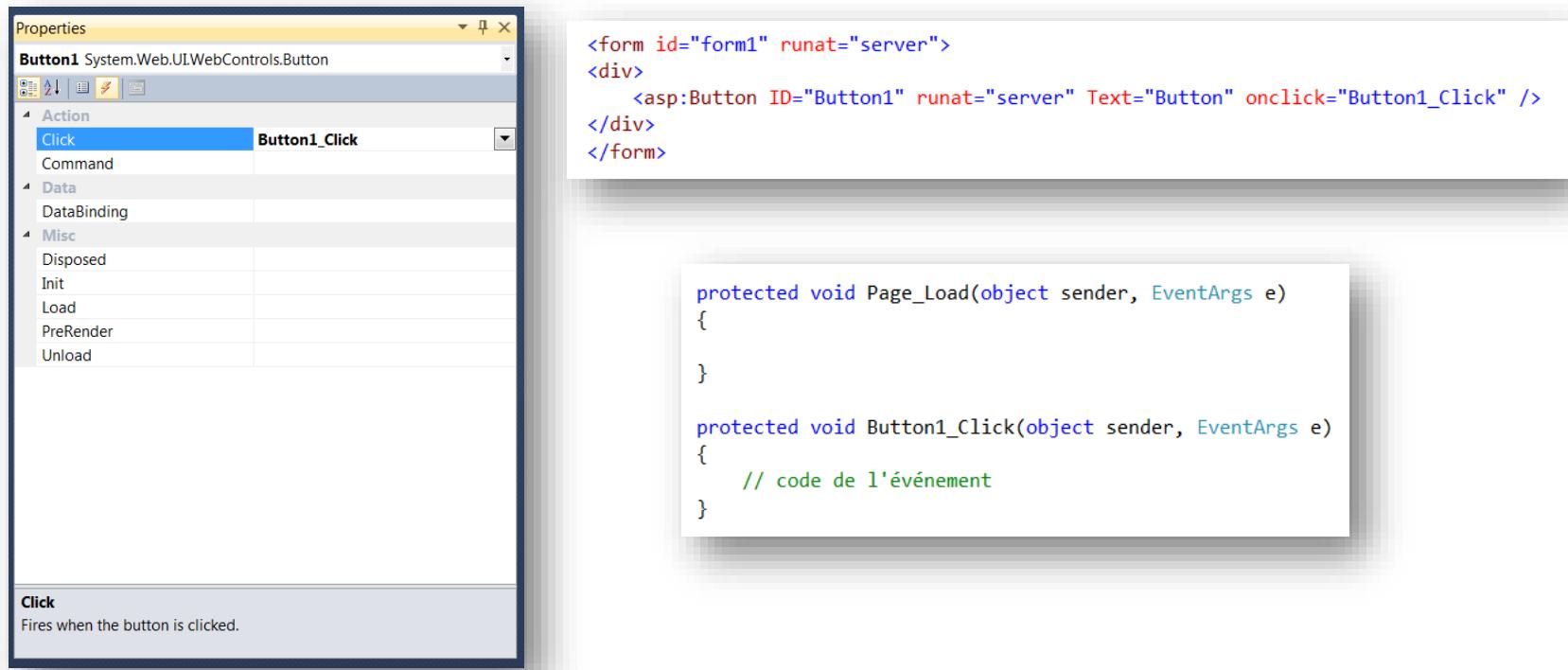
75

- Contrôles ASP.net, toujours runat="server"
- Plus riches pour manipulation côté serveur
- Génération de clients RIA
- Déclenchement d'événements côté serveur (postback)
- Différents types de WebControls :
 - Eléments HTML (textbox, button...)
 - Listes complexes
 - Conteneurs (Panel, Multiview, ...)
 - Contrôles riches (Calendar, AdRotator, FileUpload...)
 - Contrôles de validation (Validator)
 - Contrôles utilisateur (UserControls)

Contrôles web : événement

76

- Déclenchement d'événements côté serveur
- Le code des événements est exécuté APRES le page_load (cf cycle de vie)



Travaux pratiques

77

- Mise en œuvre de contrôles web simples (textbox, button, radio, listbox...) avec déclenchement d'événements côté serveur.

Contrôles de listes de données

78

- Listes liées à une source de données (databinding)
- Fonctionnalités de pagination, édition...

Contrôle	Groupement de Affichage						Tri
	Paging	données	flexible	Update,Delete	Insert		
ListView	X	X	X	X	X	X	X
GridView	X			X			X
DataList		X	X				
Repeater			X				

Contrôles conteneur

79

- Contrôles de présentation, destinés à contenir d'autres contrôles
- Panel : conteneur, devient un DIV côté client
- PlaceHolder : idem panel, mais n'affiche que son contenu (pas de DIV généré pour lui-même)
- Multiview : permet de gérer l'affichage d'une vue spécifique (View) parmi un ensemble de vues
- Wizard : permet de gérer l'enchaînement d'une série d'étapes (WizardStep), pour remplir un formulaire en plusieurs étapes par exemple

Contrôles riches

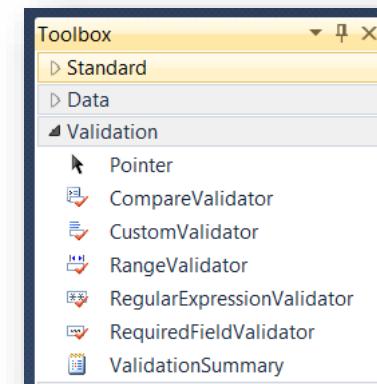
80

- **AdRotator** : Destiné à l'affichage aléatoire de publicités, avec gestion de la fréquence d'affichage. Fonctionne avec les fichiers « Advertisement » (xml), mais aussi avec un datasource standard.
- **Calendar** : Affiche un calendrier navigable et paramétrable (affichage, comportement), qui permet de sélectionner une date ou une série de dates (semaine, mois...)
- **FileUpload** : Contrôle d'envoi de fichier, permet de manipuler facilement le fichier envoyé côté serveur.

Contrôles de validation

81

- Validation des données saisies par l'utilisateur
 - Contrôles côté client
 - Affichage des messages d'erreur
 - Gestion des différents types de validation
- Validateurs ASP.net
- Attributs principaux :
 - ControlToValidate
 - ErrorMessage
 - ValidationGroup
- SECURITE : La validation doit toujours être refaite côté serveur



Validation : Exemple (1)

82

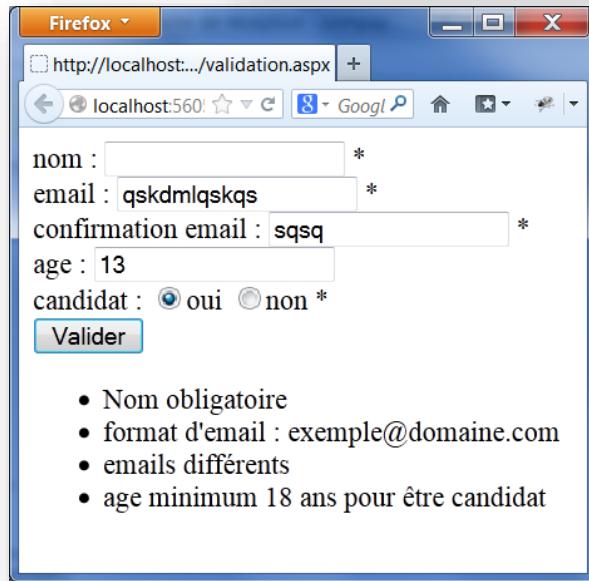
```
<form id="form1" runat="server">
<div>
nom : <asp:TextBox ID="txtNom" runat="server"></asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator1" ControlToValidate="txtNom"
        runat="server" Text="*" ErrorMessage="Nom obligatoire"></asp:RequiredFieldValidator>
<br />
email : <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>
    <asp:RegularExpressionValidator ID="RegularExpressionValidator1" ControlToValidate="txtEmail"
        ValidationExpression="([a-zA-Z0-9_\-.]+)(\([0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.\)|([a-zA-Z]{2,4})[0-9]{1,3})"
        runat="server" Text="*" ErrorMessage="format d'email : exemple@domaine.com"></asp:RegularExpressionValidator>
<br />
confirmation email : <asp:TextBox ID="txtEmail2" runat="server"></asp:TextBox>
    <asp:CompareValidator ID="CompareValidator1" ControlToValidate="txtEmail2" ControlToCompare="txtEmail"
        runat="server" Text="*" ErrorMessage="emails différents"></asp:CompareValidator>
<br />
age : <asp:TextBox ID="txtAge" runat="server"></asp:TextBox>
    <asp:RangeValidator ID="RangeValidator1" ControlToValidate="txtAge"
        Type="Integer" MinimumValue="0" MaximumValue="120"
        runat="server" Text="*" ErrorMessage="age entre 0 et 120"></asp:RangeValidator>
<br />
candidat :
    <asp:RadioButton ID="radCandidatOui" runat="server" GroupName="radCandidat"/>oui
    <asp:RadioButton ID="radCandidatNon" runat="server" GroupName="radCandidat"/>non
    <asp:CustomValidator ID="CustomValidator1" ClientValidationFunction="validerCandidat" OnServerValidate="ValidationCandidat"
        runat="server" Text="*" ErrorMessage="age minimum 18 ans pour être candidat"></asp:CustomValidator>
<br />
    <asp:Button ID="Button1" runat="server" Text="Valider" />
<br />
    <asp:ValidationSummary ID="ValidationSummary1" runat="server" />
</div>
</form>
```

Validation : Exemple (2)

83

```
<script language="javascript" type="text/javascript">
    function validerCandidat(sender, args) {
        if (document.getElementById("radCandidatOui").checked && document.getElementById("txtAge").value < 18) {
            args.IsValid = false;
        }
        else {
            args.IsValid = true;
        }
    }

```



Firefox

http://localhost:5601/validation.aspx

nom : *

email : *

confirmation email : *

age :

candidat : oui non *

Valider

- Nom obligatoire
- format d'email : exemple@domaine.com
- emails différents
- age minimum 18 ans pour être candidat

```
protected void ValidationCandidat(object source, ServerValidateEventArgs args)
{
    if (radCandidatOui.Checked && Int32.Parse(txtAge.Text) < 18)
    {
        args.IsValid = false;
    }
    else
    {
        args.IsValid = true;
    }
}
```

Contrôles utilisateur

84

- **UserControl** : fichiers ascx
- Même structure qu'un webform (.ascx, .ascx.designer.cs, .ascx.cs)
- Hérite de `System.Web.UI.UserControl`
- Permet de créer des contrôles complexes ou des groupes de contrôles réutilisables.
- Un contrôle utilisateur s'intègre dans une page comme un contrôle standard
- Possibilité de définir des événements spécifiques

Travaux pratiques

85

- Mise en oeuvre des contrôles de validation côté client et côté serveur avec les contrôles Wizard. Exemple du contrôle FileUpload... Mise en oeuvre de l'Ajax avec UpdatePanel.

Conception de la structure d'un site Web

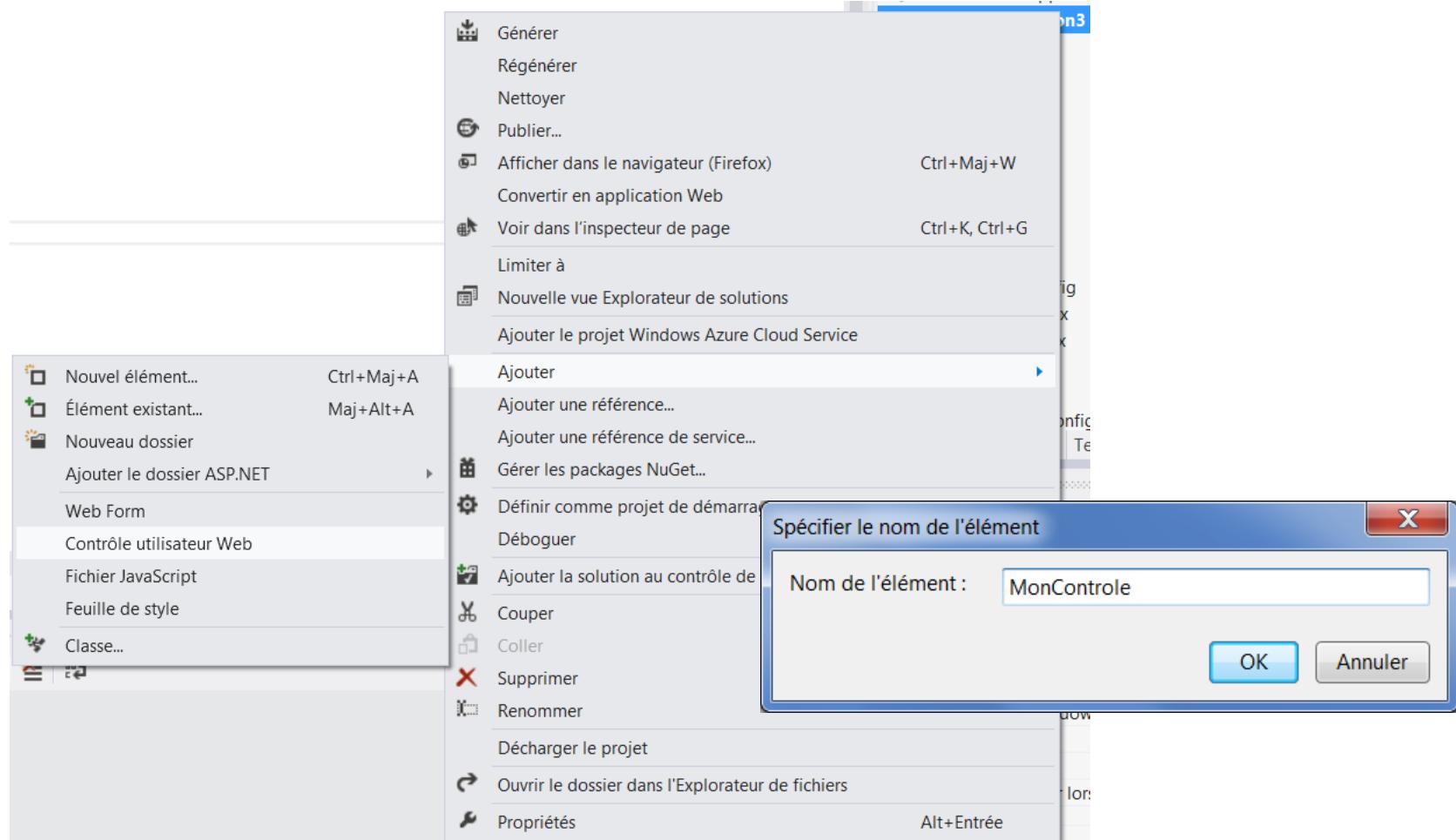
Contrôles utilisateur

87

- Contrôles utilisateur = fragments de page
- Réutilisables dans plusieurs pages
- Intégrables comme des contrôles serveur standard
- Extension .ascx
- Même architecture que les pages aspx
- Exemple : box login, résumé panier, ...

