Thomas **Petillon**

les Cahiers du Programmeur

ASP.NET

ISBN : 2-212-11210-6

EYROLLES

Architecture d'une page ASP.NET

3

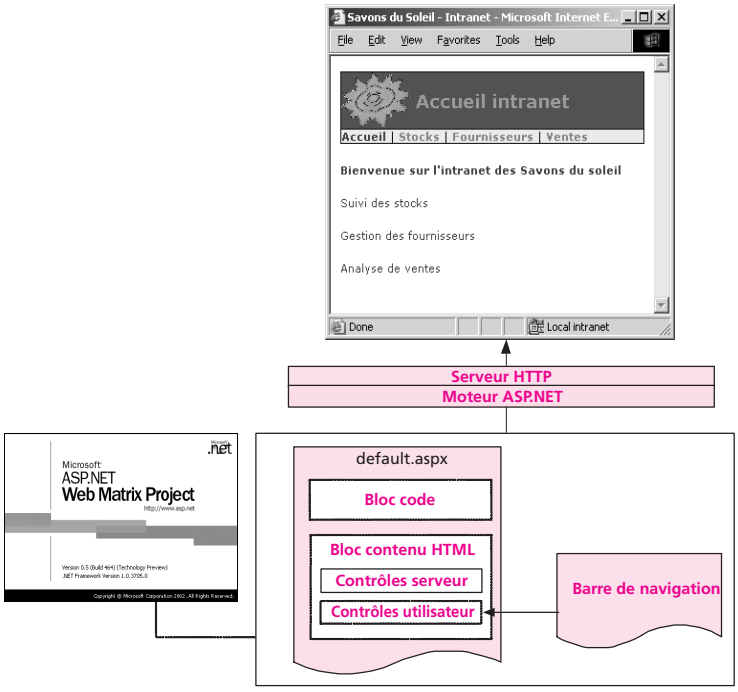
ASP.NET

Contrôle serveur | contrôle utilisateur | Code behind | Web Matrix | attribut CssClass

SOMMAIRE

MOTS-CLÉS

- Contrôle serveur
- Contrôle utilisateur
- Code behind
- Web Matrix
- Directive <%@Page ...%>
- Directive <%@Register...%>
- Directive <%@Control...%>
- Attribut CssClass



Dans ce chapitre, nous présentons l'architecture générale d'une page ASP.NET : quels sont les concepts principaux du nouveau modèle de programmation qui leur est associé, comment s'effectue en pratique la séparation entre le code et le contenu graphique, comment le développement devient plus simple et plus productif grâce à l'utilisation de *contrôles serveur* et *contrôles utilisateur*. Puis nous mettons en pratique ces connaissances pour réaliser la barre de navigation de notre étude de cas.

La page Web vue comme une interface classique

Le principe fondamental d'ASP.NET est de considérer une page Web non plus comme un document HTML mais comme une interface graphique classique d'application client-serveur.

Tout l'intérêt d'ASP.NET est de permettre au développeur d'implémenter une page Web comme un assemblage de *contrôles* graphiques (listes, boutons…) réagissant à des *événements* (changement de sélection dans une liste, clic sur un bouton…), tout en assurant la traduction de cette vision conceptuelle en HTML standard.

Prenons un exemple inspiré de notre étude de cas, l'interface de consultation de la liste des produits des Savons du Soleil, classés par famille de produits :

- Dans le cadre d'une application client-serveur classique, cette interface aurait été implémentée avec un langage objet, sous la forme d'une boîte de dialogue contenant des contrôles – une liste déroulante affichant la liste des familles et une grille de données affichant la liste des produits – avec une cinématique associée à des événements : remplissage des contrôles lors du chargement de l'interface, mise à jour de la liste des produits dans le cas d'un changement de sélection de famille.
- Dans le cadre d'une application Web classique, cette interface aurait été implémentée sous la forme d'une page Web dynamique, contenant du HTML mêlé à du langage de script, sans possibilité de gérer de manière transparente un changement de sélection de famille de produits.

Dans le passé, il fallait donc choisir entre une interface classique, riche en fonctionnalités implémentées avec un langage puissant, mais imposant un déploiement sur le poste client, et une interface Web (plus limitée en possibilités graphiques) n'intégrant pas ou peu de gestion événementielle et implémentée avec un langage de script, même si cela éliminait le problème du déploiement.

Désormais, la technologie ASP.NET permet de profiter des avantages de l'approche classique tout en bénéficiant de l'architecture Web : implémentation des pages Web comme un assemblage de contrôles, utilisation de langages évolués, gestion événementielle, performance liée à la compilation (figure 3-1).

Rassurez-vous, rien de magique dans tout cela, si ce n'est une restructuration complète du modèle de programmation de pages Web dynamiques, initialement introduit par ASP et désormais principalement articulé autour des mécanismes suivants :

- les contrôles serveur ;
- le séparation entre code et contenu graphique ;
- les contrôles utilisateur ;
- la gestion des événements « côté serveur ».

Contrôle

Un contrôle est un élément graphique paramétrable ; l'utilisation de contrôles rend le code plus modulaire et accroît la productivité du développement.

Programmation événementielle

Modèle de programmation consistant à organiser le code d'une application sous la forme de gestionnaires d'événements, c'est-à-dire de traitements exécutés lorsque tel ou tel événement survient. Technique couramment utilisée lors du développement d'interfaces graphiques.

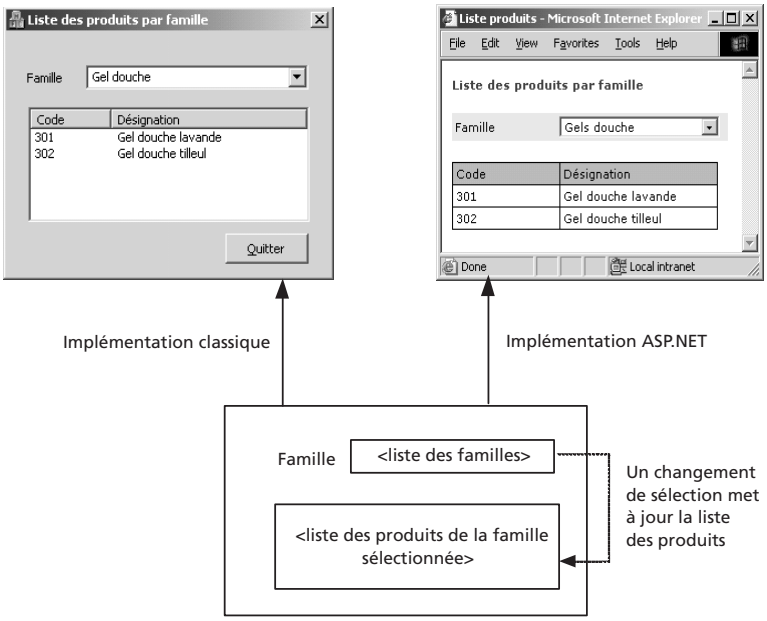


Figure 3–1 Deux implémentations d'une même vision conceptuelle

Nous allons détailler et mettre en pratique les trois premiers mécanismes dans la suite de ce chapitre, tandis que nous traiterons la gestion des événements dans le chapitre suivant.

À retenir

ASP.NET permet d'appréhender une page Web comme une interface graphique d'application client-serveur lors de la phase de développement, tout en produisant des pages HTML standards lors de la phase d'exécution.

Un contenu HTML mieux organisé grâce aux contrôles serveur

Comme pour une interface client classique, le développement d'une page ASP.NET se déroule en deux phases distinctes :

- la *réalisation de la maquette*, qui consiste à positionner les éléments graphiques sur la page en isolant, en particulier, les composants paramétrables ;
- l'*implémentation de la cinématique*, qui consiste à spécifier les actions à réaliser en fonction des événements qui surviennent.

Poursuivons avec notre exemple de page Web qui affiche la liste des produits ; cette page contient deux éléments paramétrables :

- une liste déroulante contenant les familles de produits, à remplir lors du chargement de la page ;
- un tableau de données contenant les produits de la famille sélectionnée, à remplir lors du chargement de la page et à actualiser lors de chaque changement de sélection dans la liste des familles.

