



Expert
EXPO

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





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.

Avant-propos

Chapitre 1

Visual Studio 2019 et .NET

1. Nouveautés de Visual Studio 2019	15
1.1 Installation	17
1.2 Interface du logiciel	17
1.2.1 La page de démarrage	18
1.2.2 Les fenêtres de Visual Studio	19
1.2.3 Les activités liées au développement	27
1.2.4 Les packages NuGet	33
1.3 Gestion du code	35
1.3.1 Le mode plan et les régions	35
1.3.2 La refabrication (refactoring)	37
1.3.3 Les extraits de code (code snippets)	38
1.4 Documentation	40
2. C# 8 en bref	40
2.1 Classes partielles	41
2.2 Méthodes anonymes	42
2.2.1 Les événements internes	42
2.2.2 Les fonctions auxiliaires	45
2.2.3 Simplifier l'écriture du code	47
2.3 L'inférence de type	49
2.4 Les expressions lambdas	49
2.5 Classes dynamiques et types anonymes	50
2.6 Extension de classes sans héritage	51

2 _____ ASP.NET avec C#

sous Visual Studio 2019

2.7	Types nullables	52
2.8	Itérateurs	53
2.8.1	Itérateur en C#1	54
2.8.2	Itérateur à partir de C#3	55
2.9	Généricité	56
2.9.1	Définir un type générique	57
2.9.2	Spécialisation partielle	59
2.9.3	Utilisation d'un type générique	59
2.9.4	L'espace de noms System.Collections.Generic	60
2.9.5	L'interpolation	60
3.	Les variantes de .NET	61
3.1	.NET Core	61
3.2	.NET Standard	63

Chapitre 2

Les sites web ASP.NET

1.	Le modèle de compilation	65
1.1	Du CGI au modèle ASP.NET 1.X	65
1.1.1	L'interface CGI	66
1.1.2	Les pages dynamiques ASP	69
1.2	Des classes partielles pour les pages	70
1.2.1	Structure d'une page ASPX	70
1.2.2	Modifications d'une page ASPX	72
1.3	Les assemblages référencés	73
1.3.1	Références dynamiques	74
1.3.2	Références explicites dans le fichier Web.config	74
1.4	Le cache de construction	75
1.5	Les applications web de Visual Studio	76
2.	Le rôle du serveur web	78
2.1	Le serveur IIS.	78
2.2	Le serveur de développement ASP.NET	79

3. Le pipeline HTTP de IIS	80
3.1 Fonctionnement de IIS.	80
3.1.1 Premiers pas sous HTTP avec Telnet	80
3.1.2 Détail du traitement IIS.	82
3.2 La classe HttpContext	84
3.3 La classe HttpApplication	85
3.3.1 Cycle de vie de l'application.	85
3.3.2 Ajouter un fichier Global.asax.	86
3.3.3 Créer un module HTTP	90
3.4 Les gestionnaires (handlers) HTTP	92
3.4.1 Créer un handler ASHX.	93
3.4.2 Créer un handler dans une DLL.	96

Chapitre 3 Les Web Forms

1. Présentation des Web Forms	99
1.1 Structure d'une page ASPX	100
1.1.1 Style imbriqué, en ligne et séparé	104
1.1.2 Les scriptlets	107
1.1.3 Hiérarchie des contrôles.	109
1.1.4 Ajouter dynamiquement des contrôles.	112
1.1.5 Objets intrinsèques.	113
1.2 Cycle de vie d'une page.	115
1.2.1 Le cycle nominal.	115
1.2.2 Identifier les requêtes de type postback	118
1.3 Les contrôles web	119
1.3.1 Les balises HTML.	119
1.3.2 L'attribut runat="server"	121
1.3.3 Les contrôles HTML.	122
1.3.4 Les contrôles web	123
1.3.5 Les contrôles à base de modèles (template)	127
1.3.6 Les contrôles utilisateurs et les contrôles personnalisés.	127