Contrôles utilisateur - Crédit

88



(c) Benoit Chauvet 2013

Contrôles utilisateur - Crédit

89

```
Form1.aspx.cs      WebForm2.aspx      MonControle.ascx.cs      MonControle.ascx      WebForm1.aspx
<%@ Control Language="C#" AutoEventWireup="true" CodeBehind="MonControle.ascx.cs"
           Inherits="WebApplication3.MonControle" %>



<asp:Label ID="lblNom" runat="server" Text=""></asp:Label>
    <br />
    <asp:Button ID="btnLogout" runat="server" Text="Déconnexion" OnClick="btnLogout_Click" />


```

Contrôles utilisateur - Crédit

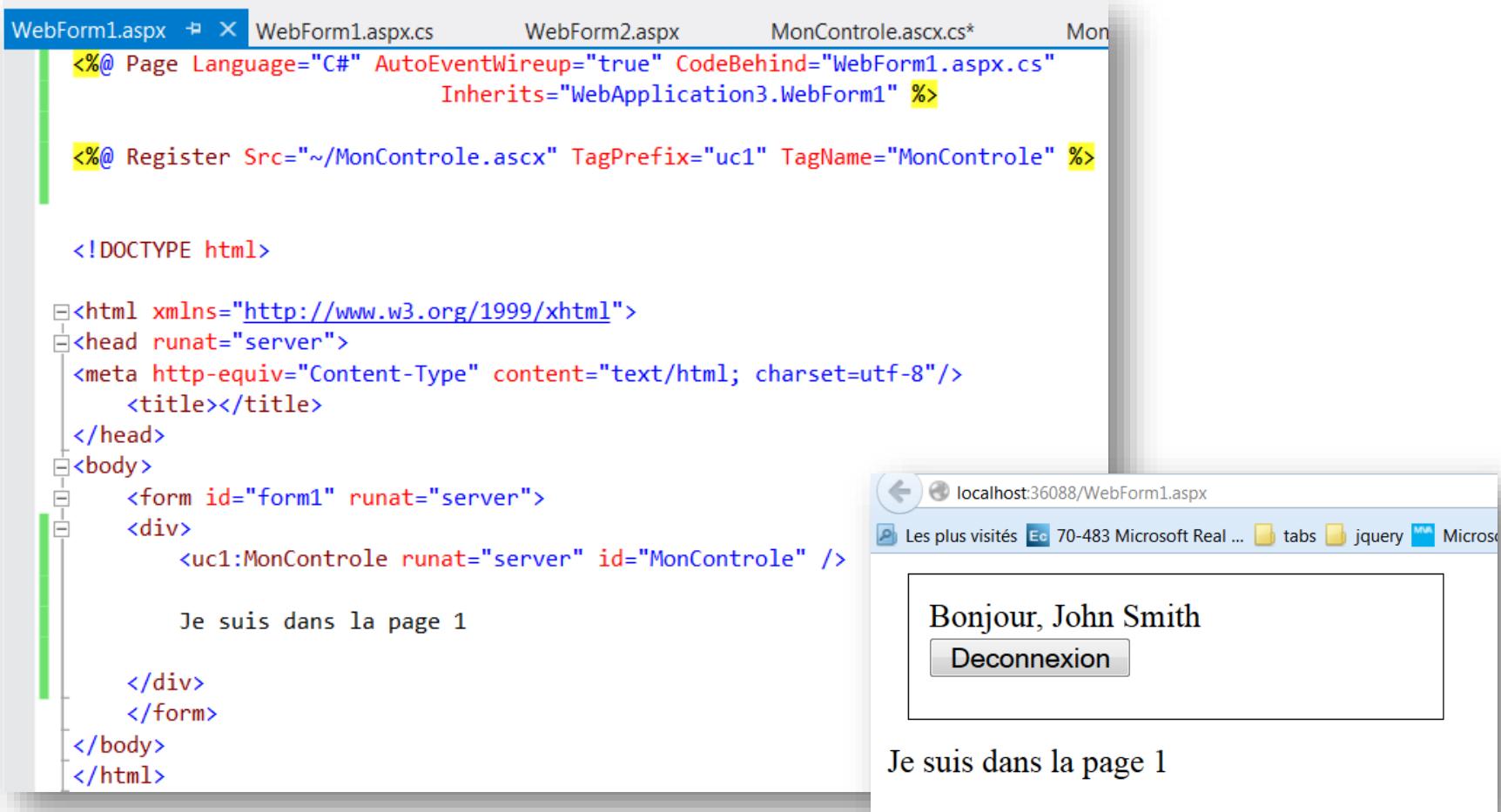
90

```
namespace WebApplication3
{
    public partial class MonControle : System.Web.UI.UserControl
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                lblNom.Text = "Bonjour, " + Session["username"].ToString();
            }
        }

        protected void btnLogout_Click(object sender, EventArgs e)
        {
            // Déconnexion de l'utilisateur...
        }
    }
}
```

Contrôles utilisateur - Utilisation

91



The image shows a screenshot of a web development environment. On the left, the code for `WebForm1.aspx` is displayed in a code editor. The code includes the following ASPX markup:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
    Inherits="WebApplication3.WebForm1" %>

<%@ Register Src="~/MonControle.ascx" TagPrefix="uc1" TagName="MonControle" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <uc1:MonControle runat="server" id="MonControle" />
            Je suis dans la page 1
        </div>
    </form>
</body>
</html>
```

On the right, a browser window shows the rendered output of the page. The browser address bar shows `localhost:36088/WebForm1.aspx`. The page content includes a user control instance and a message: "Bonjour, John Smith" and "Deconnexion". Below the browser window, the text "Je suis dans la page 1" is displayed.

Contrôles utilisateurs - Propriétés

92

The screenshot shows a Microsoft Visual Studio IDE interface with the following details:

- Top Navigation Bar:** Shows tabs for "MonControle.ascx.cs*", "WebForm1.aspx.cs", "WebForm2.aspx", and "Default.aspx".
- Code Editor (MonControle.ascx.cs):** Displays the C# code for the user control. It defines a partial class "MonControle" that inherits from "System.Web.UI.UserControl". The class contains three properties: "Formule" (String), "ShowButton" (bool), and "Nombre" (int). The "Page_Load" event handler checks if the page is not a post-back. If not, it sets the "Formule" property to "Bonjour, ". It then updates the "lblNom" label with the concatenated value of "Formule" and "Session["username"]". The "btnLogout" button's visibility is set to "ShowButton".
- Bottom Navigation Bar:** Shows tabs for "MonControle.ascx", "WebForm1.aspx", "WebForm1.aspx.cs", "WebForm2.aspx", "MonControle.ascx", and "Default.aspx".
- Code View (WebForm1.aspx.cs):** Displays the ASPX page code. It includes an instance of the "MonControle" user control with attributes: "runat='server'", "id='MonControle'", "Formule='Hello, '", "ShowButton='false'", and "Nombre='5'".

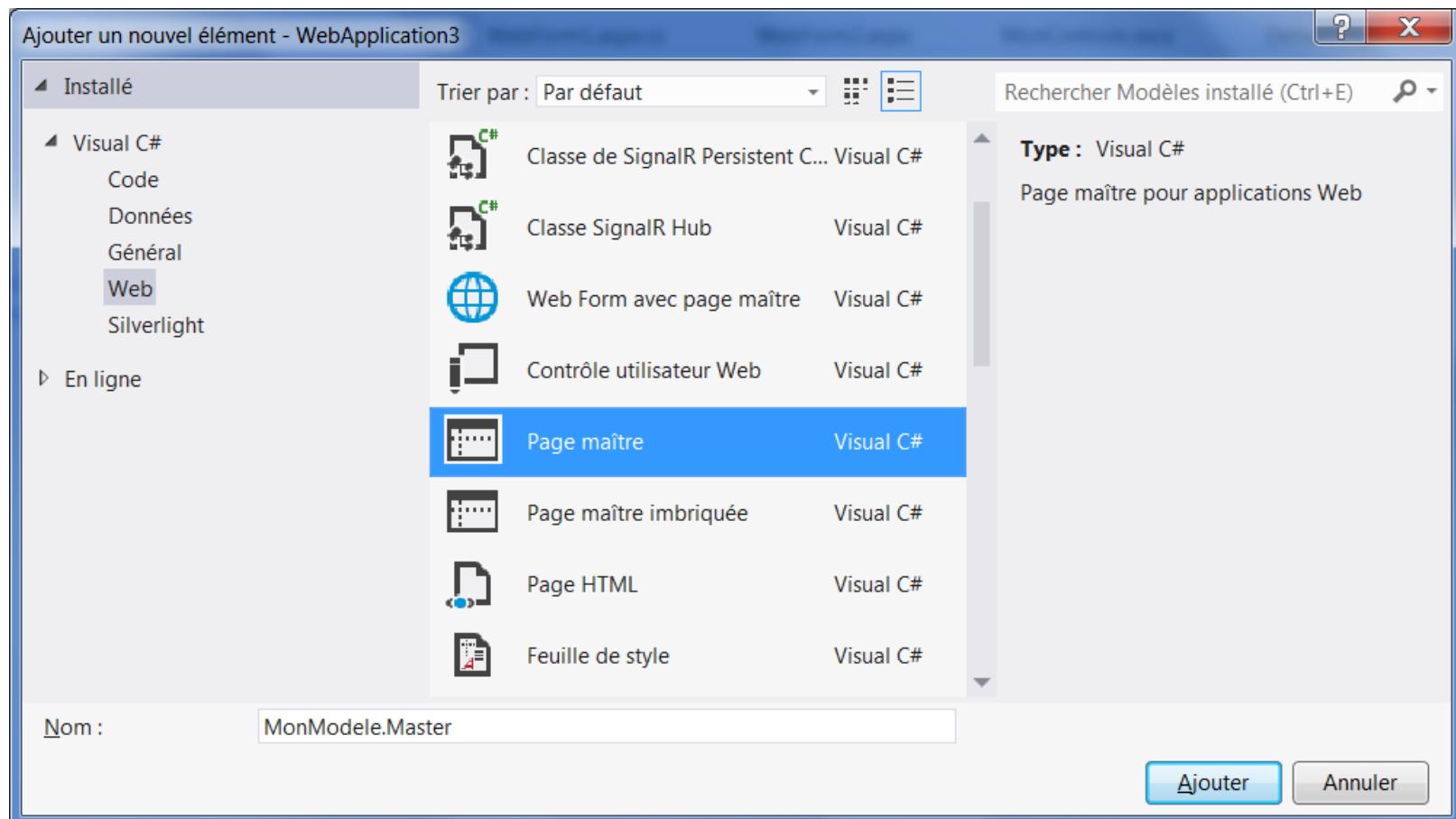
Pages Maîtres

93

- Les pages maîtres (masterpages) constituent le modèle des pages d'un site
- Une masterpage regroupe les éléments communs à toutes les pages
- « Arche » du site : header, footer, sidebar, etc...
- Contient des éléments asp.net standard et des emplacements de contenu à remplir par les pages (ContentPlaceHolder)

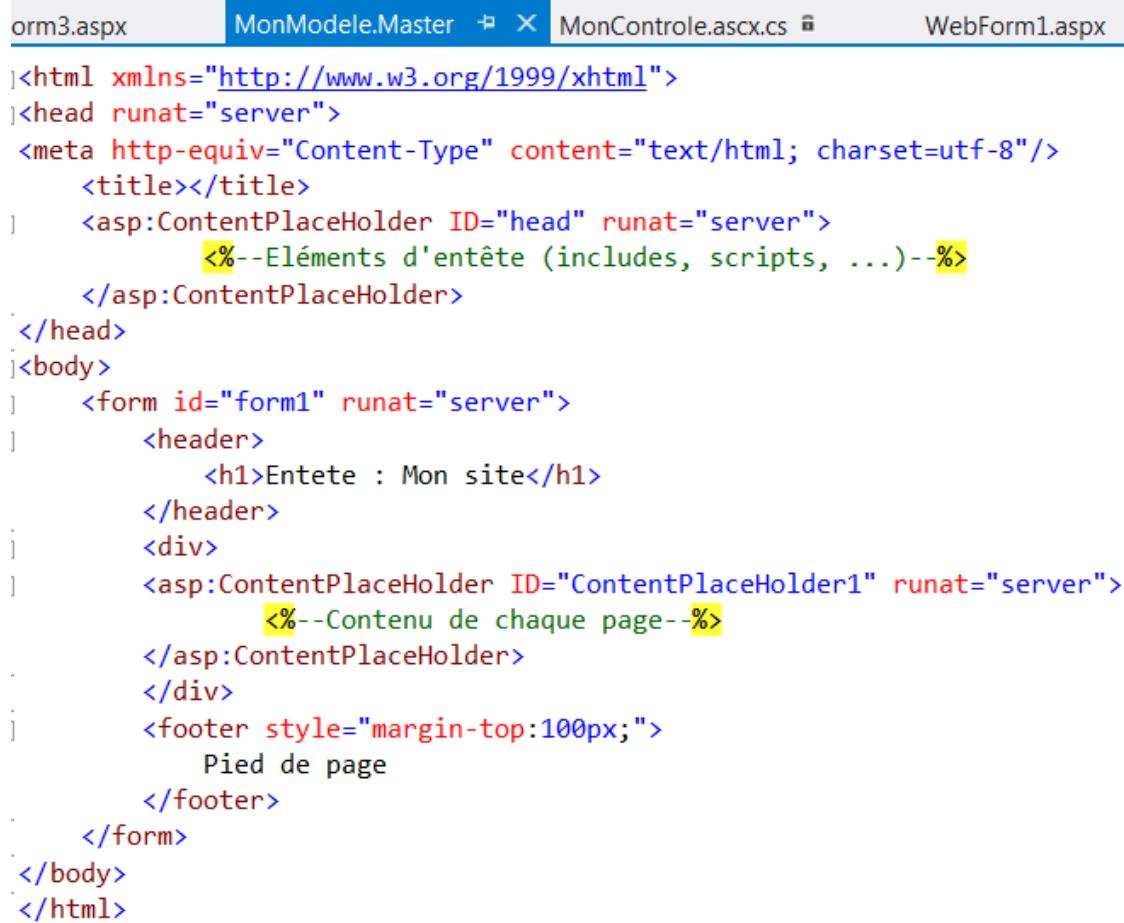
Pages Maîtres - Création

94



Pages Maîtres - Création

95

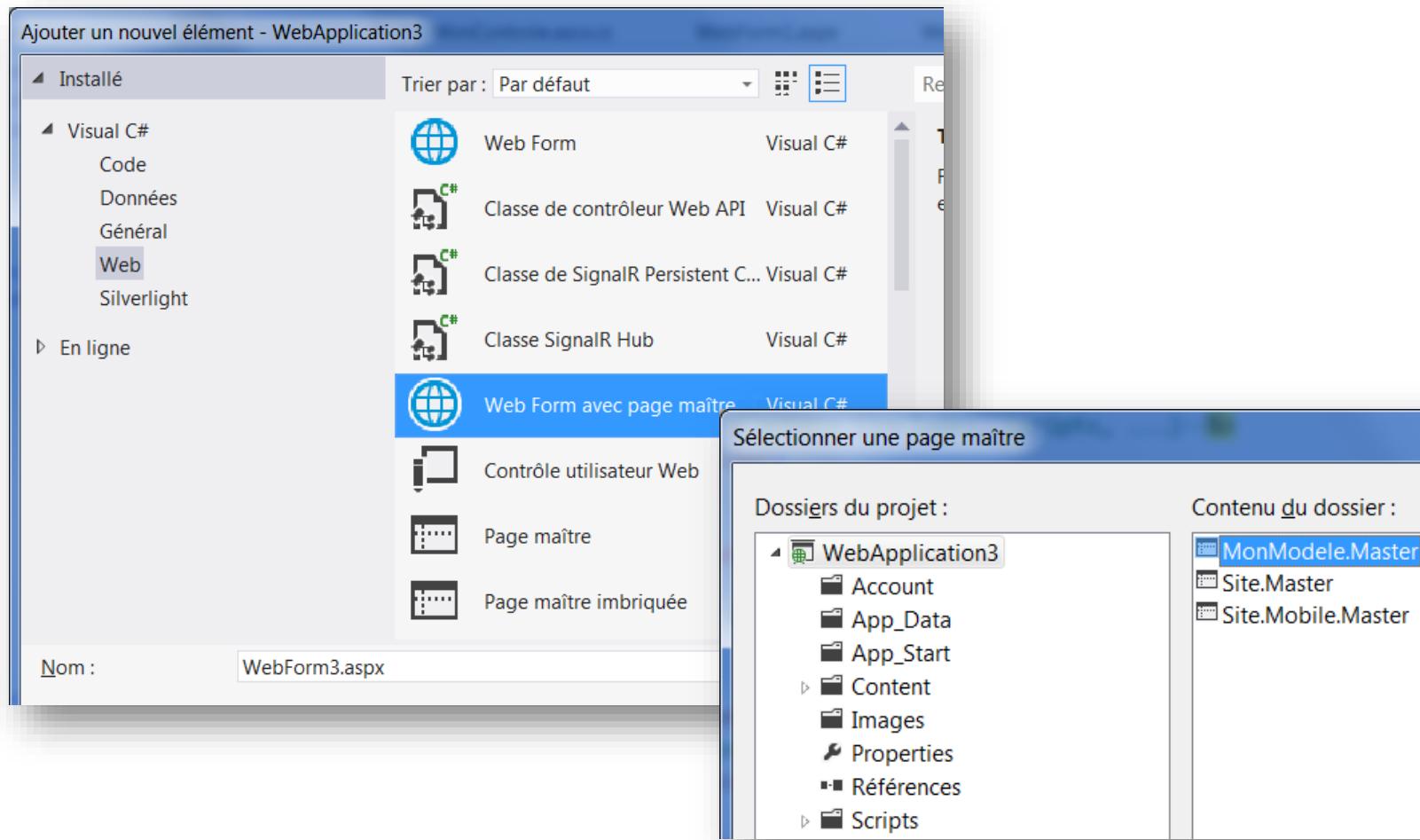


```
orm3.aspx      MonModele.Master  ✖  MonControle.ascx.cs  WebForm1.aspx

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <asp:ContentPlaceHolder ID="head" runat="server">
        <%--Eléments d'entête (includes, scripts, ...)--%>
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
        <header>
            <h1>Entete : Mon site</h1>
        </header>
        <div>
            <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
                <%--Contenu de chaque page--%>
            </asp:ContentPlaceHolder>
        </div>
        <footer style="margin-top:100px;">
            Pied de page
        </footer>
    </form>
</body>
</html>
```

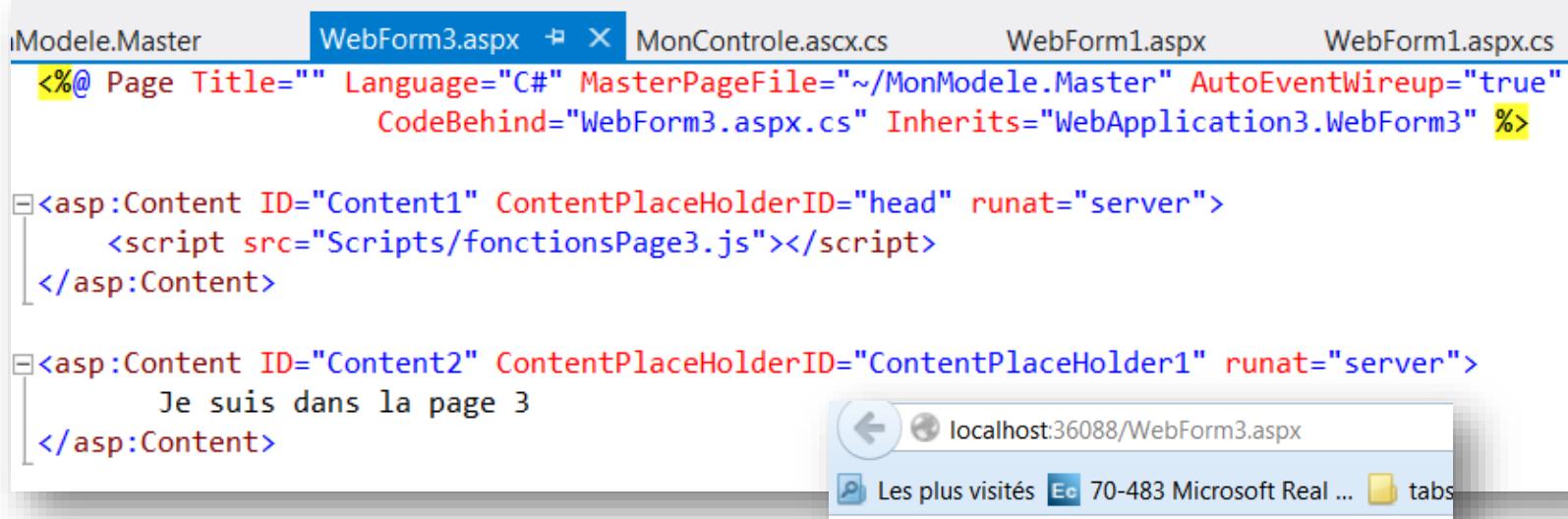
Pages Maîtres - Utilisation

96



Pages Maîtres - Utilisation

97