APARTÉ

Un mode de programmation familier ?

Les développeurs d'applications client-serveur classiques (Visual Basic, Visual C++, Delphi…) sont familiers de ce modèle de programmation : ils découvriront dans ce chapitre comment ASP.NET l'applique dans le développement Web.
Les développeurs plus habitués à la réalisation de pages Web fondées sur des langages de scripts (VBScript, JScript…) découvriront la puissance de cette nouvelle approche, notamment en termes de productivité et de structuration du code.

L'implémentation correspondante dans une page ASP.NET consistera en un document HTML classique, dans lequel on insérera des balises spéciales pour chacun des composants paramétrables :

- une balise `<asp:DropDownList>` pour la liste des familles ;
- une balise `<asp:DataGrid>` pour la liste des produits.

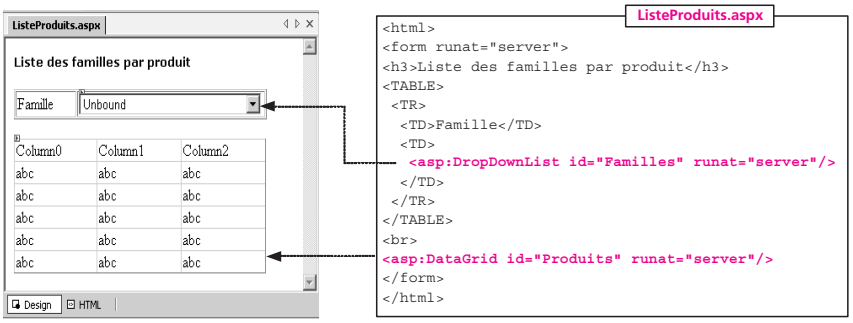


Figure 3–2 Conception de la maquette d'une page ASP.NET

Dans le vocabulaire ASP.NET, ces balises correspondent à des *contrôles serveur* : elles seront remplacées par de l'HTML généré dynamiquement lors de l'exécution de la page (figure 3-2).

Contrôle serveur

Composant graphique paramétrable, que l'on peut insérer dans une page ASP.NET et qui encapsule la génération de HTML. Les contrôles serveur participent à la séparation entre code et contenu graphique et accroissent la productivité du développement en facilitant la réutilisation.

Plus d'informations sur les contrôles serveur

Les contrôles serveur sont des éléments graphiques déclarés, positionnés dans la partie HTML de la page et accessibles sous forme d'objets dans la partie code.

La déclaration d'un contrôle serveur s'effectue de la manière suivante :

```
<asp:TypeControle runat = "server"
 id="NomDuControle"></asp:TypeControle>
```

On peut aussi employer la syntaxe abrégée :

```
<asp:TypeControle runat = "server"
 id="NomDuControle" />
```

Chaque contrôle serveur propose des propriétés permettant de contrôler son apparence et son comportement (taille, couleur, format, source de données liées, etc.). Lors de l'exécution de la page, les contrôles serveur génèrent un contenu HTML, en fonction de leur paramétrage et du navigateur utilisé par le client (adaptation du HTML généré en fonction des possibilités du navigateur).

Lors d'un « aller-retour » de la page suite à un événement donné, les contrôles serveur conservent leur valeur (grâce au mécanisme du **`VIEWSTATE`**). Enfin, il est possible d'associer des gestionnaires d'événements à un contrôle serveur.

Rassemblés dans l'espace de nommage **`System.Web.UI`**, les contrôles serveur fournis avec le framework .NET peuvent être répartis en plusieurs groupes :

- Les **contrôles HTML** : correspondant aux éléments HTML standards (boutons, zones de texte, liens…) , ils offrent l'avantage d'exposer leurs attributs sous forme de propriétés. Ils sont employés lorsqu'on souhaite contrôler l'apparence des éléments HTML depuis le code.
- Les **contrôles Web** : équivalents aux contrôles HTML standards, ils proposent des propriétés personnalisées qui aident au développement. Ils sont très fréquemment employés dans les pages ASP.NET.
- Les **contrôles de validation** : ils facilitent les opérations de validation de formulaires en vérifiant le contenu des champs (diverses règles de validation possibles) et en indiquant les champs erronés. De plus, le code de validation généré est adapté au navigateur du client.
- Les **contrôles liés à des données** : très utilisés dans les applications Web qui manipulent des données, ils permettent la présentation de listes de données et l'édition d'enregistrements.
- Les **contrôles complexes** : on regroupe dans cette catégorie les contrôles évolués tels que le calendrier ou le contrôle `AdRotator`, qui permet d'afficher des publicités en alternance.
- Les **contrôles mobiles** : ensemble de contrôles générant du contenu adapté à des terminaux mobiles (téléphone, Pocket PC…).

Enfin, notons qu'il est possible d'implémenter des contrôles serveur spécifiques ; il existe d'ailleurs un marché important de contrôles serveur proposés par des vendeurs indépendants.

Par exemple, le contrôle `DataGrid` (grille de données) sera remplacé à l'exécution par un tableau HTML :

```
<table>
   <tr>
   <td >Code</td><td>Désignation</td>
   </tr><tr>
      <td>101</td><td>Savon miel</td>
   </tr><tr>
      <td>102</td><td>Savon miel & amandes</td>
   </tr><tr>
      <td>103</td><td>Savon lavande</td>
   </tr>
</table>
```

Nous aurons l'occasion, au cours de cet ouvrage, de détailler les différents types de contrôles serveur ainsi que leurs possibilités.

Pour l'instant, nous n'avons parlé que de la partie graphique de la page… Mais où est le code qui régit la cinématique de la page ? En l'occurrence, où et comment indique-t-on que les listes de familles et de produits doivent être remplies à partir des tables correspondantes de la base de données, et qu'un changement de sélection de la famille doit provoquer le rechargement de la liste des produits ? Nous allons répondre à cette question sans plus attendre, dans le paragraphe qui suit.

Un code plus structuré grâce à la séparation du contenu HTML et de la cinématique

Avec les techniques de développement Web classique, le code qui régit le comportement de la page est *mélangé* au contenu HTML ; dans le cas d'une page ASP.NET, ce code est *séparé physiquement* du contenu HTML.

Le développeur dispose de deux techniques pour la localisation du code :

1. placer le code au sein du même fichier `.aspx` que le contenu HTML (*code in-line*) ;
2. placer le code dans un fichier séparé (*code behind*).

Alternative 1 - Placer le code et le HTML de la page dans un même fichier

Intéressons-nous dans un premier temps à la première technique, illustrée à la figure 3-3, qui est celle que nous emploierons le plus souvent dans cet ouvrage. Elle correspond à un découpage du fichier `.aspx` en deux blocs distincts :

- le bloc « HTML », composé d'un mélange de balises HTML classiques et de balises spécifiques ASP.NET correspondant généralement à des contrôles serveur ;
- le bloc « code », encadré par deux balises `<script>`, qui contient l'implémentation du paramétrage des contrôles et de la cinématique de la page ;

WEB MATRIX FACE À VS.NET

Gestion du code

Une des différences principales entre Web Matrix et Visual Studio.NET se situe dans la technique de gestion du code : Web Matrix a choisi de placer le code et le contenu HTML dans un fichier unique, tout en offrant des vues logiques separées, grâce à un système d'onglet, alors que VS.NET a adopté la technique du code behind, probablement à cause de son analogie avec Visual Basic.

SYNTAXE ASP.NET

Spécifier le langage utilisé

La directive <%@ Page Language…%> permet, entre autres, de spécifier le langage utilisé dans la page (VB.NET, C# ou JScript.NET). En l'absence de cette directive, le langage par défaut est VB.NET, sauf si les paramètres par défaut ont été changés dans le fichier web.config de l'application.

SYNTAXE ASP.NET

Position du bloc <script>

Le bloc de code est généralement placé en haut du fichier .aspx, avant le bloc de contenu HTML. Néanmoins, il est techniquement possible de le placer où on le souhaite : par exemple en dessous, voire au milieu du bloc HTML. On peut même répartir le code entre plusieurs blocs, à condition que chacun soit encadré par des balises <script>.