1.4	Navigation entre les pages	128
1.4.1	Les liens hypertextes	128
1.4.2	Redirections par serveur	129
1.5	Postback et cross postback	130
1.6	Les callback	132
1.7	Validation des entrées utilisateur	138
1.7.1	Principe de la validation	138
1.7.2	Les contrôles de validation	140
1.7.3	La validation personnalisée	144
1.7.4	La validation discrète	145
2.	Organiser la présentation	149
2.1	Thèmes et skins	149
2.1.1	Les feuilles de style CSS	149
2.1.2	D'autres approches des CSS	150
2.1.3	Les thèmes	152
2.1.4	Les skins	155
2.2	Les contrôles utilisateurs .ascx	157
2.2.1	Créer un contrôle utilisateur	158
2.2.2	Utiliser un contrôle utilisateur	159
2.2.3	Ajouter des propriétés et des événements	160
2.3	Les pages maîtres (master pages)	164
2.3.1	Créer une page maître	165
2.3.2	Créer une page de contenu	168
2.3.3	Programmer les pages maîtres et les pages de contenu	171
2.3.4	Appliquer dynamiquement une page maître	173
3.	Les composants personnalisés	174
3.1	Fonctionnement des composants personnalisés	174
3.1.1	Les types de composants personnalisés (custom controls)	174
3.1.2	Création d'une bibliothèque de composants	175
3.1.3	Mise au point du composant ColoredPad	176
3.1.4	Enregistrement et tests	184

3.2	NumericTextBox, un composant dérivé de TextBox.	186
3.2.1	Création du contrôle	186
3.2.2	Propriétés et événements	186
3.2.3	Rendu	188
3.3	ChartControl, un composant graphique utilisant GDI+	189
3.3.1	Fonctionnement	189
3.3.2	Rendu	191
3.3.3	Intégration et tests	191
3.4	PictureBrowser, un composant basé sur un modèle.	192
3.4.1	Fonctionnement	193
3.4.2	Implémentation du composant	195
3.4.3	Les modèles	196
3.4.4	Le rendu	197
3.4.5	Les événements	200
3.4.6	Informations relatives à la conception dans Visual Studio	201
3.4.7	Utilisation du composant	202
3.5	Des ressources incorporées aux DLL	204
4.	AJAX	206
4.1	Du callback à AJAX	206
4.2	Le gestionnaire de script ScriptManager	207
4.3	Le composant UpdatePanel	210
4.3.1	Fonctionnement	210
4.3.2	Mise en œuvre	211
4.3.3	Gestion des erreurs	212
4.3.4	Les triggers	214
4.4	Le composant UpdateProgress	215
4.5	Le Timer	216
4.6	La programmation objet avec JavaScript	217
4.6.1	Insertion de code JavaScript dans une page	217
4.6.2	Créer des objets et des classes JavaScript	219
4.6.3	Le style AJAX	222
4.6.4	Des classes dérivées.	223

4.6.5	Implémenter des interfaces	224
4.7	Introduction à jQuery	225
4.7.1	Installation	225
4.7.2	Parcourir le DOM	226
4.7.3	Intervenir sur la page	227
4.7.4	Les plugins	229
5.	Les services web en Web Form	231
5.1	Création d'un service web ASMX	232
5.2	Utilisation d'un service web ASMX depuis un Web Form	235

Chapitre 4

Les sites web MVC

1.	L'approche MVC	237
1.1	Le design pattern MVC	237
1.2	Les évolutions de MVC	239
2.	Les sites ASP.NET MVC	239
2.1	Création d'un site	239
2.2	Organisation des répertoires	240
2.3	Création du modèle	241
2.4	Définition du contrôleur	244
2.5	Ajout des vues	246
3.	Définition des routes	249
4.	Aller plus loin	250
4.1	D'une action à l'autre	250
4.2	Mise à jour du modèle et redirection	256
4.3	Validation	256
5.	Le moteur de rendu Razor et les vues	258
5.1	La syntaxe C# dans les vues CSHTML	258
5.1.1	Principes de base	258
5.1.2	Les balises Action	261
5.1.3	Les méthodes de formulaires	263