```
MonModele.Master WebForm3.aspx MonControle.ascx.cs WebForm1.aspx WebForm1.aspx.cs
<%@ Page Title="" Language="C#" MasterPageFile="~/MonModele.Master" AutoEventWireup="true"
   CodeBehind="WebForm3.aspx.cs" Inherits="WebApplication3.WebForm3" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
  <script src="Scripts/fonctionsPage3.js"></script>
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
  Je suis dans la page 3
</asp:Content>
```

Entete : Mon site

Je suis dans la page 3

Pied de page

Rôles des dossiers spéciaux

98

- App_Data : Données locales à l'application web (bdd .mdf, fichiers xml, ...)
- App_Code : Classes (.cs) utilisées pour les sites web (non compilées par défaut)
- App_Themes : Mise en forme (.css, images, ...) déclinable en plusieurs thèmes
- Images : Images du site
- Scripts : Fichiers javascript (.js)

Autres éléments spéciaux

99

- **Properties**
 - AssemblyInfo : Paramètres de l'assembly (description, guid, version,...)
- **References**
 - Références des assemblies utilisées par l'application (définies dans le .sln)
- **Account**
 - Modèles de page pour l'authentification (Login, Register, ChangePassword...)
- **Global.asax** : gestion des événements de l'application
- **Web.config** : configuration de l'application web

Feuilles de style CSS

100

- Les feuilles de style (CSS) permettent de définir la présentation des éléments de la page :
 - Polices, couleurs, styles de paragraphe et titres...
 - Bordures, dimensions, positionnement...
- Définition centralisée de styles, applicables à différents éléments du site
- Lien vers une css dans la section <head> de la page

CSS - Cration

101

```
Form4.aspx      aut.css  ✎ ✎ Skin1.skin
body {
    background-color: orange;
    color: brown;
    font-family: Arial;
}

h1 {
    text-transform: uppercase;
    text-decoration: underline;
}
```

```
/* Classe */
.boite1 {
    border: 2px solid #996633;
    padding: 5px;
    margin: 5px;
    background-color: #FFCC66;
    width:150px;
}

/* ID */
#boutonValider {
    background-color: #FFCC00;
    border: 2px outset #993300;
    margin:5px;
}
```

CSS - Utilisation

102

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <link href="App_Themes/Automne/aut.css" rel="stylesheet" />
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <h1>Exemple CSS</h1>
            <div class="boite1">
                Contenu de ma box
            </div>
            <asp:Button ID="boutonValider" runat="server" Text="Valider" />
        </div>
    </form>
</body>
</html>
```

EXEMPLE CSS

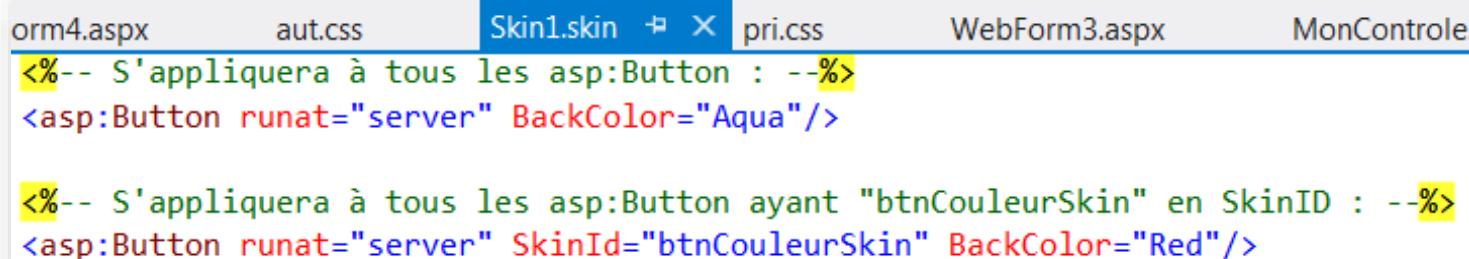
Contenu de ma box

Valider

Skins

103

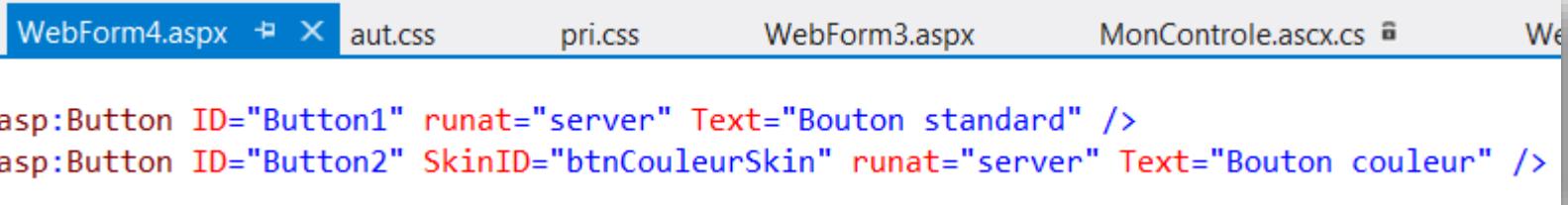
- Un fichier .skin permet de définir les propriétés visuelles des contrôles web d'ASP.NET
- Exemple :



orm4.aspx aut.css Skin1.skin pri.css WebForm3.aspx MonControle.ascx.cs

```
<%-- S'appliquera à tous les asp:Button : --%>
<asp:Button runat="server" BackColor="Aqua"/>

<%-- S'appliquera à tous les asp:Button ayant "btnCouleurSkin" en SkinID : --%>
<asp:Button runat="server" SkinId="btnCouleurSkin" BackColor="Red"/>
```



WebForm4.aspx aut.css pri.css WebForm3.aspx MonControle.ascx.cs

```
<asp:Button ID="Button1" runat="server" Text="Bouton standard" />
<asp:Button ID="Button2" SkinID="btnCouleurSkin" runat="server" Text="Bouton couleur" />
```

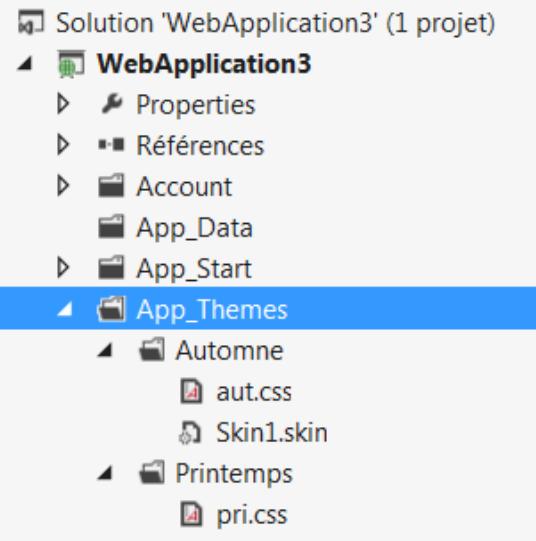
Thèmes

104

- Un thème contient des fichiers .css et .skin
- Un site peut avoir plusieurs thèmes, paramétrable au niveau de chaque page ou de l'ensemble du site
- Chaque thème est un sous-dossier du dossier « themes »
- L'ensemble des .css et des .skin du theme sont appliqués automatiquement aux pages concernées

Thèmes - exemple

105



The screenshot shows two instances of the Visual Studio code editor. The top instance is for 'WebForm4.aspx' and the bottom instance is for 'WebForm4.aspx'. Both files contain the following code:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="WebForm4.aspx.cs"
Inherits="WebApplication3.WebForm4"
Theme="Printemps" %>
```

On the right side of the screenshot, there are two green boxes labeled 'EXAMPLE CSS' containing a button labeled 'Contenu de ma box' and a button labeled 'Valider'.

Travaux pratiques

106

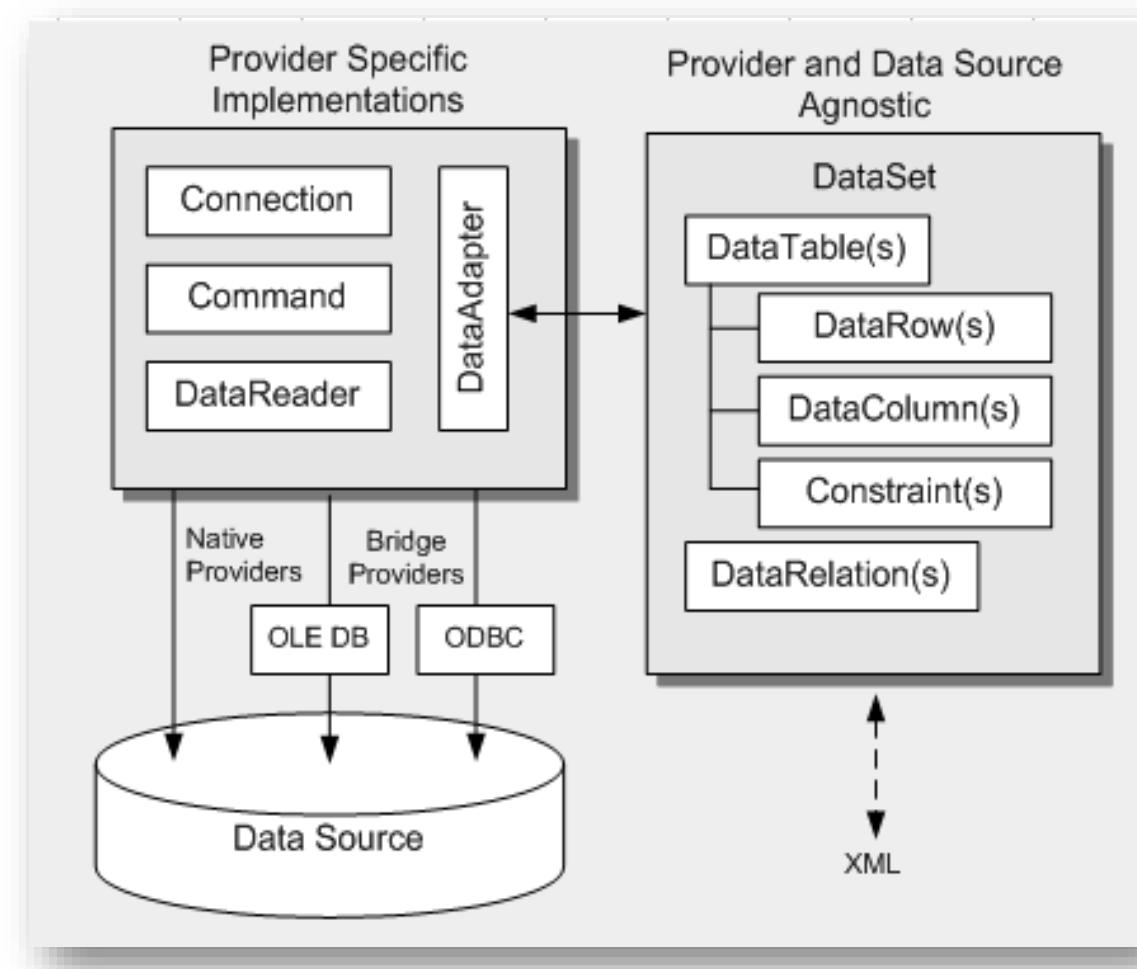
- Réalisation d'une application de e-commerce.

Contrôle de sources de données

ADO.NET - Présentation

108

- ADO.net : ensemble de composants du framework pour la manipulation de données
- Namespace System.Data



ADO.NET – Fournisseurs de données

109

- Fournisseurs disponibles dans le framework :
 - Fournisseur SQL Serveur – (SQL)
 - Fournisseur OLE DB – (OLEDB)
 - Fournisseur ODBC – (ODBC)
- Fournisseurs tiers :
 - Oracle Data Provider (ODP)
 - MySql...
 - Assemblies téléchargeables chez les tiers
- Proposent des implémentations d'ADO.NET basées sur les mêmes objets de base (préfixés).
 - SQL : System.Data.SqlClient
 - ODBC : System.Data.Odbc
 - OLE DB : System.Data.OleDb
 - Oracle : System.Data.OracleClient

Connexion à une base de données

110

- Objet Connection : gère la connexion à la base à partir d'une chaîne de connexion
 - Chaîne de connexion (propriété ConnectionString)
 - Méthodes Open () & Close()

```
// using System.Data.SqlClient;

// Création d'une connexion
SqlConnection cnx = new SqlConnection();
// Affectation d'une chaîne de connexion :
cnx.ConnectionString = "Data Source=localhost;Initial Catalog=Market;Integrated Security=True";
// Ouverture de la connexion :
cnx.Open();
// ...
// échanges avec la base de données
// ...
// Fermeture de la connexion :
cnx.Close();
```

Chaînes de connexion

111

- Représente les paramètres de connexion à la base, sous forme couples clé/valeur, stockés dans une string
- Gestion intégrée de pool de connexions
- Paramètres courants :
 - Data source : ip ou nom du serveur de données
 - Initial catalog : nom de la base de données
 - Integrated security :
 - true : utilisation du compte windows pour l'authentification
 - false : nom d'utilisateur et mot de passe à fournir
 - User ID : nom d'utilisateur (integrated security = false)
 - Pwd : mot de passe (integrated security = false)

Chaînes de connexion

112

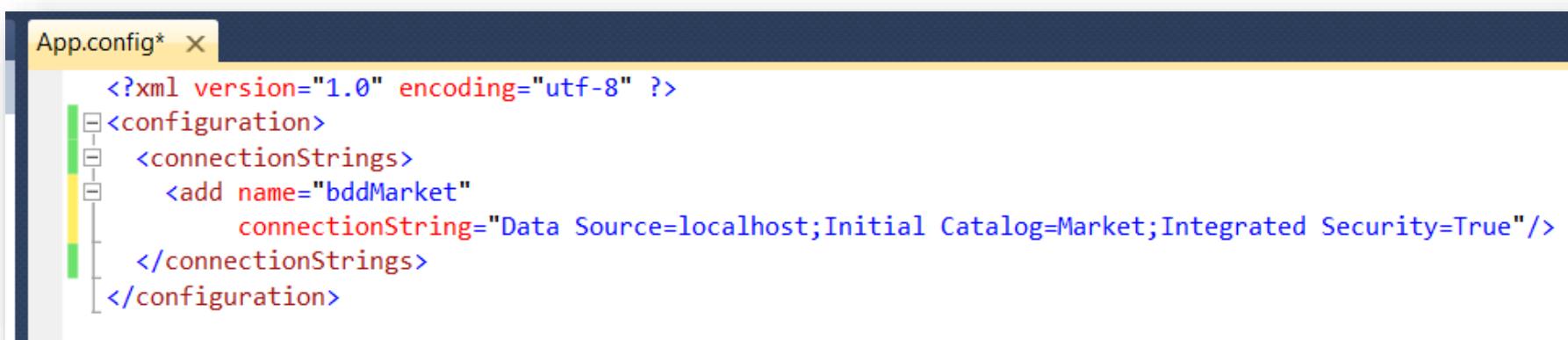
□ Paramètres courants (suite)

- **Connect Timeout** : durée en secondes d'attente de réponse du serveur à la demande de connexion.
- **Connection LifeTime** : durée de vie d'une connection dans un pool de connections (0 = infini)
- **Max Pool Size** : nombre max de connections dans le pool
- **Min Pool Size** : nombre min de connections dans le pool
- **Connection Reset** : indique si la connection est réinitialisée lors de sa remise dans le pool
- **Pooling** : indique si la connection peut être extraite d'un pool

Chaînes de connexion

113

- Dans une application .net, on stocke les chaînes de connexion dans un fichier de configuration (app.config)
- Pour l'utiliser, il faut ajouter dans le projet une référence à l'assembly System.Configuration



```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <connectionStrings>
    <add name="bddMarket"
         connectionString="Data Source=localhost;Initial Catalog=Market;Integrated Security=True" />
  </connectionStrings>
</configuration>
```

```
// Affectation d'une chaîne de connexion :
cnx.ConnectionString = System.Configuration.ConfigurationManager
                      ..ConnectionStrings["bddMarket"].ConnectionString ;
```

Mode connecté

114

- Ouverture d'une connexion à chaque lecture et mise à jour de la base
- Lecture basée sur le DataReader
- SqlDataReader, OdbcDataReader, OracleDataReader....
- Envoi d'une commande à la base
- Objet Command : Gère la commande envoyée à la base de données (requête, procédure stockée...) et les éventuels paramètres associés

Objet Command

115

- Propriété **Connection** : connexion associée
- Propriété **CommandText** : texte de la commande
- Propriété **CommandType**
 - Text : requête (par défaut)
 - StoredProcedure : procédure stockée
- Méthodes d'exécution :
 - ExecuteScalar : Requête renvoyant une seule donnée (ex : SELECT COUNT(*) FROM CLIENTS)
 - ExecuteReader : Requête renvoyant un jeu de résultats dans un DataReader (ex : SELECT * FROM CLIENTS)
 - ExecuteNonQuery : Requête de mise à jour (ex : DELETE FROM CLIENTS WHERE id = 1)

Objet Command - Exemple

116