Notons que ces deux blocs sont, en général, précédés d'une directive <% Page …%>, facultative mais utilisée dans la majorité des cas, qui permet de spécifier un certain nombre d'attributs de la page.

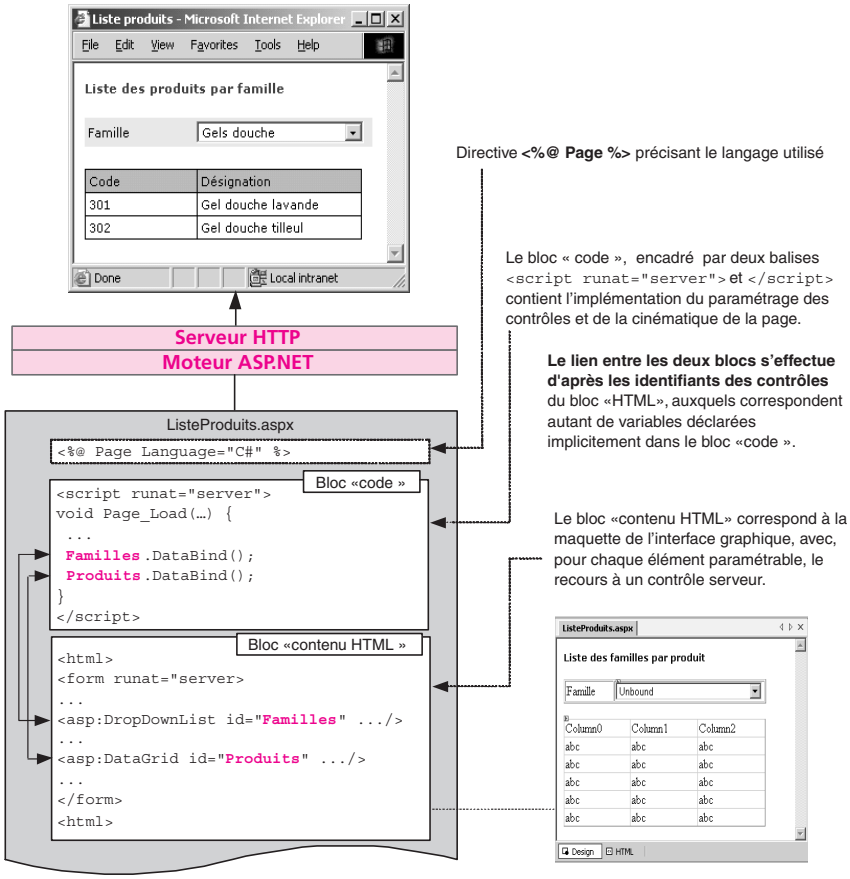


Figure 3–3 Architecture d'une page ASP.NET

ASP.NET

Compatibilité avec ASP

S'il est recommandé, pour une meilleure organisation, de placer l'intégralité du code de la page dans le bloc <script>, il est toutefois possible de placer du code au sein du contenu HTML, à l'intérieur d'un bloc <% …%>, du fait de la compatibilité ascendante avec ASP.

Sans entrer dans le détail du code, soulignons dès à présent deux points d'architecture fondamentaux :

- **Le lien entre entre les contrôles serveur et le code s'effectue par l'intermédiaire des identifiants** : à chaque contrôle serveur contenu dans le bloc « HTML » correspond une variable déclarée *implicitement* dans le bloc « code », le nom et le type de cette variable correspondant respectivement au nom et au type du contrôle.
- **Le code de la page est organisé en gestionnaires d'événements** : chaque portion de code s'exécute en réaction à un événement donné (ici, en l'occurrence, l'événement Page_Load, correspondant au chargement de la page).

Ce dernier principe nous permettra, par exemple, d'implémenter le rechargement de la liste des produits lorsque la sélection de la famille change, comme nous le verrons dans le prochain chapitre.

Alternative 2 - Placer le code de la page dans un fichier séparé

Comme on l'a indiqué plus haut, une technique alternative (dite *code behind*) consiste à placer le code associé à la page dans un fichier distinct du fichier `.aspx` principal (figure 3-4).

Le principal avantage de cette séparation du contenu graphique et du code en deux fichiers distincts est de pouvoir faire travailler en parallèle un concepteur HTML et un développeur à la production d'une même page – à condition qu'ils se soient préalablement mis d'accord sur les types et les noms des contrôles.

À quoi ça ressemble ?

Sans entrer dans le détail de la syntaxe, nous présentons ici l'extrait du code correspondant à l'affichage du contenu d'une table dans un tableau HTML – en l'occurrence, notre liste de produits – en comparant la version ASP et la version ASP.NET, afin que le lecteur puisse dès maintenant voir « à quoi ça ressemble ».

Dans la version ASP, les éléments HTML et les instructions en langage de script sont entremêlées :

Version ASP

```
<%
Set conn = Server.CreateObject("ADODB.Connection")
conn.Open "<Chaîne de connexion>"
Set rs = conn.Execute("SELECT * FROM Produit")
%>

<table border="1">
<%
Do While Not rs.EOF
   %>
   <tr>
      <td><%= rs.Fields("Code").Value %></td>
      <td><%= rs.Fields("Designation").Value %>
      </td>
   </tr>
   <%
   rs.MoveNext
Loop
%>
</table>
<%
   rs.Close
   conn.Close
%>
```

Dans la version ASP.NET, on note la séparation claire entre le code (entre les balises `<script>`) et la partie graphique (entre les balises `<html>`), ainsi que l'organisation du code en gestionnaires d'événements (ici, un seul gestionnaire : Page_Load) :

Version ASP.NET (C#)

```
<script runat="server">

void Page_Load(Object sender, EventArgs e)
{
   SqlConnection conn ;
   SqlCommand cmd;
   SqlDataReader reader;

   string SQL = "SELECT * FROM Produit";

   conn = new SqlConnection(<Chaîne de connexion>);
   conn.Open();

   cmd = new SqlCommand(SQL,conn);
   reader = cmd.ExecuteReader();

   Produits.DataSource = reader
   Produits.DataBind();
   reader.Close();
   conn.Close();
}

</script>

<html>
<asp:DataGrid id="Produits" runat="server"/>
</html>
```

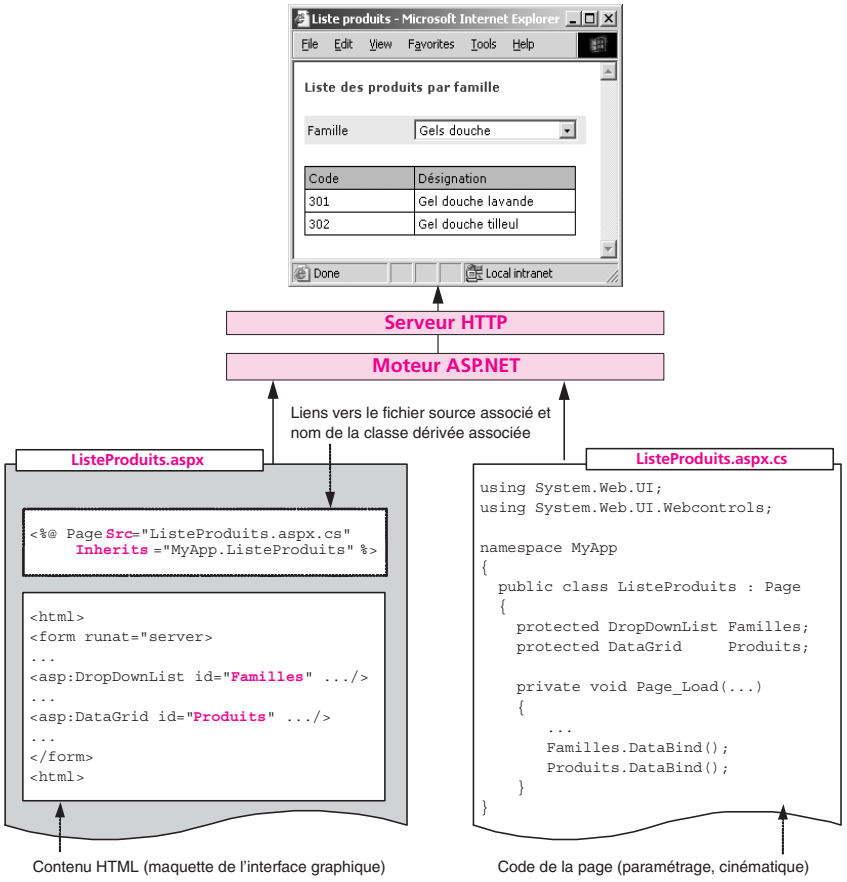


Figure 3–4 Architecture d'une page ASP.NET, avec code dans un fichier à part

Un second avantage, plus théorique : cette option fait apparaître clairement la nature technique des pages ASP.NET car le fichier code associé doit contenir la définition d'une classe dérivée de `System.Web.UI.Page` ; il faut déclarer, *explicitement* cette fois, des variables membres correspondant aux contrôles serveur placés dans le fichier `.aspx`.