5.1.4	Créer ses propres extensions HTML	264
5.2	Structure et organisation des vues	265
5.2.1	Les gabarits Layout	265
5.2.2	Les vues partielles	267
5.2.3	Rendu des scripts et des bundles	267
6.	Sécurisation des sites MVC	268
6.1	Authentification	268
6.2	Autorisations	270
7.	Les Single Page Applications (SPA)	272
7.1	Utiliser les Web API	272
7.1.1	Créer un projet Web API	272
7.1.2	Établir un modèle et un contrôleur	274
7.1.3	La page unique	275
7.2	Utiliser KnockOut pour la liaison de données	277

Chapitre 5 ASP.NET Core

1.	Un site web ASP.NET Core	281
1.1	Création du projet	281
1.2	Contenu du projet	283
2.	Configuration	285
2.1	Les fichiers Program et Startup	285
2.1.1	Program	285
2.1.2	La classe Startup	285
2.2	La configuration JSON	288
2.2.1	appSettings.json	288
2.2.2	launchSettings.json	289
2.2.3	Les bundles	290
2.3	Gestion des packages	292
2.4	Application de thèmes avec Bootstrap	294

3. Développement MVC	295
3.1 Les contrôleurs web	295
3.2 Les vues	296
3.3 Les Web API	296
3.3.1 Créer un contrôleur Web API	296
3.3.2 Utiliser un service Web API depuis une page	300
3.4 Le package Identity	302
3.4.1 Activer l'authentification	302
3.4.2 Personnaliser les pages de gestion de compte utilisateur	306
4. Définir des environnements d'exécution	309
4.1 Détection de l'environnement d'exécution	309
4.2 Définition d'environnements	310

Chapitre 6

L'accès aux données avec ADO.NET

1. Les bases d'ADO.NET	313
1.1 Le mode connecté	313
1.1.1 La connexion	314
1.1.2 La commande	316
1.1.3 Le DataReader	318
1.1.4 Les paramètres	321
1.1.5 Les transactions	322
1.2 Les bases de données SQL Server	326
1.2.1 Les déclinaisons du logiciel SQL Server	326
1.2.2 Création de bases	327
1.2.3 Création de tables	330
1.2.4 Les vues	331
1.2.5 Les procédures stockées	332
1.3 Rendre transparent l'accès aux bases	333
1.3.1 Le mode déconnecté	334
1.3.2 DataAdapter et TableAdapter	336

1.3.3 Le mapping objet-relationnel et les frameworks spécialisés	343
1.3.4 Les fabriques ADO.NET	343
2. Accès aux données à base de fournisseurs	347
2.1 Introduction au développement par fournisseurs	347
2.1.1 Contrôles sources de données en mode fournisseur	348
2.1.2 Contrôles de présentation des données	349
2.2 Les sources SqlDataSource et AccessDataSource	350
2.2.1 La requête de sélection	350
2.2.2 Les requêtes de mises à jour	353
2.2.3 Les paramètres	354
2.2.4 Le cache	357
2.3 Le fournisseur ObjectDataSource	358
2.3.1 Principe	358
2.3.2 Mise en œuvre	359
2.3.3 Paramètres de création	363
2.3.4 Gestion du cache	364
2.3.5 Une version avancée	364
2.4 Le fournisseur XmlDataSource	371
2.5 LinqDataSource	375
2.5.1 Un DAO pour LinqDataSource	375
2.5.2 Le contexte de données .edmxl	377
2.5.3 Les événements de LinqDataSource	381
2.6 EntityDataSource	382
2.6.1 Le framework Entity	382
2.6.2 Créer le modèle conceptuel	384
2.6.3 Requêter avec LINQ to Entities	389
3. Les composants graphiques de présentation des données	390
3.1 Le composant GridView	390
3.1.1 Présentation tabulaire des données	390
3.1.2 Les opérations de sélection et de navigation	393
3.1.3 Les clés et les opérations de mise à jour	394
3.1.4 Les formatages et les tris	396