```
SqlCommand cmd = new SqlCommand();
cmd.Connection = cnx;
cmd.CommandText = "SELECT * FROM ARTICLES";
cmd.CommandType = System.Data.CommandType.Text;
// exécution de la commande :
SqlDataReader dr = cmd.ExecuteReader();
```

```
SqlCommand cmd2 = new SqlCommand();
cmd2.Connection = cnx;
cmd2.CommandText = "SELECT COUNT(*) FROM ARTICLES";
cmd2.CommandType = System.Data.CommandType.Text;
// exécution de la commande :
int nbArticles = (int)cmd2.ExecuteScalar();
```

```
SqlCommand cmd3 = new SqlCommand();
cmd3.Connection = cnx;
cmd3.CommandText = "UPDATE ARTICLES SET prix = 20 WHERE id = 1";
cmd3.CommandType = System.Data.CommandType.Text;
// exécution de la commande :
int nbLignesAffectees = (int)cmd3.ExecuteNonQuery();
```

Objet Command - paramètres

117

- Passage de paramètres à une commande : collection de SqlParameter

```
SqlCommand cmd3 = new SqlCommand();
cmd3.Connection = cnx;
cmd3.CommandText = "UPDATE ARTICLE SET prix = 20 WHERE id = @idArticle";
// ajout d'un paramètre :
SqlParameter p_idArticle = new SqlParameter("@idArticle", 1);
cmd3.Parameters.Add(p_idArticle);

cmd3.CommandType = System.Data.CommandType.Text;
int nbLignesAffectees = (int)cmd3.ExecuteNonQuery();
```

DataReader - Lecture

118

- Dans le cas du ExecuteReader(), on récupère un DataReader
- Méthode Read()
 - true si ligne lue
 - false si fin du jeu d'enregistrements
- Accès aux champs de l'enregistrement en cours par un indexeur (index ou libellé de la colonne)
- Méthode Close() : à appeler à la fin de la lecture

DataReader - Exemple

119

```
SqlCommand cmd = new SqlCommand();
cmd.Connection = cnx;
cmd.CommandText = "SELECT id, libelle, prix FROM ARTICLE";
cmd.CommandType = System.Data.CommandType.Text;
// exécution de la commande :
SqlDataReader dr = cmd.ExecuteReader();
while (dr.Read())
{
    string libelleArticle = (string)dr["libelle"]; // ou dr.GetString(1);
    int idArticle = (int)dr["id"]; // ou dr.GetInt32(0);
    float prixArticle = (float)dr["prix"]; // ou dr.GetFloat(2);
}
dr.Close();
```

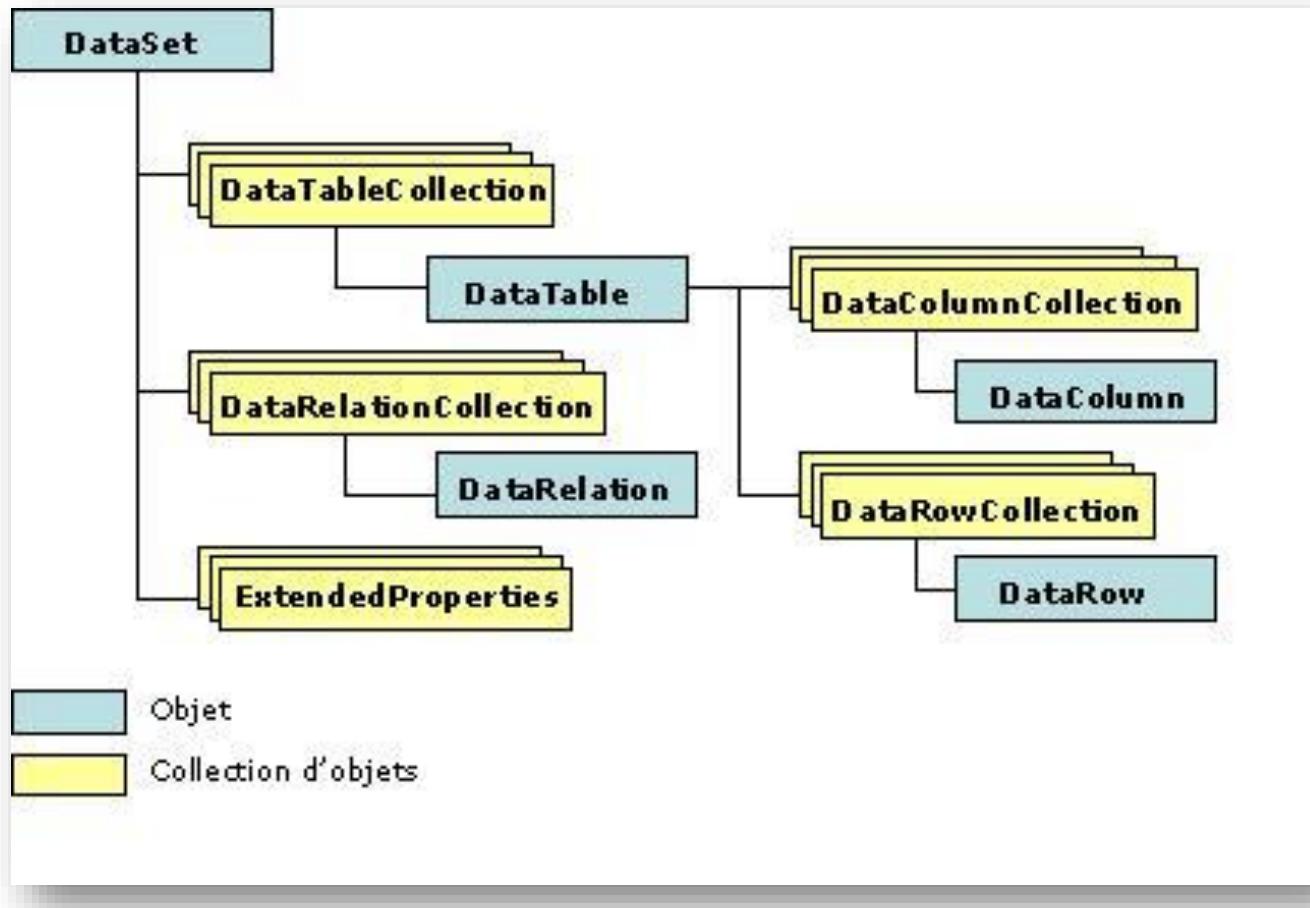
Mode déconnecté – Le DataSet

120

- Utilisation de l'objet DataSet qui représente un extrait de la base de données
- Namespace System.Data
- Accès initial à la base pour créer le DataSet
- Hors connexion, modifications sur le DataSet
- Possibilité de persistance des DataSet (enregistrement sous forme de fichier)
- Synchronisation du DataSet avec la base de données
- Un DataSet peut être alimenté par d'autres moyens qu'une base de données

DataSet

121



DataSet - Chargement

122

- Pour alimenter un DataSet, on utilise l'objet DataAdapter
- Propriété SelectCommand : requête de sélection (objet Command)

```
// Création d'une connexion
SqlConnection cnx = new SqlConnection();
// Affectation d'une chaîne de connexion :
cnx.ConnectionString = System.Configuration.ConfigurationManager
                      .ConnectionStrings["bddMarket"].ConnectionString;
// création du DataAdapter :
SqlDataAdapter da = new SqlDataAdapter();
// Affectation de SelectCommand :
da.SelectCommand = new SqlCommand("SELECT id, libelle, prix FROM ARTICLE", cnx);
// Création d'un DataSet :
DataSet ds = new DataSet();
//Remplissage du DataSet
da.Fill(ds, "Article");
// récupération de la DataTable chargée dans le DataSet :
DataTable dt = ds.Tables["Article"];
```

Modification d'un DataSet

123

Modifications sur le DataSet, en mode non connecté : à ce stade, la base de données n'est pas impactée

```
// Ajout d'un enregistrement :  
DataRow dr = ds.Tables["Article"].NewRow();  
dr["libelle"] = "nouvel article";  
dr["prix"] = 12.5f;  
ds.Tables["Article"].Rows.Add(dr);  
  
// modification d'un enregistrement :  
DataRow dr2 = ds.Tables["Article"].Rows[2];  
dr2.BeginEdit();  
dr2["Libelle"] = "lunettes de piscine";  
dr2.EndEdit();  
  
// suppression d'un enregistrement :  
ds.Tables["Article"].Rows[1].Delete();
```

DataAdapter - Update

124

- Propriété UpdateCommand pour répercuter les modifications :

```
SqlCommand ucmd = new SqlCommand();
ucmd.CommandText = "UPDATE ARTICLE SET libelle = @libelle, prix = @prix WHERE id = @id";
ucmd.Connection = cnx;
// ajout d'un paramètre :
ucmd.Parameters.Add(new SqlParameter("@libelle", SqlDbType.NVarChar, 50, "libelle"));
// variante 1 :
ucmd.Parameters.Add("@prix", SqlDbType.Float, 8, "prix");
// variante 2 :
SqlParameter p_id = new SqlParameter();
p_id.ParameterName = "@id";
p_id.SqlDbType = SqlDbType.Int;
p_id.Size = 4;
p_id.SourceColumn = "id";
ucmd.Parameters.Add(p_id);
// ajout de la commande au DataAdapter :
da.UpdateCommand = ucmd;
```

DataAdapter - Insert

125

- Propriété InsertCommand pour répercuter les insertions :

```
SqlCommand icmd = new SqlCommand();
icmd.CommandText = "INSERT INTO ARTICLE (libelle, prix) VALUES (@libelle, @prix)";
icmd.Connection = cnx;
// ajout d'un paramètre :
icmd.Parameters.Add(new SqlParameter("@libelle", SqlDbType.NVarChar, 50, "libelle"));
// variante 1 :
icmd.Parameters.Add("@prix", SqlDbType.Float, 8, "prix");
// ajout de la commande au DataAdapter :
da.InsertCommand = icmd;
```

DataAdapter - Delete

126

- Propriété DeleteCommand pour répercuter les suppressions :

```
SqlCommand dcmd = new SqlCommand();
dcmd.CommandText = "DELETE FROM ARTICLE WHERE id = @id";
dcmd.Connection = cnx;
// ajout d'un paramètre :
dcmd.Parameters.Add(new SqlParameter("@id", SqlDbType.Int, 4, "id"));
// ajout de la commande au DataAdapter :
da.DeleteCommand = dcmd;
```

CommandBuilder

127

- Alternative à la création manuelle des commandes :
objet CommandBuilder :

```
SqlCommandBuilder b = new SqlCommandBuilder(da);

da.InsertCommand = b.GetInsertCommand();
da.UpdateCommand = b.GetUpdateCommand();
da.DeleteCommand = b.GetDeleteCommand();
```

Envoi des modifications à la base

128

- Méthode `Update()` du `DataAdapter` – *ne pas confondre avec `UpdateCommand`*
- Exécute les `Insert`, `Update` et `Delete` en fonction de l'état du `DataSet`
- Propriété `RowState` d'une `DataRow` :
 - `Unchanged`
 - `Added`
 - `Deleted`
 - `Modified`

```
// mise à jour de la base :  
da.Update(ds, "Article");
```

Mécanisme de DataBinding.

129

- Liaison entre un contrôle ASP.NET et une source de données (DataSource)
 - Exemples de contrôles liés : DropDownList, GridView, ListView, ...
 - Exemples de DataSources : DataTable, Collection d'objets, tableau, fichier xml...
- 2 étapes :
 - Définition de la DataSource
 - Liaison effective des données (DataBind)

Databinding XML / GridView

130

❑ Fichier xml de données

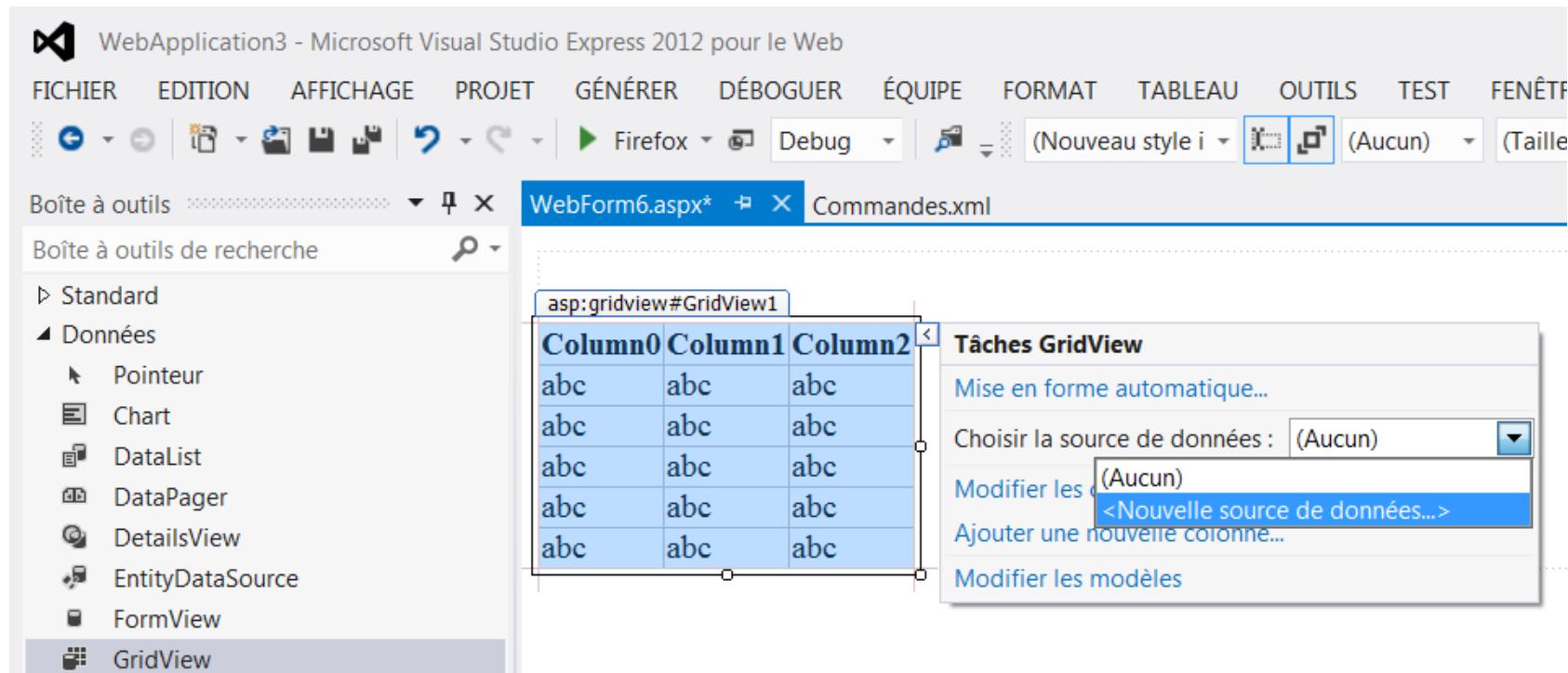
Commandes.xml ✎ X