Par convention, on nomme généralement le fichier code associé à une page d'après le nom du fichier `.aspx`, auquel on ajoute l'extension propre au langage (`.vb` pour VB.NET, `.cs` pour C#) : par exemple, `MyPage.aspx.vb` (ou `.cs`) pour le code associé à `MyPage.aspx`.

Le lien entre le fichier `.aspx` et le code associé s'effectue par l'intermédiaire de l'attribut `Inherits` de la directive `<%@ Page %>`, qui doit obligatoirement spécifier le nom de la classe correspondant à la page. L'attribut `Src`, facultatif, permet d'indiquer au compilateur *Just-In-Time* le nom du fichier source, qui est alors compilé à la volée lors de l'exécution de la page ; en l'absence de directive `Src`, le moteur ASP.NET recherche l'implémentation de la classe associée à la page parmi les assemblages disponibles.

Assemblage

Un assemblage (*assembly* en anglais) désigne une brique binaire élémentaire d'une application .NET. C'est en quelque sorte l'équivalent d'un composant COM dans le monde « Windows DNA » mais avec des fonctionnalités supplémentaires (regroupement logique de plusieurs fichiers DLL ou EXE dans un assemblage, gestion des versions, description de l'assemblage par des métadonnées…).

ASP.NET Différences entre Inherits, Src, et Codebehind

Il existe parfois une confusion entre ces trois attributs, tous utilisés dans la directive <%Page...%> et relatifs au placement du code dans une page à part, mais dont les significations sont différentes :

- `Inherits` est le seul attribut qui soit obligatoire dans le cas de placement du code dans un fichier séparé. Il doit spécifier le nom de la classe associée à la page (dérivée de `System.Web.UI.Page`).
- `Src` est un attribut facultatif spécifiant le chemin du fichier source associé à la classe. Si cet attribut est spécifié, le moteur ASP.NET effectuera une compilation à la volée (*JIT compilation*) de la page indiquée. Dans le cas contraire, le moteur recherchera la classe associée à la page parmi les assemblages disponibles.
- `Codebehind` est le plus trompeur des trois : bien qu'il porte le nom dont on qualifie la technique de séparation du code et du contenu, ce n'est pas un attribut ASP.NET : c'est en réalité un attribut spécifique de Visual Studio.NET utilisé par l'éditeur HTML pour savoir quel est le fichier source associé au fichier graphique.

Cette technique a néanmoins pour inconvénient de rendre le déploiement plus complexe : nombre de fichiers plus important, nécessité de compiler à l'avance les sources si l'on n'utilise pas l'attribut `Src`. En outre, notons qu'il est difficile d'échanger des pages ASP.NET entre deux outils de développement n'ayant pas adopté la même technique de gestion du code – en l'occurrence, entre Visual Studio.NET et Web Matrix.

Faciliter la réutilisation avec les contrôles utilisateur

La question de la réutilisation de « morceaux de pages » se pose fréquemment en développement Web : par exemple, on souhaite souvent placer une même barre de navigation sur toutes les pages d'un site ou utiliser un même composant (boîte de dialogue Login ou Recherche) dans plusieurs sites.

Pour faire cela avec une technologie classique comme ASP, il fallait utiliser une instruction de type `<!--#include file… -->` permettant d'inclure un fichier à un endroit donné. Ce mécanisme permettait, dans une certaine mesure, d'organiser le code mais présentait un certain nombre d'inconvénients : en particulier, il y avait risque de *collisions de noms* entre les variables du fichier principal et du fichier inclus.

Pour pallier ce type d'inconvénients, ASP.NET a introduit la notion de *contrôle utilisateur*, autrement dit de *morceau de page Web réutilisable* : du point de vue du développeur, un contrôle utilisateur s'implémente, à quelques détails près, comme une page ASP.NET normale ; du point de vue de la page utilisatrice, ce contrôle est vu comme un objet externe.

La figure 3-5 illustre l'utilisation d'un contrôle utilisateur – une barre de navigation :

- le contrôle doit être implémenté dans un fichier d'extension `.ascx` et doit comporter une directive `<% Control…%>` à la place de la directive `<% Page…%>` ;
- la page utilisatrice doit enregistrer le contrôle avec une directive `<%Register %>`, qui précise le nom du contrôle (ici : `Header`) et son espace de nommage (ici : `SDS` pour « Savons du Soleil ») ;
- enfin, il peut être fait référence au contrôle, à l'endroit où l'on souhaite l'insérer, comme pour un contrôle serveur normal : `<SDS:Header…runat="server">`.

Voici les principaux avantages offerts par les contrôles utilisateur :

- les noms utilisés par le contrôle sont *encapsulés* et ne risquent pas d'entrer en collision avec ceux de la page utilisatrice ;
- le contrôle est un objet pouvant exposer des propriétés : par exemple, l'index de la rubrique active d'une barre de navigation peut être paramétré depuis l'extérieur ;
- les contrôles sont organisés dans des espaces de nommage : plusieurs contrôles peuvent porter le même nom s'ils sont dans des espaces de nommage différents.

Contrôle utilisateur

Morceau de page Web encapsulé dans un composant graphique réutilisable. Très faciles à implémenter, les contrôles utilisateur permettent la réutilisation d'éléments graphiques constitutifs d'une application (barre de navigation, menu, etc.) et remplacent en quelque sorte les `<!-- #Include -->` des pages ASP).

Pagelets

Les contrôles utilisateur (*user controls*) sont parfois dénommés *pagelets*, que l'on peut approximativement traduire par « morceaux de page ».

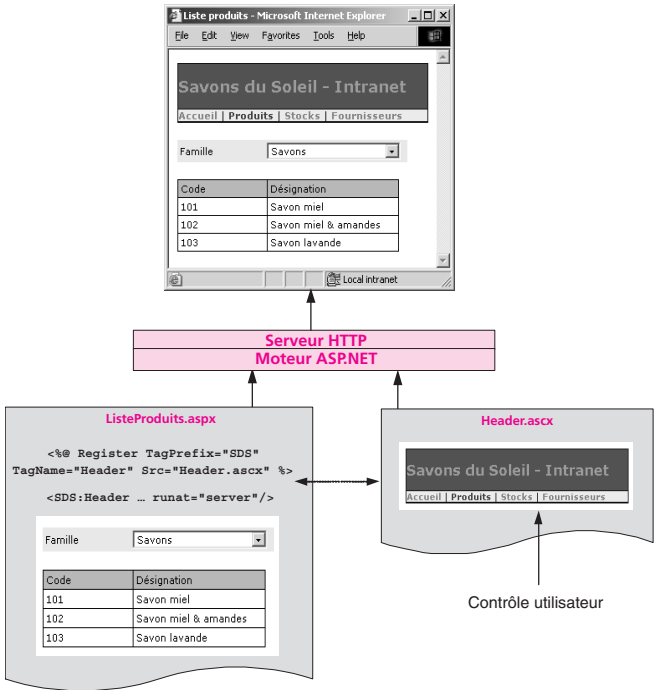


Figure 3–5 Utilisation d'un contrôle utilisateur

Il est à présent temps de mettre en pratique ces notions de contrôle serveur et contrôle utilisateur, en réalisant la barre de navigation qui sera utilisée dans notre étude de cas.

