INFORMATIQUE
TECHNIQUE

Expert

ASP.NET avec C# sous Visual Studio 2019

Conception et développement d'applications web

En téléchargement

exemples de code

Version en ligne
OFFERTE !
pendant 1 an

Brice-Arnaud GUÉRIN

eni

Table des matières

Les éléments à télécharger sont disponibles à l'adresse suivante :
http://www.editions-eni.fr
Saisissez la référence ENI de l'ouvrage **EI19CASP** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Chapitre 2
Les sites web ASP.NET

Chapitre 3
Les Web Forms

Chapitre 4
Les sites web MVC

Chapitre 5
ASP.NET Core

Chapitre 6
L'accès aux données avec ADO.NET

Chapitre 7
Gestion de l'état

Chapitre 8
Personnalisation et sécurisation

Chapitre 3
Les Web Forms

1. Présentation des Web Forms

Les formulaires web (Web Forms) représentent la partie la plus visible des sites web ASP.NET et par conséquent la plus populaire. Ils reposent sur un partage des responsabilités de type **MVC** : modèle, vue, contrôleur. Lorsqu'un formulaire est écrit en utilisant le style **code séparé**, la page HTML .aspx est chargée de l'affichage (vue), la classe C# porte les données et effectue des calculs (modèle), tandis que le serveur d'applications ASP.NET coordonne l'ensemble (contrôleur). Cette analyse rassurera sans doute les développeurs Java quant à l'organisation des sites web ASP.NET.

D'un autre côté, les formulaires web sont le résultat de la transposition par Microsoft du modèle Visual Basic 6 en une façon originale et productive de développer des interfaces graphiques sur support Internet. Le succès de ce modèle est tel que Sun l'a repris à son compte concernant la technologie développement web JSF (*Java Server Faces*).

1.1 Structure d'une page ASPX

Le chapitre Les sites web ASP.NET a mis à jour la structure d'une page ASPX sous l'angle du modèle de compilation. Il s'agit maintenant d'exposer sa structure logique.

Étudions le code figurant dans une page Default.aspx :

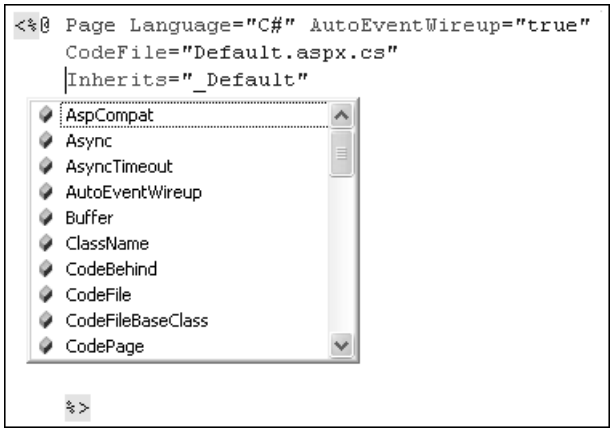
```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="_Default" %>

<!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>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
    </form>
</body>
</html>
```

Ce code est constitué de trois parties : une directive de page, une déclaration de DTD et du code XHTML.

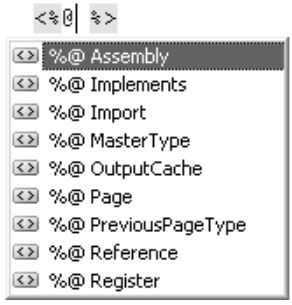
La directive de page

Les directives organisent la lecture d'une page ASPX par le serveur d'applications. Dans la page Default.aspx, l'attribut **Language** désigne le langage – C#, VB.NET, C++ – utilisé pour écrire les scriptlets. D'autres attributs sont également présents, servant à la communication avec la page de code-behind (**AutoEventWireup**, **CodeFile**, **Inherits**), à appliquer des thèmes, à démarrer les traces... Nous découvrirons l'usage de ces attributs au fur et à mesure de notre étude.

Par chance, Visual Studio propose les différents attributs applicables en utilisant la combinaison de touches [Ctrl][Espace].



D'autres directives sont également disponibles pour puiser des ressources dans l'environnement de la page : stratégies de cache, composants, assemblages, types de pages maîtres...



Les DTD

Les définitions de type de documents (*Document Type Definition*) sont établies par le consortium W3C. Il s'agit d'une norme applicable aux documents SGML, XML et HTML qui fixe les règles syntaxiques et sémantiques de construction d'un document à base de tags (marqueurs).

Les navigateurs sont souvent assez tolérants vis-à-vis du respect des DTD. Avec la version ASP.NET 1.X, le flux HTML de sortie était compatible avec la DTD **HTML transitionnel niveau 4**. Excepté l'attribut MS_POSITIONNING qui n'était pas filtré, le code HTML était tout à fait standard. Il est vrai qu'une page ASPX contient des balises spéciales (`<asp:label>` par exemple) qui sont traduites en une séquence HTML accessible au navigateur.

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

La version 2.0 apporte la conformité avec **XHTML**, une déclinaison assez stricte du langage HTML. Les puristes peuvent se rendre sur le site du W3C et passer une page ASP.NET à la moulinette de vérification située à l'adresse http://validator.w3.org. Les pages doivent être conformes à la DTD annoncée.