```
<?xml version="1.0" encoding="utf-8" ?>
<commandes>
    <commande id="1" dateCommande="15/06/2013" client="Bill Thorvalds"></commande>
    <commande id="2" dateCommande="15/06/2013" client="Steeve Gates"></commande>
    <commande id="3" dateCommande="15/06/2013" client="Linus Jobs"></commande>
    <commande id="4" dateCommande="16/06/2013" client="Bill Thorvalds"></commande>
    <commande id="5" dateCommande="18/06/2013" client="Steeve Gates"></commande>
    <commande id="6" dateCommande="18/06/2013" client="Linus Jobs"></commande>
    <commande id="7" dateCommande="18/06/2013" client="Bill Thorvalds"></commande>
    <commande id="8" dateCommande="18/06/2013" client="Steeve Gates"></commande>
    <commande id="9" dateCommande="20/06/2013" client="Linus Jobs"></commande>
    <commande id="10" dateCommande="20/06/2013" client="Bill Thorvalds"></commande>
    <commande id="11" dateCommande="20/06/2013" client="Steeve Gates"></commande>
    <commande id="12" dateCommande="22/06/2013" client="Bill Thorvalds"></commande>
    <commande id="13" dateCommande="22/06/2013" client="Steeve Gates"></commande>
    <commande id="14" dateCommande="23/06/2013" client="Linus Jobs"></commande>
    <commande id="15" dateCommande="24/06/2013" client="Linus Jobs"></commande>
</commandes>
```

Databinding XML / GridView

131

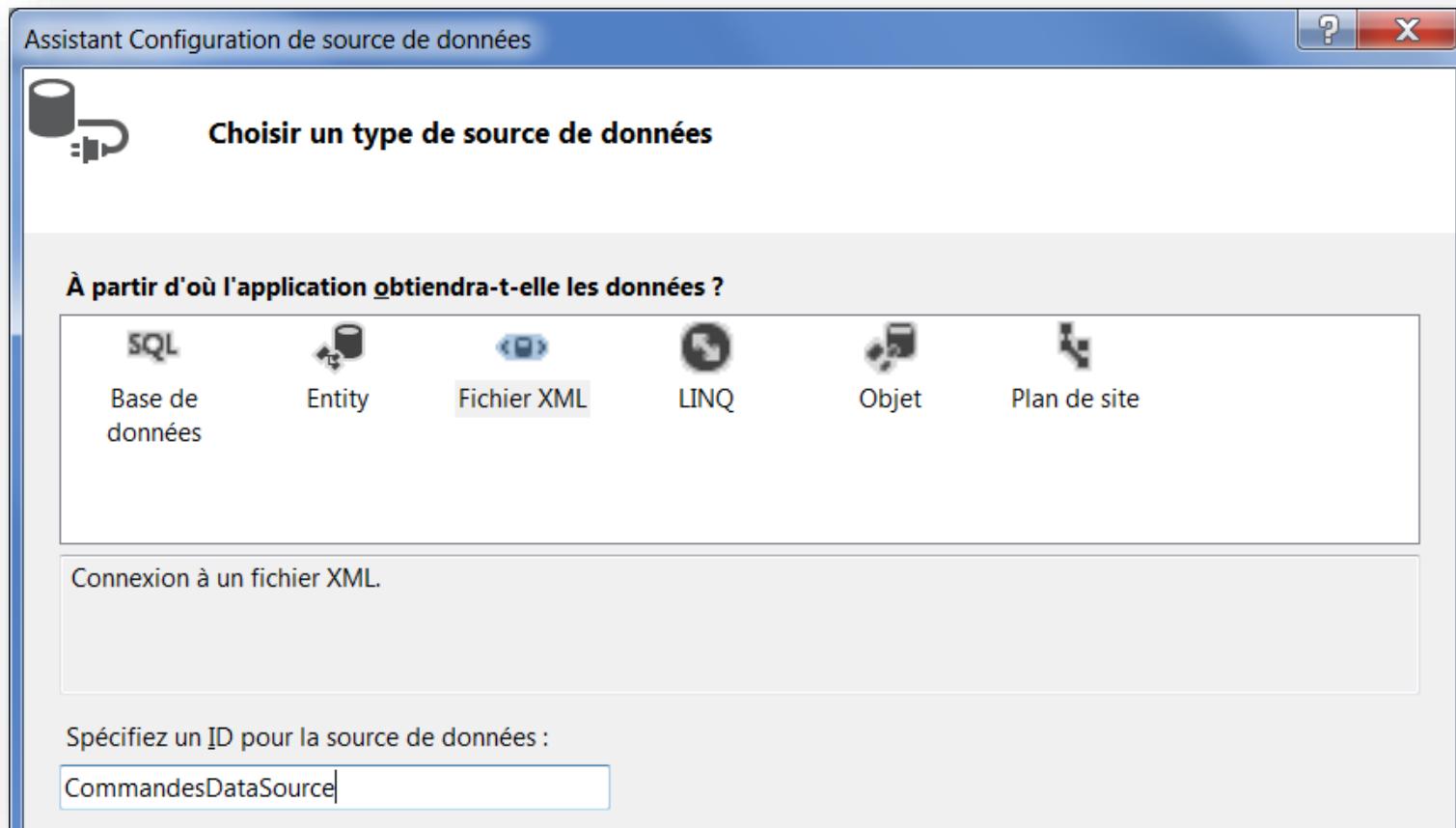
□ Ajout du gridview à la page (mode design)



Databinding XML / GridView

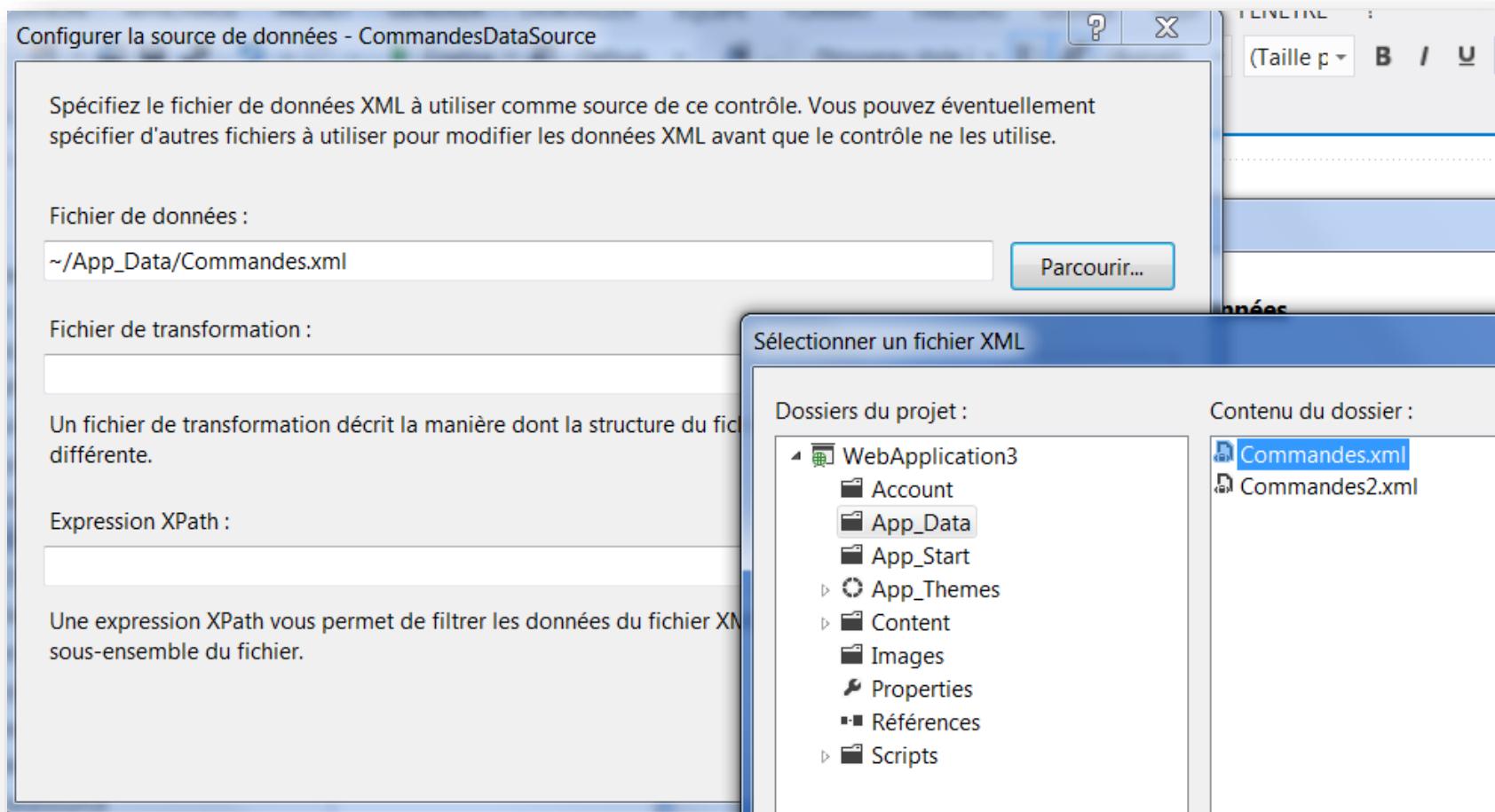
132

□ Choix de la source de données



Databinding XML / GridView

133



Databinding XML / GridView

134

□ Résultat :

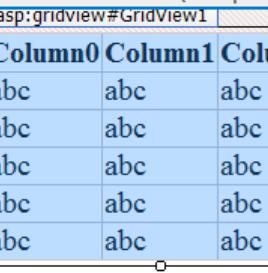
```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataSourceID="CommandesDataSource">
  <Columns>
    <asp:BoundField DataField="id" HeaderText="id" SortExpression="id" />
    <asp:BoundField DataField="dateCommande" HeaderText="dateCommande" SortExpression="dateCommande" />
    <asp:BoundField DataField="client" HeaderText="client" SortExpression="client" />
  </Columns>
</asp:GridView>

<asp:XmlDataSource ID="CommandesDataSource" runat="server" DataFile="~/App_Data/Commandes.xml"></asp:XmlDataSource>
```

id	dateCommande	client
1	15/06/2013	Bill Thorvalds
2	15/06/2013	Steeve Gates
3	15/06/2013	Linus Jobs
4	16/06/2013	Bill Thorvalds

Source de données SQL

135

dbo.ARTICLE : Table(ben-pc.Market) ARTICLE : Requête(ben-pc.Market) WebForm7.aspx*  Tâches GridView

Assistant Configuration de source de données

Choisir un type de source de données

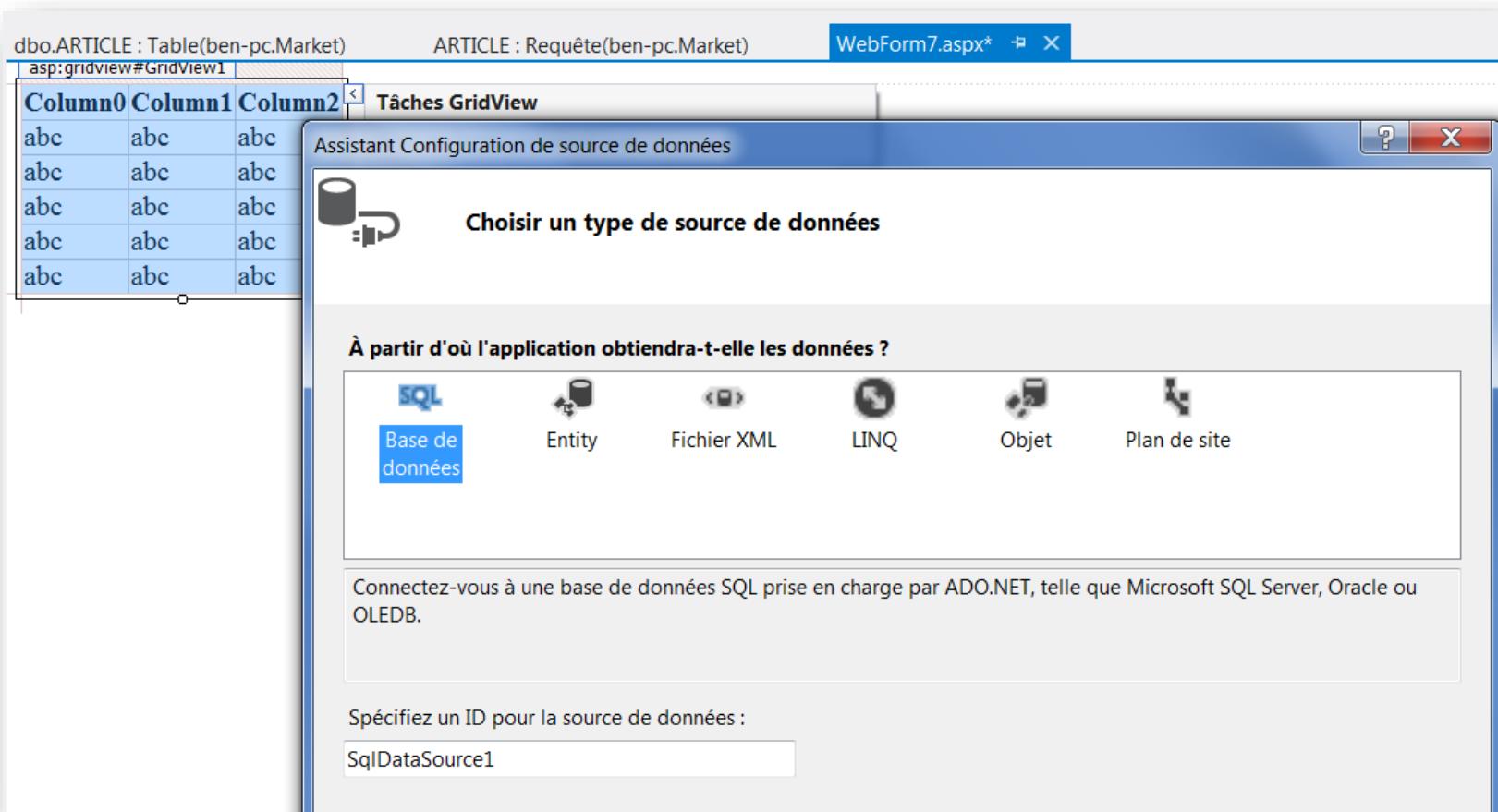
À partir d'où l'application obtiendra-t-elle les données ?

SQL Base de données Entity Fichier XML LINQ Objet Plan de site

Connectez-vous à une base de données SQL prise en charge par ADO.NET, telle que Microsoft SQL Server, Oracle ou OLEDB.

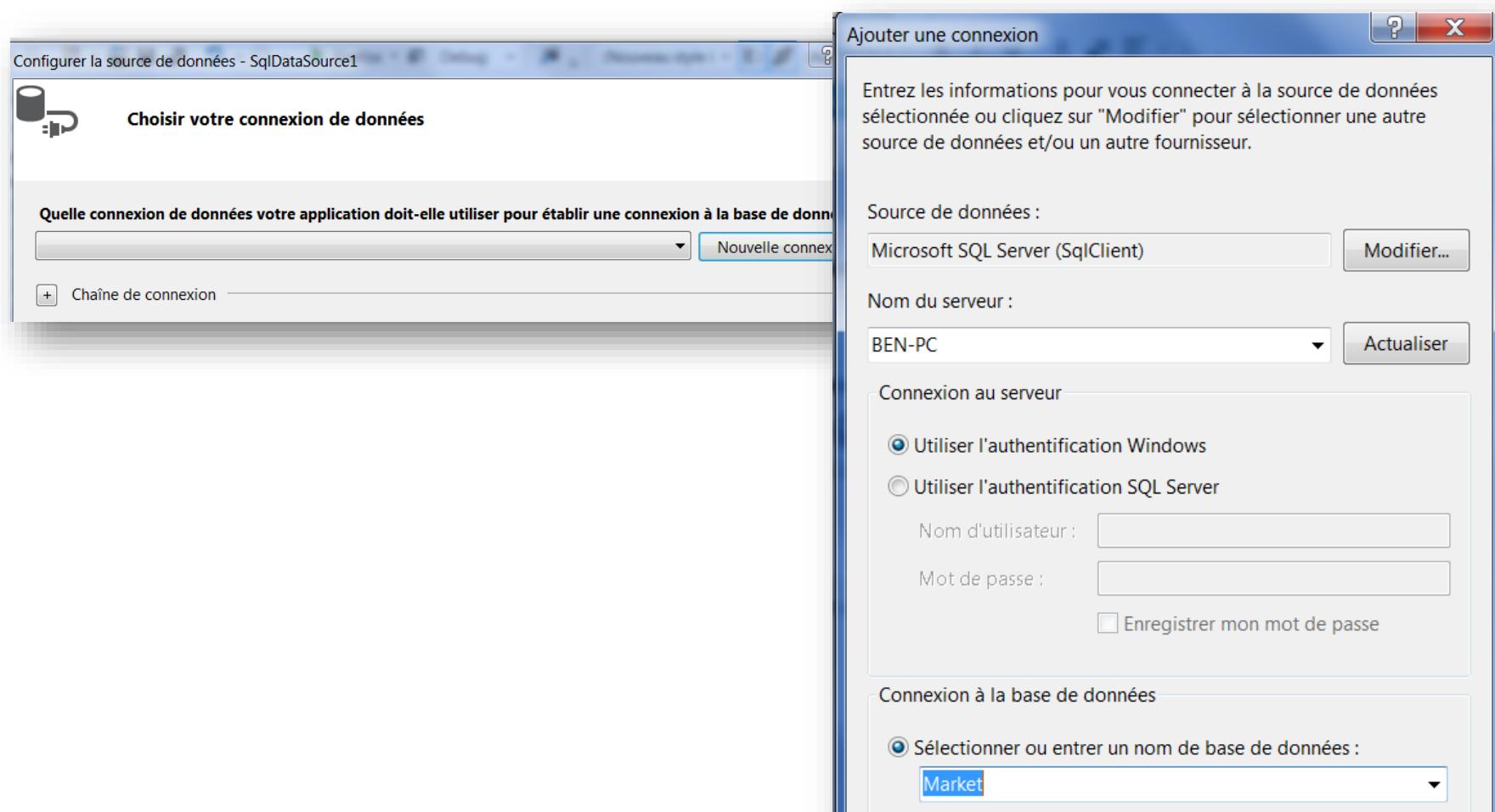
Spécifiez un ID pour la source de données :

SqlDataSource1



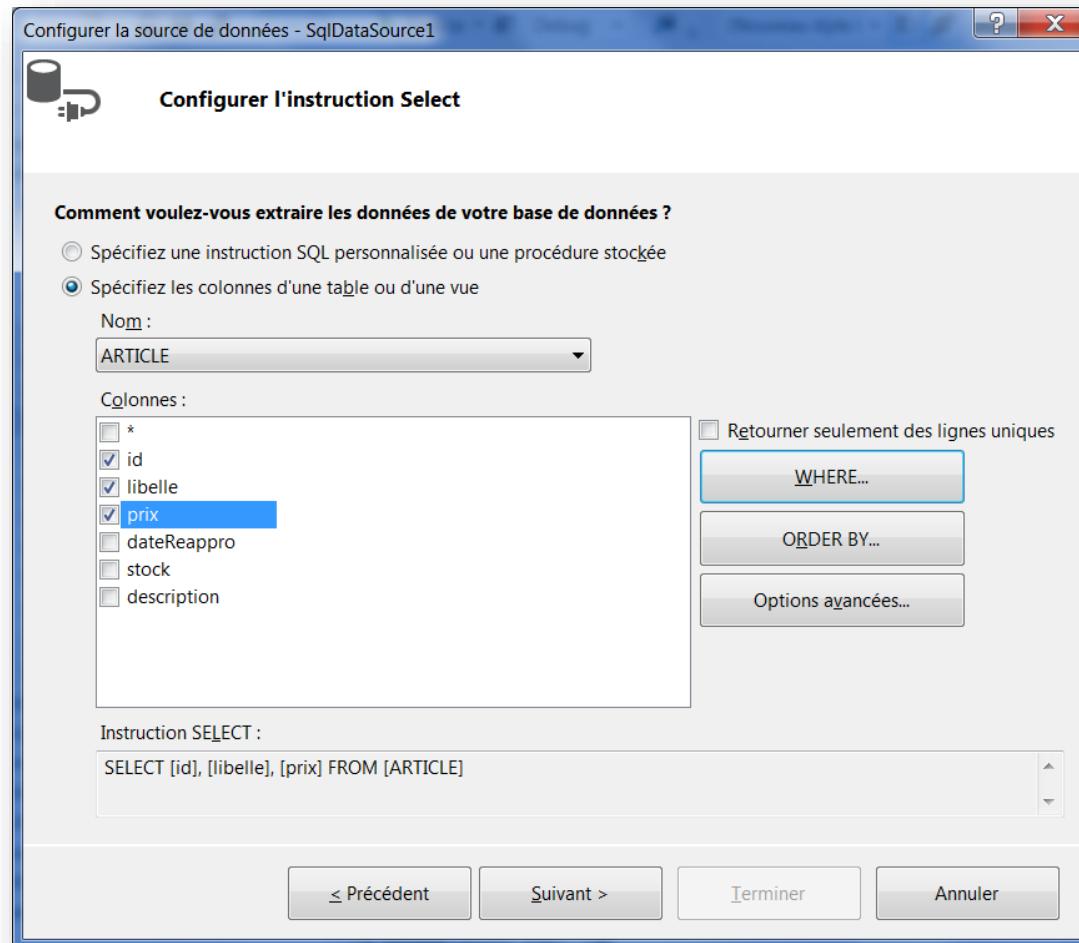
Source de données SQL

136



Source de données SQL

137



Source de données SQL

138

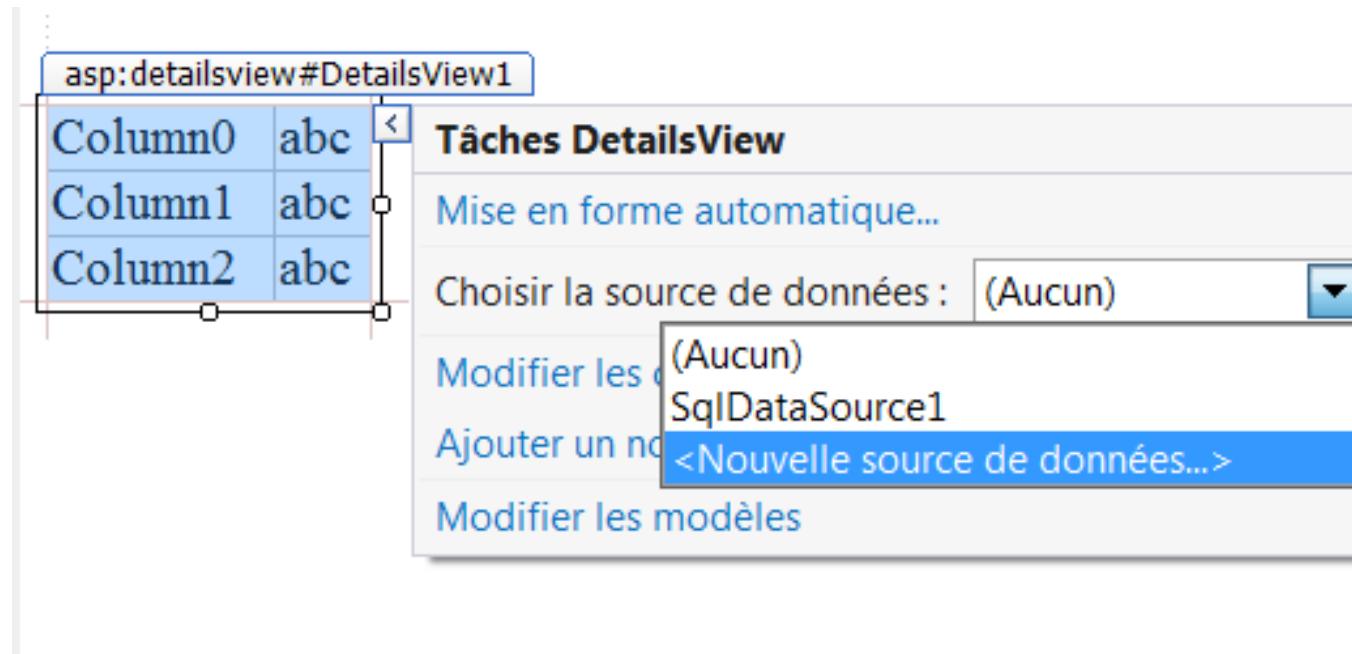
□ Résultat

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataKeyNames="id"
    DataSourceID="SqlDataSource1">
    <Columns>
        <asp:BoundField DataField="id" HeaderText="id" InsertVisible="False" ReadOnly="True"
            SortExpression="id" />
        <asp:BoundField DataField="libelle" HeaderText="libelle" SortExpression="libelle" />
        <asp:BoundField DataField="prix" HeaderText="prix" SortExpression="prix" />
    </Columns>
</asp:GridView>
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="Data Source=BEN-PC;Initial Catalog=Market;Integrated Security=True"
    ProviderName="System.Data.SqlClient"
    SelectCommand="SELECT [id], [libelle], [prix] FROM [ARTICLE]"></asp:SqlDataSource>
```