Mise en pratique : réalisation d'une barre de navigation

Dans cette seconde partie de chapitre, nous allons réaliser un contrôle utilisateur de type « barre de navigation », qui nous servira pour la suite de l'étude de cas : après une rapide prise en main de l'environnement de développement Web Matrix, nous réaliserons en deux temps la barre de navigation – partie graphique et implémentation - puis nous verrons comme l'intégrer au sein d'une page existante, en mettant en place le squelette de notre intranet.

Création de la barre de navigation

Notre projet est donc de réaliser une barre de navigation semblable à celle illustrée à la figure 3-6, qui sera présente dans toutes les pages de l'intranet et permettra à l'utilisateur de naviguer d'une rubrique à l'autre ; cette barre de navigation devra afficher le titre de la rubrique en cours et mettre en surbrillance la rubrique active.

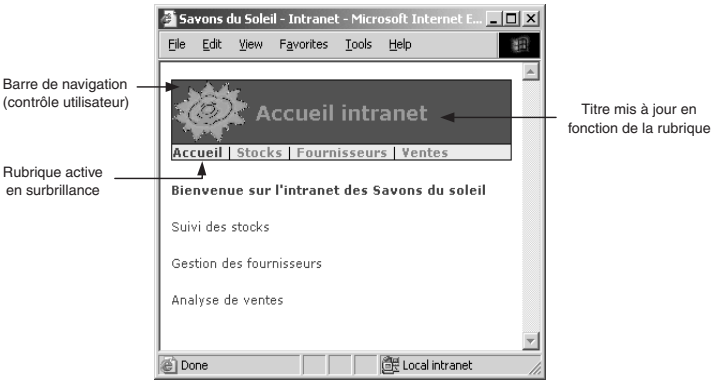


Figure 3–6 Le projet de barre de navigation

Voyons maintenant comment utiliser Web Matrix pour créer ce contrôle utilisateur.

Création d'un contrôle utilisateur avec Web Matrix

Démarrez Web Matrix (les instructions d'installation se trouvent dans le chapitre précédent) :

- une boîte de dialogue apparaît, proposant différents types de fichiers (figure 3-7) ;
- choisissez ASP.NET User Control ;
- indiquez votre répertoire de travail dans le champ Location ;
- spécifiez un nom de fichier, par exemple : `NavBar.ascx` ;
- enfin, indiquez le langage que vous souhaitez utiliser (C# ou VB.NET).

Une fois la page créée, l'interface principale de Web Matrix apparaît (figure 3-8).

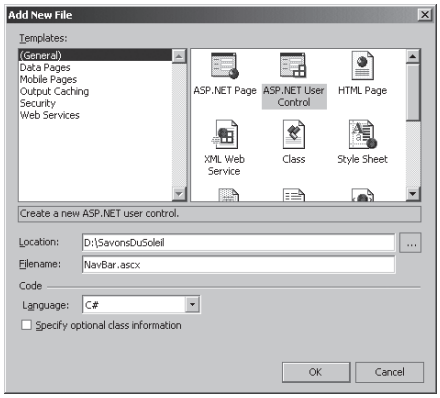


Figure 3–7 Choix du type de fichier dans Web Matrix

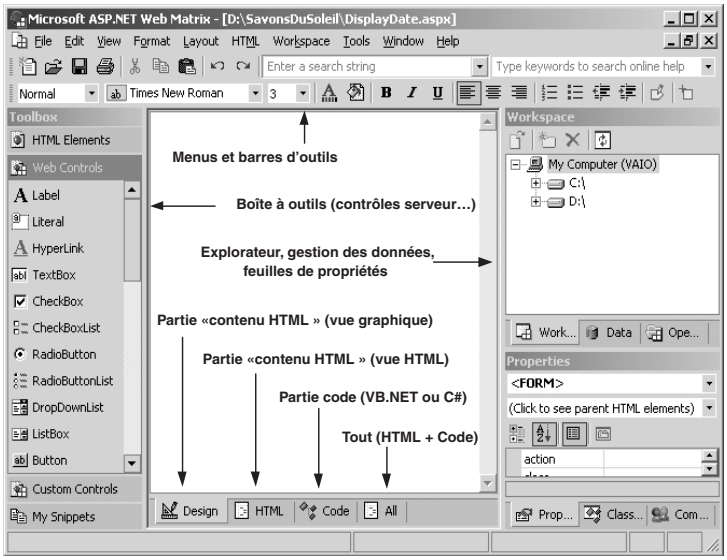


Figure 3–8 L'interface principale de Web Matrix

WEB MATRIX

Pas de Undo en mode design

À l'heure actuelle, la fonctionnalité Undo est inactive dans l'onglet Design de Web Matrix. Peut-être cette fonctionnalité sera-t-elle implémentée dans la version finale ?

La partie centrale de l'interface est occupée par un éditeur de texte, qui fait apparaître clairement la séparation entre contenu graphique et code grâce à quatre onglets :

- l'onglet Design permet une édition du contenu HTML en mode graphique ;
- l'onglet HTML contient le code HTML correspondant ;
- l'onglet Code contient le code ASP.NET de la page (VB.NET ou C#) ;
- l'onglet All affiche le fichier complet, tel qu'il est stocké sur le disque.

De part et d'autre de cette zone centrale sont situées un certain nombre de fenêtres utilitaires :

- boîtes à outils « contrôles » sur la gauche ;
- explorateurs (fichiers, données) et feuilles de propriétés sur la droite.

Pour l'instant, tous les onglets sont vides, à l'exclusion de l'onglet All qui commence par une directive destinée à indiquer au moteur ASP.NET que le contenu du fichier correspond à un contrôle utilisateur :

```
<%@ Control Language="<VotreLangage>" %>
```

Passons maintenant à la réalisation de notre barre de navigation, qui s'effectuera, comme il se doit, en deux temps : réalisation de la partie graphique puis implémentation du code.

Réalisation de la partie graphique de la barre de navigation

Pour commencer, nous allons réaliser la partie graphique de notre barre de navigation, en nous plaçant dans l'onglet Design de Web Matrix.

Figure 3–9 Partie graphique de la barre de navigation

RAPPEL **Télécharger le code source**

Tout le code source de l'application, ainsi que les divers fichiers annexes, dont les images, sont disponibles en téléchargement à l'adresse :

- http://www.savonsdusoleil.com

D'un point de vue HTML, notre contrôle est une table comportant deux lignes :

- la première contiendra le logo de la société et le titre ;
- la seconde les liens HTML vers les rubriques.

Le titre et les liens devant être paramétrables en fonction de la rubrique, nous allons les implémenter avec des *contrôles serveur*, de type `<asp:Label>` pour le titre et de type `<asp:Hyperlink>` pour les liens.

La marche à suivre pour insérer une table est très simple :

1. choisissez Insert Table dans le menu HTML : une boîte de dialogue apparaît (figure 3-10) ;
2. spécifiez le nombre de lignes et de colonnes (en l'occurrence : 2 lignes, 1 colonne).

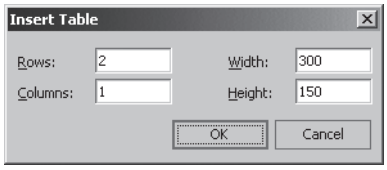


Figure 3–10 Insérer une table

Malheureusement, les possibilités de paramétrage de la taille du tableau sont relativement réduites : la boîte de dialogue n'accepte que des valeurs en pixels. Dans notre cas, nous souhaitons que la table occupe 100% de la largeur de la page et que la hauteur ne soit pas spécifiée.

Pour cela, deux solutions sont possibles :

- utiliser la boîte de dialogue Propriétés pour modifier le style de la table (figure 3-11) ;
- modifier directement le contenu HTML (`<table style="WIDTH:100%">`).