3.1.5	Les colonnes modèles	398
3.1.6	La liaison bidirectionnelle	399
3.1.7	Gérer les jointures.	400
3.2	Le composant DetailsView	404
3.2.1	Présentation du DetailsView	404
3.2.2	Les événements.	405
3.3	Le composant FormView	406

Chapitre 7

Gestion de l'état

1.	Les différents moyens pour maintenir l'état	407
1.1	Les champs cachés	407
1.2	Le ViewState	408
1.2.1	Utiliser le ViewState dans un Web Form	409
1.2.2	Contrôler l'application du ViewState	410
1.3	La chaîne de requêtes (Query String) et les URI	411
1.4	Les cookies.	412
2.	Les sessions	413
2.1	Utilisation de l'objet Session	413
2.1.1	Mémorisation d'un objet et recherche	414
2.1.2	Initialisation de l'objet Session	414
2.1.3	Sécurisation du jeton de session	415
2.2	Sessions sans cookie et délai d'abandon de session.	415
2.2.1	Sessions sans cookie	415
2.2.2	Timeout	416
2.3	Services de conservation des données en session	416
2.3.1	Le processus en mémoire InProc	417
2.3.2	Le service Windows ASP.NET State Service.	418
2.3.3	Le service SQL Server	419
2.3.4	Services personnalisés.	420

3. Les objets Application et Cache.....	421
3.1 L'objet Application.....	421
3.1.1 Utilisation.....	421
3.1.2 Verrouillage.....	421
3.2 Le cache de données d'applications Cache.....	422
3.2.1 Les dépendances de temps.....	422
3.2.2 Le callback.....	424
3.2.3 Les dépendances fichiers.....	425
3.2.4 Les dépendances SQL sous SQL Server.....	426
3.3 Le cache HTML.....	428
3.3.1 Cache de sortie.....	428
3.3.2 Fragments de pages en cache.....	430
3.3.3 Les substitutions.....	430
3.3.4 Les profils de cache.....	431

Chapitre 8 Personnalisation et sécurisation

1. Sécurisation des sites ASP.NET.....	433
1.1 Le modèle de sécurisation du site.....	433
1.1.1 Les objets de la sécurité.....	433
1.1.2 L'authentification.....	434
1.1.3 Les autorisations.....	435
1.2 Sécurisation en mode Windows.....	436
1.2.1 Activation du mode d'authentification.....	436
1.2.2 Configuration de IIS.....	437
1.2.3 Autorisations.....	438
1.3 Sécurisation en mode Forms.....	439
1.3.1 Activation du mode Forms et création d'une page de connexion.....	440
1.3.2 Endossements de rôles.....	442
1.3.3 Le mode Forms sans cookie.....	445
1.3.4 Autorisations.....	445

1.4	Le fournisseur MemberShip	445
1.4.1	Fonctionnement du fournisseur	445
1.4.2	Utiliser AspNetSqlMembershipProvider	448
1.5	Sécurisation en comptes d'utilisateurs individuels	451
1.6	Le répertoire Account	453
1.7	Le référentiel local d'utilisateurs	455
1.8	Activer un référentiel externe	457
1.9	Le fournisseur de rôles	460
1.9.1	AspNetSqlRoleProvider	460
1.9.2	WindowsRoleTokenProvider	461
1.10	Les contrôles intégrés	462
2.	Présentation personnalisée	463
2.1	Les profils utilisateur	463
2.1.1	Formation du profil	463
2.1.2	Utilisation du profil	464
2.1.3	Groupage et types complexes	465
2.2	Navigation au sein du site	467
2.2.1	Le fichier de définition du site	467
2.2.2	Le fournisseur SitemapProvider, l'API Sitemap et le SitemapDataSource	468
2.2.3	Les contrôles associés à la navigation	469
2.2.4	Filtrer l'affichage selon le rôle de l'utilisateur	469
2.3	Internationalisation	471
2.3.1	Les ressources globales	471
2.3.2	Les ressources locales	473
2.3.3	Le composant Localize	474
2.3.4	Localisation des validations	475
3.	Les WebParts	476
3.1	Du site web au portail	476
3.2	Créer un portail	476
3.2.1	Le gestionnaire WebPartManager	477
3.2.2	Les zones WebPartZone	478
3.2.3	Les éléments WebPart	479