Afficher une vue Maître/Détail

139

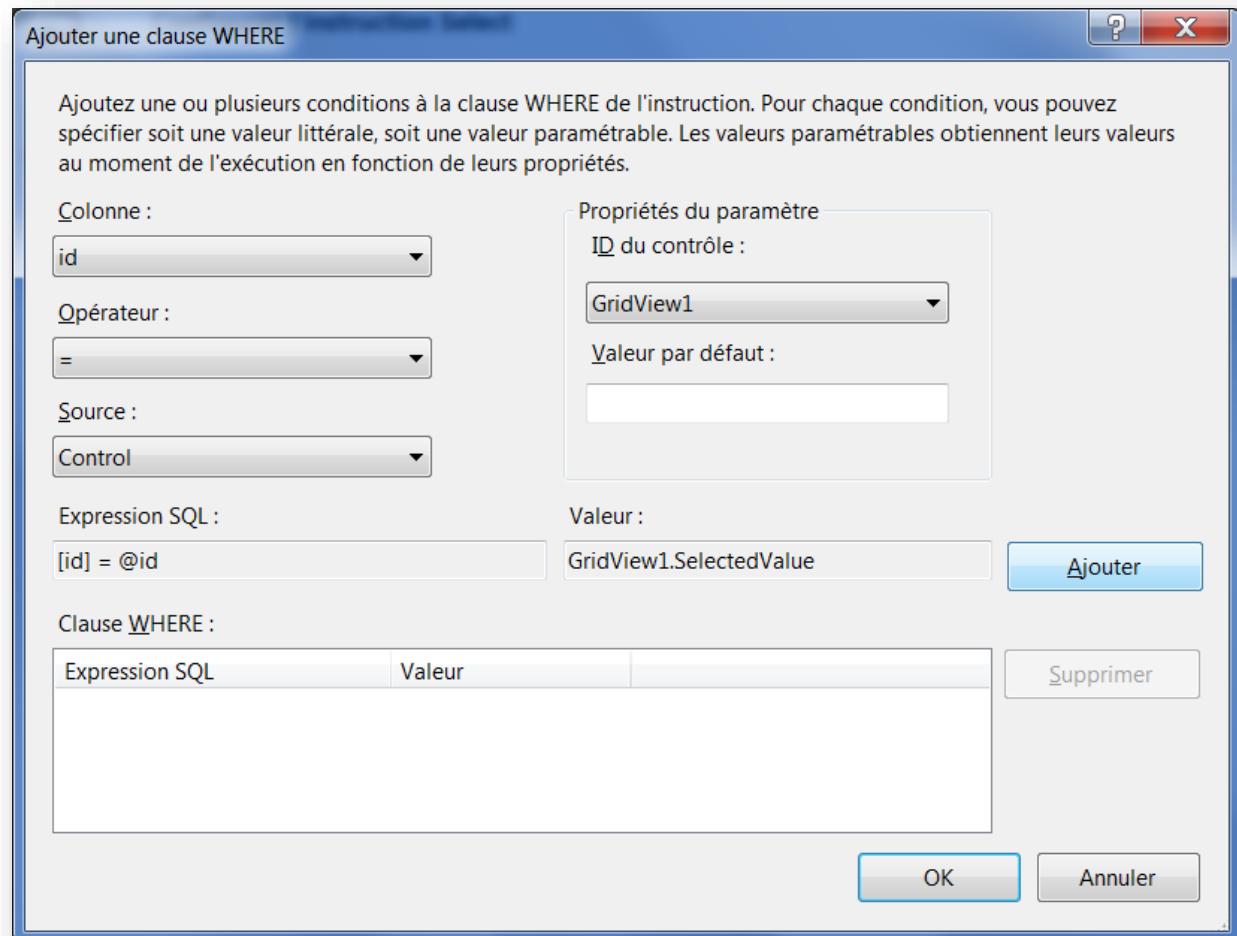
- Afficher les détails d'une ligne de grid view
- Utilisation du contrôle DetailsView



Afficher une vue Maître/Détail

140

Source de données liée au gridView (Clause WHERE)



Afficher une vue Maître/Détail

141

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataKeyNames="id"
    DataSourceID="SqlDataSource1">
    <Columns>
        <%-- Ajout du bouton selectionner au grid view : --%>
        <asp:CommandField ShowSelectButton="True" />
```

```
<asp:DetailsView ID="DetailsView1" runat="server" AutoGenerateRows="False" DataKeyNames="id" DataSourceID="SqlDataSource2"
    Height="50px" Width="125px">
    <Fields>
        <asp:BoundField DataField="id" HeaderText="id" InsertVisible="False" ReadOnly="True" SortExpression="id" />
        <asp:BoundField DataField="libelle" HeaderText="libelle" SortExpression="libelle" />
        <asp:BoundField DataField="prix" HeaderText="prix" SortExpression="prix" />
        <asp:BoundField DataField="dateReappro" HeaderText="dateReappro" SortExpression="dateReappro" />
        <asp:BoundField DataField="stock" HeaderText="stock" SortExpression="stock" />
        <asp:BoundField DataField="description" HeaderText="description" SortExpression="description" />
    </Fields>
</asp:DetailsView>
<asp:SqlDataSource ID="SqlDataSource2" runat="server" ConnectionString="<%$ ConnectionStrings:MarketConnectionString %>"
    SelectCommand="SELECT * FROM [ARTICLE] WHERE ([id] = @id)">
    <SelectParameters>
        <asp:ControlParameter ControlID="GridView1" Name="id" PropertyName="SelectedValue" Type="Int32" />
    </SelectParameters>
</asp:SqlDataSource>
```

Travaux pratiques

142

- Utilisation des contrôles SqlDataSource, GridView et DetailsView pour présenter les données d'entreprise (par exemple les articles d'un catalogue d'un site de e-commerce).

Gestion de la sécurité

Les types d'attaques

144

- Un site web est plus exposé aux attaques qu'une application standard
- Plusieurs types d'attaques : Injections SQL, Spoofing, Répudiation, déni de service...
- Il existe des moyens de se prémunir de ces attaques

Injections SQL

145

- Consiste à envoyer des instructions SQL dans un champ de saisie
- Exemple : formulaire de login
- La requête sous jacente est :
 - `SELECT id FROM Users WHERE name = 'username'`
`AND password = 'password';`
- Saisie de « dupont' -- » dans le champ login >>
 - `SELECT id FROM Users WHERE name = 'dupont' -- '`
`AND password = 'password';`

Injections SQL : solutions

146

- Toujours vérifier les entrées utilisateur, coté client ET serveur (vérifications de type, expressions régulières...)
- Ne jamais créer de requêtes SQL dynamiques à partir de concaténations de chaînes de caractères
- Préférer les procédures stockées ou les commandes paramétrées
- Les informations sensibles doivent être stockées de manière encryptée
- L'application ne doit jamais accéder à la base avec les droits d'administrateur

Autres types d'attaques

147

- Spoofing = Usurpation d'identité (IP, cookies...)
 - Utiliser des mécanismes d'authentification forte
 - Protéger les cookies par SSL (Secure Socket Layer)
- Falsification de données (tampering)
 - Utiliser des clés de hashage pour vérifier l'intégrité des données
- Répudiation : attaques « contre la responsabilité »
 - Utiliser des signatures digitales, sécuriser les logs

Contrôle applicatif des accès et des droits.

148

- ASP.net propose des mécanismes de gestion de l'authentification, des rôles et des autorisations d'accès
- Authentification :
 - Windows (compte utilisateur)
 - Forms (Formulaires de login / password)
- Autorisations :
 - Paramétrage des droits d'accès aux différentes ressources du site

Gestion de la sécurité basée sur les formulaires

149

□ Crédit d'un formulaire de login (login.aspx) :

```
<form id="form1" runat="server">


username : <asp:TextBox ID="txtUsername" runat="server"></asp:TextBox>
    password : <asp:TextBox ID="txtPassword" runat="server"></asp:TextBox>
    <asp:Button ID="btnLogin" runat="server" Text="Login" OnClick="btnLogin_Click" />


</form>
```

```
using System.Web.Security;

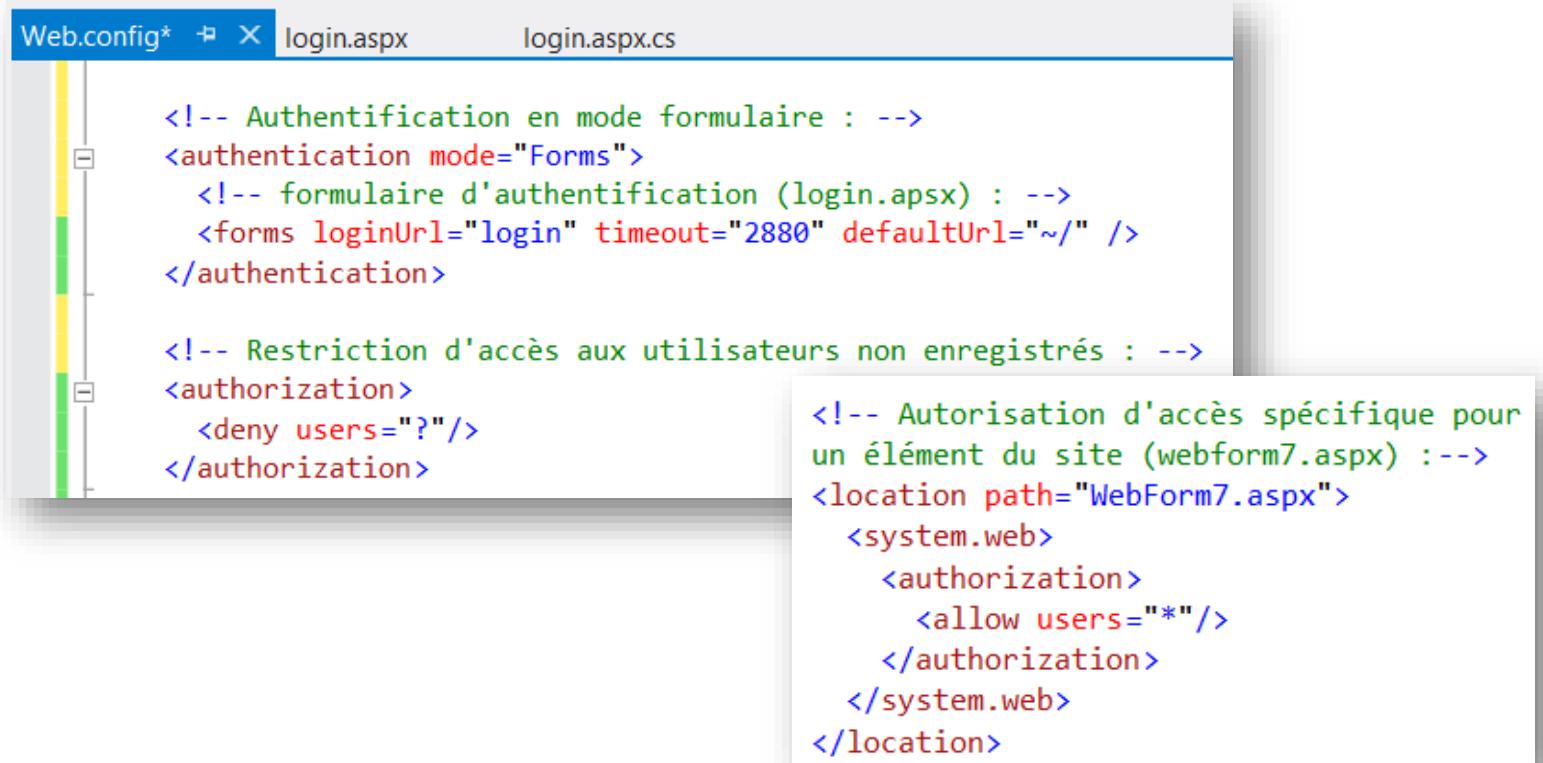
protected void btnLogin_Click(object sender, EventArgs e)
{
    if (txtUsername.Text == "admin" && txtPassword.Text == "test")
    {
        FormsAuthentication.RedirectFromLoginPage("admin", false);
    }
}
```

Gestion de la sécurité basée sur les formulaires

150

□ Configuration du site (fichier web.config) :

- deny / allow : autoriser / refuser
- Users = ? / * : inconnus / tous



```
<!-- Authentification en mode formulaire : -->
<authentication mode="Forms">
    <!-- formulaire d'authentification (login.aspx) : -->
    <forms loginUrl="login" timeout="2880" defaultUrl="/" />
</authentication>

<!-- Restriction d'accès aux utilisateurs non enregistrés : -->
<authorization>
    <deny users="?" />
</authorization>

<!-- Autorisation d'accès spécifique pour
     un élément du site (webform7.aspx) :-->
<location path="WebForm7.aspx">
    <system.web>
        <authorization>
            <allow users="*" />
        </authorization>
    </system.web>
</location>
```

Gestion de la sécurité basée sur les formulaires

151

```
// Informations sur l'utilisateur courant :  
  
string s = User.Identity.Name; // nom d'utilisateur  
bool b = User.Identity.IsAuthenticated; // indique si l'utilisateur est authentifié  
  
// Gestion des services d'authentification :  
  
// redirige l'utilisateur authentifié avec ou sans persistance de session (true/false) :  
FormsAuthentication.RedirectFromLoginPage(s, true);  
// déconnecte l'utilisateur :  
FormsAuthentication.SignOut();
```

Travaux pratiques

152

- **Mise en œuvre d'une authentification et d'un accès restreint, ainsi que d'une restriction d'accès pour un dossier Web.**

Configuration et déploiement

Gestion des exceptions.

154

- Le fichier global.asax permet de centraliser la gestion des exceptions, par exemple pour logger l'erreur ou rediriger vers une page personnalisée.