Nous allons maintenant placer une image dans la première ligne de la table. Pour cela :

1. sélectionnez l'élément Image dans la boîte à outils Eléments HTML (figure 3-12) et le faire glisser vers la première ligne de la table ;
2. utilisez la boîte de dialogue Propriétés pour spécifier la source de l'image (`soleil.gif`) ;
3. spécifiez également la valeur `center` pour l'attribut `align`.

Enfin, spécifiez deux couleurs différentes pour les deux lignes de la table, par exemple :

- couleur verte pour la ligne supérieure (logo et titre) : `bgcolor="#00c000"` ;
- couleur jaune pour la ligne inférieure (rubriques) : `bgcolor="#ffff80"`.

Nous en avons fini avec la partie purement HTML de l'interface graphique ; nous allons maintenant passer à la mise en place des contrôles serveur.

Notre barre de navigation fera appel à deux types de contrôles serveur :

- un contrôle de type `Label` pour le titre ;
- quatre contrôles de type `HyperLink` pour les liens vers les rubriques.

Pour créer ces contrôles, on utilisera la boîte à outils Contrôles Web (figure 3-13) qui, lorsqu'on sélectionne puis place un contrôle, génère automatiquement la balise `<asp :TypeControle...>` correspondante dans le contenu HTML.

Commençons par le titre :

1. sélectionnez un contrôle serveur de type `Label` dans la barre d'outils ;
2. faites-le glisser vers la première ligne de la table, à droite du soleil (voir figure 3-9) ;
3. ajustez les caractéristiques du contrôle à l'aide des barres d'outils : gras, taille de police « 5 » et couleur jaune.

Plaçons maintenant les liens hypertexte :

1. sélectionnez un contrôle serveur de type `Hyperlink` dans la barre d'outils ;
2. faites-le glisser vers la deuxième ligne de la table (voir figure 3-9) ;
3. répétez trois fois l'opération, en séparant les contrôles par une barre verticale (« | »).

Servez-vous de l'onglet HTML de Web Matrix pour voir le code qui a été généré : une balise `<asp:TypeControle id="NomControle" runat="server">` a été créée pour chaque contrôle serveur.

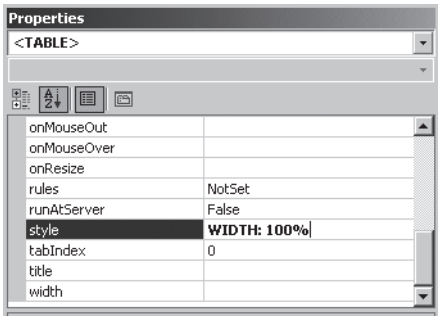


Figure 3–11 Modifier les propriétés de la table

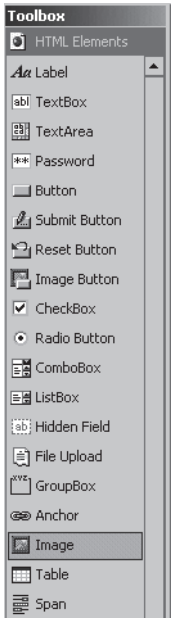


Figure 3–12 La boîte à outils Eléments HTML

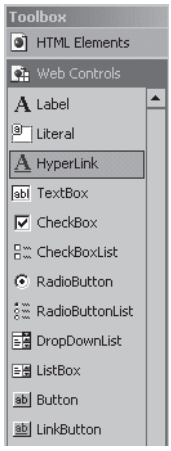


Figure 3–13 La boîte à outils Contrôles Web

ALTERNATIVE Utilisation d'un autre éditeur HTML

Si vous le préférez, il est tout à fait possible de réaliser le contenu HTML à l'aide d'un autre éditeur plus complet (Dreamweaver, FrontPage…), puis de revenir à Web Matrix pour placer les contrôles serveur et implémenter le code.

TERMINOLOGIE Contrôles Web ou contrôles serveur ?

Web Matrix regroupe sous le vocable de « contrôle Web » l'ensemble des contrôles serveur, à l'exclusion des contrôles mobiles, tandis que, dans la documentation .NET, le terme « contrôles Web » se réfère parfois uniquement aux équivalents des contrôles HTML standards (boutons, liens, zones de texte…).

Changer le nom des contrôles avec Web Matrix

Il existe deux moyens pour changer le nom d'un contrôle avec Web Matrix : modifier la rubrique (ID) dans la feuille de propriétés du contrôle, ou modifier directement la balise correspondante dans l'onglet HTML.

Les noms de contrôles générés par défaut sont de type `HyperLink1`, `HyperLink2`, etc. Pour plus de clarté, renommez-les conformément au tableau ci-dessous :

Nom du contrôle	Type	Rôle
Titre	<asp:Label>	Titre de la rubrique
Rubrique1	<asp:HyperLink>	Lien vers la rubrique « Accueil »
Rubrique2	<asp:HyperLink>	Lien vers la rubrique « Stocks »
Rubrique3	<asp:HyperLink>	Lien vers la rubrique « Fournisseurs »
Rubrique4	<asp:HyperLink>	Lien vers la rubrique « Ventes »

Voici, après renommage des contrôles, le contenu HTML de notre barre de navigation.

NavBar.ascx (partie graphique)

```
<table style="WIDTH: 100%">
   <tbody>
      <tr>
         <td bgcolor="#00c000">
            <img src="../img/soleil.gif" align="center" />
            <asp:Label id="Titre" runat="server" Font-Bold="True"
             Font-Size="Large" ForeColor="#ffff80"/>
         </td>
      </tr>
      <tr>
         <td bgcolor="#ffff80">
            <asp:HyperLink id="Rubrique1" runat="server"/> |
            <asp:HyperLink id="Rubrique2" runat="server"/> |
            <asp:HyperLink id="Rubrique3" runat="server"/> |
            <asp:HyperLink id="Rubrique4" runat="server"/> |
         </td>
      </tr>
   </tbody>
</table>
```

Et voilà, la partie graphique de notre contrôle utilisateur est terminée !
Nous allons maintenant passer dans l'onglet **Code** pour implémenter le paramétrage et la cinématique de notre barre de navigation.

Programmation de la barre de navigation

Lors du chargement de la barre de navigation, il faut effectuer les opérations suivantes :

- paramétrage des liens HTML vers les rubriques ;
- mise à jour du titre en fonction de la rubrique sélectionnée ;
- mise en surbrillance de la rubrique sélectionnée.

L'utilisateur de la barre de navigation doit avoir un moyen de spécifier la rubrique sélectionnée : ceci se fera par l'intermédiaire d'une *propriété* exposée par le contrôle utilisateur.

Pour réaliser la mise en surbrillance de la rubrique active, nous allons faire appel à une feuille de style, définissant une classe `active-rubrique` :

SDS.css (HTML)

```
body       { font-family: Verdana }
p          { font-size: 8pt  }
table      { font-size: 8pt }
h4         { font-size: 8pt; font-weight:bold; color: #000080 ; }
a          { font-size: 8pt; text-decoration:none; color: #000080 }
a.active-rubrique    { font-weight:bold }
```

Cette feuille de style (SDS comme « Savons du Soleil ») permet de définir quelques caractéristiques de la page : utilisation de la police Verdana (taille 8 points), liens en bleu non soulignés, lien en bleu gras pour la rubrique active. Avant de continuer, créez cette feuille de style avec un éditeur de texte ou avec Web Matrix et sauvegardez-la dans votre répertoire de travail.