Remarque

Attention, pour effectuer ce test, il faut enregistrer le flux HTML à partir du bloc-notes ouvert par la commande affichage source. La fonction Enregistrer sous - Page HTML du navigateur Internet Explorer modifie le fichier et biaise le test.

Pour le développeur de pages web, la conformité avec une version spécifique du langage HTML ne suffit hélas pas à garantir qu'une page aura la même présentation quel que soit le navigateur. D'abord, les navigateurs ont la responsabilité d'interpréter les règles de mise en page comme ils l'entendent. Le langage HTML décrit le contenu mais pas la mise en page. Ensuite, les pages comportent aussi du code JavaScript et des styles CSS qui sont diversement pris en charge par les navigateurs.

Le serveur ASP.NET 2.0 a introduit un autre changement : la notion de schéma de navigateur cible a disparu. Il est vrai que cette directive n'a pas pu accompagner l'évolution des navigateurs cités, sans compter l'apparition d'autres logiciels de navigation. À la place, les sites web ASP.NET possèdent un dossier **App_Browsers** répertoriant les caractéristiques de chaque navigateur. Cet aspect sera étudié en même temps que les composants personnalisés.

Pour certains navigateurs et programmes JavaScript intervenant sur le DOM et qui ne seraient pas compatibles avec la norme XHTML, le serveur d'applications peut être configuré pour revenir au mode HTML transitionnel. La consigne figure dans le Web.config :

```
<xhtmlConformance mode="Legacy"/>
```

L'attribut mode accepte trois valeurs :

Legacy	Ancien format HTML transitionnel
Strict	XHTML strict
Transitional	XHTML transitionnel

Le code XHTML

S'il est vrai que le serveur d'applications ASP.NET 1.X sortait un flux conforme à la DTD HTML 4 transitionnelle, la syntaxe même des pages ASPX mélangeait des séquences HTML avec des séquences XML. Visual Studio 2003 était chargé de contrôler la cohérence de l'ensemble en générant des avertissements si nécessaires, et le serveur d'applications devait opérer une lecture plus qu'attentive (donc coûteuse) pour séparer les séquences HTML et les séquences XML.

Cette fois-ci, l'élément `<html>` contient une référence à l'espace de noms XHTML :

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```

En d'autres termes, les balises d'une page ASPX doivent respecter la syntaxe XHTML. Bien entendu, les balises préfixées par `asp` (contrôles web), `uc` (contrôles utilisateurs) ou `cc` (contrôles personnalisés) ne font pas partie du vocabulaire XHTML. Mais au moins, la syntaxe est plus proche et plus précise. Et le flux de sortie reste en tout état de cause conforme à la DTD annoncée.

Enfin, Visual Studio fait de son mieux pour valider à l'avance les séquences HTML figurant dans une page ASPX. Des messages d'avertissement sont générés pour attirer l'attention du développeur sur une non-conformité.

1.1.1 Style imbriqué, en ligne et séparé

L'organisation d'une page dynamique est une simple question de style. Suivant la nature de la séquence HTML à décrire, il est préférable d'opter pour la version imbriquée ou pour la version en ligne (inline). Seul le style séparé (code-behind) change radicalement et apporte une distinction nette entre la présentation et le calcul. C'est la raison pour laquelle il est privilégié par Visual Studio.

Le style imbriqué

Ce sont les premières générations de pages dynamiques (ASP, PHP) qui ont imposé le style imbriqué. Avec les modèles de composants web ASP.NET, celui-ci n'a plus vraiment cours mais reste applicable. Il peut également servir dans le cas de contrôles à base de modèles tels que des Repeater ou des Data List.

Voici un exemple de code basé sur ce style :

```
<body>
    <form id="form1" runat="server">
    <ul>
    <%
        int i;
        string[] jours = { "Lundi", "Mardi", "Mercredi", "Jeudi",
"Vendredi", "Samedi", "Dimanche" };
        for(i=0; i<jours.Length; i++)
        {
    %>
            <li><%= jours[i] %></li>
    <%  } %>
    </ul>
    </form>
</body>
```

Le développeur doit faire de son mieux pour aligner son code comme s'il s'agissait d'un programme totalement écrit en C#.

Le style en ligne (inline)

Le style imbriqué est plutôt dévolu à la présentation. Il ne convient pas lorsque des traitements sont envisagés. La version en ligne sépare le code C# et le code HTML dans deux parties du même fichier .aspx. Des balises `<script runat="server">` indiquent au compilateur qu'il s'agit de code C#, mais elles pourraient être remplacées par des scriptlets `<% %>`.

```
<%@ Page Language="C#" %>
<script runat="server">
    // contient le code événementiel
    void traiter_click(object sender, EventArgs e)
    {
        message.Text = "Vous avez cliqué !";
    }

</script>

<!-- limite entre le code C# et le code HTML -->

<!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>Style en-ligne</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:Label ID="message" runat="server"></asp:Label>
        <asp:Button ID="cmd" runat="server" Text="Cliquer ici"
```