```
void Application_Error(object sender, EventArgs e)
{
    Exception exc = Server.GetLastError();
    Log.Add(exc.Message);
    Response.Redirect("~/MaPageErreur.aspx");
}
```

- Alternative : Elément CustomErrors du web.config

```
<!-- mode : On (activé) / Off (Message standard) / RemoteOnly (utilisateurs distants seulement) -->
<customErrors mode="On" defaultRedirect="MaPageErreur.aspx">
    <error statusCode="404" redirect="NonTrouve.aspx"/>
</customErrors>
```

Paramètres : web.config

155

- Fichier XML de configuration d'une application web
- Organisé en sections (configSections)
- Quelques sections fréquentes :
 - appSettings : paramètres personnalisés
 - connexionStrings : chaînes de connexions aux bases de données
 - system.web
 - Compilation : mode de compilation (debug, version framework...)
 - Authentication : mode d'authentification
 - Membership, profile, roleManager : gestion intégrée des comptes
- Paramètres accessibles dans le code via le namespace System.Configuration (ConnectionStringSettings, ConfigurationSettings, ...)

Compilation de l'application

156

- 2 répertoires à la racine du projet : Bin et Obj
 - Obj : éléments compilés avant édition des liens
 - Bin : assemblies compilées (dll)
- Chaque répertoire peut contenir 2 sous répertoires : Debug et Release, qui contiennent respectivement les versions de debug et de release
- Dans le répertoire Debug, on trouve un fichier pdb (Program Debug Database) pour chaque assembly : informations permettant l'exécution en mode debug

Le Web Administration Tool intégré à Visual Studio.

157

- Menu projet/configuration asp.net
- Utilitaire permettant de gérer la configuration du site (utilisateurs, rôles, paramètres personnalisés...)



The screenshot shows the ASP.NET Web Administration Tool interface. At the top, there is a blue header bar with the Microsoft ASP.NET logo and the text "Outil Administration de site Web". Below the header is a navigation menu with four items: "Accueil", "Sécurité", "Application", and "Fournisseur". The "Accueil" item is currently selected. The main content area is titled "Outil Administration de site Web". It displays the following information:
Application :/
Nom de l'utilisateur actuel :BEN-PC\BEN

Under the "Sécurité" section, there is a description: "Vous permet de définir et de modifier les utilisateurs, les rôles et les autorisations d'accès de votre site." and "Utilisateurs existants : 0".
Below the "Sécurité" section, there are two more sections: "Configuration de l'application" (with the description "Vous permet de gérer les paramètres de configuration de votre application.") and "Configuration de fournisseur" (with the description "Vous permet de spécifier où et comment stocker les données d'administration utilisées par votre site Web."). The "Configuration de l'application" section is highlighted with a purple background.

Sauvegarde des paramètres de l'application.

158

- Possibilité de stocker des paramètres personnalisés dans le `web.config`, être d'y accéder dans le code :

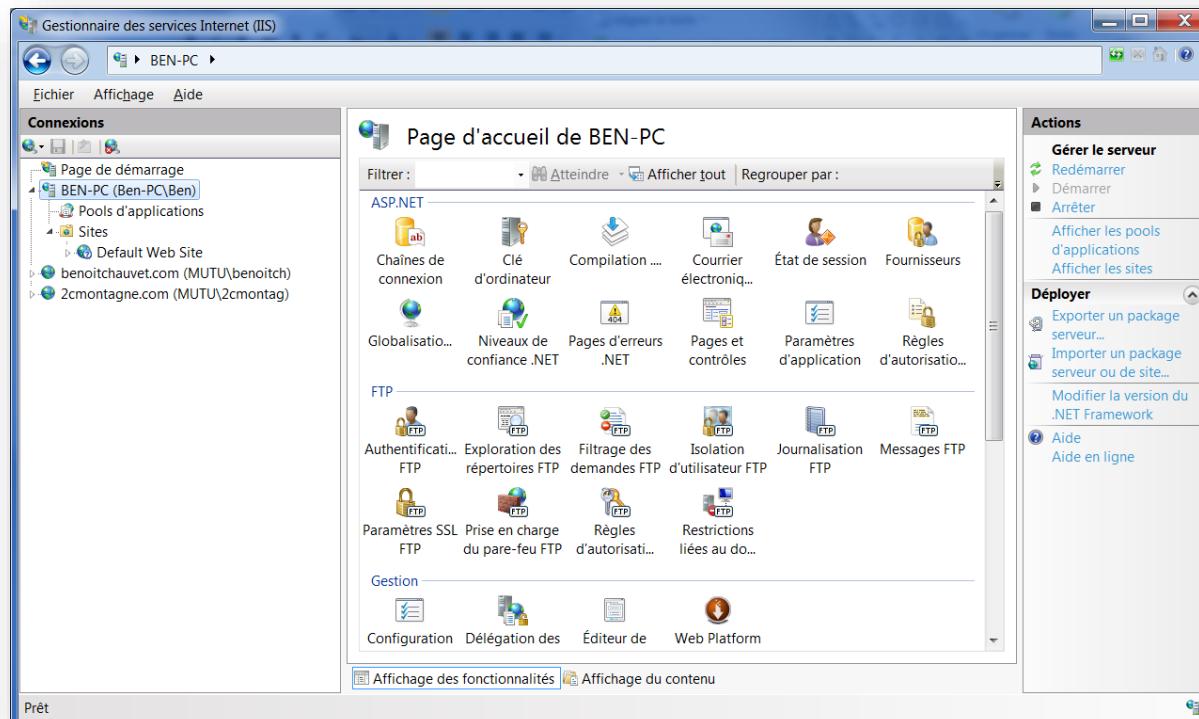
```
<appSettings>
  <add key="StockageFichiers" value="c:\monRépertoire"/>
  <add key="NomDuSite" value="Ma Boutique"/>
</appSettings>
```

```
// Lecture des paramètres :
System.Configuration.ConfigurationManager.AppSettings["NomDuSite"];
```

Le serveur Web : présentation de IIS.

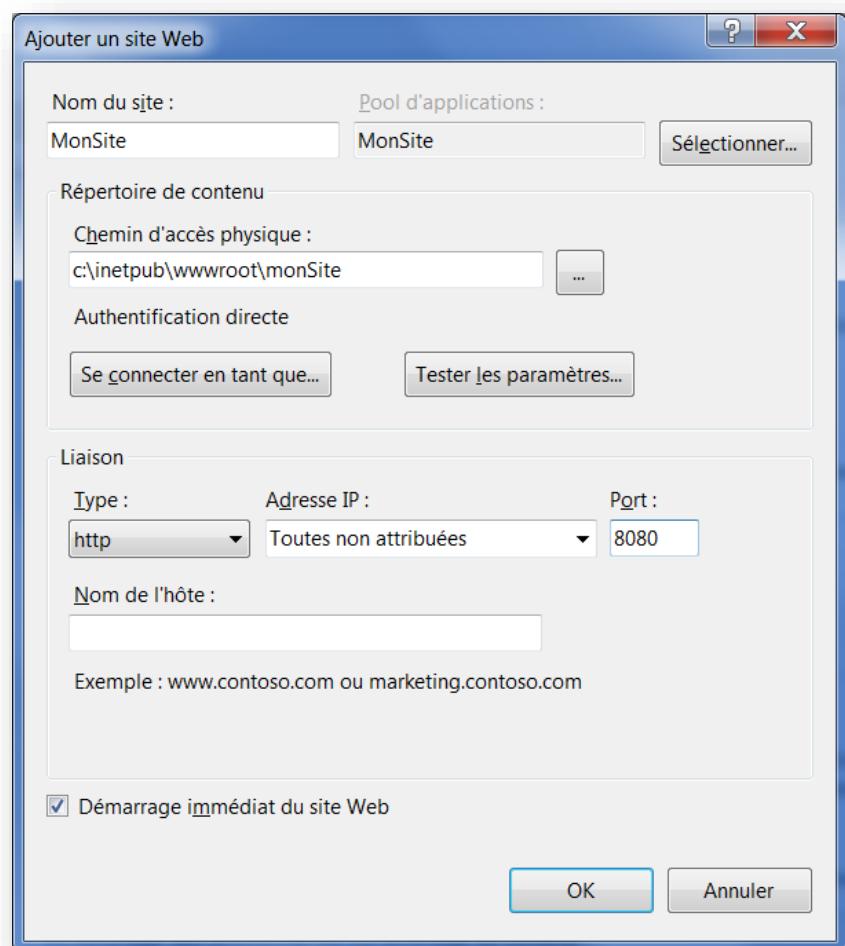
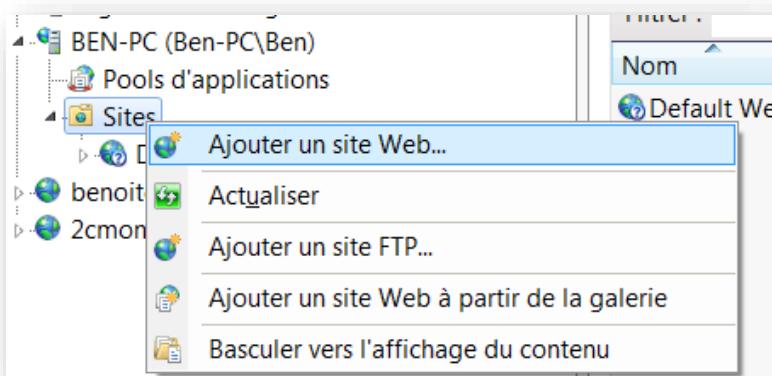
159

- Serveur web de microsoft : Internet Information Services. Version actuelle : IIS 7
- Stockage et exécution des applications web.



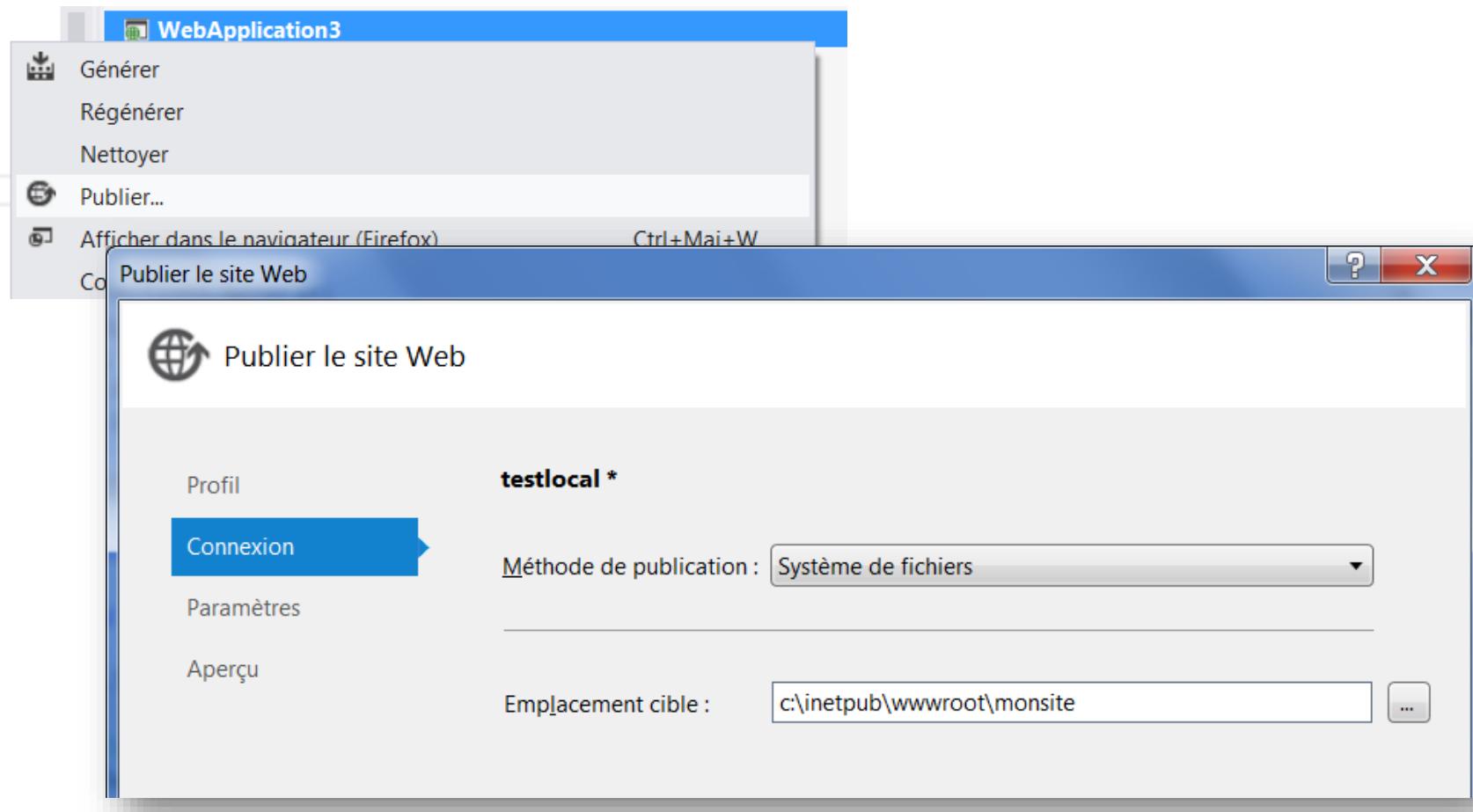
IIS : Création d'un site web

160



Publication de l'application.

161



Utilisation des services Web

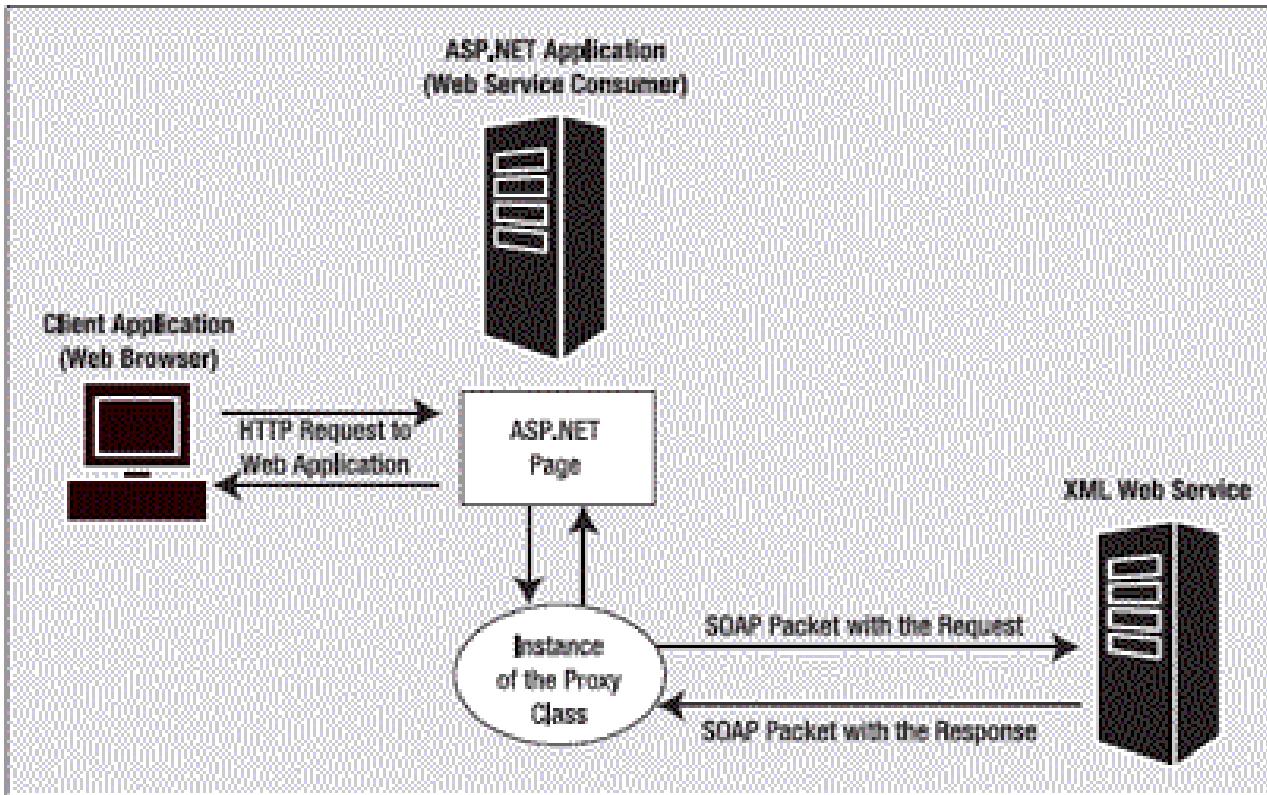
Principe des web services

163

- Fonctionnalités exécutables à distance, exposées via un registre
- Appel via http
- Format des données : XML ou JSON (JavaScript Object Notation)
- Protocole : SOAP (Simple Object Access Protocol) ou REST (REpresentational State Transfer)
- Description de l'interface : langage WSDL (Web Service Description Language).
- Service d'annuaire : UDDI (Universal Description Discovery and Integration)

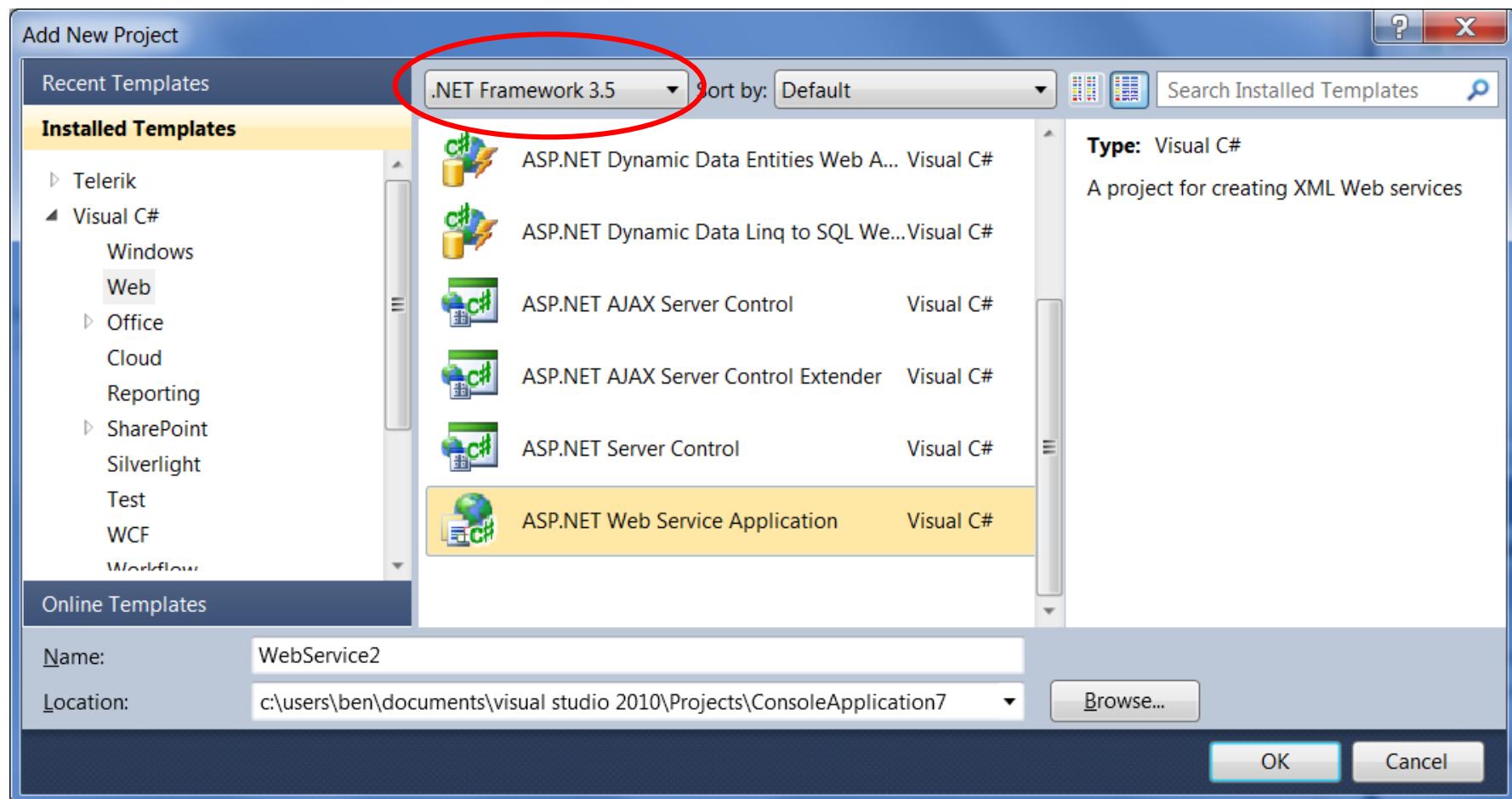
Appel d'un service Web à partir d'une requête HTTP et d'un proxy.

164



Services web ASP.NET - Crédit

165



Services web ASP.NET - Cration

166

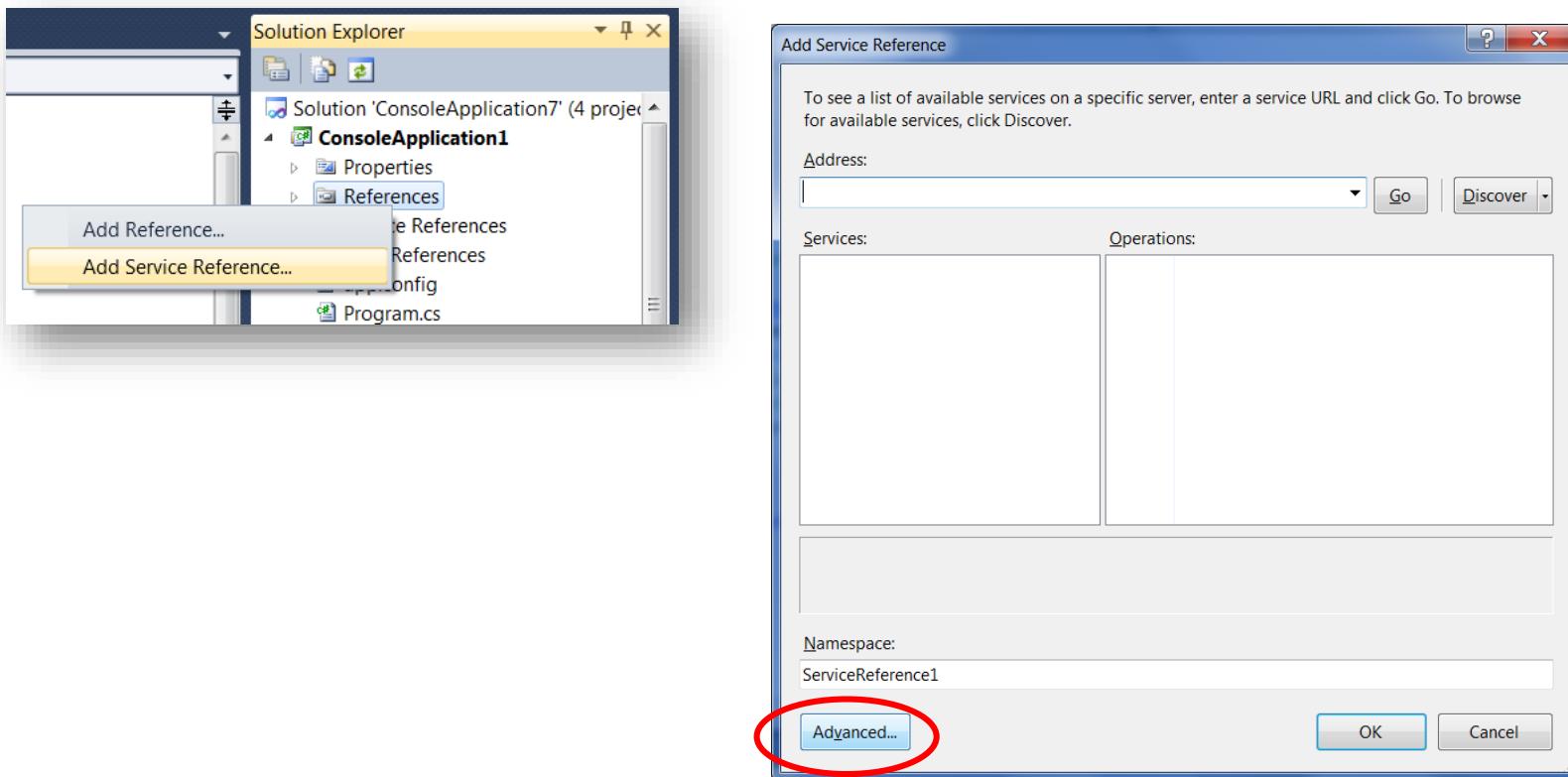
□ Implmentation du service web (Service1.asmx.cs)