Passons maintenant au code du contrôle, dont nous présentons ici deux versions, VB.NET et C# :

NavBar.ascx (code) – Version C#

```
public int SelectedIndex;
void Page_Load(Object sender, EventArgs e)
{
    Rubrique1.Text = "Accueil";
    Rubrique1.NavigateUrl = "default.aspx";
    Rubrique2.Text = "Stocks";
    Rubrique2.NavigateUrl = "stocks.aspx";
    Rubrique3.Text = "Fournisseurs";
    Rubrique3.NavigateUrl = "fournisseurs.aspx";
    Rubrique4.Text = "Ventes";
    Rubrique4.NavigateUrl = "ventes.aspx";
    switch (SelectedIndex)
    {
        case 0 :
            Titre.Text = "Accueil Intranet";
            Rubrique1.CssClass = "active-rubrique";
        break;
        case 1 :
            Titre.Text = "Suivi des stocks";
            Rubrique2.CssClass = "active-rubrique";
        break;
        case 2 :
            Titre.Text = "Gestion des fournisseurs";
            Rubrique3.CssClass = "active-rubrique";
        break;
        case 3 :
            Titre.Text = "Analyse des ventes";
            Rubrique4.CssClass = "active-rubrique";
        break;
    }
}
```

Propriété publique spécifiant la rubrique active

Mise à jour des liens

Mise à jour du titre

La propriété CssClass

Cette propriété permet de paramétrer la classe de style (CSS : *Cascading Style Sheet*) d'un contrôle serveur et correspond à l'attribut `class` d'une balise HTML. En l'occurrence, dans notre exemple, le contenu HTML généré pour le lien actif sera le suivant : `<a class="active-rubrique"...>`

Pour plus d'information sur les feuilles de style :

- http://www.w3.org/style/css

NavBar.ascx (code) – Version VB.NET

Propriété publique spécifiant la rubrique active

Mise à jour des liens

Mise à jour du titre

```
Public SelectedIndex As Int32
Sub Page_Load(sender As Object,e As EventArgs)
    Rubrique1.Text = "Accueil"
    Rubrique1.NavigateUrl = "default.aspx"
    Rubrique2.Text = "Stocks"
    Rubrique2.NavigateUrl = "stocks.aspx"
    Rubrique3.Text = "Fournisseurs"
    Rubrique3.NavigateUrl = "fournisseurs.aspx"
    Rubrique4.Text = "Ventes"
    Rubrique4.NavigateUrl = "ventes.aspx"
    Select Case SelectedIndex
        Case 0
            Titre.Text = "Accueil Intranet"
            Rubrique1.CssClass = "active-rubrique"
        case 1
            Titre.Text = "Suivi des stocks"
            Rubrique2.CssClass = "active-rubrique"
        case 2
            Titre.Text = "Gestion des fournisseurs"
            Rubrique3.CssClass = "active-rubrique"
        case 3
            Titre.Text = "Analyse des ventes"
            Rubrique4.CssClass = "active-rubrique"
    End Select
End Sub
```

L'examen de ce code est relativement rapide.

On commence par déclarer une propriété publique `SelectedIndex`, qui sera accessible depuis l'extérieur du contrôle et permettra à l'utilisateur de spécifier l'index de la rubrique active (entre 0 et 3) depuis la page qui incluera le contrôle.

Puis, dans le gestionnaire `Page_Load` exécuté lors du chargement de la page, on paramètre les liens hypertextes – mise à jour du texte et de l'adresse de lien via les propriétés `Text` et `NavigateUrl` – on spécifie le titre – mise à jour de la propriété `Text` du contrôle `Titre` – puis on attribue à la rubrique active le style `active-rubrique`, défini dans la feuille de style grâce la propriété `CssClass`.

Notre barre de navigation est maintenant terminée. Néanmoins, il n'est pas possible de la tester individuellement : il nous faut obligatoirement l'intégrer dans une page ASP.NET. C'est ce que nous allons faire dans le paragraphe qui suit en créant le squelette de notre intranet.

À propos de Page_Load

Dans le chapitre suivant, nous aurons l'occasion de parler plus amplement des événements liés à une page : `Page_Init`, `Page_Load`, etc. Dans un premier temps, retenons uniquement que `Page_Load` est appelé lors du chargement de la page.

Aparté pour les programmeurs objet

L'objet exécutable qui produit le code HTML correspondant à cette page est une instance d'une classe dérivée de `System.Web.UI.Page`. `Page_Load` est une fonction virtuelle redéfinie dans cette classe dérivée. Enfin, les contrôles serveur sont implicitement déclarés comme des variables membres (propriétés) privées .

Création du squelette de l'intranet

Notre intranet sera composé de quatre rubriques principales :

- page d'accueil ;
- suivi des stocks ;

- gestion des fournisseurs ;
- analyse des ventes.

La barre de navigation devra apparaître en haut de chacune des pages, la rubrique active étant sélectionnée (voir figure 3-14).

Commençons par réaliser la page d'accueil : pour cela, créer dans Web Matrix un nouveau fichier nommé `default.aspx`, en choisissant cette fois le type `ASP.NET Page` (c'est le choix par défaut).

L'intégration de la barre de navigation en haut de la page se fait en deux étapes :

1. enregistrement du contrôle utilisateur avec l'instruction `<% Register…%>` ;
2. positionnement du contrôle dans la page.

L'enregistrement du contrôle, indispensable à son utilisation dans la page, s'effectue à l'aide de l'instruction suivante, placée en haut du fichier (juste après la directive `<%@ Page …%>`) :

```
<%@ Register TagPrefix="SDS" TagName="NavBar" Src="NavBar.ascx" %>
```

Les significations des différents attributs, tous obligatoires, sont les suivantes :

- `TagPrefix` indique le préfixe qui sera utilisé pour référencer le contrôle, autrement dit l'espace de nommage auquel il appartient ;
- `TagName` spécifie le nom du contrôle ;
- `Src` indique le nom du fichier source correspondant au contrôle.

L'enregistrement effectué, on peut maintenant inclure la balise correspondant au contrôle, à l'endroit où l'on souhaite que la barre de navigation soit insérée :

```
<SDS:NavBar id="MyNavBar" SelectedIndex="0" runat="server">
</SDS:NavBar>
```

On note évidemment la similitude avec une balise de contrôle serveur, la différence étant qu'on a remplacé `<asp:TypeControle…>` par `<TagPrefix:TagName…>`.

Autre point important, *la propriété publique* `SelectedIndex`, déclarée dans le code du contrôle, *est accessible sous forme d'attribut de la balise du contrôle* ; dans notre cas, l'index correspondant à la page d'accueil est 0.

Terminons la réalisation de notre page d'accueil en insérant un lien vers la feuille de style externe `SDS.css` définie plus haut, indispensable pour la mise à jour de la rubrique active :

```
<link href="SDS.css" type="text/css" rel="stylesheet" />
```

Enfin, pour remplir la page, on se propose d'ajouter des liens vers les autres rubriques dans le corps de la page.

Figure 3–14 Page d'accueil de l'intranet

À propos de TagPrefix

`TagPrefix` contient généralement le nom de l'application ou le nom de la société (ici : SDS pour « Savons du Soleil »). Le principal intérêt de cet attribut est de permettre l'existence de plusieurs contrôles portant le même nom, à condition qu'ils soient dans des espaces de nommage distincts.

Le nom du contrôle est fixé depuis la page cliente

Notons que le nom de la classe du contrôle est fixé par la page utilisatrice, via les attributs `TagPrefix` et `TagName`, contrairement à ce qui se passe dans une interface classique, de type Visual Basic, où seul le nom de la variable correspondant au contrôle est paramétrable par l'utilisateur.

Voici le code complet de la partie graphique de la page d'accueil :

default.aspx (partie graphique)

```
<%@ Page Language="<VotreLangage>" Debug="true" %>
<%@ Register TagPrefix="SDS" TagName="NavBar" Src="NavBar.ascx" %>
<html>
<head>
    <title>Savons du Soleil - Intranet</title>
    <link href="SDS.css" type="text/css" rel="stylesheet" />
</head>
<body>
    <SDS:NavBar id="MyNavBar" SelectedIndex="0" runat="server">
    </SDS:NavBar>
    <p></p><p></p>
    <form runat="server">
        <h4>Bienvenue sur l'intranet des Savons du soleil
        </h4>
        <p><a href="./Stocks.aspx">Suivi des stocks</a> </p>
        <p><a href="./Fournisseurs.aspx">Gestion des fournisseurs</a>
        </p>
        <p><a href="./Ventes.aspx">Analyse des ventes</a></p>
    </form>
</body>
</html>
```

Il reste à répéter trois fois l'opération pour les autres rubriques, en modifiant uniquement le titre de la page, le nom du fichier et la valeur de `SelectedIndex`.