3.3	Les contrôles catalogues CatalogZone et PageCatalogPart . . .	480
3.3.1	Le catalogue de zones	480
3.3.2	Un menu pour changer de mode	482
3.3.3	Donner des noms aux éléments	483
3.3.4	Les éditeurs	483
3.4	Créer des éléments personnalisés	485
3.4.1	Créer un WebPart à partir d'un composant utilisateur .	485
3.4.2	Créer un WebPart personnalisé	486
3.4.3	Connecter les éléments	489

Chapitre 9

Configuration, déploiement et administration

1.	Configuration	493
1.1	Héritage de la configuration	493
1.2	Configuration de test et de production	495
1.2.1	Le gestionnaire de configuration de Visual Studio	495
1.2.2	Plusieurs fichiers de configuration Web.config	496
1.2.3	Les pages d'erreurs du fichier Web.config	497
2.	Déploiement des applications ASP.NET	497
2.1	Déploiement manuel	497
2.1.1	Création d'un répertoire virtuel	497
2.1.2	Sélection des fichiers à copier	499
2.1.3	La page par défaut	500
2.2	Déploiement par le système de copie	502
2.3	Déploiement avec Microsoft Azure	506
2.3.1	Création d'un compte Azure	506
2.3.2	Vue d'ensemble de l'interface de gestion des services . .	507
2.3.3	Création d'un projet associé à un compte Azure	508
2.3.4	Développement de l'application	510

3.	Supervision des applications ASP.NET	511
3.1	L'infrastructure de supervision Health Monitoring	511
3.1.1	La hiérarchie des événements web	511
3.1.2	La hiérarchie des fournisseurs	512
3.2	Mise en œuvre dans ASP.NET	512
3.2.1	Déclarer des événements	513
3.2.2	Déclarer des fournisseurs d'écoute	513
3.2.3	Ajouter des règles d'abonnement	513
	Index	515

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>

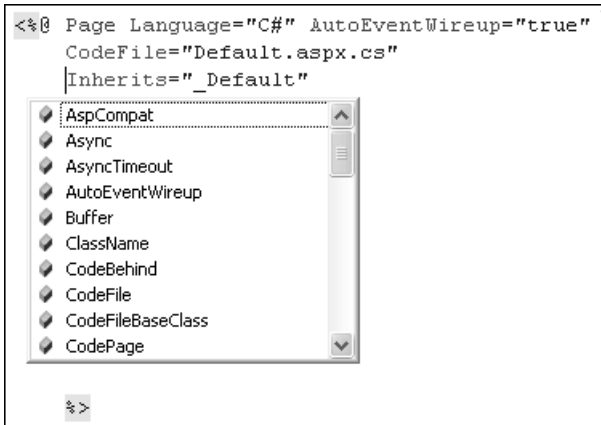
            </div>
        </form>
</body>
</html>
```

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

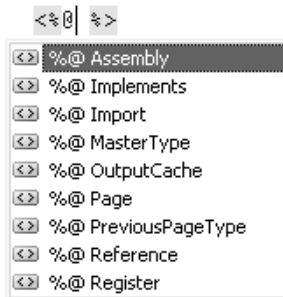
La directive de page

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_POSITIONING` 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">
```