```
/// <summary>
/// Summary description for Service1
/// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
[System.ComponentModel.ToolboxItem(false)]
// To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment
// [System.Web.Script.Services.ScriptService]
public class Service1 : System.Web.Services.WebService
{
    [WebMethod]
    public int Addition(int a, int b)
    {
        return a + b;
    }
}
```

Services Web - Utilisation

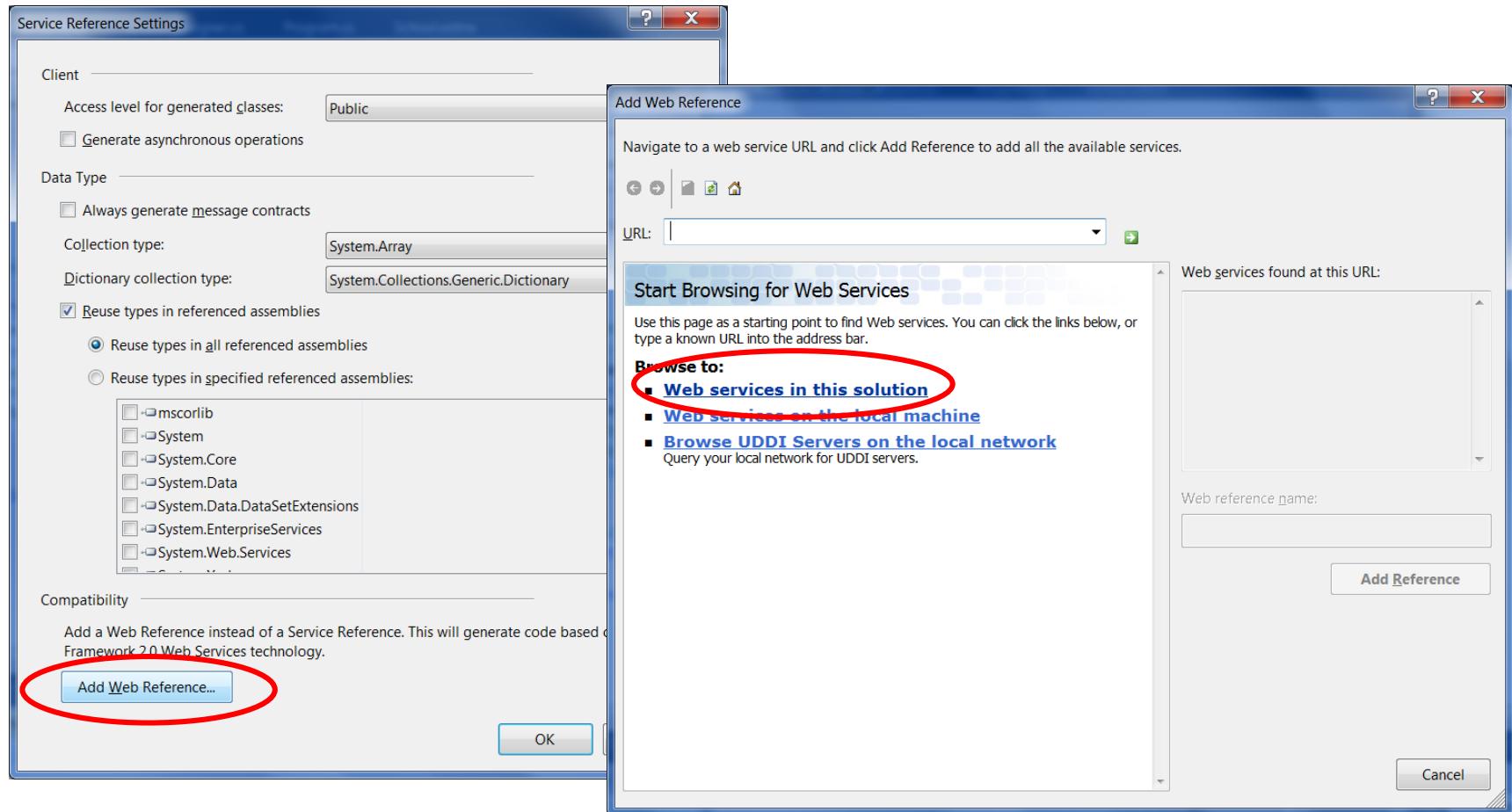
167

□ Ajout d'une référence au service dans le projet client



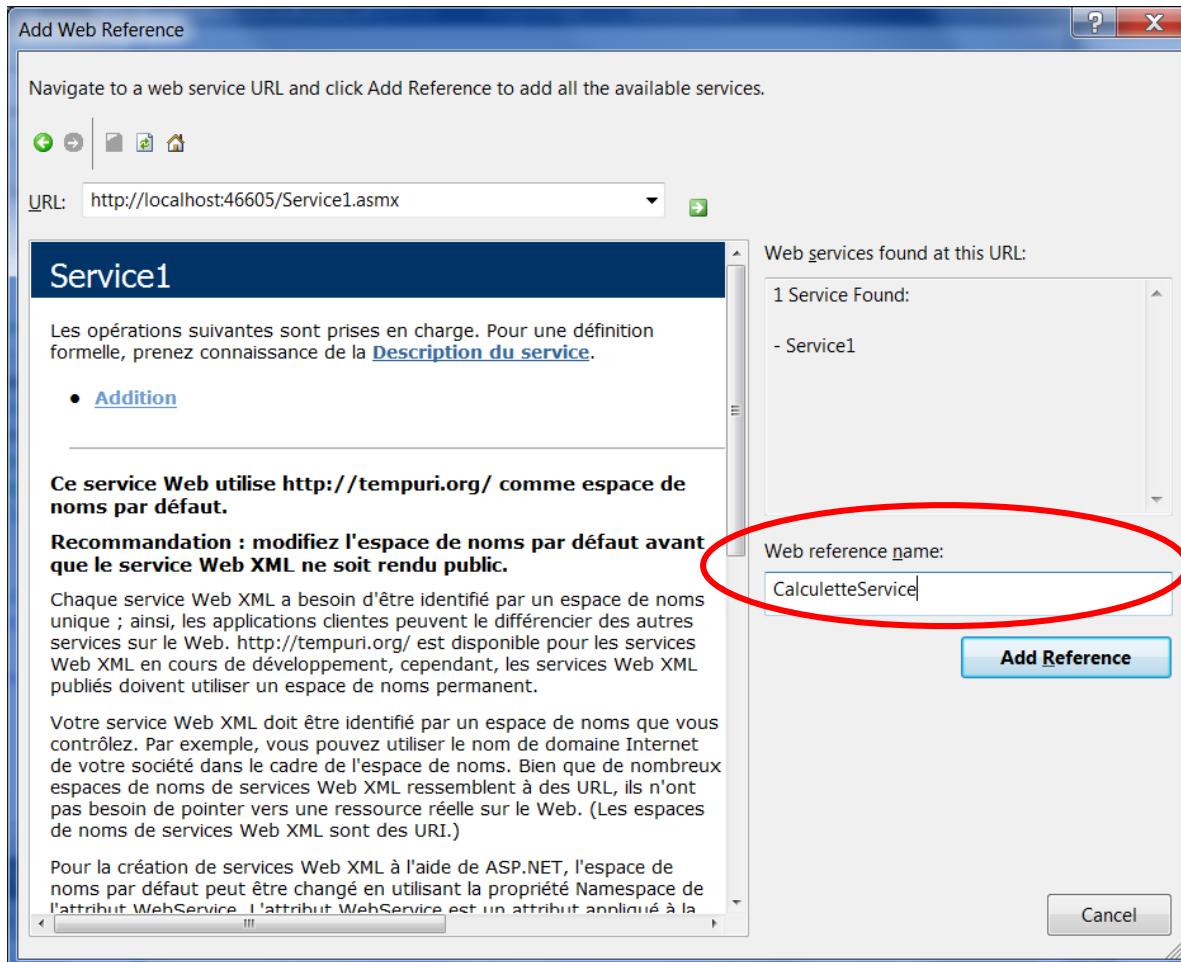
Services Web - Utilisation

168



Services Web - Utilisation

169



Services Web - Utilisation

170

- Appel des méthodes du service dans le programme client :

```
static void Main(string[] args)
{
    CalculetteService.Service1 calc = new CalculetteService.Service1();

    int resultat = calc.Addition(3, 4);
}
```

Travaux pratiques

171

- Ecriture d'un Web Service simple (recherche dans une base de données) et interrogation via une page ASP.NET et un proxy.