Rubrique	Nom du fichier	Valeur de SelectedIndex
Suivi des stocks	stocks.aspx	1
Gestion des fournisseurs	fournisseurs.aspx	2
Analyse des ventes	ventes.aspx	3

À titre d'exemple, voici le code de la partie graphique du fichier `stocks.aspx` :

stocks.aspx (partie graphique)

```
<%@ Page Language="<VotreLangage>" Debug="true" %>
<%@ Register TagPrefix="SDS" TagName="NavBar" Src="NavBar.ascx" %>
<html>
<head>
    <title>Suivi des stocks</title>
    <link href="SDS.css" type="text/css" rel="stylesheet" />
</head>
<body>
    <SDS:NavBar id="MyNavBar" runat="server" SelectedIndex="1">
    </SDS:NavBar>
    <p></p> <p></p>
    <form runat="server">
        <h4>Bienvenue dans le module de suivi des stocks </h4>
    </form>
</body>
</html>
```

Le développement du squelette de notre application est terminé : il ne reste plus qu'à le tester !

Tester l'application

Pour tester notre application, deux options sont possibles :

- utiliser le serveur HTTP léger fourni avec Web Matrix ;
- utiliser le serveur IIS.

À propos de la compilation des pages ASP.NET

Si vous avez fait une erreur dans la syntaxe du code (par exemple, vous avez omis le « ; » final d'une instruction en C#), une erreur de compilation se produira : vous verrez alors apparaître un écran similaire à celui représenté à la figure 3-15.

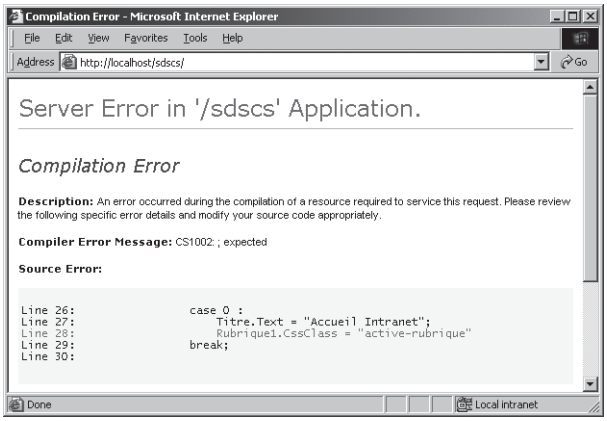


Figure 3–15 Erreur de compilation

Ceci permet au passage de souligner un des avantages du caractère *compilé* des pages ASP.NET : l'*intégralité* du code est passée en revue avant la première exécution de la page et *toutes* les erreurs de syntaxe sont détectées lors de cette phase de compilation, alors qu'avec une page interprétée (de type ASP), seules les erreurs dans les éléments interprétés étaient détectées (à l'exclusion, par exemple, des erreurs dans les branches non exécutées du code).

Néanmoins, la compilation n'élimine pas totalement le risque d'erreur à l'exécution, des *exceptions* pouvant toujours se produire.

Le résultat de la compilation de la page est un fichier exécutable – un assemblage dans le langage .NET – stocké sous la forme d'une DLL dans le répertoire Temporary ASP.NET Files situé sous le répertoire Microsoft.NET dans le dossier système (figure 3-16).

Si vous jetez un œil à ce dossier – ou plus précisément dans celui des ses sous-dossiers qui porte le nom de votre application – vous y trouverez les fichiers `.dll` correspondant aux résultats de la compilation de

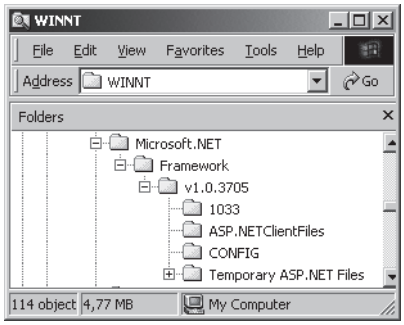


Figure 3–16 Le dossier Temporary ASP.NET Files

vos pages : chacun de ces fichiers est, en quelque sorte, un exécutable capable de « cracher » le HTML correspondant à la page ou au contrôle utilisateur.

Ces fichiers contiennent du langage IL (*Intermediate Language*) qu'il est possible de désassembler en utilisant l'utilitaire `ildasm.exe` situé dans le répertoire FrameworkSDK (figure 3-17).

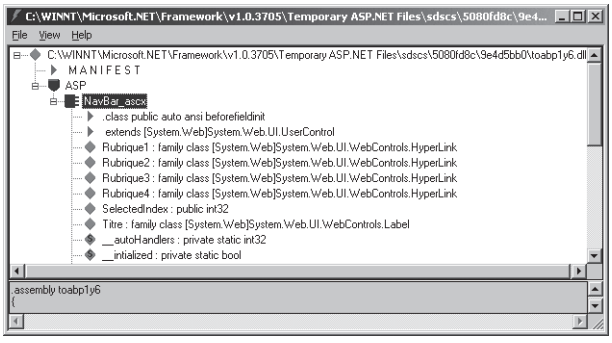


Figure 3–17 Désassemblage de la barre de navigation

Rappelons pour finir que la compilation n'intervient que lors de la première requête vers la page ; lors de requêtes ultérieures, l'assemblage en cache est réutilisé, sauf si le code source a été modifié entre temps.

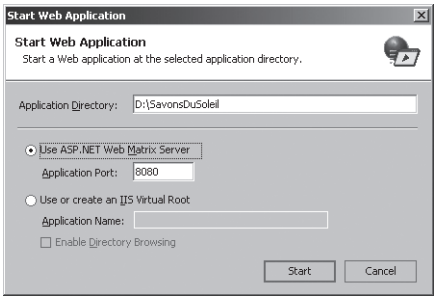


Figure 3–18 Choix du serveur HTTP

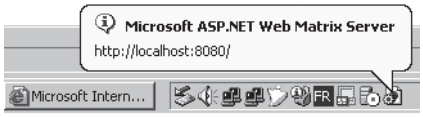


Figure 3–19 Démarrage du serveur Web Matrix

Positionnez-vous sur la page d'accueil et sélectionnez Start dans le menu View, ou utilisez la touche de raccourci F5, pour faire apparaître la boîte de dialogue permettant d'effectuer le choix du serveur HTTP (figure 3-18).

La première option permet de sélectionner le serveur Web Matrix (figure 3-19) qui a pour principal avantage d'éviter l'installation d'IIS, bien qu'il soit plus limité fonctionnellement (pas de racines virtuelles, de pages d'erreurs personnalisées, etc.).

La seconde option permet d'utiliser le serveur IIS : dans la case Application Name, saisissez le nom de la racine virtuelle de votre application. Si elle n'existe pas, Web Matrix la créera.

En résumé…

Dans ce long chapitre, nous avons présenté les principes fondamentaux de la technologie ASP.NET :

- le *modèle de programmation orienté interface* qui permet d'appréhender une page Web comme une interface graphique classique : un assemblage de contrôles graphiques réagissant à des événements ;
- la *séparation entre partie graphique et code*, qui rend le contenu HTML plus clair car il n'est plus entremêlé avec des scripts et peut faciliter, le cas échéant, la répartition du travail entre développeurs et concepteurs (designers) ;
- les *contrôles serveur*, qui augmentent la productivité du développement en encapsulant la génération de HTML et peuvent réagir à des événements ;
- les *contrôles utilisateur* qui constituent un moyen efficace et simple de réutilisation de composants graphiques.

Puis nous avons mis en œuvre ces mécanismes pour réaliser la maquette de notre intranet, en effectuant au passage un prise en main de l'environnement de développement Web Matrix.

Dans le chapitre suivant, nous allons implémenter le module de suivi des stocks de notre application, qui sera l'occasion d'illustrer deux mécanismes importants :

- l'accès à une base de données depuis une application ASP.NET ;
- la gestion des événements côté serveur.